

**GL250
ENTERPRISE
LINUX SYSTEMS
ADMINISTRATION
RHEL7 SLES12**

The contents of this course and all its modules and related materials, including handouts to audience members, are copyright ©2017 Guru Labs L.C.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Guru Labs.

This curriculum contains proprietary information which is for the exclusive use of customers of Guru Labs L.C., and is not to be shared with personnel other than those in attendance at this course.

This instructional program, including all material provided herein, is supplied without any guarantees from Guru Labs L.C. Guru Labs L.C. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

Photocopying any part of this manual without prior written consent of Guru Labs L.C. is a violation of federal law. This manual should not appear to be a photocopy. If you believe that Guru Labs training materials are being photocopied without permission, please email Alert@gurulabs.com or call 1-801-298-5227.

Guru Labs L.C. accepts no liability for any claims, demands, losses, damages, costs or expenses suffered or incurred howsoever arising from or in connection with the use of this courseware. All trademarks are the property of their respective owners.

Version: GL250S-R7S12-P06

Table of Contents

Chapter 1		
LINUX KERNEL & DEVICES	1	
Hardware Discovery Tools	2	
Configuring New Hardware with hwinfo	3	
Kernel Hardware Info – /sys/ /sys/ Structure	4	
udev	5	
Managing Linux Device Files	6	
List Block Devices	8	
SCSI Devices	11	
USB Devices	13	
USB Architecture	15	
Kernel Modules	17	
Configuring Kernel Components and Modules	19	
Handling Module Dependencies	21	
Configuring the Kernel via /proc/ Console	22	
Virtual Terminals	23	
Keyboard & locale configuration	25	
Serial Ports	27	
Random Numbers and /dev/random	29	
Lab Tasks	31	
1. Adjusting Kernel Options	32	
2. Linux Kernel Driver Compilation	34	
3. Introduction to Troubleshooting Labs	35	
4. Troubleshooting Practice: Kernel Modules	35	
Chapter 2	41	
SYSTEMD OVERVIEW	45	
System Boot Method Overview	1	
systemd System and Service Manager	2	
Modifying systemd services	3	
Systemd Service Sandboxing Features	5	
systemd Targets	6	
Using systemd	7	
Linux Runlevels Aliases	8	
Legacy Support for SysV init	10	
Lab Tasks	11	
1. Managing Services With Systemd's systemctl	12	
2. Creating a systemd unit file	13	
Chapter 3	19	
GRUB2/SYSTEMD BOOT PROCESS	1	
Booting Linux on PCs	2	
GRUB 2	4	
GRUB 2 Configuration	6	
GRUB 2 Security	8	
Boot Parameters	10	
Initial RAM Filesystem	12	
init	15	
Systemd local-fs.target and sysinit.target	16	
Systemd basic.target and multi-user.target	18	
Legacy local bootup script support	20	
System Configuration Files	21	
RHEL7 Configuration Utilities	22	
SLES12 Configuration Utilities	23	
Shutdown and Reboot	24	
Lab Tasks	25	
1. Boot Process	26	
2. Booting directly to a bash shell	30	
3. GRUB Command Line	33	
4. Basic GRUB Security	36	
5. Troubleshooting Practice: Boot Process	39	
Chapter 4	45	
SOFTWARE MAINTENANCE	1	
Managing Software	2	
RPM Features	3	
RPM Architecture	4	
RPM Package Files	5	
Working With RPMs	6	
Querying and Verifying with RPM	7	
Updating the Kernel RPM	9	
Dealing With RPM & Yum Digest Changes	10	
Yum Plugins & RHN Subscription Manager	11	
YUM Repositories	12	
YUM Repository Groups	13	
Compiling/Installing from Source	14	
Manually Installed Shared Libraries	16	
Rebuilding Source RPM Packages	17	
Lab Tasks	19	
1. Managing Software with RPM	20	

2. Creating a Custom RPM Repository	24	Advanced LVM: Striping & Mirroring	16
3. Querying the RPM Database	28	Advanced LVM: RAID Volumes	17
4. Installing Software via RPM & Source and Rebuilding SRPMs	32	SLES Graphical Disk Tool	18
5. Troubleshooting Practice: Package Management	36	RAID Concepts	19
Chapter 5		Array Creation with mdadm	20
LOCAL STORAGE ADMINISTRATION	1	Software RAID Monitoring	21
Partitioning Disks with fdisk & gdisk	2	Software RAID Control and Display	22
Resizing a GPT Partition with gdisk	5	Lab Tasks	23
Partitioning Disks with parted	8	1. Creating and Managing LVM Volumes	24
Non-Interactive Disk Partitioning with sfdisk	9	2. Creating LVM Thin Volumes	34
Filesystem Creation	10	3. Troubleshooting Practice: LVM	41
Persistent Block Devices	11	4. Creating and Managing a RAID-5 Array	42
Mounting Filesystems	12	Chapter 7	
Resizing Filesystems	14	REMOTE STORAGE ADMINISTRATION	1
Filesystem Maintenance	15	Remote Storage Overview	2
Managing an XFS Filesystem	18	Remote Filesystem Protocols	4
Swap	20	Remote Block Device Protocols	5
Filesystem Structures	21	File Sharing via NFS	7
Determining Disk Usage With df and du	22	NFSv4+	8
Configuring Disk Quotas	23	NFS Clients	9
Setting Quotas	24	NFS Server Configuration	10
Viewing and Monitoring Quotas	25	YaST NFS Server Administration	12
Filesystem Attributes	26	Implementing NFSv4	13
Lab Tasks	27	AutoFS	15
1. Creating and Managing Filesystems	28	AutoFS Configuration	16
2. Hot Adding Swap	35	Accessing Windows/Samba Shares from Linux	18
3. Setting User Quotas	37	SAN Multipathing	19
Chapter 6		Multipath Configuration	20
LVM & RAID	1	Multipathing Best Practices	22
Logical Volume Management	2	iSCSI Architecture	24
Implementing LVM	3	Open-iSCSI Initiator Implementation	27
Creating Logical Volumes	4	iSCSI Initiator Discovery	29
Activating LVM VGs	5	iSCSI Initiator Node Administration	31
Exporting and Importing a VG	6	Mounting iSCSI Targets at Boot	33
Examining LVM Components	7	iSCSI Multipathing Considerations	34
Changing LVM Components	8	Lab Tasks	36
Advanced LVM Overview	10	1. Using autofs	37
Advanced LVM: Components & Object Tags	11	2. NFS Server Configuration	42
Advanced LVM: Automated Storage Tiering	12	3. iSCSI Initiator Configuration	47
Advanced LVM: Thin Provisioning	14	4. Multipathing with iSCSI	55

Chapter 8		
USER/GROUP ADMINISTRATION		10
Approaches to Storing User Accounts	1	11
User and Group Concepts	2	12
User Administration	3	13
Modifying Accounts	4	14
Group Administration	6	16
Password Aging	7	18
Default User Files	9	19
Controlling Login Sessions	11	20
RHEL DS Client Configuration	12	22
SLES DS Client Configuration	14	24
System Security Services Daemon (SSSD)	16	25
Lab Tasks		27
1. User and Group Administration	18	28
2. Using LDAP for Centralized User Accounts	20	29
3. Troubleshooting Practice: Account Management	21	31
	24	32
	30	33
Chapter 9		
PLUGGABLE AUTHENTICATION MODULES (PAM)		34
PAM Overview	1	39
PAM Module Types	2	48
PAM Order of Processing	3	53
PAM Control Statements	4	59
PAM Modules	6	61
pam_unix	7	
pam_nologin.so	9	
pam_limits.so	10	
pam_wheel.so	11	
pam_xauth.so	12	
Lab Tasks		
1. Restricting superuser access to wheel group membership	13	
2. Using pam_nologin to Restrict Logins	14	
3. Setting Limits with the pam_limits Modules	15	
4. Using pam_limits to Restrict Simultaneous Logins	17	
	21	
	25	
Chapter 10		
SECURITY ADMINISTRATION		1
Security Concepts	2	19
Tightening Default Security	4	21
SuSE Security Checker	6	22
Security Advisories	7	25
Fine Grained Authorizations with Polkit	8	27
		29
File Access Control Lists		
Manipulating FACLs	1	
Viewing FACLs	2	
Backing Up FACLs	3	
File Creation Permissions with umask	4	
User Private Group Scheme	6	
Alternatives to UPG	7	
AppArmor	9	
SELinux Security Framework	11	
SELinux Modes	12	
SELinux Commands	14	
Choosing an SELinux Policy	16	
SELinux Booleans	18	
Permissive Domains	20	
SELinux Policy Tools	21	
SUSE Basic Firewall Configuration	24	
FirewallD	26	
Lab Tasks		27
1. User Private Groups	28	
2. Using Filesystem ACLs	29	
3. Exploring AppArmor [S12]	30	
4. Exploring SELinux Modes	31	
5. SELinux File Contexts [R7]	32	
6. SELinux Contexts in Action [R7]	33	
Chapter 11		
BASIC NETWORKING		1
IPv4 Fundamentals	2	2
TCP/UDP Fundamentals	3	3
Linux Network Interfaces	4	4
Ethernet Hardware Tools	6	6
Network Configuration with ip Command	8	8
Configuring Routing Tables	9	9
IP to MAC Address Mapping with ARP	12	12
Starting and Stopping Interfaces	13	13
NetworkManager	15	
DNS Clients	17	
DHCP Clients	19	
SUSE YaST Network Configuration Tool	21	
Network Diagnostics	22	
Information from ss and netstat	25	
Hardware and System Clock	27	
Managing Network-Wide Time	29	

Continual Time Sync with NTP	31	Chapter 14	
Configuring NTP Clients	32	MONITORING & TROUBLESHOOTING	1
Useful NTP Commands	34	System Status – Memory	2
Lab Tasks	35	System Status – I/O	3
1. Network Discovery	36	System Status – CPU	4
2. Basic Client Networking	39	Performance Trending with sar	6
3. NTP Client Configuration	43	Determining Service to Process Mapping	7
		Real-time Monitoring of Resources — Cgroups	8
		Troubleshooting Basics: The Process	9
Chapter 12		Troubleshooting Basics: The Tools	11
ADVANCED NETWORKING	1	strace and ltrace	15
Multiple IP Addresses	2	Common Problems	17
Configuring a DHCP server	4	Troubleshooting Incorrect File Permissions	18
IPv6	6	Inability to Boot	19
Interface Aggregation	8	Typos in Configuration Files	20
Interface Bonding	9	Corrupt Filesystems	21
Network Teaming	11	RHEL7 Rescue Environment	22
Interface Bridging	15	SUSE Rescue Environment	24
802.1q VLANs	17		
Tuning Kernel Network Settings	19	Lab Tasks	26
Lab Tasks	20	1. System Activity Reporter	27
1. Multiple IP Addresses Per Network Interface	21	2. Cgroup for Processes	32
2. Configuring IPv6	26	3. Recovering Damaged MBR	38
3. Troubleshooting Practice: Networking	30		
Chapter 13		Appendix A	
LOG FILE ADMINISTRATION	1	PRE-INSTALLATION CONSIDERATIONS	1
System Logging	2	Pre-Installation Considerations	2
systemd Journal	4	Hardware Compatibility	3
systemd Journal's journalctl	6	Multi-OS Booting	4
Secure Logging with Journal's Log Sealing	8	Partition Considerations	5
gnome-system-log	10	Filesystem Planning	6
Rsyslog	11	Selecting a Filesystem	7
/etc/rsyslog.conf	12		
Log Management	15	Appendix B	
Log Anomaly Detector	16	INSTALLING RHEL7	1
Sending logs from the shell	17	Anaconda: An Overview	2
Lab Tasks	18	Anaconda: Booting the System	5
1. Using the systemd Journal	19	Anaconda: Common Boot Options	6
2. Setting up a Full Debug Logfile	24	Anaconda: Loading Anaconda and Packages	7
3. Remote Syslog Configuration	26	Anaconda: Storage Options	8
4. Remote Rsyslog TLS Configuration	32	Anaconda: Troubleshooting	9
		FirstBoot	10
		Kickstart	11
		Network Booting with PXE	13

A Typical Install	15	Managing Optical Media	4
Lab Tasks	21	Tape Libraries	7
1. Linux Installation [R7]	22	Backup Examples	9
2. Automating Installation with Kickstart [R7]	27	Lab Tasks	10
		1. Using rsync and ssh for Backups	11
		2. Using tar for Backups	15
		3. Using cpio for Backups	17
		4. Creating ISO Images for Backups	19
		5. Using dump and restore for Backups	21
Appendix C			
INSTALLING SLES12	1		
YaST Install Program Interface	2		
Network Installation	3		
SLP for SUSE Linux Installation	5		
Installation Choices	6		
Kernel Crash Dump Configuration	7		
Network Booting with PXE	8		
Creating AutoYaST2 Files	10		
Using AutoYaST2 files	11		
linuxrc Automation	12		
Installation Diagnostics	13		
After The First Reboot	14		
A Typical Install	15		
Lab Tasks	21		
1. SUSE Linux Enterprise Server Installation [S12]	22		
2. Automating Installation with AutoYaST [S12]	28		
Appendix D			
MANAGE VIRTUAL MACHINES	1		
Virtualization: What and Why?	2		
Introducing libvirt	4		
libvirt: Basic Concepts	5		
libvirt: Storage Architecture	6		
libvirt: Network Architecture	8		
libvirt: Graphical Tools	10		
libvirt: Command Line Tools	12		
virsh: Basics	13		
virsh: Common Tasks	15		
virt-install	17		
Virtual Machine Guest Tools & Drivers	19		
libguestfs and guestfish	21		
Lab Tasks	23		
1. Installing a Virtual Machine	24		
Appendix E			
BACKUPS	1		
Backup Software	2		

Typographic Conventions

The fonts, layout, and typographic conventions of this book have been carefully chosen to increase readability. Please take a moment to familiarize yourself with them.

A Warning and Solution

A common problem with computer training and reference materials is the confusion of the numbers "zero" and "one" with the letters "oh" and "ell". To avoid this confusion, this book uses a fixed-width font that makes each letter and number distinct.

Typefaces Used and Their Meanings

The following typeface conventions have been followed in this book:

fixed-width normal ⇒ Used to denote file names and directories. For example, the `/etc/passwd` file or `/etc/sysconfig/directory`. Also used for computer text, particularly command line output.

fixed-width italic ⇒ Indicates that a substitution is required. For example, the string `stationX` is commonly used to indicate that the student is expected to replace `X` with his or her own station number, such as `station3`.

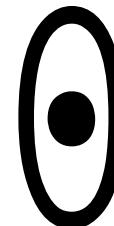
fixed-width bold ⇒ Used to set apart commands. For example, the `sed` command. Also used to indicate input a user might type on the command line. For example, `ssh -X station3`.

fixed-width bold italic ⇒ Used when a substitution is required within a command or user input. For example, `ssh -X stationX`.

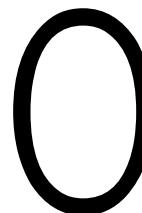
fixed-width underlined ⇒ Used to denote URLs. For example, <http://www.gurulabs.com/>.

variable-width bold ⇒ Used within labs to indicate a required student action that is not typed on the command line.

Occasional variations from these conventions occur to increase clarity. This is most apparent in the labs where bold text is only used to indicate commands the student must enter or actions the student must perform.



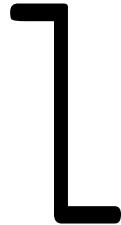
The number
"zero".



The letter
"oh".



The number
"one".



The letter
"ell".

Typographic Conventions

Terms and Definitions

The following format is used to introduce and define a series of terms:

deprecate ⇒ To indicate that something is considered obsolete, with the intent of future removal.

frob ⇒ To manipulate or adjust, typically for fun, as opposed to tweak.

grok ⇒ To understand. Connotes intimate and exhaustive knowledge.

hork ⇒ To break, generally beyond hope of repair.

hosed ⇒ A metaphor referring to a Cray that crashed after the disconnection of coolant hoses. Upon correction, users were assured the system was rehosed.

mung (or munge) ⇒ Mash Until No Good: to modify a file, often irreversibly.

troll ⇒ To bait, or provoke, an argument, often targeted towards the newbie. Also used to refer to a person that regularly trolls.

twiddle ⇒ To make small, often aimless, changes. Similar to frob.

When discussing a command, this same format is also used to show and describe a list of common or important command options. For example, the following **ssh** options:

-X ⇒ Enables X11 forwarding. In older versions of OpenSSH that do not include **-Y**, this enables trusted X11 forwarding. In newer versions of OpenSSH, this enables a more secure, limited type of forwarding.

-Y ⇒ Enables trusted X11 forwarding. Although less secure, trusted forwarding may be required for compatibility with certain programs.

Representing Keyboard Keystrokes

When it is necessary to press a series of keys, the series of keystrokes will be represented without a space between each key. For example, the following means to press the "j" key three times: **j|j|j**

When it is necessary to press keys at the same time, the combination will be represented with a plus between each key. For example, the following means to press the "ctrl," "alt," and "backspace" keys at the same time:

Ctrl|Alt+Backspace. Uppercase letters are treated the same: **Shift+A**

Line Wrapping

Occasionally content that should be on a single line, such as command line input or URLs, must be broken across multiple lines in order to fit on the page. When this is the case, a special symbol is used to indicate to the reader what has happened. When copying the content, the line breaks should not be included. For example, the following hypothetical PAM configuration should only take two actual lines:

```
password required /lib/security/pam_cracklib.so retry=3→  
    type= minlen=12 dcredit=2 ucredit=2 lcredit=0 ocredit=2  
password required /lib/security/pam_unix.so use_authtok
```

Representing File Edits

File edits are represented using a consistent layout similar to the unified **diff** format. When a line should be added, it is shown in bold with a plus sign to the left. When a line should be deleted, it is shown struck out with a minus sign to the left. When a line should be modified, it is shown twice. The old version of the line is shown struck out with a minus sign to the left. The new version of the line is shown below the old version, bold and with a plus sign to the left. Unmodified lines are often included to provide context for the edit. For example, the following describes modification of an existing line and addition of a new line to the OpenSSH server configuration file:

File: /etc/ssh/sshd_config	
-	#LoginGraceTime 2m
+	#PermitRootLogin yes
+	PermitRootLogin no
+	AllowUsers sjansen
	#StrictModes yes

Note that the standard file edit representation may not be used when it is important that the edit be performed using a specific editor or method. In these rare cases, the editor specific actions will be given instead.

Lab Conventions

Lab Task Headers

Every lab task begins with three standard informational headers: "Objectives," "Requirements," and "Relevance". Some tasks also include a "Notices" section. Each section has a distinct purpose.

Objectives ⇒ An outline of what will be accomplished in the lab task.

Requirements ⇒ A list of requirements for the task. For example, whether it must be performed in the graphical environment, or whether multiple computers are needed for the lab task.

Relevance ⇒ A brief example of how concepts presented in the lab task might be applied in the real world.

Notices ⇒ Special information or warnings **needed** to successfully complete the lab task. For example, unusual prerequisites or common sources of difficulty.

Command Prompts

Though different shells, and distributions, have different prompt characters, examples will use a \$ prompt for commands to be run as a normal user (like guru or visitor), and commands with a # prompt should be run as the root user. For example:

```
$ whoami  
guru  
$ su -  
Password: password  
# whoami  
root
```

Occasionally the prompt will contain additional information. For example, when portions of a lab task should be performed on two different stations (always of the same distribution), the prompt will be expanded to:

```
stationX$ whoami  
guru  
stationX$ ssh root@stationY  
root@stationY's password: password  
stationY# whoami  
root
```

Variable Data Substitutions

In some lab tasks, students are required to replace portions of commands with variable data. Variable substitution are represented using italic fonts. For example, X and Y.

Substitutions are used most often in lab tasks requiring more than one computer. For example, if a student on station4 were working with a student on station2, the lab task would refer to stationX and stationY

```
stationX$ ssh root@stationY
```

and each would be responsible for interpreting the X and Y as 4 and 2.

```
station4$ ssh root@station2
```

Truncated Command Examples

Command output is occasionally omitted or truncated in examples. There are two type of omissions: complete or partial.

Sometimes the existence of a command's output, and not its content, is all that matters. Other times, a command's output is too variable to reliably represent. In both cases, when a command should produce output, but an example of that output is not provided, the following format is used:

```
$ cat /etc/passwd  
. . . output omitted . . .
```

In general, at least a partial output example is included after commands. When example output has been trimmed to include only certain lines, the following format is used:

```
$ cat /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
. . . snip . . .  
clints:x:500:500:Clint Savage:/home/clints:/bin/zsh  
. . . snip . . .
```

Lab Conventions

Distribution Specific Information

This courseware is designed to support multiple Linux distributions. When there are differences between supported distributions, each version is labeled with the appropriate base strings:

R ⇒ Red Hat Enterprise Linux (RHEL)
S ⇒ SUSE Linux Enterprise Server (SLES)
U ⇒ Ubuntu

The specific supported version is appended to the base distribution strings, so for Red Hat Enterprise Linux version 6 the complete string is: R6.

Certain lab tasks are designed to be completed on only a sub-set of the supported Linux distributions. If the distribution you are using is not shown in the list of supported distributions for the lab task, then you should skip that task.

Certain lab steps are only to be performed on a sub-set of the supported Linux distributions. In this case, the step will start with a standardized string that indicates which distributions the step should be performed on. When completing lab tasks, skip any steps that do not list your chosen distribution. For example:

1) [R4] *This step should only be performed on RHEL4.*
Because of a bug in RHEL4's Japanese fonts...

Sometimes commands or command output is distribution specific. In these cases, the matching distribution string will be shown to the left of the command or output. For example:

```
$ grep -i linux /etc/*-release | cut -d: -f2  
[R6] Red Hat Enterprise Linux Server release 6.0 (Santiago)  
[S11] SUSE Linux Enterprise Server 11 (i586)
```

Action Lists

Some lab steps consist of a list of conceptually related actions. A description of each action and its effect is shown to the right or under the action. Alternating actions are shaded to aid readability. For example, the following action list describes one possible way to launch and use **xkill** to kill a graphical application:

Alt + F2

Open the "Run Application" dialog.

xkill

Launch xkill. The cursor should change, usually to a skull and crossbones.

Click on a window of the application to kill.

Indicate which process to kill by clicking on it. All of the application's windows should disappear.

Callouts

Occasionally lab steps will feature a shaded line that extends to a note in the right margin. This note, referred to as a "callout," is used to provide additional commentary. This commentary is never necessary to complete the lab successfully and could in theory be ignored. However, callouts do provide valuable information such as insight into why a particular command or option is being used, the meaning of less obvious command output, and tips or tricks such as alternate ways of accomplishing the task at hand.

[S10] \$ sux -
Password: **password**
xclock

- On SLES10, the sux command copies the MIT-MAGIC-COOKIE-1 so that graphical applications can be run after switching to another user account. The SLES10 su command did not do this.

Content

Hardware Discovery Tools	2
Configuring New Hardware with hwinfo	3
Kernel Hardware Info – /sys/	4
/sys/ Structure	5
udev	6
Managing Linux Device Files	8
List Block Devices	11
SCSI Devices	13
USB Devices	15
USB Architecture	17
Kernel Modules	19
Configuring Kernel Components and Modules	21
Handling Module Dependencies	22
Configuring the Kernel via /proc/	23
Console	25
Virtual Terminals	27
Keyboard & locale configuration	29
Serial Ports	31
Random Numbers and /dev/random	32
Lab Tasks	
1. Adjusting Kernel Options	35
2. Linux Kernel Driver Compilation	41
3. Introduction to Troubleshooting Labs	45
4. Troubleshooting Practice: Kernel Modules	50

Chapter

1

LINUX KERNEL & DEVICES

Hardware Discovery Tools

Manual discovery of hardware

- `dmesg`
- `/var/log/dmesg`
- `/var/log/boot.msg`
- `/proc/` and `/sys/`
- `udevadm`
- `lspci`, `lscpu`, `lsscsi`, `lsusb`
- `dmidecode`, `biosdecode`
- `sensors`, `sensors-detect`

Detecting New Hardware Manually

As the Linux kernel loads, it scans for hardware and then loads drivers to initialize and support the detected hardware. Examining kernel boot messages is a good way to see what hardware has been detected. You can view the current kernel messages at any time with the `dmesg` command. A copy of the kernel messages is made near the end of the boot sequence and stored so it can be examined long after the in memory messages have been overwritten.

[R7] *The following applies to RHEL7 only:*

Boot time kernel messages are kept in `/var/log/dmesg`.

[S12] *The following applies to SLES12 only:*

Boot time kernel messages are kept in `/var/log/boot.msg`.

The `/proc/` Virtual Filesystem

The running kernel exports details about detected hardware in the `/proc/` and `/sys/` filesystems. You can use the `cat` command to display the contents of the files in these filesystems. Files and directories in `/proc/` pertaining to hardware that may be useful include: `cputinfo`, `dma`, `interrupts`, `iomem`, `meminfo`, `bus`, `bus/usb/`, `ide/sdX/*`.

In many cases, utilities exist that can extract information from files in `/proc/` and display it in a more human readable fashion. For example, instead of trying to read the raw data shown in the `/proc/bus/pci/` and `/proc/bus/usb/` directories, you can use the `lspci` and `lsusb`

commands.

UDEV Hardware Database

The udev daemon maintains an internal database of the hardware it is aware of. The `udevadm` command can be used to query the database by device paths or names and return all the properties associated with the device. The entire database can also be exported as follows:

```
# udevadm info --export-db
```

Interpreting BIOS DMI Data

Information stored in CMOS often contains low-level details about system hardware. Dump the BIOS data in human readable format with the `dmidecode`, or `biosdecode`, commands:

```
# dmidecode | sed -n '/Memory Device/,//p' |>
    egrep '^[[[:space:]]*S(ize|peed|erial)'
```

Size: 2048 MB
Speed: 667 MHz (1.5 ns)
Serial Number: ED1E3C43

The `dmidecode` command can also be used to determine the current BIOS version:

```
# dmidecode -s bios-version
7IET26WW (1.07)
```

Configuring New Hardware with hwinfo

SUSE automatic detection: hwinfo

Runs at boot time and detects hardware changes

Automatically reconfigures system on addition or removal of hardware

- Uses detection routines found in the /usr/lib64/libhd.so.* libraries
- Records detected hardware in /var/lib/hardware/

Detecting and Configuring New Hardware with hwinfo

To make Linux administration easier, SLES12 provides **hwinfo**, a program which identifies and configures new hardware attached to the system.

hwinfo utilizes the /var/lib/hardware/unique-keys/ directory, where it creates and updates ASCII text files as additional hardware is added to the system.

Invoking hwinfo Manually

hwinfo can be run at any time (for example to detect and configure hot-plugged devices). By default, it will scan for nearly every type of device that it is capable of detecting. Options can be passed to limit the scan to a specific subset of devices. The following examples show using **hwinfo** to list a few details for a few specific types of hardware:

```
# hwinfo --cpu --short
cpu:
  Intel(R) Xeon(R) CPU      5150 @ 2.66GHz, 2660 MHz
  Intel(R) Xeon(R) CPU      5150 @ 2.66GHz, 2660 MHz
# hwinfo --cpu
01: None 00.0: 10103 CPU
[Created at cpu.290]
Unique ID: rdCR.j8NaKXDZtZ6
Hardware Class: cpu
Arch: X86-64
Vendor: "GenuineIntel"
```

```
Model: 6.15.6 "Intel(R) Xeon(R) CPU      5150 @ 2.66GHz"
. . . snip . . .
```

```
# hwinfo --usb --short
hub:
```

```
Linux 2.6.16.21-0.25-smp uhci_hcd UHCI Host Controller
Linux 2.6.16.21-0.25-smp uhci_hcd UHCI Host Controller
Linux 2.6.16.21-0.25-smp uhci_hcd UHCI Host Controller
Linux 2.6.16.21-0.25-smp ehci_hcd EHCI Host Controller
Cypress Hub
```

Kernel Hardware Info - /sys/

Reasons for the creation of sysfs

- Provides hardware information needed by **udev**
- Centralized location for device information
- Clean up /proc/

sysfs

- Usually mounted on /sys/

systool

- List devices in sysfs by bus, class, and topology
 - `systool -b scsi -v`
 - `systool -c scsi_host -v`
 - `systool -c scsi_disk -v`

History of the /dev/ Directory on Linux

One feature of the 2.6 kernel was improved device handling. Originally, device files in /dev/ were statically created on disk. A new system named devfs was created to allow device nodes to be automatically created. As devfs evolved, it showed several critical flaws. A new system called udev was released in the 2.6 kernel, which replaces devfs.

Reasons for the Creation of sysfs

One of the goals of **udev** was to move device naming from kernel space to user space. This simplifies the kernel and gives users more control over device naming. In order for this to be possible, information about devices needed to be exported from the kernel to a place where any application on the system could access it. The solution for this need is a virtual filesystem called sysfs.

The creation of sysfs also helped the Linux kernel developers come closer to reaching another important objective: to clean up /proc/.

The procfs virtual filesystem was originally created to provide information about processes running on the system. Over time, /proc/ became polluted with more and more information about system hardware. One goal of sysfs is to move information about hardware from /proc/ into /sys/.

sysfs

The sysfs filesystem is usually mounted on /sys/. The directory structure contains entries for all devices on the system. Programs such as **udev** look in /sys/ for information needed to load kernel modules, create device nodes in /dev/ and configure each device.

On modern Linux distributions, the sysfs filesystem is automatically mounted on /sys/ at boot.

sysfsutils

libsysfs provides an interface for querying system information from sysfs, primarily implemented in the **systool** utility. **systool** provides options for querying devices based on bus, class, and topology. The **-p** option provides the path to the sysfs file providing the information. **-c** specifies the class to query. See **-h** for a usage summary, and <http://linux-diag.sourceforge.net/Sysfsutils.html> for further information.

/sys/ Structure

Main sysfs directories

- /sys/block/
- /sys/bus/
- /sys/class/
- /sys/devices/
- /sys/module/

Other sysfs directories

- /sys/firmware/
- /sys/kernel/
- /sys/power/

/sys/ Layout

The root of the sysfs filesystem contains several directories. Important directories and their use are described below:

/sys/block/ ⇒ Provides information about each block device on the system (hard drives, CD-ROM drives, USB storage devices, etc.).

/sys/bus/ ⇒ Displays which device drivers are associated to each device. There is a directory for each system bus (PCI, SCSI, USB, etc) which in turn contain two sub-directories: devices/ and drivers/. The devices directory contains a symlink for each device on the bus. These symlinks point to the corresponding device in /sys/devices/. The drivers directory contains information on each device driver needed for the devices on that bus.

/sys/class/ ⇒ Categorizes devices by their functionality. For example, /sys/class/net/ contains information about each network interface on the system and /sys/class/input/ contains information about input devices (keyboards, mice, etc). These directories contain symlinks that point to the device location in /sys/devices/ and the device driver in /sys/bus/.

/sys/devices/ ⇒ Contains the complete hierarchy of devices on the system and how they are connected. Detailed information about each device is presented. For example, /sys/devices/ shows how a USB mouse is integrated in the system: System PCI Bus→USB Controller→USB Hub→USB Mouse

/sys/module/ ⇒ Contains a directory for each loaded kernel module. This directory contains information about the module such as

module parameter values and reference counts (how many modules are using that module).

/sys/firmware/ ⇒ Some device drivers need to load firmware into the device in order for it to function. This is accomplished in a device driver by providing a sysfs interface. That interface will appear under this directory. It can then be used by a userspace program to upload firmware (binary data) to the device driver, which (presumably) in turn, loads the firmware into the device, itself. This is about the only case where a sysfs mechanism does not provide human readable output.

/sys/kernel/ ⇒ The /sys/kernel/debug/ directory is used by kernel & driver developers for debugging. The /sys/kernel/hotplug_seqnum file is used by the hotplugging subsystem.

/sys/power/ ⇒ Controls the system power state via this directory. The /sys/power/state file can be read to show which power states are supported. Writing one of those values back to this file will place the system in that state.

/sys/fs/ ⇒ Location where each loaded filesystems can expose options. Importantly, the SELinux and CGroup filesystems are mounted there.

udev

udevd

- Reads rules into memory (`/lib,etc/udev/rules.d/*.rules`)
- Listens via netlink socket for kernel uevent messages
- Creates, deletes, or changes files under `/dev/`

udevadm

- sends messages to running `udevd`
- debug/trace udev and kernel uevent activity

Example udev rule to create a symlink in /dev to USB thumb drive with a particular serial number:

- `KERNEL=="sd*", BUS=="usb", ENV{ID_SERIAL}=="47806", SYMLINK+="MyUSBdisk"`

udev General Operation

Historically, the `/dev/` directory was static and was populated at install time, and then manually by the systems administrator as devices were added or removed from the system. In contrast, the udev system dynamically creates device files. Basic operation of udev is as follows:

1. `udevd` lexically sorts together the rule files found in `/lib,etc/udev/rules.d/*.rules` and then reads the rules into memory.
2. When the kernel detects a device (e.g. device enumeration on boot, hot-add of device, manual loading of kernel module) it emits a uevent record which is read by the running `udevd` process.
3. `udevd` creates, deletes, or changes files under `/dev/` based on the current rules.

As part of processing some rules, udev runs external programs. Additionally some hardware events are passed to higher level subsystems such as HAL and udisks (formerly DeviceKit). This allows for things like the graphical desktop reacting to a hardware change such as launching a file browser when an external disk is attached.

Managing the udev Process

The udev daemon is first started by scripts contained in the initial RAM disk during the early phases of the boot process. Init later calls additional scripts that interact with the running udev daemon and finish populating the `/dev/` directory. The `udevadm` command can be

used to interact with the running udev daemon. The following examples show some of the more common uses:

`udevadm control --reload` ⇒ Re-read rule files into memory (perhaps new rules were created that are needed for future hardware events)

`udevadm trigger --action=change` ⇒ Send change event for all subsystems rebuilding `/dev/` files based on loaded rules.

`udevadm info --export-db` ⇒ Dump the contents of the udev database showing: name, path, sysfs attributes, and environment attributes for all devices.

`udevadm monitor` ⇒ Show all kernel uevent and udev messages (useful for testing and debugging new rules).

Creating Custom udev Rules

Standard udev rules needed by the system are provided by the `/lib/udev/rules.d/*rules` files. These files should not be modified as they are replaced as newer udev packages are installed. In the rare case where it is necessary, creating a file with the same filename in the `/etc/udev/rules.d/` directory will prevent the corresponding standard `.rules` file from being processed.

Application RPMs can provide additional udev rules by placing `.rules` files in the `/etc/udev/rules.d/` directory. For custom rules, a new file such as `/etc/udev/rules.d/99-Xcustom.rules` should be created. When naming the file, remember that udev only processes files ending in the `.rules` suffix, sorts them lexicographically and that custom rules should generally be run after the standard rules.

Rule File Syntax

Within the rules files, each rule is a single line, and each line has the basic form of:

```
key==value_to_compare, key=value_to_assign
```

Comparisons match the device in question, and then assignments determine the actions taken for that device such as the name of the file to be created in /dev/. Example of common keys that are compared include: KERNEL, the kernel name for the device; ATTRS{filename}, a sysfs attribute for the device; ENV{key}, a device property value. Examples of common assignments include: NAME (network interfaces only) or SYMLINK, filename to be created in /dev/; OWNER|GROUP|MODE, permissions and ownership for the created device file. Full documentation is found in the udev(7) man page.

Using += adds the value to a key that may contain a list of entries.

Creating new rules generally start with determining the correct set of comparisons needed to uniquely identify the device in question. Use the **udevadm info** command to find useful keys and values in the /sys/ filesystem that can be matched by ATTRS or ENV keys:

First get a list of the currently attached storage devices and identify which device to query:

```
# lsscsi  
... snip ...  
# lsscsi -t  
... snip ...
```

Then use **udevadm** to query sysfs attributes and device properties that would be good candidates to uniquely identify the device:

```
# udevadm info /dev/sde | grep SERIAL  
E: ID_SCSI_SERIAL=956a7ebc-962a-1cd8dbf902bb  
E: ID_SERIAL=3600140597b4ab8962a1cd8d  
E: ID_SERIAL_SHORT=6001cb7cb4ab8962a1cd8d  
# udevadm info --attribute-walk /dev/sde  
... snip ...  
KERNEL=="sde"  
SUBSYSTEM=="block"  
... snip ...
```

Poor vs Good UDEV Rule

For instance, a simple rule might create a symlink to a device node name based on the original block device:

```
File: /etc/udev/rules.d/99-usbdrive.rules  
KERNEL=="sdb", SYMLINK+="MyUSBdisk"
```

This will create a block device /dev/usb0 file, with a symbolic link of /dev/MyUSBdisk pointing to it. However, block device names can change, and are therefore not persistent. Something unique, like a serial number or manufacturer's name, are a better target to match against:

```
File: /etc/udev/rules.d/99-usbdrive.rules  
BUS=="usb", SYMLINK+="MyUSBdisk",  
ENV{ID_SERIAL}=="36001405"
```

Advanced Rule Example

Besides controlling properties of the device node file in /dev, UDEV rules can run arbitrary commands and set attributes. For example, here is rule to set I/O scheduler, the number of I/O requests that can be outstanding, and the read ahead amount on Netgear SCSI disks:

```
File: /etc/udev/rules.d/99-netgear-tuning.rules  
KERNEL=="sd[a-z]", ACTION=="add|change", BUS=="scsi",  
SYSFS{vendor}=="NETGEAR",  
ATTR{queue/scheduler}=="noop",  
ATTR{queue/nr_requests}=="1024",  
ATTR{bdi/read_ahead_kb}="512"
```

Debug logging

Change the default log level from err to debug when troubleshooting udev rules and behavior:

```
# udevadm control --log-priority=debug
```

Messages are sent to /var/log/messages. Set the udev log priority back to err after debugging.

Managing Linux Device Files

Usually in /dev/

Creating device files

- at install time (RPM)
- **mknod**
- dynamically with udev

Device file names

Linux Devices Files

Linux provides user space access to hardware through device files. When an application accesses a device file, the kernel uses the appropriate device driver to interact with the hardware. The /dev/ directory is usually the only directory with any device files.

Historically, most device files were created during the install process (often by installing an RPM that provided the files). Device files can also be created manually using the **mknod** command. When creating device files, you must specify the type and major & minor numbers. You can optionally specify the mode and context. The major and minor numbers for common devices can be found in the kernel documentation in the devices.txt file or at <http://www.lanana.org/docs/device-list/index.html>, the Linux Assigned Names and Numbers Authority.

On an SELinux enabled distribution the proper file context must be set. This can be done with **mknod -Z** or afterward using **chcon**. This example shows setting context at the same time as the creation of the device node. Use **chcon -t null_device_t /dev/null** after the fact:

```
# mknod -Z "system_u:object_r:null_device_t" -m 666 null c 1 3
# ls -Z /dev/null
crw-rw-rw-. root root system_u:object_r:null_device_t:s0 /dev/null
```

The **fixfiles** command can be used to set the context based on the SELinux policy automatically.

Dynamically Created Device Nodes

The current solution for dynamic creation of /dev/* files is called udev. udev reads rules found in the /lib/udev/rules.d/ and /etc/udev/rules.d/ directories. These rules determine what device files are created under /dev/.

On a RHEL7 system, the majority of the files in the /dev/ directory are created based on the configuration rules in the /lib/udev/rules.d/50-udev-default.rules file.

Common Device Names

Although a file pointing to a device can be given any name—the major and minor numbers are used to map to a kernel module, not the name—naming conventions for common devices exist. These tables list information for a few common devices.

When examining this table, consider that many devices, such as SATA (Serial ATA) and USB drives (thumbdrives, external hard drives, etc), use a SCSI emulation layer. These appear as /dev/sdX devices:

Device Description	Filename	Type	Major #	Minor #
Disk 0	/dev/sda	block	8	0
Disk 1	/dev/sdb	block	8	16
Disk 2	/dev/sdc	block	8	32
Disk 0, partition 1	/dev/sda1	block	8	1
Disk 0, partition 2	/dev/sda2	block	8	2
Disk 0, partition 3	/dev/sda3	block	8	3
1st SCSI Tape (mode0)	/dev/st0	char	9	0
1st SCSI CD-ROM	/dev/sr0	block	11	0
1st Generic SCSI Device	/dev/sg0	block	21	0
NVMe Disk 0	/dev/nvme0n1	block	259	0
NVMe Disk 1	/dev/nvme1n1	block	259	2
NVMe Disk 0, partition 1	/dev/nvme0n1p1	block	259	1
NVMe Disk 1, partition 1	/dev/nvme1n1p1	block	259	3

Device Description	Filename	Type	Major #	Minor #
Floppy drive 1	/dev/fd0	block	2	0
CD-ROM drive 1	/dev/cdrom	symlink		
Null device (data sent here is discarded)	/dev/null	char	1	3
Zero device (outputs infinite stream of zeros)	/dev/zero	char	1	5
Random device (non-deterministic random number generation)	/dev/random	char	1	8
Random device (faster, less secure random number generation)	/dev/urandom	char	1	9
Physical memory access	/dev/mem	char	1	1

This table shows a collection of other important device files. Note that the /dev/random device requires entropy in the form of keyboard activity, mouse movement, interrupts, etc. to generate output. If the entropy pool runs out then the device will block (output stops) until additional entropy is available. The /dev/urandom device uses a hash algorithm (seeded by entropy from the pool) to generate a constant stream of pseudo-random data.

Persistent Disk Name

The traditional block device filename assigned to a specific disk (e.g. /dev/sda) can change depending on the order in which the kernel detects devices. Therefore, referencing disks via these names in configuration files or commands is an unsafe practice. udev provides persistent names for each disk under the /dev/disk/ directory by: device id, by filesystem label, by physical path, and by UUID. Custom udev rules can also be created to assign arbitrary persistent names of the administrator's choosing.

If installed, the **tree** command is a good way to view these:

```
# tree /dev/disk
/dev/disk
|-- by-id
|   |-- ata-HDS728080PLA380_PFDBU0SDUSRE3X -> ../../sdb
|   |-- ata-ST3160318AS_6VM6K43J -> ../../sda
|   |-- ata-ST3160318AS_6VM6K43J-part1 -> ../../sda1
|   |-- ata-ST3160318AS_6VM6K43J-part2 -> ../../sda2
|   |-- dm-name-vg0-lv_root -> ../../dm-0
...
|   |-- snip ...
|   |-- wwn-0x5000c5001b3fc5ad-part2 -> ../../sda2
|   \-- wwn-0x5000cca302f50f38 -> ../../sdb
|-- by-path
|   |-- pci-0000:00:1f.1-scsi-0:0:0:0 -> ../../sr0
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0 -> ../../sda
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0-part1 -> ../../sda1
|   |-- pci-0000:00:1f.2-scsi-0:0:0:0-part2 -> ../../sda2
|   \-- pci-0000:00:1f.2-scsi-0:0:1:0 -> ../../sdb
\-- by-uuid
    |-- 069cbe84-f9a2-476e-9dfd-affd2979e41b -> ../../sda1
    |-- 16f34d14-0312-482a-8da8-ca653c336a77 -> ../../dm-2
    |-- 4f57e937-19dc-478b-9f38-ad3e2d06340c -> ../../dm-4
    |-- 7735dc34-45d2-446a-b32c-84fe688b12da -> ../../dm-1
    |-- 7cca50b4-d3ec-425b-b784-a7301145f46e -> ../../dm-0
    \-- fec5c254-a664-40e6-a674-224b8a4995d0 -> ../../dm-3
```

3 directories, 33 files

List Block Devices

lsblk

- Lists available block devices, or information about specific block devices. Some useful options include:

-a | --all
-f | --fs
-i | --ascii
-r | --raw
-t | --topology

blkid

- Search for and print attributes of block devices

lsblk

The **lsblk** command lists block devices recognized by the system, see sysfs(2), and shows their interrelationships, (for example, a device mapped logical volume attached to a physical volume partition). Some devices are excluded by default. Use the **-a** flag to see all detected devices (other than RAM disks). The **-m** flag will show the mode of the block device file. To see the block device topology, use the **-t** flag:

```
$ lsblk -ti
```

NAME	ALIGNMENT	MIN-IO	OPT-IO	PHY-SEC	LOG-SEC	ROTA	SCHED	RQ-SIZE	RA	WSAME
sda	0	512	0	512	512	1	cfq	128	4096	0B
-sda1	0	512	0	512	512	1	cfq	128	4096	0B
`-sda2	0	512	0	512	512	1	cfq	128	4096	0B
-vg0-root	0	512	0	512	512	1		128	4096	0B
-vg0-swap	0	512	0	512	512	1		128	4096	0B
-vg0-tmp	0	512	0	512	512	1		128	4096	0B
-vg0-var	0	512	0	512	512	1		128	4096	0B
sr0	0	2048	0	2048	2048	1	cfq	128	128	0B

When run with administrative privileges, the **-f** flag will show what filesystems are being used:

```
$ lsblk -f
```

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
sda				
-sda1	xfs		5c5f8785-e17c-41b9-acd6-b183f8cac8b0	/boot
`-sda2	LVM2_member		GcJPr6-EsVn-9cLo-GHT7-z3P5-hZzs-VrPp1I	
-vg0-root	xfs		d702750a-b6b6-44e6-b09c-f33018f644c4	/
-vg0-swap	swap		264708ba-df33-4f5c-b735-4d812452d74a	[SWAP]
-vg0-tmp	xfs		6cc9fa2b-e791-4f43-a8c8-765dd4750270	/tmp
-vg0-var	xfs		d7eab8bc-93a5-4df7-97ed-92324f46fc60	/var
sr0	iso9660	RHEL-7.0 Server.x86_64	2014-05-07-03-58-46-00	/run/media/guru/RHEL-7.0 Serve

If UTF-8 tree formatting is a problem, such as for a copy-and-paste to email, use the **-i** flag to output ASCII. The **-l** flag could also be used to provide list, instead of tree, output format, and **-r** removes all formatting. The **-n** flag suppresses the column headers. To customize which headers to print, i.e. which columns to include, use the **-o** flag.

blkid

The **blkid** command is a low level utility to print block device attributes. It is similar to **lsblk** in locating block devices:

```
# blkid  
/dev/sr0: UUID="2014-05-07-03-58-46-00" LABEL="RHEL-7.0 Server.x86_64" TYPE="iso9660" PTTYPE="dos"  
/dev/sda1: UUID="5c5f8785-e17c-41b9-acd6-b183f8cac8b0" TYPE="xfs"  
/dev/sda2: UUID="GcJPr6-EsVn-9cLo-GHT7-z3P5-hZzs-VrPp1I" TYPE="LVM2_member"  
/dev/mapper/vg0-swap: UUID="264708ba-df33-4f5c-b735-4d812452d74a" TYPE="swap"  
/dev/dm-0: UUID="d702750a-b6b6-44e6-b09c-f33018f644c4" TYPE="xfs"  
/dev/dm-2: UUID="6cc9fa2b-e791-4f43-a8c8-765dd4750270" TYPE="xfs"  
/dev/dm-3: UUID="d7eab8bc-93a5-4df7-97ed-92324f46fc60" TYPE="xfs"
```

The **-p** option can be used for a superblock probe:

```
# blkid -p /dev/sda1  
/dev/sda1: UUID="5c5f8785-e17c-41b9-acd6-b183f8cac8b0" TYPE="xfs" USAGE="filesystem" PART_ENTRY_SCHEME="dos" PART_ENTRY_TYPE="0x83" PART_ENTRY_FLAGS="0x80" PART_ENTRY_NUMBER="1" PART_ENTRY_OFFSET="2048" PART_ENTRY_SIZE="1024000" PART_ENTRY_DISK="8:0"  
# blkid -p /dev/sda2  
/dev/sda2: UUID="GcJPr6-EsVn-9cLo-GHT7-z3P5-hZzs-VrPp1I" VERSION="LVM2_001" TYPE="LVM2_member" USAGE="raid" PART_ENTRY_SCHEME="dos" PART_ENTRY_TYPE="0x8e" PART_ENTRY_NUMBER="2" PART_ENTRY_OFFSET="1026048" PART_ENTRY_SIZE="71680000" PART_ENTRY_DISK="8:0"
```

SCSI Devices

Identifying devices

- /proc/scsi/scsi
- lsscsi

Adding / Removing devices with /sys/class/scsi_*

SCSI Command Protocol

- Commands supported by SCSI devices
- Carried across many transports: ATAPI, PATA, SATA, SPI, SAS, FCP, USB, Fireware SBP-2, IP (for iSCSI), etc.

Viewing and setting options

- sddparm
- sg_map

new devices on the specified SCSI host:

```
# cat /proc/scsi/scsi #see current devices
Attached devices:
Host: scsi0 Channel: 00 Id: 08 Lun: 00
  Vendor: DP      Model: BACKPLANE      Rev: 1.05
  Type: Enclosure          ANSI SCSI revision: 05
Host: scsi0 Channel: 02 Id: 00 Lun: 00
  Vendor: HP      Model: RAID CONTRLR   Rev: 1.03
  Type: Direct-Access          ANSI SCSI revision: 05
# echo 1 > /sys/class/fc_host/host_num/issue_lip && sleep 15
# echo "----" > /sys/class/scsi_host/host0/scan
# dmesg #view results of rescan
... snip ...
Vendor: SEAGATE  Model: ST3300655SS  Rev: S515
Type: Direct-Access          ANSI SCSI revision: 05
Vendor: SEAGATE  Model: ST3300655SS  Rev: S515
Type: Direct-Access          ANSI SCSI revision: 05
```

Device Identification

Every SCSI device is assigned an ID (used to differentiate commands and responses sent by devices on a shared SCSI bus). The address is in the form host (SCSI adapter number), bus (channel number), target (id number), lun (logical unit number). The **lsscsi** command can display information about what SCSI hosts and devices are seen by the kernel as shown in this example:

```
$ lsscsi -l
[0]    megaraid_sas
$ lsscsi
[0:0:8:0]  enclosu DP      BACKPLANE  1.05  -
[0:2:0:0]  disk     HP      RAID CONTRLR 1.03  /dev/sda
```

The **lsscsi** command gets its data from files in /sys/class/scsi_* which can also be read directly as an alternative.

Scanning for New SCSI Devices

In addition to the methods already discussed, a list of all SCSI devices known to the kernel can be seen by reading the /proc/scsi/scsi file. Newly added SCSI devices will not be visible to the kernel until either a reboot, or a rescan of the corresponding host. A rescan can be initiated via the /sys/class/scsi_host/host_num/scan file. If the new device is a SAN device, a loop initialization protocol (LIP) command may need to be issued to the HBA card to rescan the Fiber Channel bus as well. The following example uses the wildcard "-" (dash character) in place of bus, target, and lun and would cause the kernel to detect any

Adding and Removing SCSI Devices

To make the kernel aware of hotplugged or removed SCSI devices, send commands to either the /proc/scsi/scsi file (2.4 kernel), or within the /sys/class/scsi_host/ hierarchy (2.6 and 3.x kernel, exact file depends on kernel version). The following example identifies the SCSI id associated with the /dev/sda disk and then removes that device from the kernel's view:

```
# lsscsi | grep /dev/sda
[4:1:0:0] disk HP VIRTUAL DISK 1028 /dev/sda
# echo 1 > /sys/class/scsi_device/4\:1\:0\:0/device/delete
```

To add devices under the 2.6 kernel rescan the corresponding host as previously shown.

eSATA

A similar technique is used to safely remove an eSATA drive. First the drive is spun down, then taken offline, then removed from the kernel:

```
# hdparm -y /dev/sdc
/dev/sdc:
 issuing standby command
# echo offline > /sys/block/sdc/device/state
# echo 1 > /sys/block/sdc/device/delete
```

Viewing and Setting SCSI Mode Pages

Mode pages contain meta data about a SCSI device as defined by the SCSI (draft) standards: www.t10.org. The specific pages supported by a device will vary based on the device and transport in use. In addition, devices may support Vital Product Data (VPD) pages with additional data. The **sparm** command can be used to view and set values contained in the various mode pages supported by a SCSI device. This command is similar to the **hdparm** command which can view and set similar device and transport related settings of ATA driven devices (www.t13.org), and some overlap exists between the commands. The following example shows using **sparm** to view and set mode page values:

```
# lsscsi -g #determine SCSI generic device for first drive
[4:0:0:0] disk ATA Hitachi HUA72107 A74A - /dev/sg0
[4:0:1:0] disk ATA Hitachi HUA72107 A74A - /dev/sg1
[4:1:0:0] disk HP VIRTUAL DISK 1028 /dev/sda /dev/sg2
```

```
# sparm --enumerate #view mode pages supported by sparm
Mode pages:
. . . output omitted . . .
bc 0x1c,0x01 Background control (SBC)
ca 0x08 Caching (SBC)
cms 0x2a CD/DVD (MM) capabilities and,
      mechanical status (MMC)
. . . output omitted . . .
# sparm --page=ca /dev/sg0 #list values in Caching mode page
/dev/sg0: ATA Hitachi HUA72107 A74A
Caching (SBC) mode page:
. . . output omitted . . .
DISC 0 [cha: n]
SIZE 0 [cha: n]
WCE 0 [cha: n]
. . . output omitted . . .
# sparm -s WCE=1 /dev/sg0 #enable write cache
# sparm --page=ca -l --get=WCE /dev/sg0 #read value to verify
/dev/sg0: ATA Hitachi HUA72107 A74A
WCE 1 [cha: y] Write cache enable
```

The **sg_map** command can also show SCSI generic device mappings. The SCSI generic device can be used to send custom SCSI commands to devices.

```
# sg_map -i -x
/dev/sg0 4 0 0 0 0 ATA Hitachi HUA72107 A74A
/dev/sg1 4 0 1 0 0 ATA Hitachi HUA72107 A74A
```

The **/lib/udev/scsi_id -gud /dev/sdb** command can provide the World Wide Identifier (WWID, similar to a Fibre Channel WWN) of the SCSI lun.

```
# /lib/udev/scsi_id -gud /dev/sda
36d4ae5208a9924001b50452c1c07b2d3
```

The **sg_map** command is provided by the **sg3_utils** package and **scsi_id** comes with **systemd**.

USB Devices

lsusb

- **-v** display verbose details
e.g. bcdUSB field shows max USB version a device supports
- **-t** display physical USB device hierarchy

Kernel Messages

- dmesg
- journalctl -k
- RHEL7/SLES12: /var/log/messages

Examining Detected USB Controllers and Devices

Several applications can be used to view information regarding USB devices currently connected to the system. **lsusb** is a command line tool that displays each device and its corresponding bus and device number. Very detailed information can be displayed with the **-v** option. The **-t** option shows a physical view that includes the speeds of the various controllers and devices:

```
# lsusb
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 002: ID 1006:3002 iRiver, Ltd. iHP-120/140 MP3 Player
. . . snip . . .

# lsusb -t
/: Bus 04.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/2p, 5000M
/: Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/2p, 480M
|__ Port 2: Dev 2, If 0, Class=Mass Storage, Driver=usb-storage, 480M
/: Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
|__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/8p, 480M
/: Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
|__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
    __ Port 3: Dev 3, If 0, Class=Vendor Specific Class, Driver=, 12M
    __ Port 4: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
    __ Port 4: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
    __ Port 4: Dev 4, If 2, Class=Vendor Specific Class, Driver=, 12M
    __ Port 4: Dev 4, If 3, Class=Application Specific Interface, Driver=, 12M
    __ Port 5: Dev 6, If 0, Class=Hub, Driver=hub/6p, 480M
    __ Port 6: Dev 5, If 0, Class=Video, Driver=uvcvideo, 480M
    __ Port 6: Dev 5, If 1, Class=Video, Driver=uvcvideo, 480M
```

```
# lsusb -v  
... snip ...  
Bus 001 Device 002: ID 18d1:4ee1 Google Inc. Nexus 5  
Device Descriptor:  
    bLength          18  
    bDescriptorType   1  
    bcdUSB         2.00  
... snip ...
```

This Google Nexus 5 smartphone supports USB v2.00.

The kernel reports information about discovered USB devices in the kernel logs. With the default Syslog configuration, these messages will be written to the /var/log/messages file. Additional kernel messages can also be viewed with the **dmesg** command. Some messages will appear in both places. This can be useful in finding the device names assigned to newly added devices:

```
# dmesg  
[431912.616502] usb 3-2: new high-speed USB device number 2 using xhci_hcd  
[431912.783591] usb 3-2: New USB device found, idVendor=1006, idProduct=3002  
[431912.783602] usb 3-2: New USB device strings: Mfr=73, Product=93, SerialNumber=115  
[431912.783607] usb 3-2: Product: iRiver iHP-100 Series  
[431912.783612] usb 3-2: Manufacturer: iRiver  
[431912.783616] usb 3-2: SerialNumber: 0123456789AB  
[431912.816171] usb-storage 3-2:2.0: USB Mass Storage device detected  
[431912.816538] scsi6 : usb-storage 3-2:2.0  
[431912.816695] usbcore: registered new interface driver usb-storage  
[431912.820648] usbcore: registered new interface driver uas  
[431914.189854] scsi 6:0:0:0: Direct-Access      TOSHIBA MK2004GAL      JA02 PQ: 0 ANSI: 0 CCS  
[431914.190856] sd 6:0:0:0: [sdb] 39063024 512-byte logical blocks: (20.0 GB/18.6 GiB)  
[431914.191001] sd 6:0:0:0: Attached scsi generic sg2 type 0  
[431914.191601] sd 6:0:0:0: [sdb] Write Protect is off  
[431914.191611] sd 6:0:0:0: [sdb] Mode Sense: 00 4a 00 00  
[431914.192317] sd 6:0:0:0: [sdb] Asking for cache data failed  
[431914.192322] sd 6:0:0:0: [sdb] Assuming drive cache: write through  
[431914.615989] sdb: sdb1  
[431914.618248] sd 6:0:0:0: [sdb] Attached SCSI disk  
... snip ...
```

USB Architecture

udevd

Kernel Modules

- Chipset Modules
- Device Modules

Disabling USB Storage

00:1a.0 USB controller: Intel Corp 6 Series/C200 Series,
 Chipset Family USB Enhanced Host Controller #2 (rev 04)
00:1d.0 USB controller: Intel Corp 6 Series/C200 Series,
 Chipset Family USB Enhanced Host Controller #1 (rev 04)
0e:00.0 USB controller: NEC Corp uPD720200 USB 3.0 Host
 Controller (rev 04)

Device Specific USB Kernel Modules

Additional kernel modules exist for different USB devices. Some of the most commonly used include:

usbhid.ko ⇒ human-interface devices such as keyboards, mice, and joysticks

usb-storage.ko ⇒ mass storage devices such as external hard drives and "keychain" drives

usbip.ko ⇒ printers

usbserial.ko ⇒ USB to serial port adapters

Disabling USB Storage Devices

Site security policies sometimes require disabling support for USB storage devices. If no USB devices will be used, then disabling USB support in the BIOS, or removing the core USB kernel modules, may be an option. To disable only support for USB storage, prevent the associated module from loading by either blacklisting it, or configuring the system to run an alternate command instead:

USB Controllers and Devices

USB controllers and devices are automatically configured during installation and initialized on boot. When USB devices are plugged into the system, the kernel will load the appropriate module, and then emit a uevent message that is read by udev which then creates the corresponding device file(s).

Core USB Kernel Modules

The main kernel module for USB support is `usbcore.ko`. There are four additional kernel modules, corresponding to various chipset families. The following shows the module file names, but it is not uncommon in modern distributions to have these compiled in directly:

uhci-hcd.ko "Universal Host Controller Interface" ⇒ Kernel module for Intel/VIA chipsets (USB 1.1).

ohci-hcd.ko "Open Host Controller Interface" ⇒ Kernel module for most other non-Intel/VIA chipsets (USB 1.1).

ehci-hcd.ko "Enhanced Host Controller Interface" ⇒ Kernel module for USB 2.0 (high speed) standard (usually chips from NEC).

xhci-hcd.ko "eXtensible Host Controller Interface" ⇒ Kernel module for USB 3.0 (super speed) standard.

Since the USB host controller interfaces with the rest of the system via the PCI bus, information about which USB chipset the host has can be found in this fashion:

```
# lspci | grep USB
```

```
[R7] File: /etc/modprobe.d/blacklist.conf
```

```
[S12] File: /etc/modprobe.d/50-blacklist.conf
```

```
+ blacklist usb_storage
```

```
File: /etc/modprobe.conf
```

```
+ install usb_storage logger "Attempted USB Storage"
```

This can also be accomplished with a udev rule, perhaps best used in addition to the module blacklist. Create a file in /etc/udev/rules.d/ telling udev to ignore USB storage devices:

```
File: /etc/udev/rules.d/10-USBIgnore.rules
```

```
+ BUS=="usb", SUBSYSTEM=="block", OPTIONS+="ignore_device"
```

Kernel Modules

Kernel Modules Provide

- Hardware drivers
- Filesystems
- Network stack
- Linux Security Modules (LSM)

Discovering information about modules

- `modinfo`

Managing modules

- `lsmod`
- `insmod`
- `rmmod`

ABI compatibility

Module	Size	Used by
... snip ...		
nfsd	212097	9
exportfs	8641	1 nfsd
nfs	211785	1
lockd	63593	3 nfsd,nfs
... snip ...		

Inserting and Removing Modules

Inserting kernel modules is done with the `insmod` command. The basic syntax of the `insmod` command is:

`insmod (filename|module_name) [module_options] [...]`

The `insmod` command will attempt to load the specified module. This load will fail if the module does not detect compatible hardware on the system, or if it is dependent on modules that are not currently loaded.

Removing kernel modules is done with the `rmmod` command. Usage is `rmmod module_name`. Modules will fail to unload if they are currently in use (check the "Used by" count in the module listing). Due to the kernel data structures they use and/or modify, some modules (such as the `ipv6.ko` module) can not be unloaded once they have been loaded.

Kernel Modules

Kernel modules give the kernel access to functions, commonly called symbols, for many kernel services such as NetFilter, filesystem abstraction, hardware drivers, security, and network stacks. These modules may depend on functions from other modules. These dependencies need to be resolved, either manually using `insmod` and `rmmod`, or automatically using other `modprobe`.

Learning About Kernel Modules

Configuring a module must be done either at boot time, for a statically compiled module, or at the time that the module is loaded. Kernel modules can have various options. To identify options and other information about a module use the `modinfo` command:

- a ⇒ lists author of the module
- d ⇒ lists description of the module
- l ⇒ lists license of the module
- n ⇒ list the file name of the module
- p ⇒ lists parameters (options) for the module

Listing the modules that are currently loaded into the kernel is done with the `lsmod` command.

The following shows an example of the `lsmod` command's output which is comprised of three columns: the module name, module size, and usage count:

Persistently Loading Kernel Modules

Normally kernel modules are loaded automatically when needed via UDEV and matching hardware device ids with the hardware device ids claimed by a particular kernel module. However, the systemd modules-load.d(5) mechanism can be used to manually and persistently load kernel modules at boot by way of conf files in the /etc/modules-load.d/ directory. The following configuration file always loads the driver for the TPM hardware random number generator at boot:

File: /etc/modules-load.d/tpm-rng.conf

```
tpm-rng
```

Kernel ABI Compatibility

Upstream Linux kernel development (<http://kernel.org>) does not guarantee ABI compatibility from one version of the kernel to the next, (e.g. 3.12 to 3.13, or even 3.12.1 to 3.12.2). A driver (kernel module) may work with a kernel it was not compiled for but only if the portion of the kernel ABI that it uses has not changed.

[R7] *The following applies to RHEL7 only:*

Beginning with Red Hat Enterprise Linux 5, the Driver Update Program provides a stable ABI subset for companies to develop against. The stable ABI whitelist is provided by installing the kernel-abi-whitelists RPM package. It installs the architecture specific whitelists under the /lib/modules/kabi-rhel70/ directory. To automate the checking of third party kernel module packages, install the kabi-yum-plugins package.

[S12] *The following applies to SLES12 only:*

SUSE provides Kernel Module Packages for kernel ABI compatible driver modules. Development and testing of these modules are done through the openSUSE community. See http://en.opensuse.org/Kernel_Module_Packages and <http://en.opensuse.org/Factory>.

Configuring Kernel Components and Modules

Two methods of compiling kernel features

- Compiled into kernel binary installed under /boot/
- Separate kernel module (*.ko) installed under /lib/modules/\$(uname -r)

Configuration options can be passed to kernel binary on boot

- Interactively from GRUB command prompt
- Persistently from GRUB config file

Configuration options can be passed to kernel modules

- Interactively when loading module
- Via files in the /sys/modules/ directory
- Persistently from /etc/modprobe.d/*.conf

Kernel Configuration

In order for the Linux kernel to use any device, it needs the kernel instructions for that device. These kernel instructions are supplied in the form of a device driver. Device drivers can be statically linked into the kernel, but more typically they are modular drivers which can be loaded and unloaded dynamically in the running kernel.

If support for some device and functionality has been statically linked into the kernel, the only way you can configure that driver is by modifying the boot-loader (GRUB) configuration to pass the desired parameters when the kernel is loaded.

For example if you wanted local text consoles to run at a specific VESA mode, you could modify the kernel line in your GRUB configuration file to appending:

```
File: /etc/default/grub
→ GRUB_CMDLINE_LINUX="... video=800x600"
```

However, loadable kernel modules can be configured through parameters. Available module parameters can be shown using information from sysfs:

```
$ ls -1R /sys/module/*/parameters/
...
/sys/module/scsi_mod/parameters/:
max_luns
scsi_logging_level
```

To change tunable parameters non-persistently, write the desired value to the corresponding file via shell redirection:

```
# echo value > /sys/module/modulename/parameters/parameter
```

/etc/modprobe.d/ Directory

Modern Linux distros do not come with a /etc/modprobe.conf file by default, but instead uses files within the /etc/modprobe.d/ directory.

The /etc/modprobe.d/*.conf files, are used to configure kernel driver modules. These files support a large number of options including conditional statements. The two most commonly used configuration directives are alias and options. Aliases associate some common names with a particular kernel module. For example, this line specifies that the bond0 device uses the bonding module:

```
File: /etc/modprobe.d/bonding.conf
alias bond0 bonding
```

The options directive is used to set module parameters or activate features supported by the driver. For example, this line would enable SR-IOV and enable 32 Virtual Functions (VFs) for the Intel 10GbE card.

```
File: /etc/modprobe.d/ixgbe.conf
options ixgbe max_vfs=32
```

Handling Module Dependencies

Module Dependencies

- /lib/modules/\$(uname -r)/modules.dep
- depmod

Inserting and Removing modules

- modprobe

Module Dependencies

Some modules need functions that are provided by other modules, creating inter-dependencies. In order to use a module that needs functions provided by another, the module that provides the functions must be loaded before the module that requires them. To deal with module inter-dependencies, an administrator can manually load the modules using **insmod**.

The modprobe Command

The **modprobe** command provides an alternative to manually resolving module inter-dependencies. It provides automatic resolution of module dependencies. The **modprobe** command resolves module dependencies using a list of all modules and the symbols they require and provide. This list is created using the **depmod** command. It has a syntax similar to a Makefile and is written to the **/lib/modules/\$(uname -r)/modules.dep** file.

The **depmod** command will create a **modules.dep** file (among other files) for the currently running kernel. Once the **modules.dep** file is created the **modprobe** command can be used to insert and remove modules from the kernel.

The following example would insert the **nfs.ko** module into the kernel, including any needed dependencies, (e.g. the **sunrpc.ko** module which **nfs.ko** depends on):

```
# modprobe nfs
```

To later remove the module, you would invoke **modprobe** with the **-r** option as shown in this example:

```
# modprobe -r nfs
```

Configuring the Kernel via /proc

/proc/PID/ exposes information about each process on the system

- Files and symlinks inside each folder provide information about the process

/proc/sys/ exposes tunable kernel parameters

- view current values with **cat**
- modify with **echo**
- view and modify with **sysctl** command

Persistent tuning: /etc/sysctl.conf, /etc/sysctl.d/*.conf

- system defaults:/usr/lib/sysctl.d/*.conf
- systemd-sysctl.service

Viewing Process Information via /proc/PID/

If the /proc/ filesystem is mounted, then the running kernel uses it as an interface to expose information about itself. The original use of /proc/ was to provide information about running process on the system. Commands like **ps** depend on /proc/ as the source of process information. Inside /proc/ exists a sub-directory whose name corresponds with the PID of each process on the system, for example, /proc/31895/. Each process directory contains files that can be viewed such as cmdline, environ and status. Also, there are symlinks for root, exec and cwd that link to the process's filesystem root, executable and current working directory, respectively.

The directory /proc/PID/fd/ contains symlinks to files (when applicable) for each file handle that the process has open.

Configuring the Kernel via /proc/sys/

The majority of the files in /proc/ are read-only, but most of the files found under the /proc/sys/ hierarchy are writable by the root user, and can be used to tune various aspects of the running kernel and its modules. The /proc/sys/ directory is the only part of the /proc/ filesystem that is writable.

To view the current value of a procfs file, use the **cat** command as shown in this example:

```
# cat /proc/sys/fs/file-max  
101643
```

You can set the file to a new value using the **echo** command and redirecting the output to the file as shown in this example:

```
# echo "110000" > /proc/sys/fs/file-max
```

Using the sysctl Command

Instead of using the **cat** and **echo** commands as shown in the previous examples, you can use the **sysctl** command to view and set values in /proc/. For example, to view and modify the same value as shown before, you could execute these commands:

```
# sysctl fs.file-max  
fs.file-max = 101643  
# sysctl -w "fs.file-max=110000"
```

The **sysctl** command can also list all available kernel tuning options as shown in this example:

```
# sysctl -a  
... snip ...  
fs.overflowuid = 65534  
fs.dentry-state = 21296 18868 45 0 0 0  
fs.file-max = 101643  
fs.file-nr = 975 0 101643  
... snip ...
```

Making Tuning Changes Permanent

Changes to the values in /proc/ will not survive a reboot. To provide for permanent changes, the `systemd-sysctl.service` runs the `/lib/systemd/systemd-sysctl` command on boot. This reads settings from the `/usr/lib/sysctl.d/*.conf`, and then the `/etc/sysctl.conf` file and `/etc/sysctl.d/` directory (which overrides for values found in both files). The syntax is defined in the `sysctl.d(5)` man page.

For example, to have the `file-max` parameter set to a specific value each boot, do the following:

```
File: /etc/sysctl.conf
```

```
+ fs.file-max = 57500
```

Console

Drivers

- system driver
- frame-buffer drivers

Device Files

Serial Console

Console Log Levels

Console Drivers

The Linux system console can use either the standard VGA driver, or a video chipset specific modular frame buffer driver. The VGA driver is always present in the kernel and will bind automatically (become active) to the console if no other driver is loaded. Chipset specific frame buffer drivers can be loaded by passing the **video=driver_name** option to the kernel on boot. Alternatively, the vgacon or vesafb drivers (which support all the standard VGA and VESA video modes respectively) can be configured by passing the **vga=video_mode|ask** argument to the kernel.

If a frame buffer device is in use, the kernel will display the Tux penguin on boot. To determine exactly which driver is in use, view the values exported in `/sys/` as shown in the following example:

```
[host_using_system_driver]
# cat /sys/class/vtconsole/vtcon0/name
(S) VGA+
[host_using_vesafb_driver]
# cat /proc/cmdline
ro root=LABEL=/ vga=31b
# cat /sys/class/vtconsole/vtcon0/name
(S) dummy device
# cat /sys/class/vtconsole/vtcon1/name
(M) frame buffer device
```

Console Device File

The `/dev/console` character device is managed by the kernel and by default will point to the currently active TTY. Kernel messages are sent to `/dev/console` with the intent that the user at the physical console will see the messages (regardless of which specific virtual console is currently active). Instead of the default behavior, the console can be set to point to either a specific TTY or a serial port. For example, to have console messages always sent to the first virtual terminal pass **console=tty1** as a kernel argument. When in a graphical X session, console messages can be seen by running **xconsole**.

Serial Console

The kernel has support for running a serial console. Full documentation is found in the `/usr/share/doc/kernel-doc*/Documentation/kernel-parameters.txt` file. For example, to enable both a local TTY and serial console, pass the following kernel arguments through the bootloader: **console=tty0** **console=ttyS0,115200**. Kernel messages would then be logged to both.

When a serial console is enabled, it is also common to modify GRUB to use the same serial port for interaction and login. This is done by making the following additions:

File: /etc/default/grub

```
+ GRUB_CMDLINE_LINUX_DEFAULT="console=tty0,console=ttyS0,115200n8"
+ GRUB_TERMINAL="console serial"
+ GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1"
```

By listing two devices, console serial, whichever ever device gets the first keystroke will be used as the GRUB terminal.

On systemd enabled Linux distros, **agetty** will automatically launch a serial console if a serial console was specified as the last console on the kernel command line. This is done by the systemd `getty.target` via the `getty@.service` and `serial-getty@.service` service definitions.

Disabling Console Screen Blanking

To turn off the automatic screen blanking for the console, use the `consoleblank` kernel parameter. This is especially useful on a hung or kernel panicked systems so that the console messages can be viewed. Without this the console will blank, and because of the kernel panic, cannot be un-blanked.

File: /etc/default/grub

```
→ GRUB_CMDLINE_LINUX=". . . snip . . . consoleblank=0"
```

Network Console

The kernel has support for running a network console. This can be useful to capture kernel messages, (especially when troubleshooting kernel crashes). Assuming the kernel was built with the `CONFIG_NETCONSOLE` option, pass the

`netconsole=[src_port]@[src_ip]/[device],[dst_port]@dst_ip/[dst_mac]`

argument to the kernel on boot. Optionally, if support has been built as a module, netconsole support can be started at any time by loading the kernel module and passing the needed options as shown in this example:

```
# modprobe netconsole "netconsole=8000@10.100.0.3/eth0,514@10.100.0.4/00:1B:21:24:F8:CB"
```

Console Log Levels

Kernel code can emit messages to the system console using the `kernel printk()` function. The `/proc/sys/kernel/printk` file holds four numeric values related to which kernel messages are sent to the console, (see `proc(5)`). These can be changed by writing new values to the file, or via the `sysctl` command, (`kernel.printk` boolean). To view the current values, run either of the following:

```
$ cat /proc/sys/kernel/printk
3      4      1      7
$ sysctl kernel.printk
kernel.printk = 3      4      1      7
```

The meaning of the four values left to right, as given below top to bottom, are:

Console Log Level ⇒ messages with a higher priority than this value will be printed to the console

Default Message Log Level ⇒ messages without a priority will be printed with this priority

Minimum Console Log Level ⇒ minimum (highest priority) value that the Console Log Level can be set to

Default Console Log Level ⇒ default value for Console Log Level

An alternative, easy way, to set the console log level is with the `dmesg -n level` command. For debugging, you can cause all kernel messages to be displayed on the console with timestamps by passing the `ignore_loglevel` and `time` arguments on boot.

Virtual Terminals

Device Files

- /dev/tty
- /dev/vcs

Switching

Opening, Closing, and Locking

Scrolling

Common Changes

- font
- keymaps
- locale
- screen blanking

Switching Virtual Terminals

The first twelve TTYs can be accessed by pressing **[Alt] + [F1]** through **[Alt] + [F12]** respectively (when in X, **[Ctrl]** is also necessary). Optionally, using **[Alt] + [←]** or **[Alt] + [→]** works when not in X and can be used to access high numbered terminals (moving sequentially from one terminal to the next). Finally, the **chvt vt_num** command will switch to the specified virtual terminal.

Opening, Closing, and Locking Virtual Terminals

Additional TTYs can be started with the **openvt** command and kernel resources associated with a TTY can be freed with the **deallocvt** command. For security, the current virtual console (default), or all virtual consoles, can be locked with the **vlock** command. The following demonstrates these commands:

```
[Ctrl] + [Alt] + [F5]
station1 login: root
password: makeitso [Enter]
# tty
/dev/tty5
# openvt -sw bash
# tty
/dev/tty8
[Alt] + [←] [Alt] + [←] [Alt] + [←]
# tty
/dev/tty5
# chvt 8
```

Virtual Terminals

Although /dev/console is the primary device file associated with the system console, writes to the device are typically redirected to another device (such as a specific virtual terminal or serial device). Virtual terminals are full-screen TTY devices on the system video monitor. Up to 63 TTYs are supported corresponding to device files **/dev/tty[1-63]**. The **/dev/tty0** file is special and points to whichever TTY is currently active. Under normal operation, systemd starts a login on a tty when a user switches to the tty (with 6 TTYs being the default). This number can be changed by editing the logind config:

```
File: /etc/systemd/logind.conf
[Login]
#NAutoVTs=6
... snip ...
```

Programs can direct output to unused TTYs. For example:

```
# tail -f /var/log/messages > /dev/tty12 &
```

The current contents (text minus any attributes) of each virtual terminal can be read via the **/dev/vcs[1-x]** file. The contents of the currently active virtual terminal can be read via **/dev/vcs**. To see text plus attributes, use **/dev/vcsa[X]**. For example:

```
# cat /dev/vcs
station1 login:
```

```
# vlock -a  
The entire console display is now completely locked.  
You will not be able to switch to another virtual console.  
Please enter the password to unlock.  
root's Password: makeitso   
# exit  
# tty  
/dev/tty5  
# deallocvt 8
```

Scrolling

A scrollback buffer is maintained for the currently active virtual terminal. Scroll backward by pressing **Shift**+**Page Up**, and scroll forward with **Shift**+**Page Down**. When a switch to another virtual terminal is made, the contents of the scrollback buffer are lost.

If anything is written to the terminal while it is currently displaying something in the scrollback region, the terminal will automatically jump forward to display the new text. This can make it difficult to examine messages in the scrollback history (especially during boot, or shutdown) as the display will continue to jump away from the desired text as new messages are written. Pressing the Scroll Lock key will toggle the state of the scroll-lock flag for the currently active virtual terminal. When scroll-lock is enabled on a particular TTY, the kernel will block all writes to that TTY (causing any process attempting to write to the TTY to enter an uninterruptable sleep state waiting for the I/O write to complete). The **Shift**+**Page Up** and **Shift**+**Page Down** key combos will still work to scroll the terminal while scroll-lock is engaged. Scrollback support can be completely disabled by passing the **no-scroll** argument to the kernel on boot.

Changing Screen Blanking

By default, the virtual terminals will be automatically blanked (using APM if available) after 10 minutes of inactivity. The interval of inactivity that will trigger blanking can be set with **setterm -blank *interval_minutes***. Virtual terminal screen blanking can be disabled entirely with the following command:

```
# setterm -powersave off -blank 0
```

Keyboard & locale configuration

Console font management

- `setfont`
- `/etc/vconsole.conf`
 FONT variable

Keyboard mappings

- `loadkeys|dumpkeys|localectl`
- `/etc/vconsole.conf`
 KEYMAP variable

System locale

- `localectl`
- `/etc/locale.conf`
 LANG variable

The keyboard translation table is shared by all virtual terminals and changes will outlive a single session. For example, if a user logs into TTY1 and changes the mapping, someone later logging into TTY2 would see the same mappings (even if the user who made the change had logged off of TTY earlier).

Switch virtual terminals to use Dvorak key mappings:

[R7] The following applies to RHEL7 only:

```
# loadkeys /lib/kbd/keymaps/i386/dvorak/dvorak.map.gz
```

[S12] The following applies to SLES12 only:

```
# loadkeys /usr/share/kbd/keymaps/i386/dvorak/dvorak.map.gz
```

Disable the scroll-lock key:

```
# echo "keycode 70 = nul" | loadkeys -
```

Swap the caps-lock and left control keys:

```
# dumpkeys | sed 's/58 = Caps_Lock/58 = Control/; s/97 = Control/97 = Caps_Lock/' | loadkeys -
```

The new `localectl` command introduced with systemd can also be used to list and set keymaps for both the text console and X-11:

```
# localectl list-keymaps  
ANSI-dvorak  
amiga-de  
amiga-us
```

```
. . . output omitted . . .
# localectl set-keymap dvorak
```

To make the keymap changes persistent, modify
`/etc/vconsole.conf`:

```
File: /etc/vconsole.conf
KEYMAP="dvorak"
```

Viewing and Setting the Locale

The output of programs (language, sort order, format for date/time, currency, numbers, etc.) is formatted based on the current locale setting. The system locale can be viewed and set via the new **localectl** command. The current value of `LC_*` variables that can override can be seen with the older **locale** command:

```
# localectl
System Locale: LANG=en_US.UTF-8
    VC Keymap: us
    X11 Layout: us
# localectl list-locales | grep en_
en_AG
en_AG.utf8
en_AU
. . . snip . . .
# localectl set-locale LANG=en_GB.utf8
# locale
LANG=C
LC_CTYPE="C"
LC_NUMERIC="C"
. . . snip . . .
```

To make the locale changes persistent, modify `/etc/locale.conf`:

```
File: /etc/locale.conf
LANG="en_GB.utf8"
```

Serial Ports

```
/dev/ttys#  
minicom - TUI serial communications program  
cuteocom - GUI serial communications program  
setserial - show or change port settings  
rzsz package - Implements Zmodem, Ymodem and Xmodem file  
transfer protocols  
c-kermit - Implements Kermit file transfer protocol
```

Legacy Serial Ports

The venerable serial port has been around even longer than the PC. There are many different kinds of serial ports; GPIB, RS-232, RS-422 RS-485 being some of the most common. The big differences are in what kinds of connectors are used, which pins carry data or other signaling, the voltages used, etc.

Almost every PC ever built has at least one RS-232 serial port. This table shows the important, common parameters and names of the first 4 serial ports:

Linux Device Node	Port I/O Address	IRQ
/dev/ttys0	0x3f8	4
/dev/ttys1	0x2f8	3
/dev/ttys2	0x3e8	4
/dev/ttys3	0x2e8	3

The minicom Command

Many network devices such as routers and switches have serial ports that can be used to configure the device. The **minicom** command initializes a serial port and provides an interactive interface for working with these kind of devices. Device specific serial port settings can be stored in individual configs that are then called when connecting to that device. For example:

```
# cat /etc/minirc.cisco  
pu baudrate      9600  
pu bits          8  
pu parity        N  
pu stopbits      1  
# minicom /etc/minirc.cisco  
...serial port is initialized and connection is established...  
This system is the property of Guru Labs, L.C.  
UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED.
```

User Access Verification

Username:

. . . snip . . .

The setserial Command

Most programs that interface with a serial port, such as terminal emulators, will send the necessary configuration commands to initialize the serial port. On the rare occasions that a program does not do this, and the defaults for the port and application are incompatible, the **setserial** command can be used to configure the port. **setserial** can also be used to list the current settings of a serial port:

```
# setserial -av /dev/ttys0  
/dev/ttys0, Line 0, UART: unknown, Port: 0x03f8, IRQ: 4  
Baud_base: 115200, close_delay: 50, divisor: 0  
closing_wait: 3000  
Flags: spd_normal skip_test auto_irq
```

Random Numbers and /dev/random

Random Numbers are very important

- Used as unique IDs in website cookies for session tracking
 - Embedded in publicly accessible, yet "private" URLs
 - Used in session keys during secure connection establishment
- /dev/random — **collected pool of randomness (entropy)**
- Uses inter-interrupt timing as entropy source
 - Slow refill rate, especially on mostly idle systems
 - Includes entropy from trusted hardware RNGs with kernel 3.16+
 - Only trusts VirtIO RNG by default with kernel 3.17+

/dev/urandom — **unlimited random data**

- Merely cryptographically strong, but acceptable for most applications

Random Numbers Are Widely Used

Random numbers are used behind the scenes in many aspects of computing. Two main areas are in the generation of unique and unguessable identifiers and secondly as part of session keys for encrypted communication. These two areas consume random numbers on an ongoing basis and the demand for random numbers can fluctuate. For example if website traffic peaks at a certain time of day or hundreds of employees are connected via VPN during a certain time window. Random numbers are also used in one-off scenarios, most commonly in the generation of private/public keypairs. In any of these uses, if the random numbers are unavailable or of poor quality it either causes delays or defeats the security.

Most websites use random numbers to generate unique IDs which get sent to the user in their cookie. These unique IDs are used for session tracking and so that the website can distinguish one user from another. If someone can guess your unique ID, they can impersonate you on that website.

Similarly, most "private" content on the Internet is simply hidden behind URLs that contain random numbers within the URL with privacy coming from the inability of someone else being able to figure out the URL.

When establishing an encrypted connection using SSL, TLS, SSH or even an application protocol with built-in encryption such as ODBC or JDBC, session keys are generated using random numbers during the connection establishment phase, and then periodically after timer or

byte counters expire triggering a re-keying. Connection delays are very often due to insufficient random numbers being available to generate the session keys.

/dev/random

Computers are deterministic machines that are the very opposite of random and so obtaining or collecting entropy is a difficult problem. Linux uses inter-interrupt timing from hardware devices as random input and collects into an entropy pool accessible via /dev/random. This is why moving the mouse or typing on the keyboard can speed up public key pair generation. When /dev/random is read to obtain random numbers, a SHA hash of the contents of the pool is performed in order to hide the internal state of the pool (if any). When the pool is empty, reads will block. To get an estimate of the amount of entropy in the pool run:

```
# cat /proc/sys/kernel/random/entropy_avail  
1091
```

Unless /dev/random is fed with entropy from a hardware or software RNG, it will block and cause delays for any applications that read from it faster than it is refilled. This is particularly noticeable on mostly idle servers without a user using the keyboard and mouse.

/dev/hwrng and VirtIO RNG

Since random number generation is a difficult problem, hardware chips exists that use a physical process such as quantum mechanics, thermal noise, optical, or atmospheric noise as a source of entropy.

Linux supports multiple different hardware RNGs, but doesn't trust them by default so their entropy is available separately via /dev/hwrng. With Linux kernel 3.16 and higher, the trust is defined on a sliding quality scale from 0 to 1000, where 0 is not trusted at all and not mixed into /dev/random by the kernel, and 1000 is fully trusted and mixed in its entirety into /dev/random.

All hardware RNGs have a default quality of 0, to change this add the `rng_core.current_quality` and `rng_core.default_quality` command line parameters to a desired non-zero value up to and including 1000 in the /etc/default/grub. For example:

```
rng_core.current_quality=700 rng_core.default_quality=700
```

With Linux kernel 3.17 and higher, the qemu VirtIO RNG is trusted fully by default, so if the virtual machine has the following libvirt XML device added, it will mix the VirtIO RNG into /dev/random by default. This is a big deal for virtual machines, which are often starved for entropy sources:

```
<devices>
  ...
  <rng model='virtio'>
    <rate bytes='6000' period='300' />
    <backend model='random'>/dev/random</backend>
  </rng>
</devices>
```

rngd and the RdRand CPU Instruction

Prior to Linux kernel 3.16, when using a hardware RNG, the `rngd` daemon from the `rng-tools` package is needed. It reads from /dev/hwrng and writes to /dev/random. With kernels 3.16 and newer, `rngd` like functionality is integrated into the kernel itself. To see a list of available HW entropy sources run:

```
# rngd -fv
Available entropy sources:
  Intel/AMD hardware rng
  DRNG
[Ctrl]+[C]
```

The DRNG RNG doesn't refer to a traditional RNG, but instead refers to the new RdRand CPU instruction which returns a random number. The kernel doesn't automatically use DRNG and so `rngd` must be run if use

of DRNG is desired. To see if the CPU supports the instruction, run:

```
# grep --color rdrand /proc/cpuinfo
flags      : ...snip long list of flags... rdrand
```

/dev/urandom & Pseudorandom Number Generators

A pseudorandom number generator must be seeded with an initial random number, but then uses a deterministic algorithm to generate a stream of numbers that exhibit statistical randomness. Linux provides the /dev/urandom pseudorandom number generator which is seeded from /var/lib/systemd/random-seed at boot. The main benefit of /dev/urandom is that it provides an unlimited, high throughput non-blocking source of random numbers.

Unless a public key pair is being generated, or extreme security is needed, most applications should use /dev/urandom instead of /dev/random. For example, to configure the Java virtual machine to seed its SecureRandom class with /dev/urandom instead of /dev/random, add the following to the `java` command line invocation:

```
-Djava.security.egd=file:/dev/.urandom
```

Or for a global Java change, edit the `java.security` config file:

```
File: $JAVA_HOME/jre/lib/security/java.security
+ securerandom.source=file:/dev/.urandom
```

The haveged daemon

The hardware volatile entropy gathering and expansion (HAVEGE) daemon can be used to collect entropy from the thousands of binary volatile states that exist in modern superscalar processors and feed that entropy into /dev/random at hundreds of megabits per second. It is a good alternative if an application can't be reconfigured to use /dev/urandom and a HW RNG isn't available or fast enough.

Benchmarking Random Number Generation Performance

To determine the speed that /dev/random, /dev/urandom or /dev/hwrng can generate random numbers, run the `dd` for approximately 10 seconds:

```
# dd if=/dev/random of=/dev/null
[Ctrl]+[C]0+1825 records in
106+0 records out
54272 bytes (54 kB) copied, 10.9434 s, 5.0 kB/s
```

Lab 1

Estimated Time:
S12: 60 minutes
R7: 60 minutes

Task 1: Adjusting Kernel Options

Page: 1-35 Time: 30 minutes

Requirements:  (1 station)  (classroom server)

Task 2: Linux Kernel Driver Compilation

Page: 1-41 Time: 10 minutes

Requirements:  (1 station)  (classroom server)  (graphical environment)

Task 3: Introduction to Troubleshooting Labs

Page: 1-45 Time: 10 minutes

Requirements:  (1 station)

Task 4: Troubleshooting Practice: Kernel Modules

Page: 1-50 Time: 10 minutes

Requirements:  (1 station)

Objectives

- ❖ Configure restricted access to the kernel messages.
- ❖ Configure a persistent kernel tunable (link protections).
- ❖ Trigger the kernel VM drop cache function.
- ❖ Disable ICMP broadcast replies.

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

The Linux kernel has hundreds of tunable options that can affect the performance or security of the system. Being able to tune these options in a persistent manner is an important system administration skill.

Restrict dmesg

- 1) [S12] This step should only be performed on SLES12.

On SUSE Linux Enterprise Server, dmesg is restricted by default. Unrestrict it:

```
$ su -c 'sysctl -w kernel.dmesg_restrict=0'  
Password: makeitso   
kernel.dmesg_restrict = 0
```

- 2) Examine the current value of the kernel.dmesg_restrict kernel option and verify that unprivileged users can read from the kernel message buffer:

```
$ id -un  
guru  
$ cat /proc/sys/kernel/dmesg_restrict  
0  
$ ls -l /dev/kmsg  
crw-r--r--. 1 root root 1, 11 Oct 28 12:15 /dev/kmsg  
$ dmesg  
. . . output omitted . . .
```

Lab 1

Task 1

Adjusting Kernel Options

Estimated Time: 30 minutes

- This value means that unprivileged users can read the contents of the kernel message buffer.

- 3) Configure restricted access to the kernel message buffer and verify that permission is now denied to non-root users:

```
$ su -c 'sysctl -w kernel.dmesg_restrict=1'
```

```
Password: makeitso [Enter]
kernel.dmesg_restrict = 1
$ ls -l /dev/kmsg
crw-r--r--. 1 root root 1, 11 Oct 28 12:15 /dev/kmsg
$ dmesg
dmesg: read kernel buffer failed: Operation not permitted
$ cat /dev/kmsg
cat: /dev/kmsg: Operation not permitted
```

Even though the /dev/kmsg file still had global read permissions, the kernel now blocks non-root attempts to read from it.

Link Protection

- 4) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso [Enter]
```

- 5) Create a new sysctl config containing the default/current values for the kernel tunables related to link security:

```
# sysctl fs.protected_{hard,sym}links > /etc/sysctl.d/50-link.conf
# cat /etc/sysctl.d/50-link.conf
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
```

[R7] *The following applies to RHEL7 only:*

This is new to RHEL7.

- 6) Edit the new config so that both options are disabled:

```
# sed -i s/1/0/ /etc/sysctl.d/50-link.conf
# cat /etc/sysctl.d/50-link.conf
fs.protected_hardlinks = 0
fs.protected_symlinks = 0
```

[R7] *The following applies to RHEL7 only:*

Having these both turned off is equivalent to the behavior of RHEL 6 (and past versions).

- 7) Restart the sysctl service and verify that the kernel options change to match the config:

```
# systemctl restart systemd-sysctl  
# sysctl fs.protected_{hard,sym}links  
fs.protected_hardlinks = 0  
fs.protected_symlinks = 0
```

- 8) Create a symbolic link to test the operation of the system with the link protection kernel features disabled:

```
# su -c 'ln -s /etc/passwd /tmp/passwd' visitor  
# ls -l /tmp/passwd  
lrwxrwxrwx. 1 visitor visitor 11 Nov  3 14:28 /tmp/passwd -> /etc/passwd
```

- 9) Create a hard links to test the operation of the system with the link protection kernel features disabled:

```
# su -c 'ln /etc/shadow ~guru/' guru  
# ls -l ~guru/shadow  
-----. 2 root root 1205 Aug 25 10:37 /home/guru/shadow
```

- Mode and group ownership may vary, depending on distribution.

The fact that users can create hard links to files they otherwise have no access to is a risk. For example, consider what happens if the root user now runs chown -R guru: ~guru. The guru user would then be the owner of the /etc/shadow file.

- 10) Verify that with link protections disabled, symbolic links can be followed by a user other than the one who owns it:

```
# su -c 'cat /tmp/passwd' guru  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
. . . output omitted . . .
```

Especially in /tmp this has lead to hundreds of exploits over the years where users exploit race conditions and create links that are inadvertently followed by another user's processes.

- 11) Re-enable the new kernel link protection options and re-test:

```
# sed -i s/0/1/ /etc/sysctl.d/50-link.conf
# systemctl restart systemd-sysctl
# ls -l /tmp/passwd
lrwxrwxrwx. 1 visitor visitor 11 Nov  3 14:28 /tmp/passwd -> /etc/passwd
# cat /tmp/passwd
cat: /tmp/passwd: Permission denied
# su -c 'ln /etc/shadow ~guru/shadow2' guru
ln: failed to create hard link '/home/guru/shadow2' ->
=> '/etc/shadow': Operation not permitted
```

- Even as root the kernel prevents the link from being followed.
- Users can no longer create links to files they have no permissions to.

Dropping VM Caches

- 12) View the current size and number of cache pages allocated by the VM kernel subsystem for caching inodes and directory entries:

```
# vmstat -m | awk 'NR == 1 || /xfs_inode|dentry/'
Cache           Num  Total   Size  Pages
xfs_inode      1447    3456   1024     8
dentry         14176   26733    192    21
```

- 13) Do a recursive walk of the filesystem causing the kernel to populate those caches with more objects:

```
# ls -R / &> /dev/null
# vmstat -m | awk 'NR == 1 || /xfs_inode|dentry/'
Cache           Num  Total   Size  Pages
xfs_inode      111176  111916   1024     8
dentry         66383   68397    192    21
```

- Notice that number of cache pages has grown substantially.

- 14) Tell the kernel to drop both the inode and dentry caches and verify that the amount of memory allocated returns to roughly the previous value:

```
# sync; sysctl -w 'vm.drop_caches=2'
vm.drop_caches = 2
# vmstat -m | awk 'NR == 1 || /xfs_inode|dentry/'
Cache           Num  Total   Size  Pages
xfs_inode      1890    5104   1024     8
dentry         13828   26712    192    21
```

Configuring ICMP Broadcast Response

- 15) If your system has been configured to ignore ICMP echo-broadcasts for extra security, this kernel parameter must be disabled prior to the upcoming steps in this lab. As the root user, temporarily enable ICMP echo-broadcast support:

```
# sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0  
net.ipv4.icmp_echo_ignore_broadcasts = 0
```

- 16) ICMP ping requests sent to the broadcast address are answered by each host that listens to that broadcast address. Verify that your system is responding to broadcast pings:

```
# ping -b -c 3 10.100.0.255  
WARNING: pinging broadcast address  
PING 10.100.0.255 (10.100.0.255) 56(84) bytes of data.  
64 bytes from 10.100.0.X: icmp_seq=0 ttl=64 time=0.063 ms  
64 bytes from 10.100.0.Y: icmp_seq=0 ttl=64 time=0.173 ms (DUP!)  
64 bytes from 10.100.0.Z: icmp_seq=0 ttl=64 time=0.187 ms (DUP!)  
64 bytes from 10.100.0.W: icmp_seq=0 ttl=64 time=0.197 ms (DUP!)  
64 bytes from 10.100.0.V: icmp_seq=0 ttl=64 time=0.205 ms (DUP!)  
64 bytes from 10.100.0.U: icmp_seq=0 ttl=64 time=0.213 ms (DUP!)  
64 bytes from 10.100.0.X: icmp_seq=1 ttl=64 time=0.039 ms  
. . . snip . . .  
--- 10.100.0.255 ping statistics ---  
3 packets transmitted, 3 received, +34 duplicates, 0% packet loss, time 1000ms  
rtt min/avg/max/mdev = 0.039/0.153/0.213/0.068 ms, pipe 2
```

- Your system's IP address should appear somewhere in the output. Other systems in the classroom may or may not appear in the listing.

- 17) Alter the system so that it will not answer ICMP requests for a broadcast address by tuning the TCP/IP stack via /proc/:

```
# sysctl net.ipv4.icmp_echo_ignore_broadcasts=1  
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

- 18) Verify that the change to /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts was successfully applied by pinging the broadcast address again and verifying that the system does not respond this time:

```
# ping -b -c 3 10.100.0.255  
WARNING: pinging broadcast address  
PING 10.100.0.255 (10.100.0.255) 56(84) bytes of data.
```

```
64 bytes from 10.100.0.Y: icmp_seq=0 ttl=64 time=0.116 ms
64 bytes from 10.100.0.Z: icmp_seq=0 ttl=64 time=0.127 ms (DUP!)
64 bytes from 10.100.0.W: icmp_seq=0 ttl=64 time=0.136 ms (DUP!)
64 bytes from 10.100.0.V: icmp_seq=0 ttl=64 time=0.146 ms (DUP!)
64 bytes from 10.100.0.U: icmp_seq=0 ttl=64 time=0.156 ms (DUP!)
64 bytes from 10.100.0.T: icmp_seq=0 ttl=64 time=0.166 ms (DUP!)
. . . snip . . .
--- 10.100.0.255 ping statistics ---
3 packets transmitted, 3 received, +30 duplicates, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.021/0.164/0.234/0.053 ms, pipe 2
```

- Your system's IP address should no longer appear in this output.

Cleanup

- 19) Remove the previously created persistent kernel tuning file:

```
# rm /etc/sysctl.d/50-link.conf
```

- 20) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Build, test, and install a new driver for the currently running kernel.

Requirements

- █ (1 station) █ (classroom server) X (graphical environment)

Relevance

Sometimes it is necessary to add missing capabilities to the kernels that will be used for a particular system. This might include drivers for newer or specialized hardware. In many cases, hardware vendors provide Linux drivers in source form that can be built without a full kernel tree, separately from building the kernel. These are sometimes called "external modules".

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) As root, install packages needed to compile kernel modules:

```
[R7] # yum install -y gcc kernel-devel-$(uname -r)  
[R7] . . . output omitted . . .  
[S12] # zypper install -y gcc kernel-source  
[S12] . . . output omitted . . .
```

- 3) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

- 4) Create a directory from which to work with source packages. As the guru user, run these commands:

```
$ mkdir ~/src
```

Lab 1

Task 2

Linux Kernel Driver Compilation

Estimated Time: 10 minutes

- 5) Enter the new ~/src/ directory and extract the source code for the new module:

```
$ cd ~/src/  
$ tar xvf /labfiles/fooru-0.2.tar.bz2
```

- 6) Change into the directory for the new kernel module and list the contents:

```
$ cd fororu-0.2/  
$ ls  
. . . output omitted . . .
```

- 7) Build the driver:

```
$ make  
. . . output omitted . . .
```

- 8) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 9) Use the load-module script (as root) to test the newly built kernel module:

```
# cd ~guru/src/fooru-0.2/  
# insmod fororu.ko
```

- Remember, only root can load and unload modules for the kernel.

- 10) Use the lsmod command to verify that the module was, indeed, loaded successfully:

```
# lsmod | head -n 2  
Module           Size  Used by  
fooru            2424  0
```

- The size of the module may differ from this value.

- 11) Test the new driver by reading from the device node it is supposed to handle:

```
# head /dev/fooru  
guru  
rfc3092  
tux  
bar  
fred  
qux  
baz  
tux  
quux  
thud
```

- Remember, by default, head will only return the first 10 lines from the specified file. Also, because of the function of this particular kernel module, the output will (almost always) differ from that listed here. This command will work for any user who has access to the device node.

- 12) Unload the module and delete the corresponding device node:

```
# rmmod fooru
```

- 13) As root, use make to install the fooru module into the modules directory structure for the currently running kernel:

```
# make install  
. . . output omitted . . .
```

- 14) Load the module using the modprobe command:

```
[R7] # modprobe fooru  
[S12] # modprobe --allow-unsupported fooru
```

- 15) Verify that the module was loaded successfully and that the correct device node is present:

```
# lsmod | head -n 2  
Module           Size  Used by  
fooru            7520   0  
# ls -l /dev/fooru  
crw-rw---- 1 root root 252,  0 Apr 22 16:46 /dev/fooru
```

16) Test the driver:

```
# head /dev/fooru  
foo  
fred  
baz  
thud  
tux  
. . . snip . . .
```

Objectives

❖ Practice using the tsmenu command.

Requirements

▀ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You can use these scripts to break your system in a controlled way, then practice troubleshooting and fixing the problem.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) As the root user, invoke the tsmenu command:

```
# tsmenu
```

- 3) The first time the troubleshooting framework is started, some information about your system is needed:

Press to continue.

The Preliminary Information screen states that tsmenu is about to collect information about the system.
This will happen only once.

Select Yes then press to continue.

Confirm the correct Linux distribution was detected. Use the (left arrow) key and (right arrow) key to switch between Yes and No.

Press to continue.

Confirm your preferred Ethernet device was detected.

The Select Troubleshooting Group screen is displayed.

- 4) This first scenario is a simple HOWTO for the tsmenu command. Its function is to familiarize you with the usage of tsmenu:

Lab 1

Task 3

Introduction to Troubleshooting Labs

Estimated Time: 10 minutes

Select Troubleshooting Group #0.

Use the (up arrow) key and (down arrow) key to select a group of troubleshooting scenarios.

Select OK then press .

Use the (left arrow) key and (right arrow) key to choose whether to switch to the next or previous screen.

The Select Scenario Category screen is displayed.

- 5)** Troubleshooting Group #0 contains only one scenario category:

Select the Learn category.

Pick the scenario category to view.

Select OK then press .

Continue to the next screen.

The Select Scenario Script screen is displayed.

- 6)** The Learn category contains only one scenario:

Select the learn-01.sh scenario.

Pick the break script to run.

Select OK then press .

Continue to the next screen.

The Break system? screen is displayed.

- 7)** The system is about to be broken. Before breaking the system, read the description of the problem to solve in this scenario:

Read the scenario description.

Make sure you're prepared to "break the system now."

Select Yes then press .

Run the break script.

Wait for the break script to run.

Some break scripts can take up to a couple minutes to run.

The SYSTEM IS BROKEN! screen is displayed.

- 8) The tsmenu command is now locked on the selected scenario and will not permit another scenario to run until the current scenario is solved:

Contemplate the fact that you have just deliberately broken your own system.

Life is funny sometimes, isn't it?

Select OK then press .

Begin the troubleshooting process.

The tsmenu command stops running. Depending on the scenario, a reboot may be required before the problem is noticeable. In these cases, the system will reboot automatically after you press .

- 9) It is possible to re-read the scenario description two different ways. First, the description is saved in a text file. Display the contents of this file:

```
# cat /problem.txt  
... output omitted ...
```

Second, re-run the tsmenu command.

- 10) Each time the tsmenu command runs, it checks to see if the current problem has been solved. If the problem hasn't been solved, tsmenu will provide information about the current scenario instead of presenting a list of new scenarios.

As the root user, re-invoke the tsmenu command:

```
# tsmenu
```

The Scenario Not Completed screen is displayed.

- 11) It is not possible to run another break script until the current scenario has been finished.

Select OK then press .

- 12) If unsure of how to proceed, the tsmenu command can provide hints. It doesn't immediately reveal the solution, but instead presents gradual hints in the order of a realistic troubleshooting process.

View all of the learn-01.sh hints:

Select the Hint menu item, then select OK and press **[Enter]**.

View a hint for the current problem.

Read the first hint.

Select OK then press **[Enter]**.

Return to the scenario menu.

Press **[Enter]** then read the second hint.

Press **[Enter]** to return to the scenario menu.

Press **[Enter]** then read the third hint.

Press **[Enter]** to return to the scenario menu.

Notice that the total number of hints available is indicated and previous hints are re-shown.

- 13) Instead of closing and re-running the tsmenu command to check if the current scenario problem has been solved, it's possible to re-check the problem by using the scenario menu's Check menu item:

Select Check, then select OK and press **[Enter]**.

Check if the problem scenario is solved.

Note the scenario is not completed.

Select OK then press **[Enter]**.

Return to the scenario menu.

- 14) If the problem scenario has not been solved and tsmenu won't let a new scenario be selected, carefully review the requirements in the scenario description. If still unsure about how to proceed then consult the instructor.

Re-read the scenario description, then close tsmenu:

Select Description, then select OK and press **[Enter]**.

View the scenario description.

Re-read the scenario description.

Select OK then press **[Enter]**.

Return to the scenario menu.

Select Cancel then press **[Enter]**.

Close tsmenu and return to the command line.

- 15) Solve the scenario problem by creating the required file:

```
# touch /root/solved
```

- 16) Launch the tsmenu command again:

```
# tsmenu
```

- 17) As usual, tsmenu checks to see if you've solved the current problem. Once the problem is solved, the Troubleshooting Group screen is unlocked and another scenario can be explored:

Note that you've completed the scenario.

Select OK then press **Enter**.

The Troubleshooting Group screen is now unlocked.

Select Cancel then press **Enter**.

Close tsmenu and return to the command line.

Objectives

- ❖ Practice troubleshooting kernel module issues.

Requirements

- ☒ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You use these scripts to break your system in controlled ways, and then you troubleshoot the problem and fix the system.

Notices

- ❖ The tsmenu program requires root access to the system and will need to be run from a root shell.

- 1) Use tsmenu to complete the kernel module troubleshooting scenario:

Troubleshooting Group	Scenario Category	Scenario Name
Group 5	Kernel	kernelmodule-01.sh

Lab 1

Task 4

Troubleshooting Practice: Kernel Modules

Estimated Time: 10 minutes

Content

System Boot Method Overview	2
systemd System and Service Manager	3
Modifying systemd services	5
Systemd Service Sandboxing Features	6
systemd Targets	7
Using systemd	8
Linux Runlevels Aliases	10
Legacy Support for SysV init	11
Lab Tasks	12
1. Managing Services With Systemd's systemctl	13
2. Creating a systemd unit file	19

Chapter

2

SYSTEMD OVERVIEW

System Boot Method Overview

Runlevel-driven - AT&T System V init

- Each runlevel can have a unique defined list of services to **start** and **stop**
- Used by most commercial Unix systems and Linux distributions

Event-driven - Upstart

- Originally created for Ubuntu, also used with RHEL6
- Builds upon SysV style, launches scripts based on events

Dependency-driven - Systemd

- Parallelizes as much as dependencies allow
- Unit files replace SysV init scripts
- Used in RHEL7, SLES12, Ubuntu 15.04

init Styles

After the Linux kernel runs **/sbin/init**, and **init** reads its configuration file and starts all programs listed there, bringing the system up to a predefined working state, or runlevel, two styles of **init** are commonly used. BSD-derived **init** programs only have a multi-user mode and a single-user mode, while System V-derived **init** programs usually have a variety of different predefined runlevels which can be selected.

System V-style **init** programs offer administrators more flexibility than BSD-style **init** programs, since they make it possible to configure the system in a variety of different ways. These different runlevels can then be selected as necessary, depending upon circumstances. Because of this flexibility, almost all Unix systems have moved away from the BSD-style **init** and use a variant of System V-style **init**.

Upstart builds upon the features of the System V-style initialization scheme. It adds support for responding to events. Events may be triggered when services are started or stopped, or by other processes on the running system.

Upstart has no native notion of runlevels. The event-driven approach that has been taken to managing services and jobs makes the SysV runlevel approach obsolete. However, since a lot of work will need to take place before all software will be converted to this new mode of operation, Upstart has created SysV runlevel compatibility which is used by default.

Many Linux distributions have signaled their intent to move to systemd as RHEL, SLES, Debian and Ubuntu already have.

Systemd Targets replace SysV runlevels

Systemd uses targets to group units together. It also maintains backwards compatibility using target aliases. Unlike runlevels multiple targets can be active at the same time.

systemd System and Service Manager

Provides strict control of daemons and speedy booting

Natively uses "unit" files

- Compatible with SysV init scripts

Uses socket and D-Bus activation for starting daemons

- Aggressive parallelization with dependency support

Offers on-demand starting of daemons and daemon monitoring

- Captures all STDOUT and STDERR from daemons
- uses Linux cgroups to track daemon processes
- controls all environmental runtime properties for daemons

Maintains mount and automount points

systemctl - Administration command

Used in RHEL7, SLES12, Debian 8, and Ubuntu since 15.04

systemd Features

Linux systems historically used the SysV init bootup system, or more recently the Upstart system. systemd replaces these earlier systems and offers the following features and benefits:

- Compatibility with existing SysV init scripts.
- Fast booting with aggressive parallelization (avoids most strict ordering of daemon startup by using socket and D-Bus activation methods)
- Keeps track of processes using the cgroup feature of the Linux kernel (not by their PIDs).
- Records everything a daemon sends to STDOUT, STDERR, or via `syslog()`.
- Standardizes many aspects of the bootup process and managing services making it easier for system administrators who manage heterogeneous environments.

Unit Files

Although systemd has compatibility with SysV init scripts placed in the `/etc/init.d/` directory, its native equivalents are called "unit" files. SysV init scripts are a mix of configuration settings (such as command line arguments either hardcoded into the script, or read from `/etc/sysconfig/*`) and shell code. In contrast, unit files are simple, declarative descriptions that are usually less than 10 lines, and do not include any code.

To modify a unit file, copy it from `/lib/systemd/system/` to `/etc/systemd/system/` keeping the filename the same and edit it

there. The files in `/etc/systemd/system/` override the `/lib/systemd/system/` files and will not be touched by the package management system. An example unit file follows:

File: `/lib/systemd/system/dbus.service`

[Unit]

Description=D-Bus System Message Bus

Documentation=man:dbus-daemon(1)

Requires=dbus.socket

[Service]

ExecStart=/usr/bin/dbus-daemon --system --address=systemd:
... ommitted ...

Documentation for unit files can be found in `systemd.unit(5)`, `systemd.exec(5)`, and `systemd.service(5)`.

systemd Targets

Unit files ending in `.target` are used for grouping units together and also define certain synchronization points used during boot up. In effect, they are a more flexible replacement for the SysV init runlevel concept. Unlike runlevels, more than one target may be active at the same time.

Socket Activation

In the traditional SysV init bootup sequence, the order daemons are started is carefully defined so that dependencies are satisfied. For

example, many daemons send syslog messages (via the socket file `/dev/log`) when they start so the syslog daemon must be started before anything attempts to use it. The design systemd uses is that it creates all the sockets for all daemons in one step, and then starts all the daemons. If one daemon requires another, it will connect to the pre-created socket and send the request which will then be queued by the Linux kernel until the other daemon is ready to dequeue all the messages and process them. The result is that daemons, even those with interdependencies, can be started in parallel.

Another benefit is that a daemon can be configured to be auto-spawned by **systemd** when a request is made to its socket file. Finally, because the socket files are not created by the daemon itself, a daemon can exit or crash, then be restarted without affecting client processes or dropping any request.

Filesystem Management

On a box using systemd, the `/etc/fstab` will only contain entries for disk based filesystems. All the kernel based filesystems are handled by `.mount` unit files.

```
$ ls -1 /lib/systemd/system/*.mount
/lib/systemd/system/dev-hugepages.mount
/lib/systemd/system/dev-mqueue.mount
/lib/systemd/system/media.mount
/lib/systemd/system/proc-fs-nfsd.mount
/lib/systemd/system/proc-sys-fs-binfmt_misc.mount
/lib/systemd/system/sys-fs-fuse-connections.mount
/lib/systemd/system/sys-kernel-config.mount
/lib/systemd/system/sys-kernel-debug.mount
/lib/systemd/system/var-lib-nfs-rpc_pipefs.mount
```

Normally, disk based filesystems listed in the `/etc/fstab` are checked and mounted before they are used. systemd offers the alternative that filesystems can be configured to be checked and mounted when they are first accessed. This can speed boot, especially for filesystems not needed during boot such as `/home`. This is done by editing the `/etc/fstab` and adding the `comment=systemd.automount` mount option to the filesystem.

Tracking Processes Belonging to a Daemon

When stopping or restarting a daemon cleanly, it is important that all the running processes associated with a daemon are stopped. It is

also useful for the Systems Administrator to be able to identify which processes belong to each daemon (especially given the proliferation of processes on modern Linux systems). In certain circumstances such as **cron** or web server CGI processes it can be complex to track back the inheritance. Linux has always supported the Unix feature of process groups and tracking of the parent PID, however these decades old methods have limitations and allow a process to "escape" supervision.

The Linux kernel 2.6.24 added a new feature called "control groups" (cgroups) that was primarily created by engineers at Google. The use of cgroups allows for labeling, control, isolation, accounting, prioritization, and resource limiting of groups of processes. By default, systemd only uses cgroups for labeling and tracking which processes belong to which daemon. Every child process inherits the cgroup of its parent and can't escape that cgroup unless it is privileged. This way systemd can kill a daemon and all processes it created with certainty. View the **systemd** created cgroups with **systemd-cgls** either in their entirety or for an individual service:

```
$ systemd-cgls
. . . output omitted . . .
$ systemd-cgls systemd:/system.slice/dbus.service
systemd:/system.slice/dbus.service:
- 627 /bin/dbus-daemon --nofork --systemd-activation
- 656 /usr/libexec/polkit-1/polkitd --no-debug
- 708 /usr/sbin/modem-manager
```

With systemd, when users login to interactive sessions, all processes for that user are placed into a cgroup unique to that login session. This way all processes for users are tracked and **systemd** can be configured to terminate all user processes on logout so that straggler processes don't hang around. This is done with the following edit:

File: /etc/systemd/login.conf	
[Login]	+ KillUserProcesses=yes

Modifying systemd services

Method 1 - Fully replace the default unit file.

- `systemctl edit --full unit_name.service`
- Copies `/lib/systemd/system/unit_name.service` to `/etc/systemd/system/`

Method 2 - Create a `unit_name.service.d/` directory with supplemental configurations.

- `systemctl edit unit_name.service`

Method 3 - Create a new unit file referencing the default unit

After changing a `unit_name.service` file run `systemctl daemon-reload`

- Not needed when using `systemctl edit`

Persistent changes to unit files

The systemd unit files which exist under `/lib/systemd/system/` should never directly be edited, since updates applied to the system from the package manager will cause any changes made to the `/lib/systemd/` unit files to be lost. However, there are multiple ways to modify the default systemd unit files where the changes will remain persistent after updates.

Fully replacing a default unit file

To fully replace the existing unit file under the `/lib/systemd/system/` directory use the command `systemctl edit --full unit_name.service` which will copy the unit file from `/lib/systemd/system/` to `/etc/systemd/system/` keeping the filename the same and opening a text editor. Once the text editor is exited, the command `systemctl daemon-reload` will automatically be performed.

Creating a `unit_name.service.d` directory

Another option is to create a directory named `/etc/systemd/system/unit_name.service.d/`, within this directory multiple supplementary .conf configuration files can be placed. The files will be processed in lexicographical order. Using the command `systemctl edit unit_name.service` will automatically create this directory and open a text editor, the file created by the text editor will be saved as `override.conf`, after the text editor is exited the command `systemctl daemon-reload` will automatically be performed.

Referencing an existing unit file in a new unit file

Using the `.include` directive can include external unit files. A common method to modify existing units is to include the default unit file then override specific settings. For example, modify the crond service to start with a higher nice value so it doesn't interfere with other important system processes:

```
File: /etc/systemd/system/crond.service
.include /lib/systemd/system/crond.service

[Service]
Nice=5
```

As with any change to a unit file after creating or editing the above file, the command `systemctl daemon-reload` will need to be executed and for the changes to take effect the service would need to be restarted with `systemctl restart unit_name.service`.

If the environmental variables `$VISUAL`, `$EDITOR` and `$SYSTEMD_EDITOR` are left unset, when executing `systemctl edit` the text editor `nano` will be started.

Gathering Service Information

Run the command `systemctl status unit_name.service` to display the service state, cgroup heirarchy, and a few log lines. Run the command `systemctl cat unit_name.service` to output the service unit file as it is currently known to the running systemd process.

Systemd Service Sandboxing Features

Common Directives

- PrivateTmp
- PrivateNetwork
- ProtectSystem
- ProtectHome
- CapabilityBoundingSet

Reference Man Pages

- `systemd.exec(5)`
- `systemd.unit(5)`
- `systemd.service(5)`
- `systemd.directives(5)`
- `capabilities(7)`

Systemd Service Sandboxing Features

Within a unit file it is possible to specify restrictions on the execution environment of any processes started either directly or indirectly by a systemd unit file. Common sandbox directives include:

PrivateTmp ⇒ Runs a service with private /tmp and /var/tmp directories.

PrivateNetwork ⇒ Runs a service in a private network namespace with only a loopback interface.

ProtectSystem ⇒ Mounts /boot, /etc, and /usr read-only.

ProtectHome ⇒ Sets /root and /home read-only or inaccessible.

ReadOnlyDirectories ⇒ Mount a directory as read-only for the service

InaccessibleDirectories ⇒ Prevents the service from accessing specified directories.

The man page `systemd.directives(7)` has a full list of possible directives and references other man pages which contain descriptions of the directives.

If a service retains the Capability CAP_SYS_ADMIN it could unsandbox itself by remounting the filesystem device. To prevent this set the directive `CapabilityBoundingSet=~CAP_SYS_ADMIN` which will strip the CAP_SYS_ADMIN from any process started by systemd. Using Capability Bounding will often cause services which require root access to the system to fail in new and exciting ways. This can be witnessed in the example on this page, the root user is unable to mount filesystems when using ssh.

The following example demonstrates the use of the `PrivateTmp` directive being applied to the `sshd` service. Any files inside of the global /tmp namespace are not visible to an ssh client, any /tmp files an ssh client creates are not visible to the host system. Any files created in /tmp by the `sshd` service will be deleted when the service or system is restarted.

```
# systemctl edit --full sshd
```

```
File: /etc/systemd/system/sshd.service
```

```
Restart=on-failure
RestartSec=42s
+ CapabilityBoundingSet=~CAP_SYS_ADMIN
+ PrivateTmp=yes
```

```
# systemctl restart sshd
```

```
# ls /tmp
```

```
. . . output omitted . . .
```

```
# ssh localhost
```

```
root@localhost's password: makeitso 
```

```
# ls /tmp
```

```
# touch /tmp/TESTFILE
```

```
# ls /tmp
```

```
TESTFILE
```

```
# mount /dev/mapper/vg0-tmp /tmp
```

```
mount: permission denied
```

```
# exit
```

```
# ls /tmp
```

systemd Targets

Targets allow grouping of units

- More flexible replacement for the SysV runlevel concept
- Defined in unit files ending in .target
- Use a *unit_name.wants/* directory to track grouped units

Runlevel targets provided for backwards compatibility

Special hardcoded targets provide core systemd functionality

Changing targets (runlevels)

- `systemctl isolate graphical.target`

Viewing & Changing the default boot target

- `systemctl get-default`
- `systemctl set-default desired.target`

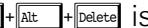
Defining Units Wanted by a Target

Any unit, including a .target unit, can explicitly specify dependencies with a `Wanted=` line within the unit file. In order to allow for easy modification of explicit dependencies without modifying the unit file, each unit can instead have a *unit_name.wants/* directory that can contain symlinks to the desired unit files. This is analogous to the SXX symlinks with the SysV init rcX.d/ directories. Since systemd has the concept of shipped configuration versus custom configuration, for each unit with a *unit_name.wants/* directory, there are two directories where the symlinks may reside. As an example, with the multi-user.target, the following two directories are consulted:
`/lib/systemd/system/multi-user.target.wants/`, and
`/etc/systemd/system/multi-user.target.wants/`

Special Targets

There are approximately 50 special targets that are internal to **systemd**, which are documented in the `systemd.special(7)` man page. Some examples include:

network-online.target ⇒ Units that absolutely require networking up and running should depend on `network-online.target` via a `Wants=network-online.target` line

ctrl-alt-del.target ⇒ Started by **systemd** when  is pressed on the console. Typically aliased to `reboot.target`

rescue.target ⇒ Alternate boot target that runs bare minimum services and a rescue shell like the legacy runlevel 1

sigpwr.target ⇒ Started when **systemd** receives the SIGPWR signal,

which is usually sent by the UPS monitoring daemons when power fails

default.target ⇒ What **systemd** starts at boot by default. Usually aliased (symlinked) to `multi-user.target` or `graphical.target`

Changing Targets

The equivalent of changing runlevels it to isolate targets. This starts all a target's dependencies and stops all others. This is done with the `systemctl isolate` command. For example:

```
# systemctl isolate graphical.target
```

This only works on target units that have the `AllowIsolate=yes` setting configured.

Default Boot Target

The symlink `/etc/systemd/system/default.target` controls what target to boot to. The `/etc/inittab` is not consulted.

To boot to a text mode login, run this command:

```
# systemctl set-default multi-user.target
. . . output omitted . . .
```

To boot to a graphical mode login, run this command:

```
# systemctl set-default graphical.target
. . . output omitted . . .
```

Using systemd

Starting and Stopping a service:

- `systemctl start unit_name.service`
- `systemctl stop unit_name.service`
- `systemctl restart unit_name.service`

Enabling and Disabling a service:

- `systemctl enable unit_name.service`
- `systemctl disable unit_name.service`
- `systemctl mask unit_name.service`

Listing services and their state:

- `systemctl status unit_name.service`
- `systemctl -a --type=service`
- `systemctl list-unit-files --type=service`

Enabling a Service

When enabling a service, a symlink is created in the `service_name.wants/` directory of the default target. When using `systemctl` to do this, it helpfully prints to the screen what it is doing. For example, the following enables the `autofs.service`:

```
# systemctl enable autofs
ln -s '/usr/lib/systemd/system/autofs.service' '/etc/systemd/system/multi-user.target.wants/autofs.service'
```

Disabling a Service

To disable a service you can either use `disable` or `mask`. The difference is with `mask`, it isn't possible to start the service manually. For example:

```
# systemctl disable autofs
rm '/etc/systemd/system/multi-user.target.wants/autofs.service'
# systemctl mask autofs
ln -s '/dev/null' '/etc/systemd/system/autofs.service'
# systemctl start autofs
Failed to issue method call: Unit autofs.service is masked.
```

To undo a `mask` operation, use `unmask`:

```
# systemctl unmask autofs
rm '/etc/systemd/system/autofs.service'
```

Checking on a Service

The **status** option is quite powerful. It shows the current status, a list of all the processes, and the last 10 lines of logged output from the daemon. For example:

```
# systemctl status crond
crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled)
     Active: active (running) since Thu, 05 Jul 2012 13:33:53 -0600; 4h 16min ago
       Main PID: 647 (crond)
          CGroup: name=systemd:/system/crond.service
                    \ 647 /usr/sbin/crond -n
```

Jul 05 13:33:53 mentor.gurulabs.com /usr/sbin/crond[647]: (CRON) INFO (running with inotify support)

Obtain a list of services that failed at boot:

```
# systemctl --failed
```

Obtain the status of a service on a remote server:

```
# systemctl -H root@server1.example.com status autofs
```

Listing all Services

To see a list of all possible services and what their state is, use the following command:

```
# systemctl list-unit-files --type=service
UNIT FILE                                     STATE
abrt-ccpp.service                            enabled
abrt-oops.service                            enabled
abrt-vmcore.service                           enabled
abrtd.service                                enabled
accounts-daemon.service                      disabled
acpid.service                                enabled
arp-ethers.service                           enabled
iscsi.service                                 masked
. . . snip . . .
```

Linux Runlevels Aliases

Maintenance runlevels: runlevel 1, s (or single)

- Booting into maintenance mode: Use runlevel s (or 1)
- Switching running system into maintenance mode: Use only 1

Historically the default runlevel specified in /etc/inittab

- Commonly, the default runlevel is either:
 - 3 – Multi-user with networking
 - 5 – Same as 3, but adds a GUI

Systemd respects these runlevel numbers via aliases

Standard Linux Runlevels

Most Linux distributions ship with several predefined runlevels, typically:

Runlevel	Description
0	halt
1	Single-user mode with runlevel preparation
2	Full multi-user, without networking
3	Full multi-user mode with text mode login
4	Custom or local; unused by default
5	Full multi-user mode with graphical X11 login
6	reboot
s,S,single	init's single-user mode

While the concept and use of runlevels is common to most Unix systems, the exact number of runlevels available, and the use of each, differs. Do not assume that a specific runlevel number maps to the same function on the various Unix systems. A good example is Solaris where runlevel 5 is functionally equivalent to Linux's run level 0 and will shut down the machine.

Systemd support for the runlevel concept

In systemd, targets are used as replacements for runlevels. The following table shows the runlevel to target mapping:

Runlevel	Target
0	poweroff.target
1	rescue.target
2	multi-user.target
3	multi-user.target
4	multi-user.target
5	graphical.target
6	reboot.target

For each named target in the table, there is a corresponding runlevelX.target alias. The alias symlinks can be seen with this command:

```
# ls -al /usr/lib/systemd/system/runlevel?.target
```

The **init** or **telinit** commands can also be used with the runlevel aliases. The **who -r** and **runlevel** command will display the corresponding runlevel alias.

Legacy Support for SysV init

Scripts located in /etc/init.d/

- LSB function library /lib/lsb/init-functions

Starting and stopping services with the SysV Init scripts

```
# service daemon start  
# service daemon stop
```

Controlling automatic start

- Manual Configuration
 - Change symbolic links in rc#.d/ directories
 - Change disable = line in /etc/xinetd.d/service files
- R7/S12: Easy configuration using **chkconfig**
 - list
 - chkconfig service on|off

LSB Standardized Init Script

The Linux Standards Base Core Specification defines required init script actions, including those provided by the standard functions library **/lib/lsb/init-functions**, and comment conventions that define how an init script is installed related to the runlevels it runs in. The follows shows an example of LSB required comment block keywords and arguments:

File: /etc/init.d/foo

```
### BEGIN INIT INFO  
# Provides: foo  
# Required-Start: $local_fs $network $remote_fs $syslog  
# Required-Stop: $local_fs $network $remote_fs $syslog  
# Default-Start: 2 3 4 5  
# Default-Stop: 0 1 6  
# Short-Description: start and stop foo  
# Description: foo is baz when it's not foobar.  
### END INIT INFO
```

Parameter	Description
restart	Performs a stop followed by a start.
reload	Causes daemon to re-read its configuration.
condrestart	Restarts the daemon only if already running.
status	Displays daemon PID and execution status.

The service Program

The parameter passed to the **service** command is the name of the desired script in the /etc/init.d/ directory:

```
# service sendmail restart
```

Shutting down sendmail:

[OK]

Starting sendmail:

[OK]

The chkconfig Command

The **chkconfig** command can be used to view what is currently configured to start on boot as well as to enable or disable a specific service. It manages both SysV init scripts as well as **xinetd** services. For example:

```
# chkconfig --list service
```

```
# chkconfig postfix on
```

```
# chkconfig finger off
```

Using SysV Init Scripts to Control Services

SysV Init scripts are located in the /etc/init.d/ directory. All SysV Init scripts take the arguments start and stop. Most Linux distributions' SysV Init scripts support additional arguments:

Lab 2

Estimated Time:

S12: 30 minutes

R7: 30 minutes

Task 1: Managing Services With Systemd's systemctl

Page: 2-13 Time: 15 minutes

Requirements:  (1 station)

Task 2: Creating a systemd unit file

Page: 2-19 Time: 15 minutes

Requirements:  (1 station)

Objectives

- ❖ Use systemctl to view the status of the system.
- ❖ Use systemctl to list and modify service states.
- ❖ Observe how systemd socket activation works

Requirements

- ▀ (1 station)

Relevance

Knowing how to manage services is critical to maintaining a stable and secure system.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Use systemctl and see if any units have failed to activate during and since the last boot:

```
# systemctl --failed  
UNIT          LOAD   ACTIVE SUB   DESCRIPTION  
rngd.service loaded failed failed Hardware RNG Entropy Gatherer Daemon
```

• The output you see may vary.

LOAD = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB = The low-level unit activation state, values depend on unit type.

1 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.

- 3) If you had any failed units, use systemctl to examine the status of the unit. If not, **skip** to the next step.

```
# systemctl status rngd.service  
. . . output omitted . . .
```

If the output is truncated, run the command again with **-l** appended to the end.

Lab 2

Task 1

Managing Services With Systemd's systemctl

Estimated Time: 15 minutes

- 4) Use systemctl to view a list of all the enabled services:

```
# systemctl list-unit-files --type service --state enabled  
. . . output omitted . . .
```

- 5) Use systemctl to view a list of all the disabled services:

```
# systemctl list-unit-files --type service --state disabled  
. . . output omitted . . .
```

- 6) Use systemctl to list all the known active service units with a short description of each one:

```
# systemctl list-units --type service  
. . . output omitted . . .
```

- 7) Use systemctl to get a list of all targets and their state:

```
# systemctl list-units --type target  
. . . output omitted . . .
```

Remember, unlike SysV init runlevels, multiple targets can be activated at the same time.

- 8) Use systemctl to get of all the dependencies of the graphical.target:

```
# systemctl list-dependencies graphical.target  
graphical.target  
. . . output omitted . . .
```

Remember, unlike SysV init runlevels, multiple targets can be activated at the same time.

- 9) Use systemctl to examine the status of the OpenSSH server:

```
# systemctl status sshd -l  
sshd.service - OpenSSH Daemon  
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)  
     Active: active (running) since Thu 2014-12-04 08:45:07 MST; 5h 55min ago
```

```
Main PID: 1494 (sshd)
CGroup: /system.slice/sshd.service
`-1494 /usr/sbin/sshd -D

Dec 04 08:45:07 station2 sshd-gen-keys-start[1481]: Checking for missing server keys in /etc/ssh
Dec 04 08:45:08 station2 sshd[1494]: Server listening on 0.0.0.0 port 22.
Dec 04 08:45:08 station2 sshd[1494]: Server listening on :: port 22.
Dec 04 10:51:49 station2 sshd[1940]: Accepted keyboard-interactive/pam for root from 10.100.0.254 port 41195 ssh2
Dec 04 10:51:49 station2 sshd[1940]: pam_unix(sshd:session): session opened for user root by (uid=0)
```

Notice how all the standard output from the sshd command is viewable.

- 10) Use systemctl to programmatically check to see if a service is enabled or disabled:

```
# systemctl is-enabled sshd
enabled
# echo $?
0
# systemctl is-enabled iscsid
disabled
# echo $?
1
```

- 11) [S12] This step should only be performed on SLES12.

Install tftp, and view the files contained within the package:

```
# zypper install -y tftp
. . . output omitted . . .
# rpm -ql tftp
```

Notice how the package contains both an Xinetd service file as well as systemd unit files. The daemon should only be enabled via one of the methods.

- 12) [R7] This step should only be performed on RHEL7.

Install the tftp client and server, and view the files contained within the server package:

```
# yum install -y tftp tftp-server
```

```
. . . output omitted . . .
# rpm -ql tftp-server
```

Notice how the package contains both an Xinetd service file as well as systemd unit files. The daemon should only be enabled via one of the methods. The systemd method uses tftp.socket activation to automatically start the tftp.service.

- 13) View the service's Xinetd status with chkconfig, and the Xinetd service file:

```
# cat /etc/xinetd.d/tftp
. . . output omitted . . .
# chkconfig --list tftp
. . . output omitted . . .
```

Observe that the tftp is disabled via the Xinetd method.

- 14) When running the TFTP server under systemd, it uses socket activation. Once the tftp.socket unit is enabled, the tftp.service will be started automatically when a connection is made to the tftp socket. View the current status of the tftp.socket unit and the tftp.service.

```
# systemctl status tftp.socket
tftp.socket - Tftp Server Activation Socket
  Loaded: loaded (/usr/lib/systemd/system/tftp.socket; disabled)
  Active: inactive (dead)
    Listen: [::]:69 (Datagram)

# systemctl status tftp.service
```

- The .service suffix could be left off, since it is assumed by default.

- 15) Enable and the view the status of the tftp.socket unit:

```
# systemctl enable tftp.socket
# systemctl status tftp.socket
```

- 16) See if anything is listening UDP port 69, then start the tftp.socket and check the port again:

```
# lsof -i :69
# systemctl start tftp.socket
# lsof -i :69
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
systemd 1 root 22u IPv6 32053      0t0    UDP *:tftp
```

- 17) Confirm that the TFTP server is not yet running:

```
# systemctl is-active tftp
inactive
# systemctl status tftp
. . . output omitted . . .
```

- 18) Run the tftp client utility and connect to the server and attempt to download a file called foo (which doesn't exist). Observe that the tftp.socket automatically starts the service.

```
# tftp localhost -c get foo
Error code 1: File not found
# systemctl is-active tftp
active
# systemctl status tftp
tftp.service - Tftp Server
   Loaded: loaded (/usr/lib/systemd/system/tftp.service; static)
     Active: active (running) since Thu 2014-12-04 15:09:16 MST; 4min 59s ago
       Main PID: 3743 (in.tftpd)
      . . . snip . . .
```

- 19) Manually stop the tftp.service, pay attention to the warning:

```
# systemctl stop tftp
Warning: Stopping tftp.service, but it can still be activated by:
          tftp.socket
```

20)

Cleanup

Stop and disable the tftp.socket:

```
# systemctl disable --now tftp.socket
```

Objectives

- ❖ Configure and test a systemd unit file.
- ❖ Create an environment configuration file for the systemd unit.

Requirements

- ▀ (1 station)

Relevance

Systemd unit files are used to control services during and after boot time. Often it is necessary to be able to control services which do not have a native unit files.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Install the socat and netcat packages:

```
[R7] # yum install -y socat nmap-ncat  
[S12] # zypper install -y socat netcat-openbsd  
. . . output omitted . . .
```

- 3) Create a shell script wrapper to start socat listening on a network port:

```
File: /usr/local/sbin/daytimed  
+#!/bin/bash  
+PORT="$1"  
+/usr/bin/socat -U -4 -6 TCP-LISTEN:${PORT},fork EXEC:"/bin/date -R"
```

- 4) Make the newly created daytimed script executable:

```
# chmod +x /usr/local/sbin/daytimed
```

- 5) Create an Environment File for systemd which will contain customizable settings for the socat/daytime server:

Lab 2

Task 2

Creating a systemd unit file

Estimated Time: 15 minutes

File: /etc/daytimed

```
+ # Additional Parameters to pass to the daytime unit file.  
+ PORT="13"
```

- 6) Create the systemd unit file:

File: /etc/systemd/system/daytime.service

```
+ [Unit]  
+ Description=A simple daytime server  
+ After=network.target  
  
+ [Service]  
+ EnvironmentFile=/etc/daytimed  
+ ExecStart=/usr/local/sbin/daytimed ${PORT}  
  
+ [Install]  
+ WantedBy=multi-user.target
```

Notice that custom user created unit files are created under /etc/systemd/system/, while unit files installed by a package manager are placed under /usr/lib/systemd/system/.

- 7) Use systemctl to have the newly created daytimed service start automatically at boot, and manually start the daytimed now:

```
# systemctl enable --now daytime
```

- 8) Ensure that the daytimed service has started:

```
# systemctl status daytime  
daytime.service - A simple daytime server  
   Loaded: loaded (/etc/systemd/system/daytime.service; enabled)  
     Active: active (running) since Thu 2015-08-20 13:55:34 MDT; 2min ago  
       Main PID: 28287 (daytimed)  
          CGroup: /system.slice/daytime.service  
                  |-28287 /bin/bash /usr/local/sbin/daytimed 13  
                  |-28288 /usr/bin/socat -U -4 -6 TCP-LISTEN:13,fork EXEC:/bin/date -R
```

- If any errors are reported, check for typos in all previously created files before proceeding.

Aug 20 13:55:34 stationX.example.com systemd[1]: Starting A simple daytime server...

Aug 20 13:55:34 stationX.example.com systemd[1]: Started A simple daytime server.

- 9) Confirm socat is listening for incoming connections on TCP port 13:

```
# ss -pant 'sport = :13'  
State      Recv-Q Send-Q  Local Address:Port      Peer Address:Port  
LISTEN      0        5          :::13                  ::::*      users:(("socat",28288,3))
```

- 10) Try and connect to the daytimed service, if the connection is successful the current date and time should be printed:

```
[R7] # ncat --recv-only localhost 13  
[S12] # nc localhost 13  
Thu, 20 Aug 2015 14:14:47 -0600
```

- 11) Systemd has a command to output the service unit file as it is currently known to the running systemd process:

```
# systemctl cat daytime  
[Unit]  
Description=A simple daytime server  
After=network.target  
  
[Service]  
EnvironmentFile=/etc/daytimed  
ExecStart=/usr/local/sbin/daytimed ${PORT}  
  
[Install]  
WantedBy=multi-user.target
```

Cleanup

- 12) Once you are done testing your new systemd service, stop and disable it so that it won't interfere with future labs:

```
# systemctl disable --now daytime
```

- 13) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```


Content

Booting Linux on PCs	2
GRUB 2	4
GRUB 2 Configuration	6
GRUB 2 Security	8
Boot Parameters	10
Initial RAM Filesystem	12
init	15
Systemd local-fs.target and sysinit.target	16
Systemd basic.target and multi-user.target	18
Legacy local bootup script support	20
System Configuration Files	21
RHEL7 Configuration Utilities	22
SLES12 Configuration Utilities	23
Shutdown and Reboot	24
Lab Tasks	
1. Boot Process	26
2. Booting directly to a bash shell	30
3. GRUB Command Line	33
4. Basic GRUB Security	36
5. Troubleshooting Practice: Boot Process	39

Chapter

3

GRUB2/SYSTEMD BOOT PROCESS

Booting Linux on PCs

Booting is a critical and complex sequence

- Linux can have very infrequent reboots (long uptimes)
- Little opportunity for SysAdmin to become familiarized
- Familiarity required for troubleshooting bootup errors

Main Actors

- System BIOS or UEFI
- Sector 0 of boot device (BIOS) or UEFI boot manager shim
- GRand Unified Bootloader (GRUB)
- Initial ramdisk
- Linux kernel
- **/sbin/init** launches bootup scripts from /etc/

System Boot Procedure

Understanding the exact sequence of events that occur during a boot of Linux greatly improves troubleshooting skills, enabling quick resolution of boot-time problems.

The Extensible Firmware Interface (EFI, originally the Intel Boot Initiative) provides a replacement for the traditional BIOS, initially supporting Itanium. HP worked on updating the LILO boot loader in 2003 with EFI support as a separate project: ELILO. On newer Linux systems, GRUB 2 is used instead of ELILO. Unified EFI (UEFI) was created as an industry-wide specification to replace the Intel-only EFI specification. The GUID Partition Table (GPT) format is included in the UEFI specification and, unlike MBR, supports disks/LUNs larger than 2TB and a straightforward approach to partition types.

Linux supports the traditional BIOS/MBR mode of booting as well as BIOS/GPT and UEFI/GPT. The new UEFI secure boot feature has been used on PCs since Windows 8. Though many Intel64 systems allow for disabling secure boot, commercial Linux distributions provide options for booting Linux with UEFI secure boot.

BIOS/MBR Boot

Following is the Linux boot process using BIOS/MBR:

1. System BIOS performs three tasks:
 - ❖ Power-On Self Test (POST)
 - ❖ Initial hardware setup and configuration
 - ❖ Loads option ROM from add-in cards (SCSI, SAN HBA, RAID)
 - ❖ Selects boot device and executes MBR from device
2. First stage GRUB (`boot.img`) in MBR (446 bytes); it loads:
 - ❖ stage1.5 GRUB (`core.img`) using int13 BIOS calls, which has embedded GRUB 2 modules needed to access /boot.
3. Second stage GRUB
 - ❖ Reads and uses configuration file or displays GRUB command prompt
 - ❖ Loads initial ram disk (usually specified)
 - ❖ Loads, decompresses, and executes selected Linux kernel from hard drive with command line arguments
4. Linux kernel
 - ❖ Initializes and configures hardware using drivers statically compiled into the kernel
 - ❖ Decompresses the initramfs image and mounts it
 - ❖ Runs `init` script from initramfs image
 - ❖ `init` script loads kernel modules and performs tasks necessary to mount the real root filesystem
 - ❖ Mounts the root partition passed to the kernel by the boot loader using the `root=` kernel command-line option (usually read only) as the root partition, replacing the `initrd`
 - ❖ Executes `/sbin/init`

UEFI/GPT Boot

Following is the Linux boot process using UEFI/GPT which differs from BIOS/MBR boot until the second stage GRUB is loaded:

1. UEFI firmware performs five tasks:
 - ❖ Power-On Self Test (POST)
 - ❖ Loads UEFI option ROM from add-in cards (SCSI, SAN HBA, RAID)
 - ❖ UEFI boot manager consults NVRAM variables for default UEFI boot entry
 - ❖ Firmware initializes the hardware required for booting
 - ❖ UEFI boot manager executes UEFI application from default boot entry
2. UEFI SecureBoot shim (`shim.efi`) on EFI System Partition (ESP); it loads:
 - ❖ stage1.5 GRUB (`grubx64.efi`) which has embedded GRUB 2 modules needed to access /boot.
 - ❖ Note that the shim is signed in case SecureBoot is enabled. It is used in either case.
3. Second stage GRUB and subsequent steps proceed as before.

System Boot Daemons

systemd is a replacement for the traditional `init` daemon. On a systemd system, `/sbin/init` is a symlink to `/lib/systemd/systemd`. It is an asynchronous, highly parallelized system that uses target dependencies for ordering. When it starts, it activates all dependencies of the `/etc/systemd/system/default.target`, which is a symlink for either `multi-user.target` or `graphical.target`. See `bootup(7)`.

[S12] *The following applies to SLES12 only:*

On SLES12, there are several other things that `init` takes care of:

- ❖ Runs `/etc/init.d/boot.d/` scripts (S symlinks to `/etc/init.d/boot.*`) on boot
- ❖ Runs `/etc/init.d/rc#.d/script start` for the relevant runlevel
- ❖ Runs `/etc/init.d/after.local` only at boot time
- ❖ Launches getty programs (e.g. `mingetty`)

UEFI Secure Boot

Secure Boot is a security standard to prevent "unauthorized" software, rootkits and low-level malware from being loaded during the boot process. It is usually enabled on modern desktop PCs, but not on servers. Examine the appropriate UEFI NVRAM variable to determine if SecureBoot is enabled. A value of 1 indicates it is enabled:

```
# od -An -t u1 /sys/firmware/efi/vars/SecureBoot-8be4df61-93ca-11d2-aa0d-00e098032b8c/data  
1
```

On a Linux system using Secure Boot, the following restrictions apply:

- ❖ Kernel modules must be signed
- ❖ kexec and thus kdump are disabled
- ❖ Debug interfaces such as SystemTap are disabled
- ❖ Access to `/dev/kmem` and `/dev/mem` is blocked, even for root
- ❖ Suspend to disk hibernation is disabled
- ❖ Low level hardware interfaces such as I/O ports and PCI BAR access is disabled

GRUB 2

GRUB rewrite with many new features

Provides a menu of all available kernels which can be booted

Can interactively find kernels to boot

- Filesystem support: ext{2,3,4}, XFS, Btrfs, JFS, ReiserFS, VFAT, and many others via modules
- Hardware support: Uses the BIOS or UEFI for I/O

Can also boot other OSes for multi-boot setups

Provides support for serial consoles

Can pass options at boot prompt

RHEL7/SLES12: use 2 in the GRUB command and file names

- e.g. `grub-install` vs `grub2-install`

GRUB 2 Features

GRUB 2 is a rewrite of the earlier GRUB bootloader (now referred to as GRUB Legacy). It has a far more modular design and introduces many new features such as:

- ❖ A more sophisticated config syntax that supports shell-script like variables, conditionals, and loops.
- ❖ Persistent storage of state across reboots (depending on exact configuration).
- ❖ The ability to read files from a greater number of filesystems such as Btrfs, HFS+, and NTFS.
- ❖ The ability to read files from LVM and RAID devices.
- ❖ The ability to find kernels using filesystem labels and UUIDs.
- ❖ A more sophisticated user based security model that includes both regular and privileged users.
- ❖ UEFI support (including GUID Partition Table) for booting from GPT.

16bit BIOS Limitations and the /boot filesystem

GRUB itself has no hardware drivers such as IDE, SATA, SCSI, or hardware RAID. In order to do I/O operations such as when loading the kernel or initramfs, it must make use of the BIOS int13 I/O functions or UEFI . Add-on controller cards such as SCSI, or hardware RAID automatically extend the BIOS/UEFI with drivers during bootup. Often a message such as "SCSI BIOS installed" is seen on the screen when this occurs.

Because GRUB uses the BIOS/UEFI for I/O, any files (such as GRUB stage 2, kernel, or initramfs) that are read must be readable by the BIOS. The original PC BIOS referenced drives by a series of numbers in the form of Cylinder, Head, Sector (CHS), and had a 1024 cylinder maximum. This meant that the BIOS could only read the first 512MB of the hard drive. The BIOS specification was updated to use 28bit Logical Block Addressing (LBA) to read the drive which provides a new limit of 137GB beyond which the BIOS cannot read. The latest 48bit LBA specification used in modern BIOSes and UEFI has a limit of 128PB. Even though 48bit LBA is now widely adopted, because of inertia, it is still common to have a /boot/ filesystem at the beginning of the drive.

[R7] *The following applies to RHEL7 only:*

The /boot filesystem used to store the files accessed by GRUB must be on its own partition and not stored within LVM or software RAID (other than RAID1 which need not be supported). This is because, although GRUB 2 now supports RAID, LVM, and Btrfs subvolumes, the supporting tools and Anaconda haven't yet been updated to support boot files on anything except for a separate partition.

Installing GRUB 2

Installation of the boot loader is normally done by the OS installer. In the event that you need to manually reinstall it (perhaps due to damage to its files, or you are switching from an earlier boot loader such as GRUB Legacy or LILO) use the `grub2-install` script. This script will:

- ❖ Copy the various GRUB modules and images from `/usr/lib/grub/arch/*` to `/boot/grub2/`.
- ❖ Call `grub2-mkimage` to create the bootable GRUB core.img file for this architecture and installs it into the correct location depending on if BIOS/MBR or UEFI/GPT is being used.

An example of installing to the MBR of the first disk would be:

```
# grub2-install /dev/sda
```

GRUB Disk and Partition Nomenclature

In the GRUB configuration file or command line, GRUB has a unique way of referencing disks and partitions. It starts numbering from 0 for disks. The first BIOS/UEFI detected hard drive is known as `hd0`, the second as `hd1`, etc. It doesn't matter if the drive is on a SAN, or locally attached via PATA, SATA, or SCSI. Partitions are described by a string indicating the disk label type (`msdos`, or `gpt`), and a number starting with 1 for the first partition. A partition and drive are referenced by GRUB using drive-comma-partition syntax:
`(hd1,msdos2)`. That is, the second partition on the second detected hard drive. The mapping between the BIOS detected hard drives, and the Linux device files, is kept in the `/boot/grub2/device.map` file.
The following table shows a comparison between GRUB and Linux kernel partition naming schemes:

Description	GRUB 2	GRUB Legacy	Linux Kernel
First primary partition on the first BIOS detected drive using MBR label.	<code>(hd0,msdos1)</code>	<code>(hd0,0)</code>	<code>/dev/sda1</code>
First logical partition on the first BIOS detected drive using MBR label.	<code>(hd0,msdos5)</code>	<code>(hd0,4)</code>	<code>/dev/sda5</code>
Second partition on the third drive using GPT label.	<code>(hd2,gpt2)</code>	Can not boot from GPT	<code>/dev/sdc2</code>

GRUB 2 Configuration

Configuration file:

- BIOS/MBR boot → /boot/grub2/grub.cfg
- UEFI/GPT boot → /boot/efi/EFI/distro/grub.cfg

grub.cfg is built by grub2-mkconfig that runs

- /etc/grub.d/* scripts which
- source /etc/default/grub

GRUB_DEFAULT — default menuentry to boot

- GRUB_DEFAULT=0 — by position
- GRUB_DEFAULT='Linux (3.17.2)' — by name
- GRUB_DEFAULT=saved' — by the saved_entry variable

GRUB 2 Configuration Overview

The /boot/grub2/grub.cfg configuration file is used by GRUB 2 and replaces the menu.lst or grub.conf file used by GRUB Legacy. This config file is created by the **grub2-mkconfig** script which sources the /etc/default/grub file for important local options and then runs every executable file in /etc/grub.d/* appending together their output to produce the config.

The arguments passed to the kernel can be adjusted by modifying the GRUB_CMDLINE_LINUX=kernel_args variable in the /etc/default/grub file. When a new config is generated, this list of arguments is used for every Linux kernel detected. To pass arguments only to the kernel of a specific menu entry create a complete menu entry via a custom script in /etc/grub.d/.

After editing /etc/default/grub, use **grub2-mkconfig** to create the actual grub.cfg:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.10.0-123.x86_64
Found initrd image: /boot/initramfs-3.10.0-123.x86_64.img
done
```

The /etc/grub.d/ scripts

The following is a definition of the most important /etc/grub.d/* scripts:

00_header ⇒ Processes /etc/default/grub

01_users ⇒ Red Hat Enterprise Linux user/password security

10_linux ⇒ Creates entries for installed Linux kernels

30_os-prober ⇒ Creates entries for Windows or other OSes found on other partitions

40_custom ⇒ Where user defined custom entries should be defined

41_custom ⇒ Includes /boot/grub2/custom.cfg if it exists

42_password ⇒ SUSE Linux Enterprise Server user/password security

The /etc/default/grub file

General configuration of GRUB 2 is done via variables in the /etc/default/grub file. The most commonly modified variable is GRUB_CMDLINE_LINUX, however there are other useful variables to be aware of:

GRUB_TIMEOUT ⇒ How many seconds before default entry is booted

GRUB_DEFAULT ⇒ Controls which menu entry to boot by default. Can be a menuentry or menuentry number, but is normally set to saved so that the saved_entry variable in the /boot/grub2/grubenv which is managed by **grub2-set-default** to define the default boot entry

GRUB_DISABLE_RECOVERY=true ⇒ Disable generation of recovery mode menu entries

GRUB_BADRAM ⇒ Use output from Memtest86 v2.3 or Memtest85+ v1.6 and higher to list bad RAM regions to avoid. Makes use of the standard memmap kernel parameter.

GRUB_CMDLINE_LINUX_RECOVERY ⇒ SUSE Linux Enterprise Server

specific setting for optional alternate recovery kernel command line

Booting Floppy Disc or ISO Images

Although most systems do not ship with floppy disc drives today, some vendors still distribute software updates such as BIOS or firmware updates via this method. GRUB is capable of booting disc images by using the **memdisk** helper program (usually located in the `/usr/share/syslinux` directory). After copying the **memdisk** program and disc image (optionally compressed with gzip) to the `/boot` directory, add a new stanza to the GRUB configuration file like the following:

File: `/etc/grub.d/40_custom`

```
menuentry "Boot floppy image" {
    linux16 /memdisk
    initrd16 /floppy_image_filename
}

menuentry "Boot iso image" {
    linux16 /memdisk iso
    initrd16 /iso_image_filename
}

menuentry "Boot raw floppy image" {
    linux16 /memdisk raw
    initrd16 /floppy_image_filename
}
```

The **root= boot parameter**

The **root= boot** parameter is used to define the location of the root filesystem whether it be on a partition, a software RAID volume, or an LVM logical volume. With GRUB 2 the **root=** doesn't need to be manually specified in the `GRUB_CMDLINE_LINUX` variable in `/etc/default/grub`. Instead it will automatically insert a `root=` using the UUID of the root filesystem and then the initramfs will be responsible for finding that filesystem.

If you wish to manually set the `root=` parameter in the `GRUB_CMDLINE_LINUX` variable then also add `GRUB_DISABLE_LINUX_UUID=true` to `/etc/default/grub`.

The `root=` parameter can accept a wide variety of syntaxes for specifying the location of the device containing the root filesystem. The following table show representative example:

Method	Parameter
Linux Device Node	<code>root=/dev/sdb1</code>
Physical ID	<code>root=/dev/disk/by-id/ata-VEN_SER-part2</code>
Filesystem Label	<code>root=LABEL=/</code>
Filesystem UUID	<code>root=UUID=e4857f8a-036d-4b5e-b4ed-9d16a</code>
LVM Logical Volume	<code>root=/dev/mapper/vg_server1-lv_root</code>
Software RAID Volume	<code>root=/dev/md1</code>

State stored in the GRUB Environment Block

GRUB 2 has a 1024 byte file, `/boot/grub2/grubenv`, which is used to save state across reboots. For example, when using `GRUB_DEFAULT=saved` with `GRUB_SAVEDEFAULT=true` the last menuentry selected becomes the default menuentry. Also if the `grub2-set-default` command is used to set the `saved_entry` variable, it will be stored in the file as well. To see what is stored in the file, use:

```
# grub2-editenv list
. . . output omitted . . .
```

To reset the file and clear out all stored state, run:

```
# grub2-editenv create
. . . output omitted . . .
```

The environment block file is not supported when using `/boot` on top of LVM or software RAID.

GRUB 2 Security

New user based security model

- superuser has full access
- One or more users can be defined

One or more users can be assigned to a menu entry

Uses Password-Based Key Derivation Function 2 (PBKDF2)

- **grub2-mkpassword-pbkdf2**
- Use key stretching to make brute force attacks more difficult
- Tunable strength with iteration count parameter -c

Defaults to 10,000 iterations

R7: Installer can define a GRUB 2 superuser

S12: Yast boot loader module can define a GRUB 2 superuser

GRUB 2 User Accounts

GRUB 2 introduces the idea of normal user accounts, and superusers. Each user will have access (via the associated password) to all menu entries that list that user. Superusers, if defined, have access to all entries, including the ability to edit menu entries and access the GRUB 2 command line. Menu entries marked --unrestricted can be booted (but not edited), by any user.

Passwords associated with users are stored within the grub.cfg and can be either clear text, or encoded using the PBKDF2 (Password-Based Key Derivation Function 2 -- as described in RFC 2898). The PBKDF2 based passwords are created using the **grub2-mkpassword-pbkdf2** command (replaces the **grub-md5-crypt** program used in GRUB Legacy):

```
# grub2-mkpassword-pbkdf2
Enter password: secret_passphrase 
Reenter password: secret_passphrase 
Your PBKDF2 is grub.pbkdf2.sha512.10000.→
A0A6350800292A00. . . snip . . .
```

This key is then used as part of the password_pbkdf2 statement within the GRUB configuration:

Securing Menu Entries

Anything more complex than having a superuser defined requires manual GRUB 2 configuration. Each menu entry can have one of the following security configurations:

- **--unrestricted** ⇒ No user/password needed to boot entry. Only superuser can edit entry.
- **--users ""** ⇒ Only superuser can boot the entry. This is the default if --users is omitted.
- **--users user1,user2,...** ⇒ Only user(s) in the list plus the superuser can boot the entry.

[R7] *The following applies to RHEL7 only:*

On Red Hat Enterprise Linux, all menu entries automatically have the --unrestricted option set. This is done within the /etc/grub.d/10_linux file. With this configuration already in place, by simply adding a superuser definition, you can control access to the GRUB 2 command line and interactive editing of menu entries.

Example configuration

[R7] File: /etc/grub.d/01_users
[S12] File: /etc/grub.d/42_password
<pre>+ cat <<EOF + set superusers="root" + password_pbkdf2 root grub.pbkdf2.sha512.10000.- A0A6350800292A00. . . snip . . + password jada a_plain_text_password + password bailey a_plain_text_password + menuentry Maintenance --users jada,bailey { + set root='hd0,msdos1' + linux /vmlinuz-3.11.0 ro root=/dev/mapper/vg0-root- init=/bin/sh + initrd /initramfs-3.11.0.img + + } + menuentry Linux --unrestricted { + set root='hd0,msdos1' + linux /vmlinuz-3.11.0 ro root=/dev/mapper/vg0-root + initrd /initramfs-3.11.0.img + + } + EOF</pre>

The configuration shown would require the username of root with its associated password to access the command line. Either the jada or the bailey credentials are required to boot the maintenance mode. The --unrestricted menu entry can be booted without credentials.

Boot Parameters

Boot options can be passed either interactively at the GRUB prompt, or persistently by configuring them in the GRUB configuration file. The kernel will interpret and act on all parameters which it understands. Options passed to the kernel on boot are exported in the /proc/cmdline file. Any program can parse this file and take action based on parameters.

```
$ cat /proc/cmdline
ro root=/dev/mapper/vg0-lv_root ro fsck.mode=force
```

Parameters are intended to be acted on by four different software components. On a typical Linux machine the only parameter that is required is the **root=** parameter. The **ro** parameter causes the root filesystem to be initially mounted read-only, a good practice for reliable booting by allowing both clean and dirty filesystems to be mounted.

Kernel Parameters

A variety of different parameters can be specified to modify the behavior of the kernel. The full documentation is available in the `kernel-parameters.txt` which is part of the kernel source or the `kernel-command-line(7)` manpage. The kernel parameter **quiet** is used to suppress all non-critical kernel messages sent to the screen during boot up. The **log_buf_len** parameter controls the size of the kernel log buffer that is displayed with `dmesg`. By default the log buffer is often too small for machines with long uptimes. To increase the buffer to 4,096 KBytes (up from a typical 128KBytes or

Boot Parameters

Passed on kernel command line

- Persistently defined in GRUB configuration file

Viewable after boot via /proc/cmdline

Four types of parameters

- kernel parameters - (e.g. `console=ttyUSB0`)
- initramfs dracut parameters - (e.g. `rd.debug`)
- **systemd** parameters - (e.g. `systemd.unit=multi-user.target`)
- Systemd unit parameters - (e.g. `fsck.mode=force`)

Kernel parameter documentation: `bootparam(7)`

Dracut documentation: (e.g. `dracut.cmdline(7)`)

Systemd boot parameters documentation: (e.g. `kernel-command-line(7)`)

512KBbytes) use **log_buf_len=22**.

The vast majority of the available kernel parameters are related to modifying the behavior of device drivers. Since most device drivers are compiled as modules, they are configured via **modprobe**.

The crashkernel Kernel Parameter

In order to save a copy of the memory for debugging purposes when the kernel crashes, the **crashkernel=128M@16M** parameter can be used to specify how much memory to reserve for the fail-safe kernel (128MB) and at what memory location (16MB).

Alternatively, in new kernels the **crashkernel=auto** parameter will reserve 1/32 of physical memory if the system has 4GB or more of RAM. For example, a system with 12GB of memory would have 384MB of memory reserved.

Initial Ramdisk Parameters

When booting the system, the **init** script inside of the initial ramdisk can make use of boot parameters. On systemd systems with dracut, **init** is a symlink to systemd. The processing of the **root=** values are done by the script. Since the main role of the initial ramdisk is to access and mount the real root filesystem, many options pertain to that procedure when the root filesystem is accessed via atypical methods, such as NFS.

Documentation for valid initial ramdisk parameters can be found in the `dracut.cmdline(7)` manpage. The **rd.debug** parameter causes the **init** script to print each line of the script on the screen as it is being executed. Examples of common ramdisk parameters include:

- rd.lvm.lv=VGname/LVname** ⇒ Activate LVM Volume Group and specified Logical Volume
- rd.luks=0** ⇒ Disable auto-detection of LUKS encrypted partitions
- rd.md=0** ⇒ Disable auto-detection of Linux software RAID volumes
- rd.dm=0** ⇒ Disable auto-detection of software RAID volumes that use hardware assisted boot redundancy
- rd.plymouth=0** ⇒ Defines key mapping, useful during initial ramdisk shell access
- vconsole.keymap** ⇒ Defines key mapping, useful during initial ramdisk shell access

systemd Parameters

Once the initial ram disk has finished making the root filesystem available, the `/sbin/init` command is run and processes parameters meant for it (if any). On a Linux box using systemd, `/sbin/init` is a symlink to `/lib/systemd/systemd`. Normally no parameters are specified for a typical boot, but the `/etc/systemd/system/default.target` symlink can be overridden with `systemd.unit=rescue.target` or `systemd.unit=emergency.target`. A root password is required at login when booting to either the rescue or emergency target.

For backwards compatibility, a single number representing a SysV init runlevel, such as 3 or 5 can be passed as well which will be mapped to the closest systemd target.

systemd.unit parameter	Description	Alternative parameter
<code>rescue.target</code>	Mounts all filesystems and starts important services, but doesn't activate network interfaces. Equivalent to single user mode.	1, s, S, and single
<code>emergency.target</code>	Only root filesystem is mounted, and read-only at that. Only most critical services are started.	-b or emergency

Systemd Unit Parameters

Some systemd unit files make use of kernel command line parameters. Some of the parameters are listed in the following table:

- fsck.mode=force** ⇒ Forces filesystem check on clean filesystems
- plymouth.enable=0** ⇒ Disable the plymouth bootsplash entirely
- systemd.journald.forward_to_console=1** ⇒ Sends journal daemon log messages to the console instead of to syslog. Useful for troubleshooting boot failures.

Initial RAM Filesystem

Distributions ship lightweight generic kernels

- Nearly all features and drivers compiled as modules

Mounting the root filesystem (/) during boot

- May require filesystem module
- May require hardware driver: HW RAID, SAN HBA, SCSI
- May require software configuration: LVM, Software RAID, Encrypted /, iSCSI, De-hibernation

Kernel can't hard code handling all possible boot variations

Solution: Initial RAM filesystem (aka early user space or initramfs)

- Loaded into memory by GRUB
- Does what's required to mount /

Dracut: `mkinitrd` and `mkinitramfs` replacement

Solving Boot Time Catch-22s

During bootup, the / filesystem must be mounted. However, many catch-22 scenarios can exist that prevent the / filesystem from being mounted. Consider the scenario where the XFS kernel module is file residing in the /lib/modules/ directory tree which is on a filesystem formatted with XFS. The system needs to be able to support XFS in order to load the XFS module. Or consider the scenario where the driver for a hardware RAID controller is inside of the hardware RAID volume. Technically these module/driver issues can be solved by compiling the modules into the kernel directly, however, this would mean specially compiled system specific kernels which is considered a non-optimal solution by Linux vendors.

Besides (or in addition to) the above scenarios, certain system configurations require commands to run in order to make the root filesystem available. For example, if the root filesystem is located on a LVM logical volume then LVM commands to scan for and activate the LVM volume group must be run during the bootup sequence.

Linux solves these boot problems by use of an initial RAM filesystem. The initial RAM filesystem is stored in a file on the /boot/ filesystem. The initial RAM filesystem is loaded into memory by GRUB with the help of the BIOS/UEFI that has been extended by option ROMs loaded from I/O controllers.

During boot, the Linux kernel executes /init in the initial RAM filesystem which is a symlink to a systemd binary that runs in initramfs mode. In older Linux distributions, /init was typically a

script. In either case, whatever is required to mount the real root filesystem is done.

Initial Ramdisk Implementation

Originally Linux used an actual filesystem, typically ext2 in a compressed file as the initial ramdisk, this is known as the `initrd` scheme. Modern Linux uses a compressed `cpio` archive to store the initial ramdisk contents, known as the `initramfs` scheme. The `initramfs` scheme is lighter weight since the kernel doesn't have to have any filesystem drivers compiled into it.

Because of the `cpio` format, it is easy to see or extract out the contents of the initramfs either using a combination of `/usr/lib/dracut/skipcpio` and `cpio` or the `lsinitrd` command.

```
# cd /boot  
# lsinitrd initramfs.img  
init  
... snip ...  
# mkdir /tmp/initramfs; cd /tmp/initramfs  
[R7] The following applies to RHEL7 only:  
# /usr/lib/dracut/skipcpio /boot/initfs.img | zcat | cpio -id  
[S12] The following applies to SLES12 only:  
# /usr/lib/dracut/skipcpio /boot/initfs.img | xzcat | cpio -id  
# ls -al
```

```
drwxr-xr-x 22 root root 4096 Feb 29 14:55 .
drwxrwxrwt  6 root root 4096 Feb 29 14:55 ..
drwxr-xr-x  2 root root 4096 Feb 29 14:55 bin
. . . snip . . .
```

Creating an Initial Ramdisk

Normally an initial ramdisk is created automatically whenever a kernel package is installed or updated. This is done via the post-install script embedded into the kernel package. Traditionally each Linux distribution created their own custom tool, typically called **mkinitrd**, for creating an initial ramdisk and the scripts inside of it.

The Dracut project aims to provide a distribution-neutral initial ramdisk standard for use by all Linux distributions.

One of the design decisions of Dracut is that it can optionally build initramfs files which are generic by embedding all kernel modules and configuration scripts that could be needed to mount the root filesystem. During runtime, Dracut will inspect the system and load what's appropriate for the hardware. This means that an initramfs built on one computer will work on a different computer as long as the distribution, exact kernel version, and CPU architecture is the same.

The **dracut** command has a location configuration directory `/etc/dracut.conf.d/` that can be used to override settings from the vendor configuration directory `/lib/dracut/dracut.conf.d/`.

Dracut functionality to handle specific boot scenarios is handled by individual modules located in the `/usr/share/dracut/modules.d/` directory. When **dracut** is run, it consults its configuration, inspects the system (if `hostonly="yes"`) and generates an initramfs.

[S12] *The following applies to SLES12 only:*

SLES12 configures dracut in the `/etc/dracut.conf.d/01-dist.conf` file. Ubuntu configures dracut in the `/etc/dracut.conf.d/10-debian.conf` file. In this file, `hostonly="yes"` is set.

To re-generate an initramfs for the currently running kernel:

```
# dracut --force
```

To generate an initramfs for a specific kernel:

```
# dracut /boot/initramfs.img kernel_version
```

The **dracut rd.*** parameters in the GRUB configuration file will be host specific. In typical scenarios, the parameters are optional, but will speed up the boot and make the boot more reliable if a hard drive from another Linux system is added to the system. Without parameters Dracut will operate in full autodetect-everything mode.

Dracut also includes a wrapper script called **mkinitrd** for backwards compatibility.

Generic Initial Ramdisk

One option that can be set is `hostonly="no"` which causes Dracut to build a generic initial ramdisk, instead of host specific. This ramdisk can then be used to boot the system, even if a catch-22 scenario is created after installation. For example, if a SCSI controller is changed to a different model or brand. Notably, no host specific `/etc` files will be included either (such as `/etc/modprobe.d/` files).

[R7] *The following applies to RHEL7 only:*

To automatically configure **dracut** to generate the `hostonly="no"` initial ramdisks, install the **dracut-config-generic** RPM.

Alternatively, to leave the standard initial ramdisk host specific, yet have a generic rescue initial ramdisk GRUB boot menu option, set `GRUB_DISABLE_RECOVERY="false"` in `/etc/default/grub` and install the **dracut-config-rescue** RPM.

[S12] *The following applies to SLES12 only:*

To override the default `hostonly="yes"` in the `/etc/dracut.conf.d/01-dist.conf`, create a file `99-local.conf` in the same directory with the line `hostonly="no"`.

Interactive Initial Ramdisk

There are some initial ramdisk parameters that can aid in bootup troubleshooting. These parameters can be added interactively from the GRUB menu.

rd.break ⇒ Break to a shell prompt within the initramfs environment before switching to the real root filesystem. When done with shell, type **exit** to continue boot.

rd.break=pre-mount ⇒ Break to a shell prompt within the initramfs environment before the real root filesystem is mounted. When done with shell, type **exit** to continue boot.

rd.shell ⇒ If the real root filesystem can't be mounted, drop to a shell instead of kernel panic.

init

First userspace process launched by kernel

- PID 1
- All other processes are children of init
- Troubleshooting Tip: Pass `init=/bin/sh` to launch `/bin/sh` instead of `init`

Modern Linux distributions have standardized on systemd

- On boot, systemd recursively activates all units that are dependencies of the `default.target`
`default.target` is usually aliased to `multi-user.target` or `graphical.target`
`local-fs.target` → `sysinit.target` → `basic.target` → `multi-user.target` (optionally) → `graphical.target`

Overriding init

The Linux kernel will normally start the program `init` after it boots up, will initialize all system hardware, and mount the root filesystem. Occasionally, it can be beneficial to start an alternate program in place of `init`; this can be done by passing to the kernel an option specifying the name of the alternate program to run, such as `init=/bin/bash`, telling the kernel to start the `bash` shell in place of `init`. Using `bash` (or another shell) instead of `init` is often useful when troubleshooting system boot problems.

With `init=/bin/bash`, no systemd units are activated or scripts from /etc/ are run, so some minimal configuration is required in order to access all the filesystems read-write:

```
# /usr/sbin/load_policy -i # If using SELinux  
# mount -o remount,rw /  
# /usr/lib/systemd/systemd-udevd --daemon  
# /sbin/vgchange -ay           # If using LVM  
# mount -a
```

Emergency Access Without root Password

Prior to RHEL7, booting to single user mode provided root access without prompting for a password. Now with systemd, the root password is required no matter what `systemd.unit=` or corresponding backwards compatibility alias is used. Passing a shell as the alternate init program (as shown above) is the suggested method if access is required and the root password is unknown.

The init Program

On all Unix systems, the final action taken by the kernel during system boot-up is to start an initial program, usually `/sbin/init`. This initial program always has a process ID of 1, and its successful operation is absolutely essential for the stability of the system. In fact, most Unix kernels will crash if this initial program ever stops running for any reason.

With systemd, `init` is a symlink to `../lib/systemd/systemd`, on boot it activates the `default.target`. To do so it works backwards and recursively activates all units that are dependencies of `default.target`.

The main boot logic is `default.target` depends on `basic.target` which depends on `sysinit.target`. A complete chart of the unit interdependencies that come into play during boot can be seen in the `bootup(7)` man page.

Viewing the default.target dependency tree

The `list-dependencies` option to the `systemctl` command. Use `--all` to recursively see the entire dependency tree.

```
# systemctl list-dependencies  
... output omitted ...  
# systemctl list-dependencies --all  
... output omitted ...
```

Systemd local-fs.target and sysinit.target

R7: Replaces /etc/rc.sysinit

S12: Replaces /etc/init.d/boot

local-fs.target **first target after initramfs, pulls in units that:**

- Checks/remounts /, Activates swap, Updates/activates disk quotas and DM RAID (if in use)
- Dynamically generate mount units for each regular filesystem in /etc/fstab via **systemd-fstab-generator**

sysinit.target **next major target processed during boot**

- Starts **systemd-udevd**
- Configures kernel parameters: /etc/sysctl.conf & /etc/sysctl.d/
- Launches plymouth for graphical boot

Critical low-level boot tasks

During the boot up of a Linux system, there are critical low-level tasks that must be performed no matter if the system is booting to a text or graphical login screen, or if the Apache web server will be started or not. With systemd, these tasks are defined as systemd targets or services that are dependencies of the local-fs.target and sysinit.target. Some of the key tasks including checking and remounting read-write the root filesystem, checking and mounting the other regular filesystems, mounting the various kernel virtual filesystems, applying persistent kernel tuning parameters, launching UDEV via **systemd-udevd**, and launching the Journal via **systemd-journal**.

The dependencies are either created with symlinks in the sysinit.target.wants/ and local-fs.target.wants/ directories, or via Before= lines in the unit files themselves.

The local-fs.target target

To see what local-fs.target pulls in via Before= lines, run:

```
# grep -lr '^Before.*local-fs\.target' /usr/lib/systemd/system/  
/usr/lib/systemd/system/quotaon.service  
/usr/lib/systemd/system/systemd-fsck-root.service  
/usr/lib/systemd/system/systemd-quotacheck.service  
/usr/lib/systemd/system/systemd-remount-fs.service  
... snip ...
```

Systemd generators

In addition to these hard coded target dependencies, systemd supports dynamically generated units via "generators" which are programs the live in /usr/lib/systemd/system-generators/ that create systemd unit files under the /run tmpfs filesystem in the /run/systemd/generator/ directory tree.

The processing of /etc/fstab

The **systemd-fstab-generator** generator is how systemd handles the processing and mounting of the regular filesystems listed in the /etc/fstab. See **systemd.mount(5)** for special mount options that can be used in /etc/fstab. When run, it dynamically creates a mount unit corresponding to each filesystem. For example:

```
# cd /run/systemd/generator/local-fs.target.requires/  
# cat boot.mount  
# Automatically generated by systemd-fstab-generator  
  
[Unit]  
SourcePath=/etc/fstab  
Before=local-fs.target  
Requires=systemd-fsck@dev-disk-by\x2duuid-52941340.service  
After=systemd-fsck@dev-disk-by\x2duuid-52941340.service  
  
[Mount]  
What=/dev/disk/by-uuid/52941340  
Where=/boot  
Type=xfs
```

The sysinit.target target

To see what units the sysinit.target pulls in, run the following command:

```
# ls -l /usr/lib/systemd/system/sysinit.target.wants/ /etc/systemd/system/sysinit.target.wants/
lrwxrwxrwx. 1 root root 20 Oct 24 14:59 cryptsetup.target -> ../cryptsetup.target
lrwxrwxrwx. 1 root root 22 Oct 24 14:59 dev-hugepages.mount -> ../dev-hugepages.mount
lrwxrwxrwx. 1 root root 19 Oct 24 14:59 dev-mqueue.mount -> ../dev-mqueue.mount
lrwxrwxrwx. 1 root root 28 Oct 24 14:59 kmod-static-nodes.service -> ../kmod-static-nodes.service
. . . snip . . .
```

Although most systemd-using Linux distributions will be similar, there will be some distribution specific units.

Systemd basic.target and multi-user.target

basic.target is next target after sysinit.target finishes

- R7: Starts `firewalld`, saves `dmesg` output, performs SELinux relabel if requested, performs reconfiguration via `firstboot` if requested
- S12: Restores sound card state, generates dynamic udev rule to create `/dev/root` symlink

multi-user.target last major target, processed after basic.target

- Primary target for standard services
`agetty`, `postfix`, Apache, `ntpd`, `rsyslogd`, `sshd`, etc.

graphical.target used if aliased to default.target

- pulls in multi-user.target as a dependency
- Has `Wants=display-manager.service` to launch GUI Login

`/usr/lib/udev/write_dev_root_rule` to generate the `/run/udev/rules.d/10-root-symlink.rules` UDEV rule file that creates the `/dev/root` symlink needed by some commands such as `dracut`.

multi-user.target

The multi-user.target is the last major grouping of systemd units. This is how standard Linux services get started. When enabling a service unit for a daemon, behind the scenes, a symlink is created in the `/etc/systemd/system/multi-user.target.wants/` directory to the unit file. Distribution shipped unit files live in the `/usr/lib/systemd/system/` directory.

To see what units the multi-user.target pulls in, run the following command:

```
# ls -l /usr/lib/systemd/system/multi-user.target.wants/ >
    /etc/systemd/system/multi-user.target.wants/
. . . output omitted . . .
```

Booting to Text mode or GUI Login

When systemd starts, it activates the `default.target` and anything it depends on. The `default.target` is an alias to either `multi-user.target` or `graphical.target`. The alias is determined by which unit file the `/etc/systemd/system/default.target` symlink points to. Use the `ls -l` command to see where the symlink points to, or run:

```
# systemctl get-default
```

basic.target

The `basic.target` is the third grouping of systemd units during boot, following `local-fs.target` and `sysinit.target`. Many Linux distributions use this for distribution specific boot tasks or other mid-level tasks that should be performed before the standard services are started.

To see what units the `basic.target` pulls in, run the following command:

```
# ls -l /usr/lib/systemd/system/basic.target.wants/ >
    /etc/systemd/system/basic.target.wants/
. . . output omitted . . .
```

[R7] The following applies to RHEL7 only:

- ⊗ The `rhel-autorelabel.service` runs the `/lib/systemd/rhel-autorelabel` script if the file `/.autorelabel` exists.
- ⊗ The `rhel-configure.service` runs the `/lib/systemd/rhel-configure` script if the file `/.unconfigured` exists.

[S12] The following applies to SLES12 only:

- ⊗ The `alasa-(restore|state).service` uses one of two different ALSA sound card state management schemes to restore the sound card state.
- ⊗ The `systemd-udev-root-symlink.service` runs the

multi-user.target

To change the alias, manually change the symlink, or run:

```
# systemctl set-default graphical.target  
rm '/etc/systemd/system/default.target'  
ln -s '/usr/lib/systemd/system/graphical.target' ↵  
    '/etc/systemd/system/default.target'
```

To override the default.target alias from GRUB, append
systemd.unit=desired.target to the kernel command line.

Legacy local bootup script support

Provides simple, easy way of run things on boot
systemd-rc-local-generator creates

- rc-local.service for backwards compatibility
If legacy script exists and is executable

R7: Legacy script /etc/rc.d/rc.local

- Called after network.target

S12: Legacy script /etc/init.d/boot.local

- Called after basic.target

Legacy local halt script also supported

Legacy init.d scripts also supported

- All legacy SysV init scripts subject to a 5 min timeout

Creating custom unit files

To carry out boot tasks on the system, custom unit files can be created in the /etc/systemd/system/ directory. Symbolic links can then be created in the /etc/systemd/system/multi-user.target.wants/ to the unit file. However, creating full-blown systemd unit files can be overkill. For example, some tasks need to be executed once when the system boots up, but never need to be killed, or monitored.

One-Time Bootup Tasks

systemd maintains backwards compatibility with the legacy local bootup script using the **systemd-rc-local-generator** generator. If the legacy local bootup script exists and is executable, it will generate

/run/systemd/generator/multi-user.target.wants/rc-local.service.

[R7] *The following applies to RHEL7 only:*

The legacy local bootup script is **/etc/rc.d/rc.local**

[S12] *The following applies to SLES12 only:*

The legacy local bootup script is **/etc/init.d/boot.local**

Example contents from a local bootup script file

```
# disable VT console blanking
setterm -blank 0 </dev/console> /dev/console 2>&1
# enable the power aware CPU schedule
```

echo 1 > /sys/devices/system/cpu/sched_mc_power_savings

One-Time Halt Tasks

systemd maintains backwards compatibility with the legacy local halt script using the **systemd-rc-local-generator** generator. If the legacy local halt script exists and is executable, it will generate /run/systemd/generator/final.target.wants/halt-local.service.

[R7] *The following applies to RHEL7 only:*

The legacy local halt script is **/usr/sbin/halt.local**.

[S12] *The following applies to SLES12 only:*

The legacy local halt script is **/etc/init.d/halt.local**.

System Configuration Files

The /etc/ Directory

- Standard system configuration file location
- Example: /etc/localtime, /etc/auto.master

The /etc/sysconfig/ Directory

- Distribution specific extension
- Example: /etc/sysconfig/clock, /etc/sysconfig/autofs
- Not used in Ubuntu

The /etc/ Directory

Unix tradition and standards dictate that system configuration files be stored in the /etc/ directory. This includes global configuration files such as /etc/localtime and /etc/resolv.conf, as well as daemon specific configuration files such as /etc/auto.master and /etc/crontab.

The /etc/sysconfig/ Directory

Not all system details have a universally recognized configuration file location. In addition, not all daemons can be fully controlled using their standard configuration file. For that reason, many Linux distributions such as Red Hat and SuSE use the /etc/sysconfig/ directory as a semi-standard location to store additional system settings. Ubuntu, on the other hand, has done away with the /etc/sysconfig/ directory entirely. This directory includes global configuration files such as /etc/sysconfig/clock as well as daemon specific configuration files such as /etc/sysconfig/autofs.

Most configuration files in /etc/sysconfig/ are sourced by SysV scripts in /etc/init.d/ when starting a service or by systemd unit files with the EnvironmentFile= option. As a result, most files in /etc/sysconfig/ utilize shell variables. For example, the following shows how to use `irqbalance`'s -d option to enable debugging.

File: /etc/sysconfig/irqbalance

-	#IRQBALANCE_ARGS=
+	IRQBALANCE_ARGS="-d"

[R7] *The following applies to RHEL7 only:*

Red Hat provides documentation for the majority of files in /etc/sysconfig/ in the file /usr/share/doc/initscripts-*/sysconfig.txt.

[S12] *The following applies to SLES12 only:*

SUSE provides documentation for the majority of files in /etc/sysconfig/ in the directory /usr/share/doc/packages/sysconfig/.

RHEL7 Configuration Utilities

firstboot

- runs on the first boot after installation
- non-root user creation, authentication config, date & time, and kdump

Configuration Utilities

- **setup**
- **system-config-***

firstboot

The **firstboot** utility runs first thing after installation is complete and the system has rebooted. It allows for common post-install configuration tasks. If desired, **firstboot** can be prevented from running post-install by using Kickstart: add `firstboot --disable` to the Kickstart file. It is also possible to re-run **firstboot** by running `touch /.unconfigured` then rebooting. All configuration performed by **firstboot** can also be performed by hand or using other tools.

System Configuration Utilities

The **setup** command provides a convenient front-end menu interface for accessing a suite of system configuration utilities provided by Red Hat Enterprise Linux. **setup** has shows a dynamically generated menu depending on what **system-config-*tui** configuration tools are installed and with configuration files in `/etc/setuptool.d`. With the appropriate **system-config** packages installed, the **setup** menu will consist of: Authentication, Keyboard, System services, and Firewall configuration.

The configuration tools available in **setup** can also be run individually from the command line. Most of these configuration tools are named following the pattern **system-config-***. Because of this it is easy to see what utilities are available by using the command completion feature of the shell, for example:

```
# system-config-[Tab][Tab]  
system-config-authentication  system-config-printer  
system-config-date          system-config-printer-applet
```

system-config-firewall
system-config-firewall-tui
system-config-kdump
system-config-keyboard
system-config-kickstart
system-config-language

system-config-users

Also included is **nmtui**, a TUI for managing network interfaces.

SLES12 Configuration Utilities

YaST

- Graphical Interface: `yast2`
 - Text Mode (ncurses): `yast`
 - Installing Software: `yast -i package`
- /usr/sbin/config.* scripts**
- Rewrites configuration files
 - Previously handled by SuSEconfig in SLES11 and older

A list of currently installed YaST modules can be obtained by passing the `-l` option to the `yast` command:

```
# /sbin/yast -l
```

Software Installation with YaST

By using the `-i` option to `yast`, software can be installed with dependency resolution. Package source defaults to the operating system installation source(s).

Configuration File Management

Some configuration files in SLES12 are actually rewritten whenever their daemon is restarted or `yast` is run. This behavior can be surprising and frustrating if one is unaware and tries to edit rewritten files by hand.

Some configuration files in SLES12 are managed by config scripts in `/sbin/` and variables in `/etc/sysconfig/` to rewrite configuration files. For example, files in the `/etc/postfix/` directory are recreated based on the contents of `/etc/sysconfig/postfix` by `/usr/sbin/config.postfix`.

Other daemons have their own logic for rewriting config files. For example, before `/etc/init.d/apache2` starts Apache it runs `/usr/share/apache2/get_module_list` to rewrite `/etc/apache2/sysconfig.d/loadmodule.conf` based on the contents of `/etc/sysconfig/apache2`.

System Configuration with YaST

One significant differentiator that sets SUSE Linux Enterprise Server apart from other Linux distributions is YaST (Yet another Setup Tool). YaST has deep and wide integration with the entire system. It is not only the installer, but can also be used to manage the system after install. YaST can be used to configure global system and individual daemons settings. It can also be used to install software. Because of YaST's modular design, installing new RPMs can add new functionality to YaST.

YaST has two primary interfaces: graphical and ncurses. The graphical mode can be launched by running `yast2` from the command line or by choosing Computer → YaST from the application menu. YaST's text mode uses the ncurses library to approximate a similar interface to the graphical mode. Text mode can be launched by running `yast` from the command line; it will also be launched if graphical mode can not be started. This is useful for systems where X is unneeded, for example a firewall or dedicated server. It's also useful for remote system configuration and administration even over low bandwidth connections.

YaST Modules

If the name of the YaST module is known it can be accessed directly by specifying the module name after the `yast2` command or the `yast` command. For example, to start graphical YaST and load the User and Group Administration module:

```
# /sbin/yast2 users
```

Shutdown and Reboot

Activating poweroff.target provides a graceful shutdown

- `systemctl poweroff`
- `shutdown`
- Legacy commands: `poweroff`, `halt`, `init 0`

Activating reboot.target provides a graceful reboot

- `systemctl reboot`
- `shutdown -r`
- Legacy commands: `reboot`, `init 6`

Instant, non-graceful shutdown and reboot

- `systemctl -ff halt` or `halt -f`
- `systemctl -ff reboot` or `reboot -f`

The shutdown Command

The **shutdown** command is the preferred method of rebooting or shutting down Linux. Using shutdown you can schedule a reboot or shutdown at some time in the future, and **shutdown** will automatically notify all logged-in users of the impending action. As the time approaches, it will alert logged-in users with a greater frequency and urgency.

With **shutdown**, a message can be specified that is sent to all logged-in users along with each alert.

The basic (different init systems may include other options) syntax of the **shutdown** command is:

shutdown [-crhHkP] now|+m|hh:mm [warning-message]

-k ⇒ Don't really shutdown; only send the warning messages to everybody.

-r ⇒ Reboot after shutdown.

-h ⇒ Halt after shutdown.

-c ⇒ Cancel an already running shutdown. With this option it is of course not possible to give the time argument, but you can enter an explanatory message on the command line that will be sent to all users.

time ⇒ When to shutdown, most often **now**, or an exact time (e.g. **21:00**), or minutes in the future (e.g. **+5**). This is a required argument.

warning-message ⇒ Message to send to all users. Surround with

quotes if messages contains spaces.

The reboot, poweroff, and halt Commands

The **reboot** and **halt** commands, unlike older versions of Unix, call the **shutdown** command with the **-r** or **-h** option, respectively.

The **poweroff** command activates the poweroff target, which will gracefully halt, then power off (if supported by the hardware).

Ctrl Alt Del

From a text terminal (i.e. not within X), press  +  +  to reboot. This will activate the ctrl-alt-del target.

Ungraceful halt or reboot

When using **systemctl** to halt, poweroff, reboot or kexec the **--force** or **-f** option can be used either once or twice. If the force option is used once, it forcibly kills all processes and then all filesystems are unmounted or remounted read-only. This provides a drastic and fast method that still leaves the filesystems clean.

If the force option is used twice, no processes are terminated in advance and no filesystems are unmounted or remounted. This is the fastest and most ungraceful method. For example:

```
# systemctl -ff reboot
```

Lab 3

Estimated Time:
S12: 45 minutes
R7: 45 minutes

Task 1: Boot Process

Page: 3-26 Time: 10 minutes

Requirements:  (1 station)  (classroom server)

Task 2: Booting directly to a bash shell

Page: 3-30 Time: 10 minutes

Requirements:  (1 station)

Task 3: GRUB Command Line

Page: 3-33 Time: 5 minutes

Requirements:  (1 station)

Task 4: Basic GRUB Security

Page: 3-36 Time: 10 minutes

Requirements:  (1 station)

Task 5: Troubleshooting Practice: Boot Process

Page: 3-39 Time: 10 minutes

Requirements:  (1 station)

Objectives

- ❖ Use GRUB to boot into the rescue.target.
- ❖ Modify kernel parameters in GRUB.

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

Properly booting Linux systems sometimes requires modifications to the default boot loader configurations provided by a distribution's install program.

Notices

- ❖ If this lab exercise is being run within a virtual environment, the use of special keystrokes may be needed to switch between virtual terminals.

- 1) Reboot the system.
- 2) When the GRUB screen appears, Press the **Esc** or **Space Bar** key to stop the countdown timer before it reaches 0 and the system boots on its own.
- 3) It is common to boot Linux into the rescue target to debug boot problems since no system or network services (which can make debugging easier). Boot the system to the rescue.target:

Select the default kernel boot option.

The default kernel boot option is generally the first one listed.

Press **e**. To modify what parameters are passed to the kernel.

Press **↓** approximately 13 times until the cursor is on the first linux line.

Press **ctrl**+**e** to move to the end of the line.

GRUB 2 supports emacs key bindings

Type **systemd.unit=rescue.target**

Be sure to have a leading space. For less typing, use one of the alternatives 1, s, S, or single

Press **ctrl**+**x**. To initiate the actual boot.

- 4) After the system has booted, enter the root password to access the system:

Welcome to emergency mode! Type "systemctl default" or ^D to enter default mode.

Lab 3

Task 1

Boot Process

Estimated Time: 10 minutes

Type "journalctl -xb" to view system logs. Type "systemctl reboot" to reboot.

Give root password for maintenance

(or type Control-D to continue): **makeitso**

- 5) At the command prompt, verify that this is indeed a root shell:

```
# id  
[R7] uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0  
[S12] uid=0(root) gid=0(root) groups=0(root)
```

With root access, it is possible to perform a filesystem check, disable services that are stopping the system from booting correctly, change the root password, fix incorrect network settings, etc.

- 6) Continue booting to the default target:

```
# systemctl default
```

- 7) Login as root, open a Terminal window and then isolate the multi-user.target to kill the graphical environment and display a text mode login:

```
# systemctl isolate multi-user.target
```

- 8) After the multi-user.target has been isolated, the login program is launched. Log in as guru:

```
stationX login: guru  
Password: work   
. . . snip . . .
```

- 9) See what "runlevel" the legacy runlevel command reports:

```
$ /sbin/runlevel  
5 3
```

- 10) Reboot the system.

```
$ su -c "systemctl reboot"
```

Password: **makeitso**

. . . output omitted . . .

- 11) When the GRUB screen appears, Press the key to stop the countdown timer before it reaches 0 and the system boots on its own.
- 12) The Linux kernel supports many optional parameters that can be passed at boot time. Pass the `printk.time=1` parameter to observe the effect it has:

Select the kernel that should be booted.

Use the arrow keys to change the selected kernel (there may only be a single kernel in the list).

Press To modify what parameters are passed to the kernel.

Press approximately 13 times until the cursor is on the first linux line.

The auto-generated GRUB 2 config file is verbose.

Press + to move to the end of the line.

GRUB 2 supports emacs key bindings

Remove quiet, along with rhgb or splash, from the list of kernel parameters.

The rhgb or splash parameter starts up the graphical bootup display and quiet suppresses almost all of the kernel's output on boot.

Then Add `printk.time=1` to the kernel parameters

This will prefix the time from boot to each message stored in kernel log buffer.

Press + To initiate the actual boot.

- 13) As the system boots, notice that kernel messages are prefixed with a timestamp.

- 14) After the system has booted, log in as **guru**

- 15) Run the dmesg command to see the contents of the kernel log buffer. Notice that kernel messages are prefixed with a timestamp

[R7] \$ **dmesg**

[S12] \$ **su -c "dmesg"**

[S12] Password: **makeitso**

. . . output omitted . . .

- 16) See what parameters were passed to the kernel on boot:

```
$ cat /proc/cmdline
[R7] BOOT_IMAGE=/vmlinuz-3.10.0-123.6.3.el7.x86_64 root=UUID=d702750a ro-
      rd.lvm.lv=vg0/root crashkernel=auto vconsole.font=latarcyrheb-sun16 rd.lvm.lv=vg0/swap vconsole.keymap=us-
      LANG=en_US.UTF-8 printk.time=1
[S12] BOOT_IMAGE=/vmlinuz-3.12.28-4-default root=UUID=579713f3-
      resume=/dev/system/swap crashkernel=224M-:112M showopts printk.time=1
```

Objectives

- ❖ Change the root password
- ❖ Interactively modify the kernel command line
- ❖ Boot directly to a bash shell
- ❖ Manually perform critical boot time tasks
- ❖ Reset the root password back

Requirements

▀ (1 station)

Relevance

Sometimes a system won't properly boot, or does boot but won't allow logging in. In those cases booting directly to a bash shell allows the system administrator to potentially correct this problem. This is useful skill to practice and memorize.

Notices

- ❖ This lab exercise sets up a common real world scenario, forgetting or not knowing the root password as a pretext for needing to boot directly to bash.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Change the root password to a random value that you do not know to simulate a common real world scenario. View the current cipher text, change to password to random number, and then view it again:

```
# grep root /etc/shadow  
root:$6$d90mMJvmz/CXqFmP$u1mA3arm. . . snip . . .  
# usermod -p $(openssl passwd -1 $RANDOM) root  
# grep root /etc/shadow  
root:$1$h00Y7V4b$jNqq6bz0s0. . . snip . . .
```

The number after the first dollar sign indicates the encryption or hash method used by the cipher text. The 1 value indicates MD5 and 6 indicates SHA-512.

Lab 3

Task 2

Booting directly to a bash shell

Estimated Time: 10 minutes

- 3) Reboot the box and then when the GRUB menu displays, press the **[Space Bar]** to stop the GRUB timer.

```
# reboot
```

- 4) Use the GRUB 2 procedure to append `init=/bin/bash` to the end of the kernel command line.

Select the kernel that should be booted.

Use the arrow keys to change the selected kernel (there may only be a single kernel in the list).

Press **[e]**. To modify what parameters are passed to the kernel.

Press **[t]** approximately 13 times until the cursor is on the first `linux` line.

The auto-generated GRUB 2 config file is verbose.

Press **[ctrl]+[e]** to move to the end of the line.

GRUB 2 supports Emacs key bindings

Delete the `rhgb` and `quiet` parameters from kernel command line if they exists.

If they are not removed, when you type at the bash prompt the characters won't echo to the screen.

Type `init=/bin/bash`

Be sure to have a space between the new `init` parameter and previous parameter.

Press **[ctrl]+[x]**. To initiate the actual boot.

- 5) Since all boot time system initialization was skipped by booting directly to bash manually preform the minimum tasks required to get all the filesystems mounted read-write:

```
[R7] # /usr/sbin/load_policy -i  
# mount -o remount,rw /  
# /usr/lib/systemd/systemd-udevd --daemon  
# /sbin/vgchange -ay  
# mount -a
```

- You may get harmless warning messages if LVM is configured to use the `lvmetad` daemon.

If you are using SELinux and don't or forget to run the `load_policy` command, then you must **touch** `/.autorelabel` before you exit from the bash shell so that the SELinux labels on any modified files are restored.

- 6) Now reset the root password back to **makeitso**. The passwd command will complain that it is based on a dictionary word, proceed anyway.

```
# passwd root  
Changing password for user root.  
New password: makeitso   
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word  
Retype new password: makeitso   
passwd: all authentication tokens updated successfully.
```

- 7) Unmount all filesystems, and remount the root filesystem read-only in preparation of rebooting. Normally this is done by systemd units but since you booted directly to bash it must be done manually.

```
# umount -a  
# mount -o remount,ro /
```

- 8) Forcefully reboot the system via the power button, or configure the Linux kernel to reboot after 1 second on a panic, and then exit the shell.

```
# echo 1 > /proc/sys/kernel/panic  
# exit
```

The Linux kernel always panics when the init program exits. Since the bash shell is init for this boot, exiting it triggers a panic.

Objectives

- » Explore the GRUB interface.
- » Attach to the device holding the /boot/ filesystem and display the contents of the GRUB configuration file.

Requirements

- (1 station)

Relevance

Sometimes, it is necessary to use GRUB to repair a damaged system and/or its boot process.

- 1) Reboot the system.
- 2) When the GRUB screen appears, Press the **[Space Bar]** key to stop the countdown timer before it reaches 0 and the system boots on its own.
- 3) The GRUB software is a powerful, pre-boot environment. Many GRUB commands can be run before the operating system is even loaded. At the GRUB screen, press **[c]** to get a command line.
- 4) View the current pager settings, and enable it if it is not set so that any output that is longer than one screen doesn't scroll by.

```
grub> echo $pager  
0  
grub> set pager=1
```

Lab 3

Task 3

GRUB Command Line

Estimated Time: 5 minutes

• If the value is 1, skip to the next step.

- 5) A GRUB prompt will be displayed, run the help command to display a list of available GRUB commands:

```
grub> help  
... output omitted ...
```

Use the **[Space Bar]** to page down through output.

- 6) Obtain detailed help on the cat and root commands:

```
grub> help cat  
Usage: cat FILE  
Show the contents of a file.  
... snip ...  
grub> cat --help  
Usage: cat FILE  
Show the contents of a file.  
... snip ...  
grub> help set  
... output omitted ...
```

- An alternate way to see the help for a command

- 7) Tell GRUB to treat the first partition on the first hard drive as its root filesystem for future commands (this should be the Linux /boot/ filesystem):

```
grub> set root='hd0,msdos1'
```

- 8) Display the contents of the /boot/grub2/grub.cfg GRUB configuration file and make note of the kernel and initrd information:

```
grub> cat /grub2/grub.cfg  
#  
# DO NOT EDIT THIS FILE  
#  
# It is automatically generated by grub2-mkconfig using templates  
... snip ...
```

- This is GRUB's cat command, not the /bin/cat command. Why /grub2/grub.cfg and not /boot/grub/grub.conf? Hint: Think about the root command.

Result: _____

- 9) Point GRUB to the Linux kernel and specify which kernel command line options to use.

Use the  filename completion feature of the GRUB shell to "type" the filename without typos:

```
[R7] grub> linux /vmlinuz-3.10.X-YY ro root=/dev/vg0/root  
[S12] grub> linux /vmlinuz-3.12.X-YY root=/dev/vg0/root
```

Replace X-YY with the correct version information for the kernel installed on the system.

10) Define which initial ramdisk to load:

Again, use the `tab` filename completion feature of the GRUB shell to "type" the filename without typos:

```
[R7] grub> initrd /initramfs-3.10.X-YY  
[S12] grub> initrd /initramfs-3.12.X-YY
```

Replace X-YY with the correct version information for the kernel installed on the system.

11) GRUB has been told everything it needs in order to boot so issue the boot command to initiate the kernel boot process:

```
grub> boot  
. . . output omitted . . .
```

If there are any error messages, it is most likely a typo in either the kernel or initrd commands. Just retype the correct command and issue the boot command again. GRUB's command line supports the use of the up and down arrow keys to recall previous commands and the left and right arrow keys to make editing those commands easier.

Objectives

- Set a GRUB password.

Requirements

- (1 station)

Relevance

GRUB is very powerful and capable. So much so, that if an attacker can get to a GRUB prompt, the attacker can read any file on any partition, thus bypassing the OS security. To prevent this attack, a password can be set in GRUB to prevent any unauthorized access to the GRUB shell.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) GRUB can be configured with a password that must be used to obtain a GRUB prompt or to modify kernel parameters. During the installation, the opportunity to set a GRUB password is presented. Since a GRUB password wasn't set during install, set one manually.

- 3) [R7] *This step should only be performed on RHEL7.*

As of RHEL7.3, the grub2-setpassword command automates GRUB superuser creation.

```
# grub2-setpassword  
Enter password: gurulabs   
Reenter password: gurulabs 
```

- 4) [S12] *This step should only be performed on SLES12.*

Generate a password hash using the grub2-mkpasswd-pbkdf2 binary:

```
# grub2-mkpasswd-pbkdf2  
Enter password: gurulabs   
Reenter password::: gurulabs   
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.EDCB9C1F8...
```

Lab 3

Task 4

Basic GRUB Security

Estimated Time: 10 minutes

Make note of the displayed password hash. Copying it to the clipboard is recommended for this lab.

5) [S12] This step should only be performed on SLES12.

As root, modify the /etc/grub.d/10_linux so that generated menu entries all have the --unrestricted option set:

File: /etc/grub.d/10_linux
- CLASS="--class gnu-linux --class gnu --class os" + CLASS="--class gnu-linux --class gnu --class os --unrestricted"

6) [S12] This step should only be performed on SLES12.

As root, define the GRUB 2 superuser:

File: /etc/grub.d/42_password
+ cat <<EOF + set superusers="root" + password_pbkdf2 root grub.pbkdf2.sha512.10000.password_from_previous_step + EOF

7) [S12] This step should only be performed on SLES12.

Make the file executable and only readable by the root user for security:

```
# chmod 750 /etc/grub.d/42_password
```

8) Regenerate the /boot/grub2/grub.cfg file:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg  
Generating grub configuration file ...  
. . . output omitted . . .
```

9) Reboot the system.

10) When the GRUB screen appears, Press **Esc** or the **Space Bar** key to stop the countdown timer before it reaches 0 and the system boots on its own.

- 11) Test the new GRUB password by attempting to edit the first menu entry:

Press **[e]** Brings up the Enter username: prompt.

Type **root** Brings up the Enter password: prompt.

Type **gurulabs** **[Enter]** Inputs the password and unlocks the interactive editing features of GRUB are now available if desired.

Press **[Esc]** Returns to the boot menu.

- 12) Select the top entry and press enter to boot it.

Cleanup

- 13) The following actions require administrative privileges. Switch to a root login shell:

```
$ su -  
Password: makeitso [Enter]
```

- 14) [S12] This step should only be performed on SLES12.

Remove the **--unrestricted** menu option:

```
# sed -i '/^CLASS/ s/ --unrestricted//' /etc/grub.d/10_linux
```

- 15) Remove the GRUB superuser definition and regenerate the grub.cfg file:

```
[R7] # rm /etc/grub.d/01_users  
[S12] # rm /etc/grub.d/42_password  
# grub2-mkconfig -o /boot/grub2/grub.cfg  
Generating grub configuration file ...  
. . . output omitted . . .
```

Objectives

❖ Practice troubleshooting boot process issues.

Requirements

☒ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You use these scripts to break your system in controlled ways, and then you troubleshoot the problem and fix the system.

- 1) Use tsmenu to complete the boot process troubleshooting scenario:

Troubleshooting Group	Scenario Category	Scenario Name
Group 1	Boot	boot-05.sh

Lab 3

Task 5

Troubleshooting Practice: Boot Process

Estimated Time: 10 minutes

Content

Managing Software	2
RPM Features	3
RPM Architecture	4
RPM Package Files	5
Working With RPMs	6
Querying and Verifying with RPM	7
Updating the Kernel RPM	9
Dealing With RPM & Yum Digest Changes	10
Yum Plugins & RHN Subscription Manager	11
YUM Repositories	12
YUM Repository Groups	13
Compiling/Installing from Source	14
Manually Installed Shared Libraries	16
Rebuilding Source RPM Packages	17
Lab Tasks	
1. Managing Software with RPM	20
2. Creating a Custom RPM Repository	24
3. Querying the RPM Database	28
4. Installing Software via RPM & Source and Rebuilding SRPMs	32
5. Troubleshooting Practice: Package Management ...	36

Chapter

4

SOFTWARE MAINTENANCE

Managing Software

Unix Packaging

- SysV packages

Linux Packaging

- Slackware Tarballs
- Debian dpkg
- Red Hat RPMs

Managing Installed Software

Unix software is typically distributed as uncompiled source text, usually provided in a Unix tape archive. To use this software, an administrator must compile and install it. If the software requires supporting packages, they must be built and installed in advance. If local customizations are required, such as porting to an unsupported platform, or bug fixes, they must be applied to the software before compiling/recompiling.

To upgrade, an administrator must find the original files for that software and replace them with the updated versions, taking care to preserve any local configuration changes, and remove any left over files.

Consider a typical Unix workstation application such as Mozilla's Firefox. Firefox consists of hundreds of files spread across a dozen different directories. When removing it, an administrator must find all those files and delete them. When upgrading, an administrator must manually replace all those files, or rely on an uninstall or upgrade script. To operate, Firefox requires that dozens of system executables and dynamically shared libraries be installed (many of which, in turn, require still other libraries).

Managing this complexity manually is certainly possible, and Unix administrators have done so successfully. However, a variety of vendors have developed tools which simplify the work necessary to cope. In the commercial Unix world, System V Unix modeled a standard packaging system which is the basis for packaging software

such as the pkg format once used with Solaris, or the DEPOT format used with HP-UX. These package systems supply tools which can be used to install, uninstall, and upgrade software obtained in a package format which contains the software and metadata about the software (such as tracking what other software needs to be installed).

In the Linux world, when the Slackware distribution was first released in 1993, it introduced the use of a rudimentary package format. Shortly thereafter, two new distributions, Debian then Red Hat, began efforts to develop more full-featured packaging software for Linux. Debian introduced the dpkg packaging system, while Red Hat introduced the RPM packaging system. Since that time, both packaging systems have become widely-utilized within the Linux community, and are used by most Linux distributions (though others exist). RPM is the de jure standard for package management in the Linux world. The LSB (Linux Standards Base) effort requires that LSB-compliant systems support installation of RPM files.

RPM Features

- Dependency tracking**
- Tracking of installed files**
- Verification of installed files**
- Non-interactive, scriptable installation**
- Queries of installed files**
- Tracking of all source and build process**
 - Pristine source preservation
- Digitally signed software**

Considering RPM Features

The RPM Package Manager, provides a variety of different features which make it an excellent package manager for use by the Linux community, and a great tool to simplify the work life of Linux administrators.

RPM provides a set of tools which can be used to carry out non-interactive, script-able installations of software. Once that software is installed, RPM provides tools which can track those installed files, making it easily possible to uninstall them later, or to upgrade them. It also provides several methods of verifying those installed files, allowing administrators to double-check that the installed software is installed correctly, and has not been inadvertently modified since installation. In addition, RPM provides a set of commands which can be used to search installed files to determine which application uses them.

Dependency Tracking

RPM has excellent dependency tracking capabilities, meaning that when used to install new software, it can first ensure that any software required for that new software to work is already installed; when used to uninstall software, it can first verify that doing so will not break any other applications.

Elegant Build Process

In addition, RPM provides a tool set which manages the entire process of patching, configuring, and compiling new applications.

When compiling an application, RPM makes it possible to start with the pristine source code for that application, produce any needed patches for that application, then script the process of applying those patches, configuring the source code, and compiling the source code to produce executables. This aspect of RPM greatly simplifies the process of maintaining custom-configured applications in the enterprise.

Digitally Signed Packages

Furthermore, RPM allows all packaged software to be signed digitally (using public-key technology). This feature allows the authenticity of software packaged for use with RPM to be verified, helping prevent the accidental installation of Trojan Horse software.

Delta RPMs

Incremental upgrades from one RPM release to the next, without requiring the download of complete, updated packages, is provided using delta RPM packages. This can provide significant bandwidth savings.

RPM Architecture

RPM package files

RPM database

- /var/lib/rpm/

RPM utilities

- rpm
- rpmquery, rpmverify, rpmsign
- rpmbuild
- rpm2cpio

RPM configuration files

- macros used during preparation and installation of RPMs

RPM Package Files

The RPM system consists of several components. Software which is to be installed using RPM must be supplied in a special format, the RPM package file. This RPM package file is an archive which contains the actual files to be installed, as well as metadata about those files which is used by the RPM system to ensure that those files are installed with the correct permissions and ownerships and in the correct locations.

RPM Database

As software is installed using RPM, the name and several other properties of each file being installed on the system are recorded in the RPM database, typically located in the /var/lib/rpm/ directory. This database contains a list of all installed applications, and the files which belong to those applications, allowing easy upgrade or uninstallation of applications at a later time. It also tracks properties of each file—such as its correct size, timestamp, and cryptographic checksum—ensuring that the file's correctness can be verified at a later date. This database also contains dependency information for every application, ensuring that administrators installing new applications can be certain that the applications the new software requires to operate are present, and that administrators removing applications can be certain that doing so will not break any other existing applications.

RPM Utilities

Several utilities are supplied for use with RPM. The basic utility used for most RPM administrative tasks is the /bin/rpm command. On systems with RPM v4.x+, many of the auxiliary functions formerly performed by the /bin/rpm command have been moved to helper utilities. For example, the /usr/bin/rpmbuild command is used to produce new RPM package files, while the /usr/bin/rpmsign command is used to sign new packages. Previously with older RPM versions those tasks were handled by the main binary, /bin/rpm. The /usr/bin/rpm2cpio command is provided for conversion of RPM package files into a standard archive file format.

RPM Configuration Files

RPM also has several configuration files. These configuration files primarily define macros (short-cut commands which are used when running RPM commands, or when preparing RPM package files).

The RPM configuration files are found in the /usr/lib/rpm/ directory. Vendor specific configuration can be found in the /etc/rpm/ directory.

RPM Package Files

Naming Conventions

- *name-version-release.architecture.rpm*

Architectures

- source: *.src.rpm*
- noarch: *.noarch.rpm*
- binary: *.x86_64.rpm*
- binary: *.i586.rpm*

Format

- cpio archive plus a binary header

Considering RPM Files

Software prepared for use with RPM must be packaged in an RPM package file. These package files should be named in the format:

name-version-release.architecture.rpm

In this package file name, the *name* indicates the software which is packaged in that RPM. Usually, this *name* is the name of the application.

In the RPM package file name, the *version* indicates the version of the software which is packaged in that RPM. This *version* is usually the version of the application which is being packaged in that RPM package file. Although it is not required by the RPM package file format or RPM utilities, this *version* field should always be a number; use of words in this field can (and has) caused problems for several front-end applications which use RPM.

The *release* in the RPM package file name is used to indicate revisions of the packaging of that particular version of that application. Sometimes, mistakes are made when preparing RPM package files of a specific version of a specific application. When that occurs, a new package file of that specific version of that specific application can be prepared which fixes those mistakes. The *release* field allows these revisions of package files to be tracked. This field should be a number, and should be increased every time the package is revised.

The *architecture* is the platform on which that RPM can be

executed, if binary. Typical values seen here include:

i586 ⇒ Pentium class 32-bit Intel-compatible CPU

i686 ⇒ Pentium Pro/II class 32-bit Intel-compatible CPU

x86_64 ⇒ any AMD64/Intel64 CPUs

ppc64 ⇒ any 64-bit PowerPC CPU

ia64 ⇒ any 64-bit Itanium CPU

s399x ⇒ 64-bit IBM zEnterprise 196 or later

noarch ⇒ the package can be used on any CPU architecture

Most 64-bit CPU architectures are capable of executing 32-bit binaries as well. As such, the RPM system supports "bi-arch", which allows having both the 32-bit and 64-bit architectures of the same package installed simultaneously.

In addition to executables, RPM package files can also be used to package platform-neutral executable code (such as programs written in an interpreted language like Lisp, Perl, or Java). RPM package files which contain code which can run on any CPU, or other files—such as documentation—which are platform-neutral are packaged with in RPM package files for which the *architecture* is noarch.

RPM package files can also be used to package application source code, patches, and scripts specifying how that source code should be configured and compiled into binary RPMs. These package files will have an architecture of *src*.

RPM package files are cpio archives with a binary header attached.

Working With RPMs

Installing without upgrading RPMs

- `rpm -i`

Upgrading/Installing & Freshening RPMs

- `-U` and `-F`
- `--oldpackage`, `--replacepkgs`, and `--force`

Uninstalling RPMs

- `rpm -e`

Useful Options

- `-v`, `-h`
- `--nodeps` ignore dependencies (not recommended)
- `--test` check for possible errors, but don't actually do anything

Installing RPMs

RPM package files are normally installed using the `-U` option to the `rpm` command. This will upgrade installed packages, as well as install new ones. This is often used with the `-v` option, which makes RPM be more verbose, and the `-h` option, which causes RPM to display progress meters:

```
# rpm -Uvh joe-3.5-92.22.x86_64.rpm
Preparing...                                           #####
1:joe                                              #####
                                                               [100%]
```

To install a new package without upgrading (such as to install two, non-overlapping, packages in parallel) use the `-i` option.

Packages can also be upgraded using the `-F` option (which is also used with the `-v` and `-h` options). This option causes the package to be upgraded if already installed and of an older version, but will not be installed if it is not already.

In order to downgrade to an older release of an installed package, use `--oldpackage` in conjunction with `-U`. Without `--oldpackage` RPM will correctly complain that a newer version is already installed. `--replacepkgs` will reinstall an installed package.

When performing upgrades, or freshening installed packages, the `rpm` utility compares the version of the RPM being installed with the version in the RPM database on the local system. If the version number of the package being installed is greater than the installed version number, RPM will upgrade. If the version number of the

package being installed is less than the installed version number, RPM will refuse to downgrade. If the versions numbers of the two are the same, then RPM compares the release numbers, and upgrades only if the release number of the new package is greater than the release number of the installed package.

In addition to using locally accessible files, the `rpm` command has the ability to take FTP, HTTP and HTTPS URLs as a filename argument.

Uninstalling RPMs

Once installed, applications can be removed using the `-e` option to the `rpm` command:

```
# rpm -e joe
```

When erasing installed software, only the name of the software (`joe`) is used. When installing new packages, or upgrading existing packages, the complete package file name (`joe-3.5.92-22.x86_64.rpm`) is used.

Overriding Default Behavior

`--force` ⇒ Install a package regardless of whether an upgrade, downgrade, or reinstall is being done, even if this replaces files from other, already installed, packages; install the packages even if some of them are already installed. `--replace{pkgs,files}`, `--oldpackage`, and `-U` are usually the better options.

`--nodeps` ⇒ do not perform the dependency checks before installing, or upgrading, a package.

Querying and Verifying with RPM

rpm -q | rpmquery has powerful query capabilities:

- files – see information about the package that supplied the file
- non-installed package files – see package contents
- system's RPM database – see information on installed packages

rpm -V | rpmverify: verify file integrity and changes

RPM GnuPG key checking

- rpm --import — Import GPG (GNU Privacy Guard) Key
- rpm -qa 'gpg-pubkey*' - List Imported Keys
- rpm -K package - Check Signature

Query package dependencies

- rpm -qRp package.rpm
- RHEL7: repoquery -R --resolve package

Querying Packages

The following examples show commonly used query options of the **rpm** command:

Example Command	Description
-qa	lists all packages installed on the system
-qi <i>packagename[s]</i>	information about package
-ql <i>packagename[s]</i>	list of files in package
-qilf <i>file[s]</i>	information and a list of files in the package that owns file
-qdp <i>packagefile[s]</i>	list documentation provided by non-installed packagefile
-qR <i>packagename[s]</i>	list packages on which this package depends
-q --scripts <i>packagename[s]</i>	list the scripts contained in given installed package
-q --changelog <i>packagename[s]</i>	list changelog contents for given installed package
-q --querytags	list all tags that can be used with --queryformat

Verifying Installed Packages

Verify will generate symbolic output to describe any changes to files in the package being verified. The **rpm(8)** man page lists the meaning of each of the symbols. A verification can be done on a single package or on all packages:

Example Command	Description
rpm -V <i>packagename [...]</i>	verify all the files provided by package.
rpm -Va	verify all the files provided by all packages currently installed
rpm -Vf <i>file [...]</i>	verify all the files provided in the package that provided file.

The **rpm(8)** man page contains definitions for all the letters used by **rpmverify** in its output.

Querying and Verifying from Packages

RPM query and verification can use a package as the data source, instead of the installed database, with the **-p** option. In addition, the **-p** optional argument can use both FTP and HTTP(S) URLs. While querying the package allows package inspection before installation, the use of verification allows either a remote or local copy, or a DVD copy that cannot be modified, to provide a simple, but limited, form of file intrusion detection. For example:

```
# rpm -Vp http://mirrors.example.com/EnterpriseLinux/7/→
  os/x86_64/Packages/acl-2.2.51-12.el7.x86_64.rpm
warning: http://mirrors.example.com/EnterpriseLinux/7/os→
  /x86_64/Packages/acl-2.2.51-12.el7.x86_64.rpm: Header→
  V3 RSApSHA256 Signature, key ID fd431d51: NOKEY
..5....T.    /usr/bin/chacl
..5....T.    /usr/bin/getfacl
..5....T.    /usr/bin/setfacl
... . snip . . .
```

RPM GPG Keys

When RPM packages are installed, upgraded, queried, or used for verification, and they indicate that they are signed by a GPG key, RPM will either verify the key using one of its imported keys or provide a warning.

To import a new GPG key to the database, use the **--import** option:

```
# rpm --import RPM-GPG-KEY
```

To retrieve a list of imported keys in the RPM database use the following **rpmquery** command:

```
# rpm -qa 'gpg-pubkey*'
gpg-pubkey-7fac5991-4615767f
gpg-pubkey-982e0a7c-4f34288f
. . . output omitted . . .
```

Once the key is discovered, it can be used with commands such as **RPM erase**:

```
# rpm -e gpg-pubkey-7fac5991-4615767f
```

To see what repositories the keys correspond to, use a custom query format:

```
# rpm -qa --qf '%{version}-%{release} %{packager}\n' 'gpg-pubkey*'
7fac5991-4615767f  Google, Inc. Linux Package,
                     Signing Key <linux-packages-keymaster@google.com>
982e0a7c-4f34288f  RPM Fusion free repository for,
                     Fedora (18) <rpmfusion-buildsys@lists.rpmfusion.org>
. . . output omitted . . .
```

Before installation, test a package against an imported key with the **-K** option:

```
# rpm -K ed-1.9-1.x86_64.rpm
ed-1.9-1.x86_64.rpm: rsa sha1 (md5) pgp md5 OK
```

Updating the Kernel RPM

Kernel Importance

- Proper operation required for system boot
- Kernel drivers required for proper hardware operation

Manual Kernel Updates

- Install new updates side by side existing kernel
Use **-i** option to **rpm**

Updating the Kernel Manually

The kernel is very special and the most critical piece of software on the computer. Because of this, updates should be treated with care. In particular, when installing a new kernel, the old kernel should be left installed. After a few weeks of running the new kernel without troubles, then the old kernel can be confidently removed.

When installing new kernels using rpm, use the **-i** option. For example:

```
# rpm -ivh /tmp/kernel-3.10.0-121.el7.x86_64.rpm
```

Then when you are ready to remove the old kernel, determine what kernels are installed:

```
# rpm -q kernel
kernel-3.10.0-118.el7.x86_64
kernel-3.10.0-119.el7.x86_64
kernel-3.10.0-120.el7.x86_64
```

If needed, remove one or more old kernels to save space on the /boot partition:

```
# rpm -e kernel-3.10.0-118
```

Updating the Kernel Automatically

The built-in package administration tool will automatically update old kernels.

[R7] *The following applies to RHEL7 only:*

YUM will preserve the previous kernel, while adding the new. By default, yum will keep three kernel installed due to the `installonly_limit=3` default setting in the `/etc/yum.conf` file:

```
# yum update kernel
```

[S12] *The following applies to SLES12 only:*

Both **zypper** and **yast** will replace the old kernel with the new. The old kernel will not be preserved. This requires a reboot. For example:

```
# zypper update kernel
```

Dealing With RPM & Yum Digest Changes

RPM

- Converting src RPMs
- Building backwards compatible RPMs

Yum

- Creating backwards compatible repositories

RPM & Yum Digest Changes

RPM and Yum use file digests, or hashes, to detect modified or corrupted files. RPM used to use MD5, and Yum used to use SHA1. Unfortunately, both hashes are vulnerable to deliberate attack.

Starting with RHEL6, both RPM and Yum use SHA256.

Converting src RPMs

Older versions of RPM are unable to recognize SHA256 as a valid checksum format. While this is generally not a problem for binaries which are usually built for a specific distribution release, it can be an issue when backporting packages. Although older versions of `rpm` do not understand SHA256, `rpm2cpio` can still be used to extract files before rebuilding the src RPM.

Building Backwards Compatible RPMs

To build backwards compatible RPMs, `rpmbuild` must be configured to use MD5.

```
$ rpmbuild -ba package.spec
  --define "_source_filedigest_algorithm md5"
  --define "_binary_filedigest_algorithm md5"
  --define "_source_payload nil"
```

Alternatively, chroot-based build systems like Mock will automatically achieve compatibility by using release specific tools.

Creating Backwards Compatible Repositories

To create backwards compatible Yum repositories, `createrepo` must be configured to use SHA1.

```
$ createrepo --checksum sha /var/ftp/repo
```

Yum Plugins & RHN Subscription Manager

Yum is extensible through plugins

- Use must be enabled in yum.conf

Common plugins include

- yum-rhn-plugin
- yum-plugin-changelog
- yum-plugin-fs-snapshot
- yum-plugin-versionlock

yum-plugin-versionlock

The versionlock plugin can be used to prevent configured packages from being updated to a newer version.

Extending Yum with Plugins

Yum can be easily extended by installing plugins. Plugins are written in Python and are usually distributed in RPM packages. The Yum default has plugins disabled, but most distributions enable their use by placing `plugins=1` in the `/etc/yum.conf` file. If enabled, Yum looks for plugins in the `/usr/share/yum-plugins/` and `/usr/lib/yum-plugins/` directories. All available plugins are documented in the System Administrators Guide:
<http://access.redhat.com/documentation/>

subscription-manager

The Red Hat Network (RHN) subscription-manager plugin to Yum provides the interface for accessing packages from the RHN update module. This allows for installing new packages and updating packages with bug fix and security updates. Specifically, RHN uses an HTTPS auto-generated, temporary URL, and the subscription-manager plugin provides access to this URL. The **subscription-manager** command is used to register with RHN before updates can be obtained. RHN configuration of Yum, and per channel modifications to the update module, can be made in the `/etc/yum/pluginconf.d/subscription-manager.conf` file.

yum-plugin-changelog

The changelog plugin can be used to view package changelogs before or after updating the system.

YUM Repositories

Creating your own YUM repository

- Obtain rpm packages
- Generate package metadata
- Make your repo available via HTTP or FTP

Maintain your YUM repository

- Re-generate package metadata when packages are updated
- Save disk space by deleting old versions of packages

Why Create YUM Repositories?

Software distribution can be vastly improved and simplified when paired with a good package management solution. Combining a good package management solution with great dependency resolution leads to (almost) pain-free software management.

If managing a number of computers running the same LSB compliant Linux distribution version, creating an on-site YUM repository is extremely useful to distribute a common set of packages to all systems. It becomes even more useful when you create custom packages, or develop your own software.

Creating YUM Repositories

Creating a YUM repository is fairly straightforward. The first step is to acquire the needed RPM package files. Mirroring a distro's FTP or HTTP software download site is a common way of getting a good start.

Place all the packages for a specific distribution version in the same directory. Different distributions, and versions, can be served by the same YUM repository by creating a separate directory for each.

The second step is to generate the package metadata that the **yum** (or distribution specific) command will download to find out which packages are served by this repository. This is done using the **createrepo** command. The **createrepo** command will strip out each package's header, generate a sub-directory named repodata/ containing the generated metadata.

The last step in creating a repository is to share this directory to the network. An example of how to create a YUM repository follows:

```
# cp *.rpm /srv/repository/  
# cd /srv/repository/  
# createrepo .  
... output omitted ...
```

Share the directory created above via FTP, HTTP, or NFS.

Some Useful Options

Useful **createrepo** command options include:

--checksum ⇒ Select checksum format used in repodata/ XML files.
-x ⇒ Exclude a file from being included during repository creation.
-g ⇒ Specify an XML file to define groups.

Repo Maintenance

When packages are added or removed from a repository, regenerate the metadata by re-running **createrepo**. Over time, a repository can accumulate multiple versions of the same packages if the normal workflow is to simply add new packages to the repository. The **repomanage** command can be used to obtain a list of old packages in a repository, where "old" means a newer version of the same package exists in the repository. The list of "old" packages can be then be deleted, for example:

```
# repomanage --old /srv/repository | xargs rm -f
```

YUM Repository Groups

YUM Repository Groups

- Groups are a collection of packages defined by the repository:
 - Ease of package installation
 - Ease of package updates
- Any RPM dependencies must be provided by repository
 - Dependencies are not defined in a YUM group

Creating Group XML File

- By hand
- Using the `yum-groups-manager` command

The following would be the resulting XML file:

File: MyGroups.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE comps PUBLIC "-//Red Hat, Inc./DTD Comps info,
//EN" "comps.dtd">
<comps>

<group>
<id>MikeMix</id>
<default>false</default>
<uservisible>true</uservisible>
<display_order>1024</display_order>
<name>TheMikeMix</name>
<description>The Mike Mix</description>
<packagelist>
    <packagereq type="mandatory">WindowMaker</packagereq>
    <packagereq type="mandatory">joe</packagereq>
    <packagereq type="mandatory">httpd</packagereq>
</packagelist>
</group>
</comps>
```

`yum-groups-manager` relies on the ability to use an existing and working `yum` configuration.

`yum-groups-manager` --id=MikeMix --mandatory WindowMaker,

joe --optional httpd --description "The Mike Mix" --save MyGroups.xml

Compiling/Installing from Source

In general, to compile from source code:

- extract the source from archive (decompressing, if necessary)
- read installation documentation
- select compile time options and run the **configure** script
- run **make** to compile
- **su** - to root
- run **make install** (usually, as root) to install software

Compiling and Installing from Source

Much of the Linux software distributed as source code uses tools called **autoconf** and **automake**. The actual steps involved to compile a complex program completely by hand can be enormous. To help aid in compiling software, the GNU team have produced several programs that automate the process. The compile process shown here assumes that the software is using these tools.

Extract the source; often the source is packaged as a compressed tarball:

```
$ tar -zxf foo-1.2.tar.gz
```

When you unpack the tarball, it will usually create a new sub-directory and place its files there. This directory should be in the \$HOME directory of the administrator, or in the rare occasion that the source should be readable to other system users, in the /usr/local/src/ directory. The /usr/src/ directory is common for Linux kernel source, and source code provided by vendors. Change to the new sub-directory and read any installation documentation found there. By convention, typical files used are README, or INSTALL (note the upper-case filenames).

```
$ cd foo-1.2/  
$ less README INSTALL
```

Now select the desired compile-time options and build the needed Makefiles. Software based on the **autoconf** system will have often have a **configure** script in the source root directory. You can see

what compile time options the software supports, and then run the **configure** script with the needed options. This will cause the **autoconf** generated **configure** script to create the needed Makefiles based on your supplied information.

```
$ ./configure --help  
... output omitted ...
```

Note the available options, especially those listed last. With some software, a separate build directory will need to be used to keep the source tree pristine.

```
$ ./configure --prefix=/usr/local options  
... output omitted ...
```

Run **make** in the root source directory. **make** will look for the master Makefile created in the previous step, and call the compiler and linker as specified by the Makefile. Lower level Makefiles will be recursively processed as the compile proceeds. Depending on the quantity of code to be compiled, this step may take a long time and can place significant load on the system.

```
$ make  
... output omitted ...
```

For software to be available to other users, it must be installed as root, and should be installed to the /usr/local/ filesystem. (Install to \$HOME, or a sub-directory of \$HOME, for personal use.) Since you need to install the software from within the source root directory, it may be convenient to run **su** without the - option. Though this will almost always work, it is still considered a better practice to use **su -**. After, run **make** using the **install** target to install the file to the appropriate locations on the system.

```
$ su -
# cd /path/to/source/build/directory/
# make install
. . . output omitted . . .
```

Manually Installed Shared Libraries

Check for library dependencies with the `ldd` command

`ld-linux.so` locates libraries to link to

- /etc/ld.so.conf
- /etc/ld.so.conf.d/*.conf
- /etc/ld.so.cache

Run `ldconfig` command after installation of new libraries

Installing Shared Libraries

On Linux systems shared libraries are normally provided via package files. In this case, the pre and post installation scripts contained within the package take care of running `ldconfig` and making additions to /etc/ld.so.conf file, or /etc/ld.so.conf.d/ directory as appropriate. If you are compiling libraries from source or manually copying precompiled libraries onto the system, then you must take care of these steps manually.

Identifying What Libraries a Program Uses

Most programs on the system build on functions found in libraries. When these programs are compiled, they can be either statically linked so that any used functions are included in the final binary, or the linker can leave these references to external functions unresolved. Programs that contain unresolved references to external functions are said to be dynamically linked. When a dynamically linked program is executed, the system must resolve any references to linked libraries—loading the libraries into memory if needed and passing the program a reference to the memory location where the library has been loaded. You can view a list of the libraries needed by a dynamically linked program, and which library files will be used to resolve those needs by running the `ldd` command:

```
$ ldd /bin/cp
    libacl.so.1 => /lib/libacl.so.1 (0x00da7000)
    libselinux.so.1 => /lib/libselinux.so.1 (0x0088d000)
. . . snip . . .
```

Configuration for the Dynamic Linker

Although standard locations exist, libraries can be installed onto the system into arbitrary directories. When a dynamically linked program is executed, the `ld-linux.so` program must locate the needed libraries. The /etc/ld.so.conf file, and `LD_LIBRARY_PATH` variable, lists the directories that will be checked for libraries. Files with a .conf extension, found in the /etc/ld.so.conf.d/ directory, also list library directories. Configuration files have one directory per line. If you install libraries into a directory not so listed, then `ld-linux.so` will not be able to find and use the new library.

Installing New Libraries

To install new libraries manually, perform the following steps:

1. Copy to the library into one of the standard library directories (/lib/, /usr/lib/, etc.) or optionally update the /etc/ld.so.conf or create a .conf in /etc/ld.so.conf.d/ to include your custom library directory.
2. Execute the `ldconfig` program to rescan the library directories and rebuild the /etc/ld.so.cache file (the cache file is used to speed run-time linking).

Rebuilding Source RPM Packages

Source RPMs contain the source code of the application, with information about how to install the software

Source RPMs can be compiled and installed directly, or can be used to produce the final RPM:

```
$ rpmbuild --rebuild package_name-version-release.src.rpm  
$ rpm -U package_name-version-release.src.rpm
```

The resulting RPM from a --rebuild can then be installed:

```
# rpm -U package_name-version-release.x86_64.rpm
```

The RPM Building Directory Structure

To create RPMs, a certain directory structure needs to be available:

Directory	Content
BUILD/	Where source tarball is extracted, patches applied and compilation takes place.
SOURCES/	Contains pristine source tarball as well as any patches and other ancillary files.
SPECS/	Contains <i>package.spec</i> file that describes and controls RPM build. The spec file also contains post/pre install/uninstall scripts.
RPMS/	Contains sub-directories, <i>athlon/</i> , <i>i386/</i> , <i>i486/</i> , <i>i586/</i> & <i>noarch/</i> . Freshly built binary RPMs placed in these directories.
SRPMS/	Freshly built <i>package_name-version-release.src.rpm</i> placed here.

The macro container files that macros are read from start with the RPM default /usr/lib/rpm/macros file, then the local system override file /etc/rpm/macros is consulted (if it exists), and finally, the per-user macro file is read, \$HOME/.rpmmacros (if it exists). Any macro found in the per-user macro file will override the same macro in the local system override file, and any macro found in the local system override file will override macros from the RPM default file.

[R7] The following applies to RHEL7 only:

On Red Hat Enterprise Linux, this can be automated with the **rpmdev-setuptree** command, part of the **rpmdevtools** package. RPM defaults to building packages in the user's (including root's) home directory under `~/rpmbuild`, and will create the directory structure if it does not already exist.

[S12] The following applies to SLES12 only:

On SUSE Linux Enterprise Server, RPM is configured to support RPM builds by root users within the `/usr/src/packages/` sub-directory. However, since building packages as root is not recommended, instead, configure a new build directory in the non-root user's home directory:

```
$ mkdir -p $HOME/rpmbuild/{BUILD,{,S}RPMS,SOURCES,SPECS}  
$ echo "%_topdir $HOME/rpmbuild" >> $HOME/.rpmmacros
```

Creating Binary RPMs from a .src.rpm

When building executable RPMs from source RPMs, some preparatory steps on the build system is required. The software needed for compilation needs to be installed, such as the GNU C Compiler, the **make** command, and the autoconf and automake tools, and of course the **rpmbuild** package. In addition to development tools, most software has build dependencies which must be present when the software is being compiled. Many applications require support libraries and header files which must be present when they are compiled. For this reason, it is best to compile software on the same operating system release as the system on which it will be installed, though cross-compiling is an option.

A *package_name-version-release.src.rpm* file contains the spec file and files for the SOURCES/ directory.

There are two ways to invoke building of source packages with **rpmbuild**. For both options, when the build has completed, the build directory is removed and then the sources and spec file for the package are removed:

- recompile** ⇒ install source package then prep, compile, and create a new source package
- rebuild** ⇒ same as above but also produces an RPM that can be installed, providing executables, libraries, etc.

The RPMFind website (<http://www.rpmfind.net/>) provides a wide variety of source and binary RPMs for most redistributable software packages.

[R7] *The following applies to RHEL7 only:*

The **yum-builddep** command can be used to download packages necessary to build a source RPM.

Building RPMs from a spec file

Some developers release their source code with a spec file. These can be used to rebuild a binary RPM much like from an SRPM file. To build RPMs from such a tarball, use the **-ta** option to the **rpmbuild** command. For example:

```
$ rpmbuild -ta /path/to/package.tar.bz2
```

Instead of with **-ta**, the spec file can be referenced directly, assuming the tarball is in the current working directory:

```
$ rpmbuild -ba /path/to/foo.spec
```

To modify a source RPM, first install it:

```
$ rpm -Uvh /path/to/name-ver-rel.src.rpm
```

The upstream source code, patches, and supporting files will be placed in the SOURCES directory, the spec file in the SPECS directory.

Lab 4

Estimated Time:
S12: 55 minutes
R7: 55 minutes

Task 1: Managing Software with RPM

Page: 4-20 Time: 10 minutes

Requirements:  (1 station)  (classroom server)  (graphical environment)

Task 2: Creating a Custom RPM Repository

Page: 4-24 Time: 15 minutes

Requirements:  (1 station)  (classroom server)

Task 3: Querying the RPM Database

Page: 4-28 Time: 5 minutes

Requirements:  (1 station)  (classroom server)

Task 4: Installing Software via RPM & Source and Rebuilding SRPMs

Page: 4-32 Time: 15 minutes

Requirements:  (1 station)  (classroom server)

Task 5: Troubleshooting Practice: Package Management

Page: 4-36 Time: 10 minutes

Requirements:  (1 station)

Objectives

- ❖ Answer a few questions about the system using RPM queries.
- ❖ Install zsh using RPM.
- ❖ Troubleshoot and repair a package using RPM verification.

Requirements

- █ (1 station) █ (classroom server) X (graphical environment)

Relevance

A core duty shared by nearly all system administrators is managing what software is installed on the system. This lab provides practice in using the RPM commands to see what has been installed on the system, verifying that installed software is functional, and installing or removing other software as needed.

- 1) Explore what packages are installed on your system using the rpm query functions. List all packages currently installed on the system:

```
$ rpm -qa | less  
. . . output omitted . . .
```

Package names are returned in database order by default, but can be sorted as needed.

- 2) What package provided the file /bin/ed?

```
$ rpm -qf /bin/ed  
. . . output omitted . . .
```

- 3) List detailed information and files installed by the ed package:

```
$ rpm -qil ed | less  
. . . output omitted . . .
```

- 4) Display a list of only the documentation files provided by the ed package:

```
$ rpm -qd ed  
. . . output omitted . . .
```

Lab 4

Task 1

Managing Software with RPM

Estimated Time: 10 minutes

- 5) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 6) [R7] *This step should only be performed on RHEL7.*

Import the RPM GPG keys so that packages to be installed will be authenticated properly.

```
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY*
```

- 7) Mount the network share that contains the installation packages:

```
[R7] # mount server1:/export/netinstall/RHEL7 /mnt/  
[S12] # mount server1:/export/netinstall/SLES12 /mnt/
```

- 8) Attempt to install a single RPM package using the NFS mounted share. Observe the RPM dependency checking in action.

```
[R7] # cd /mnt/Packages/  
[R7] # rpm -ivh amanda-[0-9]*.rpm  
[S12] # cd /mnt/suse/x86_64/  
[S12] # rpm -ivh fetchmailconf*.rpm  
error: Failed dependencies:  
... snip ...
```

The command fails with dependency errors.

- 9) Try installing the package again, this time specifying additional packages to satisfy the dependencies:

```
[R7] # rpm -ivh amanda-[0-9]*.rpm amanda-libs*.rpm  
[S12] # rpm -ivh fetchmail-*.*.rpm python-tk*.rpm • Catches both fetchmail and fetchmailconf packages.  
Preparing... ####### [100%]  
Updating / installing...  
... snip ...
```

10) Install the zsh package:

```
# rpm -ivh zsh-*x86_64.rpm  
Preparing...  
1:zsh
```

[100%]
[100%]

11) Verify that none of the files provided by the zsh package have changed. Since the zsh package was installed in the previous step, rpm will not print any errors (i.e. no output):

```
# rpm -V zsh
```

12) Modify several of the files provided by the zsh package:

```
# touch /bin/zsh  
# chown mail: /etc/zshenv  
# rm -f /etc/zshrc
```

- Change just the time stamps for the /bin/zsh.
- Change the user and group ownership.
- Delete the file.

13) Run the RPM verify command again to see the reported changes:

```
# rpm -V zsh  
.....T. /bin/zsh  
.....UG.. c /etc/zshenv  
missing c /etc/zshrc
```

- Definitions for the listed attributes can be found in the rpm(8) man page.

14) Use the rpm command with the **--replacepkgs** option to repair the corrupted zsh files:

```
# rpm -Uvh --replacepkgs zsh-*x86_64.rpm  
Preparing...  
1:zsh
```

[100%]
[100%]

15) Run the RPM verify command one final time to confirm that the files which were modified above are replaced with the original files:

```
# rpm -V zsh  
. . . output omitted . . .
```

Clean Up

- 16) Run the `umount` command to unmount the filesystem at `/mnt/`:

```
# umount /mnt/
```

- 17) Administrative privileges are no longer required; `exit` the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- >Create a custom RPM repository for installing software

Requirements

- (1 station) (classroom server)

Relevance

Creating your own RPM repository gives you an easy way to install and update RPM packages not included in your Linux distribution.

- The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- Install the Apache web server and createrepo packages (required in later Steps):

```
[R7] # yum install -y httpd createrepo  
[S12] # zypper install -y apache2 createrepo  
     . . . snip . . .
```

- Creating a custom repository can be enormously helpful in installing and maintaining a set of packages across multiple machines. Start this process by populating the RPM directory:

```
[R7] # mkdir -p /var/www/html/myrepo/  
[R7] # cd /var/www/html/myrepo/  
[S12] # mkdir -p /srv/www/htdocs/myrepo/  
[S12] # cd /srv/www/htdocs/myrepo/  
# cp /labfiles/rndcmd*.rpm ./  
# ls  
rndcmd-1.0-1.noarch.rpm
```

Lab 4

Task 2

Creating a Custom RPM Repository

Estimated Time: 15 minutes

- Create the repository using the createrepo command:

```
# createrepo ./  
     . . . snip . . .  
Saving Primary metadata
```

- If the rndcmd RPM is not found, verify you ran the above copy command.

- The . is used to refer to the directory holding the RPMs (in this case, the current directory). Strictly speaking, the / isn't needed and is usually not used.

Saving file lists metadata

Saving other metadata

The repodata/ sub-directory is created.

- 5) Examine the XML files that contain the RPM header information and other package metadata:

```
# ls -al repodata/
total 56
drwxr-xr-x 2 root root 4096 Jul 21 15:00 .
drwxr-xr-x 3 root root 4096 Jul 21 15:00 ..
-rw-r--r-- 1 root root 4579 Jul 21 15:00 2af12232-primary.xml.gz
-rw-r--r-- 1 root root 4406 Jul 21 15:00 976ad6c2-primary.sqlite.bz2
-rw-r--r-- 1 root root 1559 Jul 21 15:00 a5fa32ce-filelists.sqlite.bz2
-rw-r--r-- 1 root root 259 Jul 21 15:00 ad98821c-other.xml.gz
-rw-r--r-- 1 root root 659 Jul 21 15:00 b15e211a-other.sqlite.bz2
-rw-r--r-- 1 root root 859 Jul 21 15:00 bca6709d-filelists.xml.gz
-rw-r--r-- 1 root root 951 Jul 21 15:00 repomd.xml
```

- It is important for the XML files to be regenerated with the createrepo command whenever the RPM packages in the repository are changed, deleted, or new ones are added. Note that names may be prepended with a checksum hash.

- 6) Start Apache to serve the content:

```
[R7] # systemctl restart httpd
[S12] # systemctl restart apache2
. . . output omitted . . .
```

If stopping a service fails, it is usually because the service was not running. If that happens here, it is safe to ignore the failure. The important part is that the service starts successfully.

- 7) [R7] This step should only be performed on RHEL7.

Create a repo file pointing at your new repository:

File: /etc/yum.repos.d/myrepo.repo
+ [myrepo] + name=My yum repository + baseurl=http://localhost/myrepo/ + enabled=1 + gpgcheck=0

8) [S12] This step should only be performed on SLES12.

Add your new repository:

```
# zypper ar http://localhost/myrepo myrepo
Adding repository 'myrepo' [done]
. . . output omitted . . .
```

9) [S12] This step should only be performed on SLES12.

Because SUSE Linux Enterprise Server/Desktop by default expects signed packages, and the RPM in our newly created repository is not signed, disable GPG signature checking for the repo:

File: /etc/zypp/repos.d/myrepo.repo

[myrepo]
name=myrepo
enabled=1
+ gpgcheck=0

10) List the available packages in your repository:

```
[R7] # yum list available | grep myrepo
[R7] rndcmd.noarch           1.0-1                  myrepo
[S12] # zypper packages | grep myrepo
[S12] . . . snip . . .
[S12] | myrepo      | rndcmd | 1.0-1      | noarch
```

11) Install rndcmd, a utility to select "lucky" commands.

```
[R7] # yum install -y rndcmd
[R7] . . . snip . . .
[R7] Complete!
[S12] # zypper install -y rndcmd
[S12] . . . snip . . .
[S12] (1/1) Installing: rndcmd-1.0-1 .....[done]
```

- 12) Verify that rndcmd installed correctly by running it:

```
# rndcmd  
Your lucky command is:  
funzip
```

- rndcmd is random. Example output will likely not match. Run the command again to get a new result.

Cleanup

- 13) Make sure that your new repository doesn't cause problems with future lab tasks by removing it:

```
[R7] # rm -f /etc/yum.repos.d/myrepo.repo  
[S12] # zypper rr myrepo  
[S12] Removing repository 'myrepo' ..... [done]  
[S12] Repository 'myrepo' has been removed.
```

- 14) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Use the rpm program to query the system for information about locally installed packages and to query package files.
- ❖ Use the queryformat option of rpmquery to control the output of more advanced queries

Requirements

- █ (1 station) █ (classroom server)

Relevance

Because all software on RPM-based Linux distributions is managed via rpm, it is useful to be able to query the RPM database and determine what is installed. This lab task guides you through common rpm queries.

- 1) Use rpm to view a list of all RPMs installed on the system:

```
$ rpm -qa | sort | less  
. . . output omitted . . .
```

- 2) View a list of the last 20 packages installed and when they were installed:

```
$ rpm -qa --last | head -n 20  
rndcmd-1.0-1  
Mon 14 Feb 2011 10:50:10 AM MST  
. . . snip . . .
```

- 3) Get a list of all the kernel packages installed on your system:

```
$ rpm -qa kernel\*  
. . . output omitted . . .
```

- 4) View the changelog for the bash package:

```
$ rpm -q --changelog bash | less  
. . . output omitted . . .
```

- 5) Download a package file from server1 using the wget program:

```
$ wget -P /tmp ftp://server1/pub/antiword-*x86_64.rpm  
. . . output omitted . . .
```

Lab 4

Task 3

Querying the RPM Database

Estimated Time: 5 minutes

6) Query the RPM package file you just downloaded (without installing it).

Please note that the presence of a warning message indicates that the appropriate GPG public key has not been imported into the local RPM database.

```
$ rpm -qip /tmp/antiword*.rpm
warning: /tmp/antiword-0.37-9.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY
Name        : antiword                         Relocations: (not relocatable)
Version     : 0.37                               Vendor: Fedora Project
Release     : 9.el6                            Build Date: Tue 20 Mar 2012 01:58:20 PM PDT
Install Date: (not installed)                  Build Host: x86-12.phx2.fedoraproject.org
Group       : Applications/Text                 Source RPM: antiword-0.37-9.el6.src.rpm
Size        : 627342                           License: GPLv2+
Signature   : RSA/8, Sat 24 Mar 2012 09:20:10 AM PDT, Key ID 3b49df2a0608b895
Packager    : Fedora Project
URL         : http://www.winfIELD.demon.nl/
Summary     : MS Word to ASCII/Postscript converter
Description :
Antiword is a free MS-Word reader for Linux, BeOS and RISC OS. It converts
the documents from Word 6, 7, 97 and 2000 to ASCII and Postscript. Antiword
tries to keep the layout of the document intact.
```

```
$ rpm -qlp /tmp/antiword*.rpm | less
warning: /tmp/antiword-0.37-9.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY
/usr/bin/antiword
/usr/bin/antiword.bin
/usr/share/antiword
/usr/share/antiword/8859-1.txt
. . . snip . . .
```

7) Install the Antiword binary RPM downloaded earlier:

```
$ su -c "rpm -Uvh /tmp/antiword*.rpm"
Password: makeitso 
. . . output omitted . . .
```

8) Use the rpm command to list the files in the installed package and locate the correct binary:

```
$ rpm -ql antiword | grep bin
```

```
/usr/bin/antiword  
/usr/bin/antiword.bin
```

Using queryformat for Custom Reports (Bonus)

- 9) Build a sorted list of the number of packages for each license type (for all packages installed on your system):

```
$ rpm -qa --queryformat "%{LICENSE}\n" | sort | uniq -c | sort -rn  
[R7] 239 GPLv2+  
[R7] 228 GPLv2+  
[R7] 116 GPL+ or Artistic  
[R7] 114 MIT  
[R7] 64 BSD  
[R7] 58 GPLv3+  
[R7] 57 GPLv2  
[S12] 396 GPL v2 or later  
[S12] 91 LGPL v2.1 or later  
[S12] 83 GPL v2 or later; LGPL v2.1 or later  
[S12] 69 BSD 3-Clause  
[S12] 62 X11/MIT  
[S12] 25 GPL v2 only  
[S12] 23 Artistic License  
[S12] 16 GPL v3 or later  
. . . snip . . .
```

- 10) List all packages with size 20MB or greater, sorted by size, and including summary:

```
$ rpm -qa --queryformat "%{SIZE} %{NAME}: %{SUMMARY}\n" | sort -rn | awk '($1 >= 20*1024*1024)'  
[R7] 132628448 kernel: The Linux kernel  
[R7] 119480713 glibc-common: Common binaries and locale data for glibc  
[R7] 93937919 java-1.7.0-openjdk-headless: The OpenJDK runtime environment without audio and video support  
[R7] . . . snip . . .  
[S12] 149170791 MozillaFirefox-translations: Translations for MozillaFirefox  
[S12] 140374881 kernel-default: The Standard Kernel  
[S12] 119762706 glibc-locale: Locale Data for Localized Programs  
[S12] . . . snip . . .
```

- 11) Query the RPM database and generate an ls -l style listing for the specified files showing: permissions, ownership, size, and install date. Note the use of alternate format types (perms, and date):

```
$ for file in /bin/*
> do
> (rpm -qf $file --qf "%{FILEMODES:perms} %{FILEUSERNAME} \
>   %{FILEGROUPNAME} %10{FILESIZES} %{INSTALLTIME:date} "; echo $file)
> done | less
. . . output omitted . . .
```

Objectives

- Install software via binary RPMs, source RPMs and source code.

Requirements

- (1 station) (classroom server)

Relevance

Although modern Linux distributions ship a lot of software, it may be necessary to install third party software packages. Several distinct packaging solutions can be used to provide software on Linux. This lab covers the most common solutions.

- Because of the potential to damage the system, software should always be compiled as a non-root user. Verify that you are logged in as the guru user:

```
$ id -un  
guru
```

- Download prerequisite files to the guru user's home directory:

```
$ cd  
$ wget ftp://server1/pub/vnstat-1.13.tar.gz  
... output omitted ...  
$ wget ftp://server1/pub/cowsay-3.03-10.src.rpm  
... output omitted ...
```

- Install packages needed for rebuilding and compiling steps done later in this lab task:

```
[R7] $ su -c "yum install -y rpm-build gcc"  
[R7] Password: makeitso [Enter]  
[R7] ... output omitted ...  
[S12] $ su -c "zypper install -y gcc rpm-build"  
[S12] Password: makeitso [Enter]  
[S12] ... output omitted ...
```

Lab 4

Task 4

Installing Software via RPM & Source and Rebuilding SRPMs

Estimated Time: 15 minutes

Building an RPM from a source RPM file

- 4) [R7] This step should only be performed on RHEL7.

Create a new file with the required configuration options to turn off the automatic creation of debuginfo packages:

```
File: ~/.rpmmacros
+ %debug_package %{nil}
```

- 5) [R7] This step should only be performed on RHEL7.

Use the yum-builddep utility to inspect the cowsay source RPM to see what build requirements it has, and install those requirements:

```
$ su -c "yum-builddep --nogpgcheck -y cowsay-3.03-10.src.rpm"
Password: makeitso 
. . . output omitted . . .
```

- 6) Use rpmbuild to rebuild the cowsay package from source RPM:

```
$ rpmbuild --rebuild cowsay-3.03-10.src.rpm
. . . output omitted . . .
```

- The --target option can optimize for a specific architecture.

Near the end of the output there is a line that starts with "Wrote:"; following it is the full path to the newly created binary RPM.

- 7) Install the newly rebuilt RPM:

```
[R7] $ su -c "rpm -ivh /home/guru/rpmbuild/RPMS/noarch/cowsay-3.03-10.el7.noarch.rpm"
[S12] $ su -c "rpm -ivh /home/guru/rpmbuild/RPMS/noarch/cowsay-3.03-10.noarch.rpm"
Password: makeitso 
Preparing... ####### [100%]
Updating / installing...
 1:cowsay-3.03-10 ####### [100%]
```

- 8) The cowsay can be run interactively, or similar to the echo command:

```
$ cowsay hello, world
. . . output omitted . . .
```

Build from upstream source

- 9) Extract vnstat source:

```
$ tar -xzf vnstat-1.13.tar.gz
```

- 10) By convention, software distributed in a source tarball usually will have a text file called README or INSTALL which contain instructions on how to build and install that software. Briefly look at the contents of these files:

```
$ cd vnstat-1.13/  
$ less README  
$ less INSTALL
```

• Type **[q]** to quit.

- 11) Compile the new software by running make in the current directory. This will make the appropriate calls to the compiler and linker programs to compile the software:

```
$ make  
... output omitted ...
```

GNU programs often require the use of autoconf and automake. These tools build the makefile from a template. The template is generated from a script called configure, (and often the configure script is generated from a template). The vnstat package builds from a simpler Makefile.

- 12) For packagers, the vnstat package has a DESTDIR variable that allows the package to be installed to an alternate location. Install the package to the /tmp/ directory to see what files are installed:

```
$ make install DESTDIR=/tmp/vnstat  
... output omitted ...  
$ ls -R /tmp/vnstat/  
... output omitted ...
```

- 13) The following actions require administrative privileges. **Switch** to a root user login shell, but preserve the current working directory:

```
$ su  
Password: makeitso [Enter]
```

• We want to switch to root and stay in guru's home directory. Do not use the - option to su here.

- 14) As root, perform the final step of installation:

```
# make install  
. . . output omitted . . .
```

- 15) Initialize a database for vnstat, then use vnstat to listen on the ethernet interface:

```
# vnstat --create -i eth0  
# vnstat -l -i eth0
```

• *eth0* may not be your default network interface. Change the interface as needed.

vnstat will monitor bandwidth usage by interface. Leave the vnstat command running.

- 16) Open a new terminal and generate network activity by recursively downloading from the classroom FTP server:

```
$ wget -r -O /dev/null ftp://server1/pub  
. . . output omitted . . .
```

Type **Ctrl+C** to stop the vnstat command in the first window.

Clean Up

- 17) As root, clean up the promiscuously installed, unpackaged program:

```
# make uninstall  
. . . output omitted . . .
```

• Not all Makefiles include an `uninstall` directive. Fortunately, this one does.

Bonus

- 18) [R7] *This step should only be performed on RHEL7.*

Install the Ltris game from sources. This will require installing its build dependencies: `recode-devel`, `SDL-devel`, and `libXp`. It will also require the use of the `configure` script to build the Ltris Makefile: run `./configure --help` inside the `ltris` build directory for details. The tarball can be downloaded from <ftp://server1/pub/ltris-1.0.1.tar.gz>. The resulting `ltris` command requires the X Window System.

Objectives

❖ Practice troubleshooting common RPM related issues.

Requirements

☒ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You use these scripts to break your system in controlled ways, and then you troubleshoot the problem and fix the system.

- 1) Use tsmenu to complete the networking troubleshooting scenario:

Troubleshooting Group	Scenario Category	Scenario Name
Group 3	Package Management	rpm-01.sh

Lab 4

Task 5

Troubleshooting Practice: Package Management

Estimated Time: 10 minutes

Content

Partitioning Disks with fdisk & gdisk	2
Resizing a GPT Partition with gdisk	5
Partitioning Disks with parted	8
Non-Interactive Disk Partitioning with sfdisk	9
Filesystem Creation	10
Persistent Block Devices	11
Mounting Filesystems	12
Resizing Filesystems	14
Filesystem Maintenance	15
Managing an XFS Filesystem	18
Swap	20
Filesystem Structures	21
Determining Disk Usage With df and du	22
Configuring Disk Quotas	23
Setting Quotas	24
Viewing and Monitoring Quotas	25
Filesystem Attributes	26
Lab Tasks	27
1. Creating and Managing Filesystems	28
2. Hot Adding Swap	35
3. Setting User Quotas	37

Chapter

5

LOCAL STORAGE ADMINISTRATION

Partitioning Disks with **fdisk** & **gdisk**

fdisk/gdisk

- Edits in memory copy. Only writes to disk when instructed
- **fdisk** — Can create and edit DOS/MBR style partition tables
DOS/MBR created in 1983 supporting 2TB max disk
- **gdisk** — same UI as **fdisk** but works with GPT partition tables
Can convert MBR partition tables into GPT tables
- Doesn't always update kernel structures
use **partprobe**, **partx**, or **addpart** and **delpart**
May require a reboot

sfdisk/gdisk

- Non-interactive, Script-able via command line options/params

Supported Partition Tables

By default, Linux uses standard DOS-style partition tables. However, a wide variety of other partition table formats are supported, so it can be installed in conjunction with other OSes, or use data disks which were partitioned under other operating systems. Additional partition types currently supported include GPT, Acorn, OSF, Amiga, Atari, Macintosh, BSD, Minix, Solaris, Unixware, SGI, Ultrix, and Sun partition schemes.

DOS/MBR Style Partition Table

Long the default, this style of partition table can only address disks up to 2TB in size. It also requires the use of a special work around in the form of an Extended partition table for anything beyond 4 partitions per disk, (once restricted to 15 (SCSI) and 63 (EIDE) logical drives). Each partition is assigned a type code to help identify what it contains. The following partition type codes are used by Linux when using a DOS style partition table:

83 ⇒ Linux Filesystem
82 ⇒ Linux SWAP
fd ⇒ Software RAID
8e ⇒ LVM Physical Volume
5 ⇒ Extended partition
ee ⇒ GPT protective partition

Deleting the Partition Table

With a DOS/MBR style partition table, it is sufficient to erase the first 512 bytes of a device using **dd** which deletes both the MBR boot code as well as the partition table. However, GPT partition structures occupy the first 17 KiB of the device and a backup exists starting 16 KiB before the last sector of the disk. Thus wiping only the first part of the device is not sufficient. The easiest way to remove partition tables is to use the **-Z** option to **sgdisk** which erases both DOS/MBR and GPT tables.

```
# sgdisk -Z /dev/sda
GPT data structures destroyed!
```

Alternatively, using **dd**:

```
DOS/MBR# dd if=/dev/zero of=/dev/sda count=1 bs=512
GPT (primary)# dd if=/dev/zero of=/dev/sda count=34 bs=512
GPT (backup)# dd if=/dev/zero of=/dev/sda bs=512-
seek=$((`blockdev --getsize /dev/sda` - 33))
```

Creating Partitions with **fdisk**

The **fdisk** program has an interactive interface. The following example shows defining 3 partition records on a new disk. The first partition is a 200M primary partition of type code 83 (the default). The second is an extended partition that fills the rest of the disk. The third is a 20G logical partition inside the extended partition:

```
# fdisk /dev/sdb
. . . snip . .
Command (m for help): n
Command action
  p  primary partition (0 primary, 0 extended, 4 free)
  e  extended
Select (default p): p
Partition number (1,2,3,4, default 1): 1
First sector (2048-83886079, default 2048): 
Using default value 2048
Last sector , +sectors or +size{K,M,G} (2048-83886079, default 83886079): +200M
Partition 1 of type Linux and of size 200MiB is set

Command (m for help): n
Command action
  p  primary partition (1 primary, 0 extended, 3 free)
  e  extended
e
Partition number (2,3,4, default 2): 2
First sector (411648-83886079, default 411648): 
Using default value 411648
Last sector , +sectors or +size{K,M,G} (2048-83886079, default 83886079): 
Using default value 83886079
Partition 3 of type Extended and of size 39.8 GiB is set

Command (m for help): n
Command action
  p  primary partition (2 primary, 0 extended, 2 free)
  l  logical (5 or over)
l
Adding logical partition 5
First sector (413696-83886079, default 411648): 
Using default value 413696
Last sector , +sectors or +size{K,M,G} (413696-83886079, default 83886079): +20G
Partition 5 of type Linux and of size 20 GiB is set

Command (m for help): t
Partition number (1,2,5, default 5): 5
Hex code (type L to list codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): p
Disk /dev/sdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
```

Units = sectors 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0009da4f

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		2048	411647	204800	83	Linux
/dev/sdb2		411648	83886079	41737216	5	Extended
/dev/sdb5		413696	42356735	20971520	8e	Linux LVM

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.

The kernel still uses the old table.

The new table will be used at the next reboot.

Syncing disks.

partprobe /dev/sdb

Resizing a GPT Partition with gdisk

Scenario: A LUN has been grown on a RAID controller or SAN

- The LUN is using a GPT partition table
- Grow the last partition to include the new space

Perform SCSI rescan so that kernel sees new block device size

Relocate the GPT data structures to the (new) end of the disk

Delete the last partition after noting:

- type, name and starting location

Create new partition with same starting location

- Ending location at the end of available space

Set partition type and name if needed

Save and reboot

Resizing a LUN that has GPT Partition table

Resizing a LUN that contains a GPT partition table, and then resizing the last partition to match, is essentially the same as when using an MBR partition. The added complexity comes from GPT storing an extra copy of the partition table at the end of the disk that needs to be relocated to the new ending of the disk. In the following example, a LUN has just been resized from 2.7 TiB to 3.8 TiB, and the last partition is then resized from 2.7 TiB to 3.8 TiB.

```
# echo 1 > /sys/block/sda/device/rescan  
# gdisk /dev/sda
```

```
GPT fdisk (gdisk) version 0.8.6
```

Partition table scan:

 MBR: protective
 BSD: not present
 APM: not present
 GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p

Disk /dev/sda: 8196980736 sectors, 3.8 TiB

Logical sector size: 512 bytes

Disk identifier (GUID): 072D0A5A-D20E-4F98-84F6-E6E67642721B

Partition table holds up to 128 entries

First usable sector is 34, last usable sector is 5854986206

Partitions will be aligned on 2048-sector boundaries

Total free space is 2014 sectors (1007.0 KiB)

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	411647	200.0 MiB	EF00	EFI System Partition
2	411648	1435647	500.0 MiB	0700	
3	1435648	5854986206	2.7 TiB	8E00	Linux LVM

Command (? for help): **x**

Expert command (? for help): ?

... snip ...

e relocate backup data structures to the end of the disk

... snip ...

m return to main menu

... snip ...

? print this menu

Expert command (? for help): **e**

Relocating backup data structures to the end of the disk

Expert command (? for help): **m**

Command (? for help): **d**

Partition number (1-3): **3**

Command (? for help): **n**

Partition number (3-128, default 3):

First sector (34-8196980702, default = 1435648) or {+-}size{KMGTP}:

Last sector (1435648-8196980702, default = 8196980702) or {+-}size{KMGTP}:

Current type is 'Linux filesystem'

Hex code or GUID (L to show codes, Enter = 8300): **8E00**

Changed type of partition to 'Linux LVM'

Command (? for help): **p**

Disk /dev/sda: 8196980736 sectors, 3.8 TiB

Logical sector size: 512 bytes

Disk identifier (GUID): 072D0A5A-D20E-4F98-84F6-E6E67642721B

Partition table holds up to 128 entries

First usable sector is 34, last usable sector is 8196980702

Partitions will be aligned on 2048-sector boundaries

Total free space is 2014 sectors (1007.0 KiB)

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	411647	200.0 MiB	EF00	EFI System Partition

2	411648	1435647	500.0 MiB	0700
3	1435648	8196980702	3.8 TiB	8E00 primary

Command (? for help): ?

... snip ...

c change a partition's name

... snip ...

Command (? for help): c

Partition number (1-3): 3

Enter name: Linux LVM

Command (? for help): p

Disk /dev/sda: 8196980736 sectors, 3.8 TiB

Logical sector size: 512 bytes

Disk identifier (GUID): 072D0A5A-D20E-4F98-84F6-E6E67642721B

Partition table holds up to 128 entries

First usable sector is 34, last usable sector is 8196980702

Partitions will be aligned on 2048-sector boundaries

Total free space is 2014 sectors (1007.0 KiB)

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	411647	200.0 MiB	EF00	EFI System Partition
2	411648	1435647	500.0 MiB	0700	
3	1435648	8196980702	3.8 TiB	8E00	Linux LVM

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING PARTITIONS!!

Do you want to proceed? (Y/N): Y

OK; writing new GUID partition table (GPT) to /dev/sda.

Warning: The kernel is still using the old partition table.

The new table will be used at the next reboot.

The operation has completed successfully.

systemctl reboot

Partitioning Disks with parted

parted

- Supports many partition table types (including MBR and GPT)
- Writes changes to disk immediately!
- **parted** v2.4+ no longer has filesystem functionality
 - FS-related commands removed: **resize**, **cp**, **mkfs**, **mkpartfs**, **check** and **move**
Use native filesystem specific commands instead
- Graphical version: **gparted**
Uses libparted

The GNU Partition Editor

The GNU **parted** utility is a powerful utility that can create, destroy, resize, and copy partitions. It operates in a paranoid fashion and is designed to avoid data loss during interruptions (such as power failure). From the command line, the **parted** command can be run in interactive, or non-interactive, modes. In addition to the basic command line interface, many GUI front-ends for **parted** exist supporting things such as drag-and-drop style resizing of partitions.

Creating Partitions with parted

parted supports both MBR and GPT disk labels. The **parted** program supports both an interactive and non-interactive interface. The following example shows using the interactive interface to create a new GPT disk label and define 2 partitions. The first partition is just under 3TB in size and will be used as an LVM Physical Volume. The second is 5GB and will be an Ext3 filesystem holding /var/:

```
# parted /dev/sdb
GNU Parted 1.8.1
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel
New disk label type? gpt
(parted) mkpart primary 0 2995G
(parted) mkpart primary 2995G 3000G
(parted) name 1 "Project Blue"
(parted) name 2 "/var"
```

```
(parted) set 1 lvm on
```

```
(parted) print
```

Model: Areca SYSTEM/DATA VOL (scsi)

Disk /dev/sdb: 3000GB

Sector size (logical/physical): 512B/512B

Partition Table: gpt

Disk Flags:

Number	Start	End	Size	File system	Name	Flags
1	17.4kB	2995GB	3000GB		Project blue	lvm
2	2995GB	3000GB	5GB		/var	

```
(parted) quit
```

Information: you may need to update /etc/fstab.

Note how the fdisk program will view this same disk:

```
# fdisk -l /dev/sdb
```

... snip ...

Disk /dev/sdb: 2999.5 GB, 2999599497216 bytes
255 heads, 63 sectors/track, 364680 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb		1	267350	2147483647+	ee	EFI GPT

Non-Interactive Disk Partitioning with **sfdisk**

sfdisk — MBR partition table support

- Non-interactive, Script-able via command line options/params
- List, Create, Delete, and Resize partitions
- Uses stdin for all commands

Use sgdisk for GPT partitioned disks

Delete the 3rd partition from /dev/sda

```
# sfdisk -f -N 3 /dev/sda <<<'0,0,0'
```

Editing a Partition Table Dump

Partitions can also be edited by dumping the current partition table to a text file, editing the dumped file, then loading it back into sfdisk. For example, to create a new 5GiB partition for /dev/sda3:

```
# sfdisk -d /dev/sda > sda-partition-table
```

File: sda-partition-table

/dev/sda1 : start= 2048, size= 1024000, Id=83, bootable
/dev/sda2 : start= 1026048, size= 71680000, Id=8e
- /dev/sda3 : start= 0, size= 0, Id= 0
+ /dev/sda3 : start= 72706048, size= 10485760, Id=83
/dev/sda4 : start= 0, size= 0, Id= 0

```
# sfdisk -f /dev/sda < sda-partition-table
```

Resizing an Existing Partition

With **sfdisk** version 2.26.2 and newer, it is possible to resize partitions by relative adjustments. Assuming there is available free space after the partition, to add 1GiB to /dev/sda3 run:

```
# echo ',+1G,' | sfdisk -N 3 /dev/sda
```

To resize the third partition to the maximum possible size run:

```
# echo ', +' | sfdisk -N 3 /dev/sda
```

The Scriptable Partition Editor – **sfdisk**

The **sfdisk** command is a non-interactive tools for partitioning block devices. Being non-interactive they are controlled via command line options and parameters.

Partitions can be listed with **sfdisk -l**. Other operations such as creating, deleting and resizing partitions are done by sending commands into the command's standard input and using the **-N** to specify the desired partition to manipulate. The standard syntax is: *start-sector, number-of-sectors, partition-type*.

Creating and Deleting Partitions

To create a new partition first the *start-sector* needs to be determined. An easy way to do this is to run **sfdisk -d /dev/sda** and take the *start=* and *size=* numbers from the last partition and add them together. Next determine the *number-of-sectors* which is desired size (in bytes) of the new partition divided by the disks sector size. Since some drives use 512 byte sectors and others use 4096 byte sectors, use the **blockdev** command to get the sector size.

In this example, first determine the number of sectors in 5GiB, then create a new 5GiB partition of type 83 starting at the end of the last partition (previously determined using the technique outlined above):

```
# echo $(( (5*1024*1024*1024) / $(blockdev --getss /dev/sda) ))  
10485760  
# sfdisk -f -u S -N 3 /dev/sda <<<'72706048,10485760,83'
```

Filesystem Creation

Filesystem creation with `mkfs -t filesystem_type`

- Runs commands specific to the filesystem type requested:
`mkfs.ext4`, `mkfs.ext3`, `mkfs.xfs`, `mkfs.btrfs`, `mkfs.msdos`
- `/etc/mke2fs.conf` sets defaults for `mke2fs`

File System Creation

Just as it supports a wide variety of partition types, Linux also supports a variety of different filesystems. After creating a partition with `parted`, if not already created, a filesystem is initialized using `mkfs`.

Most Linux distributions have dropped support for ReiserFS and JFS, (Reiser4 is not supported by the mainline kernel). For distributions with compiled support (see the kernel `/boot/config` file), further information can be found at <http://jfs.sf.net>,
<http://reiser4.sf.net>, and
<http://kernel.org/pub/linux/kernel/people/jeffm/reiserfsprogs/>.

Using `mkfs`

`mkfs` is the standard utility used to create Linux filesystems. By default, `mkfs` will create ext2 filesystems, but can be used to create other filesystem types by using the `-t` option, and specifying the filesystem type to create.

```
# mkfs -t xfs /dev/sdb1
# mkfs -t btrfs /dev/sdb1
```

`mkfs` itself does not actually create any filesystems. Rather, `mkfs` is a wrapper utility which calls filesystem-specific commands. Using `mkfs -t xfs` tells `mkfs` to run `mkfs.xfs`, while using `mkfs -t ext2` (or just `mkfs`, since ext2 is the default) tells `mkfs` to run `mke2fs`.

For specific filesystem options, such as how to adjust the filesystem

block size, consult the man page for the specific filesystem creation utility.

Extended Filesystem Utilities

The ext3 filesystem is essentially just an ext2 filesystem with a journal; the default on-disk filesystem structures of ext2 and ext3 are identical. The ext4 filesystem adds significant scalability features to ext3. Ext2/3/4 filesystems can be created many different ways, using `mkfs`, `mkfs.ext{2,3,4}`, or `mke2fs`. For example, this command would create an ext3 filesystem on the first partition of the first disk by using the journal option `-j` with `mke2fs`:

```
# mke2fs -j /dev/sda1
```

There are other filesystem specific options that can be used to modify the parameters of a filesystem. For example, this would create an ext4 filesystem with exactly 750,000 inodes:

```
# mkfs.ext4 -j -N 750000 /dev/sda1
```

To create an ext4 filesystem, use any of the following commands:

```
# mkfs -t ext4 /dev/sda1
# mkfs.ext4 /dev/sda1
# mke2fs -t ext4 /dev/sda1
```

The configuration file `/etc/mke2fs.conf` is used to store commonly used groups of options for types referenced with `-t`.

Persistent Block Devices

Block Devices use:

- Filesystems
- Swap
- Raw

Block device names may change as a result of:

- Hardware failure or removal
- Software configuration change
- USB, Firewire, PC Card/ExpressCard

Potential persistent naming solutions

- /dev/disk/by-id/name
- UUID
- udev

system, but suffers from the same drawbacks as labels.

The udev system solves this problem at a lower level that is more convenient and elegant. The entire /dev/ directory is managed by udev. Most device nodes are only created when the hardware associated with them is present.

Udev works with the hotplug system so that hotplug storage such as USB, Firewire, or PC Card/Express Card storage devices have consistently named device nodes when such hardware is plugged in. Imagine trying to access files stored on a USB keychain device when you already have your portable music player attached. With udev, you can specify the music player to always be /dev/oggplayer no matter how many or in what order you attached other USB storage devices.

There are also very nice benefits to udev in a SAN environment where a given disk drive is /dev/sdb on one host, but /dev/sdd on another. By using udev in such a setting, all the hosts can have a persistent and common view of the device files.

Changing Block Devices Locations

In the face of OS changes, hardware failures, and hot-swappable storage media, block device files change based on when a device is plugged-in. Consider the scenario where a system with three SCSI drives (sda, sdb, sdc) has a failure in sdb. On reboot, the former sdc will become sdb. When using block devices on multiple USB, Firewire, or PCMCIA storage devices, the order in which they are attached determines where they appear. Regardless of the physical connection, as of kernel 2.6.19 and libata, all storage devices use the SCSI/SATA protocol. The first device attached becomes /dev/sda, the second /dev/sdb, etc.

Persistent Naming Solutions

One approach in solving this problem is to use filesystem labels in the /etc/fstab file. Using filesystem labels does not solve the problem in all cases given that some partitions do not have filesystems such as raw devices used in SAN environments and Oracle database installations. The Linux kernel and **mount** command on Red Hat Enterprise Linux and SUSE Linux Enterprise Server support filesystem labels for all distribution supported filesystems.

A better approach than labels is the Universally Unique Identifier (UUID), originally a proposal of the Open Software Foundation, and now IETF and ISO standards (RFC 4122, ISO/IEC 11578:1996, and ISO/IEC 9834-8:2005). UUIDs provide a 128-bit hash, typically generated at filesystem creation. This significantly reduces the chance of two filesystems having the same label on the same

Mounting Filesystems

mount

- **-o**
- **--bind**

umount

/etc/fstab

- List of mounts and options
- Used during boot

What is mounted or in use?

- /etc/mtab → symlink to /proc/self/mounts
- /proc/mounts → state recorded by kernel
- **fuser** → processes using a mount
- **lsof** → files open on a mount

Mounting Filesystems

A basic task of most administrators is maintaining filesystem layouts. Under Unix, new partitions can be grafted into existing filesystem structures using the **mount** command. Commonly, these mounts are listed in /etc/fstab so they will occur automatically at boot.

On older Linux distributions such as RHEL6, the **mount** and **umount** commands modify a text database, /etc/mtab, which lists all currently mounted partitions. On Linux, the kernel itself also maintains a database, /proc/mounts, which lists all current mounts. Since the kernel is always more aware of what is mounted than **mount** and **umount**, the /proc/mounts file should be considered authoritative. On newer Linux distributions, /etc/mtab is a symlink to /proc/self/mounts so it will never get out of sync with reality.

Mount Options

Mounting options are provided to the kernel at filesystem mount time. Some options are independent of the filesystem being used while others are only available on specific filesystems. Mount options can be specified with **mount -o option, ...**. The **mount** command also reads options from the /etc/fstab file. It is common to see the defaults option used in the fstab file. This has the same effect as setting the following options: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, and `async`. Other options might be useful, such as for performance. For instance, instead of `defaults`, `noatime,data=writeback` disables the access time stamp from being written each time a file is read, and `writeback` writes file metadata to the journal before the block data is

written to the filesystem. See the **mount(8)** man page for a complete list of mount options and other details.

The extended filesystem allows certain mount options (a small subset) to be stored directly in the filesystem superblock. If present, these will be used unless they are explicitly negated by options passed during the mount operation. To enable file access control lists (**acl(5)**) and extended file attributes (**attr(5)**), both common defaults in other modern filesystems, use the **acl** and **user_xattr** mount options with the **tune2fs** command or add the options to /etc/fstab.

[S12] *The following applies to SLES12 only:*

SUSE Linux Enterprise Server relies on the extended filesystem defaults. The **acl** and **user_xattr** mount options must be explicitly added.

Mounting a Filesystem in Multiple Locations

Linux supports mounting a filesystem, or arbitrary directory, to other locations at the same time. This is a powerful technique that allows you to control where directories appear. For example, if you wanted to run the BIND DNS server in a chroot, you could cause the /var/named/ directory, containing the zone files, to also appear in the chrooted directory by running:

```
# mount --bind /var/named /chroot/named/var/named
```

Unmounting Busy Filesystems

When attempting to unmount filesystems, Linux will refuse to unmount any partitions which are currently in use. A common challenge faced by system administrators is figuring out why a filesystem they need to unmount is still considered to be busy by the kernel.

Utilities which can help with this task include:

fuser ⇒ Simple tool which displays the PIDs of processes using the specified files or filesystems.

lsof ⇒ Lists information about files opened by processes. Very full featured and versatile.

Another option is to run **umount -l *mount_point*** which will cause the kernel to unmount the specified filesystem as soon as it is not busy anymore.

The kernel will automatically flush any pending writes to disk before allowing a filesystem to be unmounted. As the root user, the **sync** command can be used at any time to force all in memory data to be flushed to all disks.

Resizing Filesystems

Modify partition sizes

- `parted`
- `fdisk/gdisk`
- `sfdisk/sgdisk`
- SLES12: `cfdisk`

Modify filesystem sizes

- Ext2/3/4 - `resize2fs`
- XFS - `xfs_growfs`

Does not support filesystem reduction

- Btrfs - `btrfs filesystem resize`

Resizing Partitions

In addition to creating and deleting partitions, Linux supports the ability to modify existing partitions and filesystems. Resizing partitions is a non-trivial task requiring extensive knowledge of the partition layout and the filesystem and should not be attempted by the faint of heart. Because of this, the modern day best practice is to place filesystems inside of LVM volumes which are easier to resize.

Resizing ext Filesystems

The command `resize2fs` can be used to resize an ext3/4 filesystem. The `resize2fs` command can be used on unmounted filesystem to grow or shrink the filesystem. Mounted filesystems cannot be shrunk, only grown. The following command will resize the filesystem to a size of 10GB:

```
# resize2fs /dev/sda7 10G
```

The `resize2fs` command can also be used to grow an online (mounted) ext3/4 filesystem. This command would grow the filesystem to its maximum size within the underlying partition or LVM volume (the `-p` option prints out percentage completion bars for visual progress feedback):

```
# resize2fs -p /dev/sda8
```

Resizing XFS Filesystems

XFS filesystems can be grown when mounted, but not reduced, with the `xfs_growfs` command using the mount point as argument. If you are growing the filesystem, you must grow the underlying partition or logical volume first. Using the `-d` option causes the file system to be grown to the maximum size that the underlying device supports.

```
# xfs_growfs -d /dev/sda5
```

Resizing Btrfs Filesystems

The `btrfs filesystem resize` command can be used to grow or reduce an online (mounted) Btrfs filesystem. This command would grow the filesystem to its maximum size within the underlying partition:

```
# btrfs filesystem resize max /dev/sda6
```

Alternatively, you can grow or shrink by a specified amount, for example:

```
# btrfs filesystem resize +size /dev/sda6  
# btrfs filesystem resize -size /dev/sda6
```

Filesystem Maintenance

Viewing filesystem meta-data

- XFS **xfs_info** & Ext2/3/4 **dumpe2fs**

Reconfiguring filesystem

- Ext2/3/4 **tune2fs** highlights
 - set default mount options
 - set auto **fsck** interval and mount count
 - filesystem conversion: ext2 > ext3 or ext3 > ext4
- XFS **xfs_admin** — Parameter adjustment
- XFS **xfs_fsr** — Online Filesystem Defragger

Correcting minor filesystem problems

- **fsck** & **e2fsck/xfs_repair**
- Systemd boot parameter **fsck.mode=force**

power fails to the system. Use the **barrier=1** mount option with the extended filesystem. XFS enables this by default; to disable use the **nobarrier** mount option. If using RAID, it is typically best to leave barriers disabled, as this is managed with a battery backed cache on the RAID controller, (not applicable to software RAID).

Filesystem Labels and UUIDs

The **e2label** and **xfs_admin -l** commands can be used to display what the label is by specifying the raw device for the filesystem:

```
# xfs_admin -l /dev/sda2
label = "/boot"
# e2label /dev/sda6
/usr
```

A new label can be set thusly:

```
# xfs_admin -L newlabel /dev/sdXX
# e2label /dev/sdXX newlabel
```

With XFS a label can be cleared by using "--" as the value for the label. After labels have been set, the **mount** command can be used with labels and it will automatically locate the appropriate underlying device. For example:

```
# mount LABEL=/usr/local /usr/local
```

Filesystem labels may be used in the **/etc/fstab** file. This technique is good when the BIOS or hot-swapping reorders the physical

location of the drives; mounting will still work properly.

Filesystem UUIDs are theoretically unique identifiers for a filesystem. Because of their improved uniqueness compared to labels, the use of UUIDs in the /etc/fstab has become the preferred method. For example:

```
File: /etc/fstab
UUID=6fa884-724b-4aee-8f90-b3bb13fb9 /boot xfs defaults 1 2
```

The **findfs** command searches the disks in the system looking for a filesystem which has a label or UUID matching the specified label or UUID:

```
# findfs LABEL=/boot
/dev/sda1
# findfs UUID=6fa884-724b-4aee-8f90-b3bb13fb9
/dev/sda1
```

If a Linux machine is cloned, it is a good idea to re-generate unique filesystem UUIDs. This can be done with **tune2fs** and **xfs_admin**. Afterwards, make sure all references to the old UUIDs within configuration files are updated to the new UUIDs, especially in /etc/fstab.

```
# xfs_admin -U generate /dev/sdXX
# tune2fs -U random /dev/sdXX
```

Fragmentation

A file's block data is most efficiently accessed when it is in one place. However, as a file's data is updated, it is less likely that the data is written to aligned, contiguous free space. When data is removed, its extents (or blocks) become unallocated, and the free space is therefore fragmented. In general, Linux filesystems are considered to be resistant to fragmentation. However, under certain conditions a filesystem can fragment sufficient to warrant the need to defragment a drive. To identify free space fragmentation on an XFS filesystem, run **xfs_db**:

```
# xfs_db -c frag -v -r /dev/vg0/root
actual 62521, ideal 60297, fragmentation factor 3.56%
```

Even if a filesystem is not fragmented, certain kinds of file operations can cause fragmentation of individual files. The **filefrag** command

can identify the fragmentation of a particular file.

```
# filefrag -v /data/isos/CentOS-7-x86_64-DVD.iso
Filesystem type is: 58465342
File size of /data/isos/CentOS-7-x86_64-DVD.iso is 4333764608
(1058048 blocks of 4096 bytes)
ext: logical_offset: physical_offset:
  0: 0.. 131055: 5148188.. 5279243:
  1: 131056.. 1058047: 2241235.. 3168226:
/data/isos/CentOS-7-x86_64-DVD.iso: 2 extents found
```

Currently, there is no suitable defragmentation tool for the ext2/3/4 filesystem. Archiving the defragmented file, deleting the original, then extracting the archive file is a possible work around.

XFS does support online defragmentation via the **xfs_fsr** command. The easiest way to invoke it is without any options in which case it defrags all mounted XFS filesystems for up to two hours, making multiple passes defragmenting the top 10% fragmented files each time. If it doesn't finish in two hours, it saves its progress in /var/tmp/.fsrlast_xfs. Because of this behavior, you can enable it with a single command:

```
# ln -s /usr/sbin/xfs_fsr /etc/cron.daily/
```

tune2fs Command Options

The **tune2fs** command is used to tune different filesystem properties. The most commonly tuned settings are the following properties:

Switch	Description
-c num	By default a "just-in-case" filesystem check will be performed every 28 mounts. An alternate number of mounts can be set, or it can be set to 0 or -1 to disable mount count based checks altogether.
-i interval	By default a "just-in-case" filesystem check will be performed every 6 months. An alternate time interval can be set in days, weeks, or months. It can be set to 0 to disable time interval based checks altogether.
-j	Adds a journal to an ext2 filesystem, turning it into an ext3 filesystem.
-o [^]option,...	Set default mount options from the following list: debug, bsdgroups, user_xattr, acl, uid16, journal_data, journal_data_ordered, journal_data_writeback. Usually only user_xattr and acl are set. If run with the ^ in front of the option, that option will be removed instead of added.

tune2fs Command Examples

The following example illustrates usage of the **tune2fs** command:

```
# tune2fs -c 0 /dev/sda1
. . . snip . .
Setting maximal mount count to -1
# tune2fs -i 0 /dev/sda1
. . . snip . .
Setting interval between checks to 0 seconds
# dumpe2fs /dev/sda1 | grep "Default mount options"
. . . snip . .
Default mount options: (none)
# tune2fs -o acl,user_xattr /dev/sda1
. . . output omitted . . .
```

Convert an ext3 filesystem to ext4:

```
# tune2fs -O extents,uninit_bg,dir_index /dev/sdb1
tune2fs 1.41.12 (17-May-2010)
Please run e2fsck on the filesystem.
```

```
# e2fsck -fDC0 /dev/sdb1
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
One or more block group descriptor checksums are invalid. -->
Fix<y>? yes
Group descriptor 0 checksum is invalid. FIXED.
Group descriptor 1 checksum is invalid. FIXED.
. . . snip . .

Once ext4 features are enabled, reverting back to ext3 is not
possible.
```

Filesystem Debugging

To debug filesystems, expert level knowledge of filesystem internals is required. The **xfs_db** command can view and change lowlevel XFS filesystem details. The **debugfs** command allows manipulation of the Ext2/3/4 filesystem. Following is an example of **debugfs** use:

```
# cp /etc/fstab /boot/
# sync
# debugfs -w /dev/sda1
. . . snip . .
debugfs: ls
2 (12) . 2 (12) .. 11 (20) lost+found 65025 (12) grub
65027 (12) efi 18 (44) fstab
. . . snip . .
debugfs: cat fstab
. . . output omitted . .
debugfs: rm fstab

debugfs: lsdel
Inode Owner Mode Size Blocks Time deleted
18 0 100644 997 1/ 1 Wed Nov 5 11:53:20 2112
1 deleted inodes found.
debugfs: undel <18> newfstab
debugfs: ls
2 (12) . 2 (12) .. 11 (20) lost+found 65025 (12) grub
65027 (12) efi 18 (44) newfstab
. . . snip . .
debugfs: quit
# ls -li /boot/newfstab
18 -rw-r--r--. 1 root root 997 Nov 5 11:51 /boot/newfstab
```

Managing an XFS Filesystem

Maintenance

- **xfs_admin**
- **xfs_repair**
- **xfs_growfs**
- **xfs_info**

Backup and Restore

- **xfs_freeze**
- **xfs_copy**
- **xfsdump**
- **xfsrestore**

The XFS Filesystem

XFS is a high performance filesystem developed by Silicon Graphics Inc. The filesystem was released under the GPL in May 2000 and gained Linux support in 2001. XFS is commonly used in environments with large storage requirements of up to 9 million terabytes where the filesystem spans multiple storage devices. XFS support includes journaling, fast transactions, Direct I/O, guaranteed-rate I/O, online defragmentation (using **xfs_fsr**), and quotas.

XFS Filesystem Maintenance

The **xfs_admin** command is used to view and change parameters of an XFS filesystem. For example it can be used to view and set the filesystem label as follows:

```
# xfs_admin -L database /dev/sdb1
writing all SBs
new label = "database"
# xfs_admin -l /dev/sdb1
label = "database"
```

xfs_admin may also be used to set or view an XFS filesystem's UUID:

```
# xfs_admin -u /dev/sdb1
UUID = 0e869819-5305-4648-8bd3-8ed23e40058a
```

The **xfs_repair -n** command is used to perform a consistency check on an XFS filesystem, (replacing **xfs_check**). It can be used on either a storage device containing an XFS filesystem, or on an XFS filesystem stored in a file (such as a file created with **xfsdump**). For filesystem repair do not use **fsck** (i.e. **fsck.xfs**), but instead use **xfs_repair**.

xfs_info is the equivalent of running **xfs_growfs -n**, where the filesystem is not grown but geometry information is printed and argument checking is performed:

```
# xfs_info /mnt
meta-data = /dev/sdb1      isize=256    agcount=1, agsize=7583 blks
                  = sectsz=512   attr=2
data        = bsize=4096   blocks=7583, imaxpct=25
                  = sunit=0     swidth=0 blks
naming       = version 2   bsize=4096   ascii-ci=0
log          = internal    bsize=4096   blocks=1200, version=2
                  = sectsz=512   sunit=0 blks, lazy-count=1
realtime     = none        extsz=4096   blocks=0, rtextents=0
```

Backup & Restore

Creating backups of an XFS filesystem can be done in two ways. Either through the command **xfs_copy** or with **xfsdump**. During backup operations it is advisable to freeze the filesystem, or set it read-only, so that the backup is consistent with the actual data on disk. XFS does not directly support snapshots as it expects this functionality to be implemented by the volume manager. In the case of Linux this is usually implemented by LVM. Should LVM not be present, the command **xfs_freeze** may be used to prevent modifications to an XFS filesystem during the backup procedure:

```
# xfs_freeze -f /mnt
```

After the backup is complete, the filesystem can be thawed:

```
# xfs_freeze -u /mnt
```

The tool **xfs_copy** can be used to copy the contents of an XFS filesystem to another storage device or to a file. Before using **xfs_copy** the filesystem should be unmounted, frozen or set to read-only. Each copy made by **xfs_copy** is identical to the original filesystem except for the UUID on each new copy is unique. **xfs_copy** can copy a filesystem to one or more destinations in parallel:

```
# umount /dev/sdb1
# xfs_copy /dev/sdb1 /tmp/sdb1-xfs-backup /tmp/sdb1-xfs-backup2
Creating file /tmp/sdb1-xfs-backup
Creating file /tmp/sdb1-xfs-backup2
 0% ... 10% ... 20% ... 30% ... 40% ... 50% ... 60% ...
    ... 70% ... 80% ... 90% ... 100%
```

All copies completed.

xfsdump is used for creating full or incremental backups. The backups can be written to one or more storage devices. **xfsdump** backs up both data and metadata, including extended attributes. In its most basic form **xfsdump** is called with only the **-f** option:

```
# xfsdump -f /tmp/homedir-dump-20120702 /mnt/xfs
xfsdump: using file dump (drive_simple) strategy
. . . snip . . .
xfsdump: Dump Status: SUCCESS
```

An XFS dump can be restored using the **xfsrestore** command:

```
# xfsrestore -f /tmp/homedir-dump-20120702 /mnt/xfs
xfsrestore: using file dump (drive_simple) strategy
. . . snip . . .
xfsrestore: restore complete: 5 seconds elapsed
xfsrestore: Restore Status: SUCCESS
```

Swap

Swap partitions

- Type 0x82

Swap files

- Must be contiguous

`mkswap`

`swapon`

Swap

In addition to creating filesystems, Linux system administrators also need to pay close attention to swap usage and needs. Memory usage and swap usage can be checked using tools such as `free`, `top`, `vmstat`, `cat /proc/meminfo`, and `swapon -s`.

If additional swap is needed, either more swap partitions or more swap files can be added dynamically as needed. If using swap files, instead of swap partitions, Linux will map out the blocks provided for the swap file and then access the disk space directly, bypassing the filesystem avoiding the performance penalty.

Adding Additional Swap Space

Adding additional swap is a three-step process. First, contiguous disk space must be allocated to a file. Typically, this is done using `dd`. Either the XFS specific `xfs_mkfile` command, or the general purpose `dd` command, can be used to create a 256-megabyte empty, contiguous file:

```
# xfs_mkfile 256m /root/swapfile
# dd if=/dev/zero of=/root/swapfile bs=1024 count=256000
256000+0 records in
256000+0 records out
```

Since this file can contain memory contents, it is important the permissions are set securely. In this case the swap file is in root's home directory, and since normal users can't enter that directory, it is safe. To take a security-in-layers approach, also secure the file:

```
# chmod 600 /root/swapfile
```

The file is then converted to a swap file using the `mkswap` command:

```
# mkswap /root/swapfile
```

Setting up swapspace version 1, size = 255996 KiB
no label, UUID=ac435597-6949-4e22-9219-d8adeaf18ed9

When running `mkswap` on a partition, a UUID will be autogenerated so that the swap partition can be referenced by it in the `/etc/fstab` file.

The system is configured to swap to the new swap device:

```
# swapon /root/swapfile
```

Finally, this line is added to the `/etc/fstab` file to activate the swap file when the system is rebooted.

File: `/etc/fstab`

```
+ /root/swapfile swap swap defaults 0 0
```

When using a swap partition, it is recommended to reference it via its UUID in the `/etc/fstab`.

Filesystem Structures

Data Blocks

- The file's data.

inode Tables

- Data about the file's data.
- Note: XFS creates inodes on demand

Filesystem Structures

Linux filesystems divide disk space into two fundamental types of storage: data blocks and inode tables. As their name implies, data blocks are used to store data (some modern filesystems may use a more powerful format called "extents," but the concept is the same.) Data about the data blocks, sometimes referred to as metadata, is stored in inode tables, (a table containing multiple inodes). Each file on the system uses exactly one inode.

Historically, inodes were each 128 bytes in size however, new technologies such as SELinux have created an increasing amount of metadata for each file. For efficiency, inode size can be enlarged to store this extra metadata (avoiding an extra lookup). It is now common to see 256 byte inodes in use, essentially trading disk space for speed.

When a new filesystem is created, inode tables are pre-allocated. Some Linux filesystems can turn unused data blocks into inode tables, but others cannot. As a result, it is possible to run out of inodes before running out of disk space. Although popular, the extended filesystem (ext2, ext3, and ext4) cannot dynamically allocate new inode tables. The filesystem must be backed up and recreated with more inodes. In contrast, XFS, JFS, and Btrfs can dynamically add inodes as needed.

Determining Inode and Block Numbers

It is rarely necessary to know the actual inode number or, even rarer, the block numbers associated with a file. The inode number can be easily obtained with the `ls` or `stat` commands. Details about allocated block numbers (or extents) generally require filesystem specific commands, but `hdparm` can return basic block location info for most:

```
$ ls -li demofile
134 -rw-r--r--. 1 root root 2582 Aug 25 10:37 demofile
$ stat -c %i demofile
134
# hdparm --fibmap demofile
demofile:
      filesystem blocksize 4096, begins at LBA 0; assuming →
          512 byte sectors.
      byte_offset    begin_LBA    end_LBA    sectors
                  0        1400      1407         8
```

Determining Disk Usage With df and du

df Report disk space usage per filesystem

- **-h** human readable output
- **-i** list inode information instead of block usage
- **-T** include filesystem type
- **-H|--si** use powers of 1000 instead of 1024

du Report disk usage per file and directory

- **-h** human readable sizes
- **-s** summarize, only display total for each argument
- **-x** do not include files on a different filesystem
- **--si** use powers of 1000 instead of 1024

Determining Disk Usage by Filesystem

The **df** command shows how much disk space each filesystem is using and where it is mounted. It can also show the filesystem type when the **-T** option is added.

To get a human readable (**-h**) summary of how much disk space is available run:

```
$ df -hT
Filesystem Type Size Used Avail Use% Mounted on
/dev/hda2 ext3 252M 136M 102M 57% /
/dev/hda3 ext3 2.0G 1.6G 238M 88% /usr
/dev/hda5 ext3 2.0G 386M 1.5G 20% /usr/local
/dev/hdc1 ext3 20G 9.1G 11G 46% /home
. . . snip . . .
```

By default, **df** uses powers of 1024. To instead use powers of 1000, add the **-H** (or **--si**) option:

```
$ df -H /home
Filesystem Type Size Used Avail Use% Mounted on
/dev/hdc1 ext3 22G 9.8G 12G 46% /home
```

By default, **df** shows data block usage. To show inode usage instead, add the **-i** options:

```
$ df -hTi /home
Filesystem Type Inodes IUsed IFree IUse% Mounted on
/dev/hdc1 ext3 128K 6.4K 122K 5% /home
```

Determining Disk Usage by File

The **du** command scans the size of all files in a directory and its sub-directories, then prints a report. When **du** is not given any arguments, it reports on the current directory. Where **du** can only report on directories it can read, non-root users will get incorrect totals when scanning directories which do not grant them read or execute permission.

The following example prints a human readable (**-h**) summary (**-s**) of how much disk space the **/home/** directory is using:

```
# du -hs /home
9.1G /home
```

By default, **du** uses powers of 1024. To instead use powers of 1000, add the **--si** option:

```
# du -hs --si /home
9.8G /home
```

To show only files on the current filesystem, excluding filesystems mounted on sub-directories, add the **-x** option:

```
# du -hsx /usr
2.0G /usr
# du -hsx /usr
1.6G /usr
```

Configuring Disk Quotas

Mount options for filesystems

- `usrquota`
 - `grpquota`

Database records

- XFS treats quotas as filesystem metadata
 - Created and populated with initial values by **quotacheck** for other file systems (e.g. ext3)

Files used at the root of the filesystem: aquota.user and aquota.group

- Kernel updates values as disk writes occur

Starting and stopping accounting

- quotaon, quotaoff

that filesystem.

Except with XFS, quota configuration files (`aquota.user` and `aquota.group`) for that filesystem must be created using the **`quotacheck`** command. For example, to create the initial user and group quota database files for the `/home` filesystem you would run the following:

```
# quotacheck -cuqm /home
```

Enabling and Disabling Quotas

Kernel support for disk quotas must be activated. This is done automatically on boot with the system initialization script(s) for filesystems mounted with quota options, or it can be done manually with `quotaon`:

```
# quotaon /home
```

The kernel tracking of disk quotas can be enabled or disabled at any time using the **quotaon** and **quotaoff** commands. For example, if you wanted to temporarily disable quota accounting on the /home filesystem you would run the following command:

```
# quotaoff /home
```

that filesystem.

Except with XFS, quota configuration files (`aquota.user` and `aquota.group`) for that filesystem must be created using the **`quotacheck`** command. For example, to create the initial user and group quota database files for the `/home` filesystem you would run the following:

```
# quotacheck -cuqm /home
```

Enabling and Disabling Quotas

Kernel support for disk quotas must be activated. This is done automatically on boot with the system initialization script(s) for filesystems mounted with quota options, or it can be done manually with `quotaon`:

```
# quotaon /home
```

The kernel tracking of disk quotas can be enabled or disabled at any time using the **quotaon** and **quotaoff** commands. For example, if you wanted to temporarily disable quota accounting on the /home filesystem you would run the following command:

```
# quotaoff /home
```

Setting Quotas

Types of Quotas

- User Quotas
- Group Quotas

Types of Limits

- File Limits (soft or hard)
- Block Limits (soft or hard)
- Grace Period

setquota

- Non-interactive

edquota

- Interactive with text editor

XFS also uses `xfs_quota(8)`.

Types of Quotas

Quotas enforce how much disk space (per block) is being used by a user or group, or how many files (inodes) can be owned by a user or group. Disk and file limits can have both soft and hard limits:

Hard limits ⇒ absolute restrictions which the user cannot exceed in any way.

Soft limits ⇒ can be exceeded by the user for a period of time. After this configurable grace period is up, soft limits behave like hard limits.

setquota

Disk quotas can be configured non-interactively with the `setquota` command. This is usually the fastest and most efficient way to configure a user or group quota, especially in a script. The syntax of `setquota` is:

```
setquota username block_softlimit block_hardlimit inode_softlimit inode_hardlimit filesystem
```

The following is an example of using `setquota` to implement a 50MB block soft limit, and a 100MB block hard limit for the emcnabb user on /home:

```
# setquota emcnabb 50000 100000 0 0 /home
```

It is common for users on a system to have similar quotas. The `setquota` command can use the quota on an existing account to be a prototype for another user or group. For example, the following command would set the quota on the lmcnabb account to be the

same as the quota on the emcnabb account.

```
# setquota -p emcnabb lmcnabb /home
```

With XFS this can also be done with `xfs_quota`:

```
# xfs_quota -xc 'limit -u bsoft=50m bhard=100m emcnabb' /home
```

edquota

Disk quotas can be configured interactively with the `edquota` command. `edquota` opens a temporary text file with the default editor (defined by the `EDITOR` variable), and allows the user to modify the quota limits in the appropriate column. When the limits have been set, the changes are made active by saving the file, and exiting the editor:

```
# edquota christec
```

Viewing and Monitoring Quotas

Determining usage

- **quota**
- **repquota**

Notifying users

- **warnquota**

guru	8	50	75	28	100	125
toor	497	0	0	16	0	0

The warnquota Utility

The **warnquota** utility emails users who are over their disk quota with a friendly notification. To enable it, type:

```
# ln -s /usr/sbin/warnquota /etc/cron.daily/
```

The contents of the messages sent by **warnquota** can be changed by editing the `/etc/warnquota.conf` file.

The following shows example output from the **quota** and **repquota** commands:

```
# quota guru
Disk quotas for user guru (uid 500):
Filesystem blocks quota limit files quota limit
/dev/hda7      8      50     75      28    100    125
# repquota /tmp
*** Report for user quotas on device /dev/hda7
Block grace time: 7days; Inode grace time: 7days

File limits          Block limits
User used soft hard grace used soft hard grace
-----
root   497    0    0      16    0    0
xfs     1    0    0      2    0    0
```

Filesystem Attributes

Special attributes

- no atime
- synchronous
- no backup
- append-only
- immutable
- journaled

Manipulating

- **chattr**
- **lsattr**

Filesystem Attributes

The XFS and ext2/3/4 filesystem supports a variety of special attributes on files which can be set using the **chattr** command, or viewed using the **lsattr** command. The following list shows the most commonly used attributes and their effect when applied to a file:

- a** ⇒ Append-only: attempts to open(2) the file without the O_APPEND flag set will fail.
- A** ⇒ Do not update file access time.
- D** ⇒ The **dump** program will not backup this file when filesystem backup is performed.
- e** ⇒ Extents: reflects the use of extents with Ext4. This cannot be modified with **chattr**.
- i** ⇒ Immutable; all attempts to modify the file's data or metadata (except for atime) will fail.
- s** ⇒ Force synchronous writes for file.

Note that not all of the attributes listed in the **chattr(1)** man page are supported. See the BUGS AND LIMITATION section of the man page for details.

Setting and Displaying Attributes

The **chattr** and **lsattr** commands can be used to set and display attributes as shown in the following example, (note that the **ls** command gives no indication of, or information about, filesystem attributes):

```
# echo "initial data" > example
# chattr +a example
# ls -l example
-rw-r--r-- 1 root root 13 2009-01-20 16:29 example
# lsattr example
-----a-----e- example
# echo "new data; overwrite" > example
-bash: example: Operation not permitted
# echo "new data; append" >> example
# cat example
initial data
new data; append
```

Lab 5

Estimated Time:
S12: 35 minutes
R7: 35 minutes

Task 1: Creating and Managing Filesystems

Page: 5-28 Time: 25 minutes

Requirements:  (1 station)

Task 2: Hot Adding Swap

Page: 5-35 Time: 5 minutes

Requirements:  (1 station)

Task 3: Setting User Quotas

Page: 5-37 Time: 5 minutes

Requirements:  (1 station)

Objectives

- ❖ Use fdisk to partition free space.
- ❖ Use mkfs to create filesystems.
- ❖ Use filesystem-specific tools to manage filesystems.

Requirements

- ▀ (1 station)

Relevance

Linux supports many filesystem types. Each filesystem has individual costs, benefits, and capabilities, and is managed differently.

Notices

- ❖ The commands and output used in this lab are based on the first SCSI drive of the system. Depending on your hardware, you may need to replace /dev/sda with a different value, (e.g. /dev/hda).

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Create a record of the current partition table to make cleanup easier later:

```
# sfdisk -d /dev/sda > /root/part.table
```

- 3) Use fdisk to create three new 500M partitions. The first will be used to hold an ext4 (type 83) filesystem, the second will be used for a FAT (type c) filesystem, the last will be used for an XFS filesystem.

```
# fdisk /dev/sda  
... output omitted (intro text) ...  
  
Command (m for help): p  
... output omitted (the current partition table) ...
```

```
Command (m for help): n  
Partition type:  
  p  primary (2 primary, 0 extended, 2 free)  
  e  extended
```

Lab 5

Task 1

Creating and Managing Filesystems

Estimated Time: 25 minutes

Select (default p)

P

Partition number (3,4, default 3): **3**

First sector (x-y, default x): **Enter**

Using default value x

Last sector or +sectors or +size{K,M,G} (x-y, default y): **+500M**

Partition 3 of type Linux and of size 500 MiB is set

- This assumes that only two partitions exist. If 3 exist, an extended partition will be needed instead, and partitions 4, 5, and 6 will need to be created, the fourth taking all available space, 5 and 6 taking portions of that space.

Command (m for help): **n**

Command action

 p primary (3 primary, 0 extended, 1 free)

 e extended

Select (default e)

e

Selected partition 4

First sector (x-y, default x): **Enter**

Using default value x

Last sector or +sectors or +size{K,M,G} (x-y, default y): **Enter**

Using default value y

Partition 4 of type Extended and of size 4.9 GiB is set

- The size will likely be different, depending on your lab environment.

Command (m for help): **n**

All primary partitions are in use

Adding logical partition 5

First sector (x-y, default x): **Enter**

Using default value x

Last sector or +sectors or +size{K,M,G} (x-y, default y): **+500M**

Partition 5 of type Linux and of size 500 MiB is set

Command (m for help): **n**

All primary partitions are in use

Adding logical partition 6

First sector (x-y, default x): **Enter**

Using default value x

Last sector or +sectors or +size{K,M,G} (x-y, default y): **+500M**

Partition 6 of type Linux and of size 500 MiB is set

Command (m for help): **p**

. . . output omitted (the new partition table with the two new partitions) . . .

Command (m for help): **t**

Partition number (1-6, default 6): **5**

- Enter the number of the second 500 MiB partition created.
- Type "c" is the preferred partition type for FAT filesystems.

Hex code (type L to list codes): **c**
WARNING: If you have created or modified any DOS 6.x partitions, please see the fdisk manual page for additional information.

Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'

Command (m for help): **w**

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)

Syncing disks.

partprobe /dev/sda

- Have the kernel re-read the partition table.

4) In this lab, the three partitions will be referenced as *sdaX*, *sdaY*, and *sdaZ*.

Record the real partition numbers in the following table:

Device Name	Filesystem	Partition Number
/dev/sdaX	ext4	3
/dev/sdaY	vfat	5
/dev/sdaZ	xfs	6

5) Verify that the three partition records are seen on disk, in the kernel data structures and by udev:

```
# fdisk -l /dev/sda  
. . . output omitted . . .  
# cat /proc/partitions  
. . . output omitted . . .  
# ls /dev/sda*  
. . . output omitted . . .
```

If /proc/partitions and fdisk -l /dev/**sda** are inconsistent, or the new block devices are missing, a reboot is necessary.

- 6) Being careful to specify the correct block device file, create filesystems on the three partitions:

```
# mkfs -t ext4 /dev/sdaX
. . . output omitted . . .
# mkfs -t vfat /dev/sdaY
. . . output omitted . . .
# mkfs -t xfs /dev/sdaZ
. . . output omitted . . .
```

- 7) Add entries for the filesystems to the /etc/fstab file:

File: /etc/fstab					
+	/dev/sdaX	/mnt/ext	ext4	defaults	1 2
+	/dev/sdaY	/mnt/fat	vfat	shortname=lower	1 2
+	/dev/sdaZ	/mnt/xfs	xfs	defaults	1 2

- 8) Make the corresponding directories, mount the filesystems, and confirm that they're mounted:

```
# mkdir /mnt/{ext,fat,xfs}
# mount -a
# df -hT | grep /mnt
/dev/sdaX      ext4    490M   11M   455M   3% /mnt/ext
/dev/sdaY      vfat    510M     0    510M   0% /mnt/fat
/dev/sdaZ      xfs    497M   26M   472M   6% /mnt/xfs
# mount | grep /mnt
. . . output omitted . . .
```

9) Unmount the filesystems and find the corresponding UUIDs:

```
# umount /mnt/{ext,fat,xfs}
# blkid
. . . output omitted . . .
/dev/sdaX: UUID="sdaX-UUID" TYPE="ext4"
/dev/sdaY: SEC_TYPE="msdos" UUID="sdaY-UUID" TYPE="vfat"
/dev/sdaZ: UUID="sdaZ-UUID" TYPE="xfs"
. . . output omitted . . .
```

- It is highly recommended to write down all three UUIDs. Be sure to specify which UUID each device corresponds to.

10) Edit entries in the /etc/fstab file:

File: /etc/fstab					
-	/dev/sdaX	/mnt/ext	ext4	defaults	1 2
+	UUID=sdaX-UUID	/mnt/ext	ext4	defaults	1 2
-	/dev/sdaY	/mnt/fat	vfat	shortname=lower	1 2
+	UUID=sdaY-UUID	/mnt/fat	vfat	shortname=lower	1 2
-	/dev/sdaZ	/mnt/xfs	xfs	defaults	1 2
+	UUID=sdaZ-UUID	/mnt/xfs	xfs	defaults	1 2

11) Remount the partitions:

```
# mount -a
# df -hT | grep /mnt
/dev/sdaX ext4 490M 11M 455M 3% /mnt/ext
/dev/sdaY vfat 510M 0 510M 0% /mnt/fat
/dev/sdaZ xfs 497M 26M 472M 6% /mnt/xfs
# mount | grep /mnt
. . . output omitted . . .
```

- Notice that even though UUIDs were used in fstab, df outputs device names, not UUIDs

12) Create a file on the ext4 filesystem and set an ACL on it:

```
# touch /mnt/ext/file1
# setfacl -m u:guru:rw /mnt/ext/file1
# getfacl /mnt/ext/file1
. . . output omitted . . .
```

- 13) View the default mount options on the ext4 filesystem:

```
# tune2fs -l /dev/sdaX | grep "Default mount"  
Default mount options: user_xattr acl
```

- Notice that extended attributes and ACLs are enabled by default.

- 14) Create test files on the VFAT filesystem, and notice that the lower filesystem mount option will munge the case of some filenames:

```
# touch /mnt/fat/{TEST1,test2,Test3}  
# ls -l /mnt/fat  
total 0  
-rwxr-xr-x 1 root root 0 Mar 26 15:31 test1  
-rwxr-xr-x 1 root root 0 Mar 26 15:31 test2  
-rwxr-xr-x 1 root root 0 Mar 26 15:31 Test3
```

- Notice the case change in this filename

- 15) Change the mount options:

File: /etc/fstab				
-	UUID=sdaY-UUID	/mnt/fat	vfat	shortname=lower 1 2
+	UUID=sdaY-UUID	/mnt/fat	vfat	shortname=mixed 1 2

- 16) Remount the filesystem, and test again:

```
# umount /mnt/fat; mount /mnt/fat  
# ls -l /mnt/fat  
total 0  
-rwxr-xr-x 1 root root 0 Mar 26 15:31 TEST1  
-rwxr-xr-x 1 root root 0 Mar 26 15:31 test2  
-rwxr-xr-x 1 root root 0 Mar 26 15:31 Test3
```

- The VFAT filesystem doesn't start respecting this option with just a remount.

- The case isn't mangled this time.

If using defaults in the /etc/fstab file, RHEL7/SLES12 will default to shortname=mixed.

- 17) Create a fragmented file in the XFS filesystem:

```
# xfs_io -f -c "pwrite -R 0 50M" -c "fsync" /mnt/xfs/testfile  
. . . output omitted . . .  
# filefrag /mnt/xfs/testfile  
. . . output omitted . . .
```

- 18) Defrag the file with XFS tools:

```
# xfs_fsr /mnt/xfs/testfile  
# filefrag /mnt/xfs/testfile  
. . . output omitted . . .
```

Cleanup

- 19) Unmount the filesystems and remove them from /etc/fstab:

```
# umount /mnt/{ext,fat,xfs}  
# rm -rf /mnt/{ext,fat,xfs}  
# sed -i '/mnt/d' /etc/fstab
```

- 20) Remove the partitions created for this lab:

```
# sfdisk -f /dev/sda < /root/part.table  
. . . output omitted . . .  
# reboot
```

• Double-check this command for correctness.

If a reboot results in booting to an emergency mode (sulogin prompt), ask your instructor for help.

Objectives

- >Create and activate additional swap space.

Requirements

- (1 station)

Relevance

It is not always possible to create a dedicated swap partition. For example, in an emergency low memory situation it may be necessary to add virtual memory to a server with no uncommitted partitions. In such situations it is possible to instead use swap files.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Record the current memory usage so that you can refer to it later:

```
# (echo "BEFORE"; free) > /tmp/mem
```

- 3) Allocate a 100MB block of contiguous disk space using the dd command:

```
# dd if=/dev/zero of=/tmp/swaptest bs=1M count=100  
100+0 records in  
100+0 records out  
104857600 bytes (105 MB) copied, 0.432301 seconds, 243 MB/s
```

• if = input file; of = output file; bs = block size

```
# ls -lh /tmp/swaptest  
-rw-r--r-- 1 root root 100M Feb 6 10:19 /tmp/swaptest
```

- 4) Ensure safe permissions for the swap file. Write a swap signature into the file so that the kernel will recognize it as a valid swap space:

```
# chmod 600 /tmp/swaptest  
# mkswap -L swaptest /tmp/swaptest  
Setting up swapspace version 1, size = 102396 KiB  
LABEL=swaptest, UUID=dca9c3b7-2695-441b-b4aa-12b9259f3b7d
```

Lab 5

Task 2

Hot Adding Swap

Estimated Time: 5 minutes

- 5) Activate the new swap space and record the new memory totals:

```
# swapon /tmp/swaptst  
# (echo; echo "AFTER"; free) >> /tmp/mem
```

- 6) Compare the BEFORE and AFTER totals for memory usage:

```
# cat /tmp/mem
```

BEFORE

	total	used	free	shared	buffers	cached
Mem:	3110264	368320	2741944	0	47420	152936
-/+ buffers/cache:	167964	2942300				
Swap:	524280	0	524280			

AFTER

	total	used	free	shared	buffers	cached
Mem:	3110264	472612	2637652	0	47540	255388
-/+ buffers/cache:	169684	2940580				
Swap:	626672	0	626672			

- 7) Deactivate the temporary swap space and remove the swaptst file:

```
# swapoff /tmp/swaptst  
# rm -f /tmp/swaptst
```

- 8) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Configure and test disk quotas on the /tmp/ filesystem.

Requirements

- (1 station)

Relevance

User quotas prevent a single user or process from consuming all of a system's disk space, resulting in a more reliable environment.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) [R7] This step should only be performed on RHEL7.

Modify the entry for /tmp/ in the /etc/fstab file to include the user and group quota options:

```
File: /etc/fstab  
→ /dev/mapper/vg0-tmp /tmp xfs defaultsusrquota,grpquota 1 2
```

- 3) [S12] This step should only be performed on SLES12.

Modify the entry for /tmp/ in the /etc/fstab file to include the user and group quota options:

```
File: /etc/fstab  
→ /dev/vg0/tmp /tmp xfs defaultsusrquota,grpquota 1 2
```

- 4) Reboot the system so that the XFS /tmp filesystem will use the new mount options, since it can't be unmounted and remounted as it is in use:

```
# reboot
```

Note that with ext4 a **mount -o remount /tmp** would have been sufficient, but XFS won't apply quota options with a **remount**. After the reboot finishes, log back in and become root.

Lab 5

Task 3

Setting User Quotas

Estimated Time: 5 minutes

- 5) Verify the filesystem has the quota options set:

```
# mount -o remount /tmp/  
# mount | grep /tmp  
[R7] /dev/mapper/vg0-tmp on /tmp type xfs (rw,relatime,seclabel,attr2,inode64,usrquota,grpquota)  
[S12] /dev/mapper/vg0-tmp on /tmp type xfs (rw,relatime,attr2,inode64,noquota)
```

- 6) [S12] This step should only be performed on SLES12.

On SUSE Linux Enterprise Server, reboot to enable quotas, then verify after the reboot that quotas were enabled:

```
# reboot  
# mount | grep /tmp  
/dev/mapper/vg0-tmp on /tmp type xfs  
(rw,relatime,attr2,inode64,usrquota,grpquota)
```

- Run this after logging back in as the guru user, and switching to the root user.

- 7) Activate the quotas for, and generate a report of, the current quota settings for each user including usage statistics for the /tmp/ filesystem:

```
# quotaon /tmp/  
# repquota /tmp/  
*** Report for user quotas on device /dev/mapper/vg0-tmp  
Block grace time: 7days; Inode grace time: 7days  
          Block limits           File limits  
User        used    soft    hard   grace     used    soft    hard   grace  
-----  
root      --     36      0      0            7      0      0  
guru      --      4      0      0            4      0      0  
... snip ...
```

- Disk quotas are normally activated automatically on boot. If a reboot was required in a previous step, then skip running quotaon.

- 8) Change the block quota to set a hard limit (4th column) of 2048 blocks (2MB) for the guru user. (Note that the usage for blocks and inodes will almost certainly differ.)

```
# setquota guru 0 2048 0 0 /tmp/
```

The edquota command is an alternate, interactive method for setting the quota, which looks similar to the output of the quota command.

- 9) Test guru's disk quota by attempting to create a 4MB file:

```
# su - guru
$ cd /tmp/
$ dd if=/dev/zero of=/tmp/bigfile bs=1024 count=4096
[s12] sdaX: write failed, user block limit reached.
dd: writing `/tmp/bigfile': Disk quota exceeded
2045+0 records in
2044+0 records out
2093056 bytes (2.1 MB) copied, 0.0183077 s, 114 MB/s
```

- Results may vary. What is important is identifying that the disk quota was exceeded, and that the data written (2.1MB here) was truncated, (i.e. not 4 MB).

- 10) As the guru user, use the quota command to view defined quotas and usage:

```
$ quota
Disk quotas for user guru (uid 500):
      Filesystem    blocks   quota   limit   grace   files   quota   limit   grace
/dev/mapper/vg0-tmp        2048*       0     2048           1       0       0
```

- 11) Try copying another file to /tmp/ to see the error message generated when hitting quota limits:

```
$ cp /etc/fstab /tmp/
cp: writing `/tmp/fstab': Disk quota exceeded
```

- 12) Delete bigfile and verify that the usage reported by the quota command reflects the change:

```
$ rm /tmp/bigfile
$ quota -v
Disk quotas for user guru (uid 500):
      Filesystem    blocks   quota   limit   grace   files   quota   limit   grace
/dev/mapper/vg0-tmp        0       0     2048           1       0       0
```

- 13) As root, use the setquota command to reset the disk quota for guru back to the default (unlimited):

```
$ exit
```

```
# setquota guru 0 0 0 0 /tmp
```

- 14) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Content

Logical Volume Management	2
Implementing LVM	3
Creating Logical Volumes	4
Activating LVM VGs	5
Exporting and Importing a VG	6
Examining LVM Components	7
Changing LVM Components	8
Advanced LVM Overview	10
Advanced LVM: Components & Object Tags	11
Advanced LVM: Automated Storage Tiering	12
Advanced LVM: Thin Provisioning	14
Advanced LVM: Striping & Mirroring	16
Advanced LVM: RAID Volumes	17
SLES Graphical Disk Tool	18
RAID Concepts	19
Array Creation with mdadm	20
Software RAID Monitoring	21
Software RAID Control and Display	22
Lab Tasks	
1. Creating and Managing LVM Volumes	24
2. Creating LVM Thin Volumes	34
3. Troubleshooting Practice: LVM	41
4. Creating and Managing a RAID-5 Array	42
	23



Chapter

6

LVM & RAID

Logical Volume Management

Hierarchy of Concepts

- Physical Volumes
- Physical Extents
- Volume Groups
- Logical Volumes
- Logical Extents
- Filesystems

RHEL7: All filesystems except /boot/ can live on LVM

- Anaconda limitation

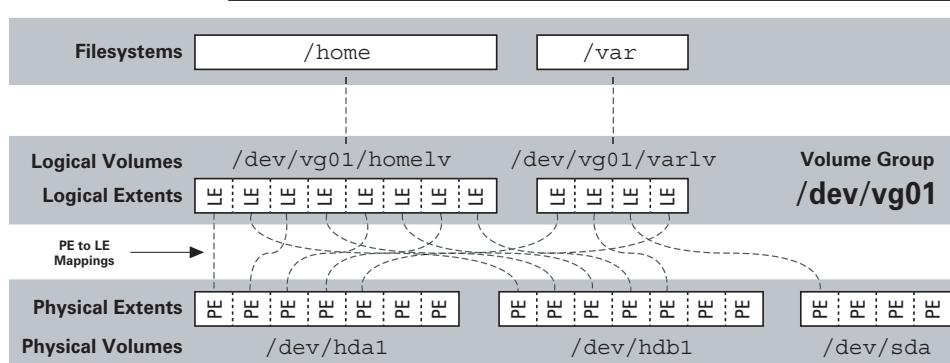
LVM2

Logical Volume Management introduces an abstraction layer between the physical disk and the filesystem, allowing filesystems to be resized easily, to use discontiguous disk space, and to span physical disks. LVM operates on top of a low level volume manager called device mapper.

Linux LVM is very similar to HP-UX LVM, and is implemented through a few conceptual layers. In LVM, the physical hard drives are called Physical Volumes. Physical Volumes are assigned to a Volume Group. A Volume Group is divided into arbitrary fixed-size Physical Extents.

Logical Volumes are block devices which are constructed out of arbitrary pools of Physical Extents (but only from within the same Volume Group). Filesystems are then built on top of Logical Volumes. Volume Groups are logical management concepts, a pool of all Physical Volumes which can be used to supply Physical Extents for Logical Volumes.

To support numerous Logical Volumes, Extents, or complex configurations of Logical Volumes, a larger metadata area should be created when building new Physical Volumes, (see `pvcreate(8)`, specifically the `--metadatasize` option).



Implementing LVM

Can be created at OS install time

Use whole block device, or optionally create GPT or MBR partition

- **gdisk /dev/device**
Set GPT partition type to 8E00
- **fdisk /dev/device**
Set MBR partition type to 8E

Create Physical Volumes

- **pvcreate /dev/device [device] [...]**

Create Volume Group

- **vgcreate VGname /dev/device [device] [...]**

Create LVM Partitions

Creating new Logical Volumes from free space is fairly straightforward. First, physical partitions are created and set to partition type 0x8E. If using **fdisk**, or **sfdisk**, make sure to update the kernel's in-memory copy of the partition table by either rebooting the system, or using the **partprobe** or **sfdisk -R** commands. The **parted**, or **gdisk** programs are preferred; capable of dealing with large partition sizes and the GUID Partition Table (GPT).

Physical Volumes

Physical Volumes (PVs) can be a disk partition, an entire disk, a meta device (RAID device) or even a loopback file. The **pvcreate** command is used to initialize the Physical Volume for use by LVM. The standard usage is:

```
# pvcreate /dev/sda3 /dev/sdb ...
```

The **pvcreate** creates a physical volume label in the 2nd sector of the device. The label contains the PV UUID, size, list of data areas, and list of metadata area locations. A metadata area stores configuration information. Every PV within a volume group contains identical information. By default one metadata area is created starting at the 5th sector. The **--metadatacopies 2** option can be used with **pvcreate** so that a second metadata area is also created at the end of the device.

Create a Volume Group

A Volume Group (VG) contains Physical Volumes (PVs) which provide Physical Extents (PEs). The size of the PEs can vary from one VG to another, and is defined at VG creation time. The default is 4MB. PEs are allocated to Logical Volumes and are mapped to Logical Extents (LEs) within the associated Logical Volume.

The PE size can only be set when the Volume Group is created. It is now common to set the PE size larger than the default. PE sizes can range from 1KB to an unlimited size. In general, the larger the size, the better the performance, (though with a larger PE size, there is less granular control of the Logical Volume). They should not be made too large, however, as a LV must have at least 1 PE allocated in order to exist. Additionally, since each PE must fit entirely on only 1 PV, if they are too large there could be a significant amount of storage space that is not accessible on the PV(s).

The size of the PEs can be set using the **-s** option to the **vgcreate** command. For example:

```
# vgcreate -s 32M volgrpname /dev/sda3 /dev/sdb ...
```

Creating Logical Volumes

Create Logical Volumes, specifying number of extents, percentage of VG free space, or explicit size

- `lvcreate -l 525 -n LVname VGname`
- `lvcreate -l 100%FREE -n LVname VGname`
- `lvcreate -L 850G -n LVname VGname`

Create filesystems

- `mkfs.xfs LV_path`

Mount the Logical Volume and test

- Edit /etc/fstab for persistence, then `mount -a`

Two legal ways to reference a Logical Volume

- `/dev/mapper/VGname-LVname`
- `/dev/VGname/LVname`

Create Logical Volumes

Multiple Logical Volumes (LVs) can be created in a Volume Group (VG) using the `lvcreate` command. Each LV has a unique name that can be specified, or automatically generated. When using `lvcreate`, Logical Volumes are created as linear logical volumes by default which means there is a 1:1 mapping between a LE to a PE. Other advanced types of volumes are supported such as legacy striping, legacy mirroring or RAID. When mirroring each LE is mapped to 2 or more PEs.

The two most commonly used options are `-L` to specify the size and `-n` to specify the name. For example:

```
# lvcreate -L 650M -n LVname VGname
lvcreate -- rounding size up to PE boundary
lvcreate -- doing automatic backup of "VGname"
lvcreate -- LV "/dev/VGname/LVname" created
```

Initialize Filesystems

Logical Volumes can be used like any other block device, and most often are used as a more flexible replacement for a partition. Filesystems are created on top of Logical Volumes, this is done with the `mkfs` command and can then be mounted no differently than without LVM. For example:

```
# mkfs -t xfs /dev/VGname/LVname
# mount /dev/VGname/LVname /some/dir
```

Do not forget to edit the /etc/fstab file and add an entry for the

new Logical Volume so that it will be mounted whenever the system is booted.

Safe Ways to Reference a Logical Volume

Since LVM is built on top of device mapper, every logical volume will appear as a device node in the /dev directory named `dm-x` where `x` is a number based on the activation order. The /dev/dm-`x` files should not be used or referenced in configuration files as the number assignments aren't guaranteed to be the same across reboots. However, the symlinks in the /dev/mapper directory to the device nodes are based on the device mapper name and are stable. For example:

`/dev/mapper/vg_loki-code.gurulabs.com` → `../dm-24`

The /dev/mapper symlinks and the /dev/dm-`x` files are created by the `10-dm.rules` UDEV file.

Additionally, UDEV creates a subdirectory under /dev named after the VG is and then creates symlinks for each LV inside of the subdirectory. Since LVs are inside VGs this method of referencing a LV correlates well. For example:

`/dev/vg_loki/code.gurulabs.com` → `../dm-24`

The subdirectory method is created by the `11-dm-lvm.rules` UDEV file.

Activating LVM VGs

VG Activation

- Initramfs **dracut**'s LVM module
 - Activates VG containing the / filesystem
- Systemd's **lvm2-activation-generator** if **lvmetad** is disabled
- UDEV triggered **pvscan** with autoactivation if **lvmetad** is enabled
- Manually via **vgchange**

Notable /etc/lvm/lvm.conf settings

- **global_filter** — controls what block devices LVM looks at
 - Use **lvmdiskscan** to test filter
- **use_lvmetad** — caches in memory LVM VG state
 - Enabled by default, huge performance win with many block devices

File: /etc/lvm/lvm.conf

```
+ global_filter = [ "a|/dev/sda.*|", "a|/dev/disk/by-id/dm.*|", "r|.*|" ]
```

Whenever the /etc/lvm/lvm.conf is modified, a new initramfs should be generated with: **dracut -f**.

Activating VGs and lvmetad

Activating a VG makes it usable. Most Linux systems have their root filesystem within a LVM LV, and as such the LVM VG containing the LV must be activated during the initramfs stage at boot time. The stock dracut LVM module located in `/usr/lib/dracut/modules.d/90lvm/` performs the work of activating the VGs and LVs referenced on the kernel command via the `rd.lvm.lv` dracut parameters which are persistently defined in the `/etc/default/grub` file.

Modern Linux systems use the **lvmetad** caching daemon by default. Without **lvmetad**, on systems with a large amount of block devices, each time an LVM command is run there can be a long delay as the current state and status must be built by examining all the unfiltered block devices. The **global_filter** setting applies to both **lvmetad** as well as the LVM command line tools whereas the **filter** setting is ignored by **lvmetad**.

With **lvmetad** running, the UDEV file `69-dm-lvm-metad.rules` automatically activates any VGs it sees, otherwise systemd's **lvm2-activation-generator** handles activation.

Running LVM Commands

Most LVM commands are symbolic links to the **lvm** command which looks at the invocation name to determine its behavior. Alternatively the **lvm** command can be used and passed the volume operation as an argument, or the **lvm** command is invoked and used interactively. For example, each of the following are equivalent:

```
# vgchange -a y      #Operation determined by link name
# lvm vgchange -a y  #Operation passed as argument
# lvm                #Interactive usage
lvm> vgchange -a y
2 logical volume(s) in volume group "vg_server1" now active
lvm> quit
Exiting.
```

Filtering Which Block Devices LVM Sees

By default LVM looks at all block devices on the system. The **global_filter** setting in the /etc/lvm/lvm.conf can be used to customize which block devices LVM looks at such as when multipathing is being used. The **lvmdiskscan** command displays what block devices LVM sees after the **global_filter** setting has been applied and can be used to fine tune and troubleshoot the setting.

When using multipath devices as LVM physical volumes, you should configure LVM to ignore the individual paths, otherwise you will get spurious error messages. This is done by adjusting the **global_filter** in the **lvm.conf** file. For the example scenario where LVM should only examine `/dev/sda` and DM devices use:

Exporting and Importing a VG

A LVM VG on SAN LUN provides lots of flexibility

- Scenario: An application is stored entirely within a dedicated VG on a dedicated LUN
Enable easy movement of application between servers

vgexport

- Marks VG exported within PV metadata
- Most LVM command ignore entirely exported VGs

vgimport

- Unexports a VG, allowing it be activated
- Hotplugged PVs can be discovered with **pvscan**

Moving Volumes Between Systems

The **vgexport** and **vgimport** commands can be used to move Volume Groups between systems, such as when a Volume Group on an external disk array is being moved from one system to another. Most LVM commands will ignore VGs marked exported, which is a flag persistently stored in the VG, and different than a VG that has simply been made temporarily unavailable with **vgchange -a n**.

On the server exporting the VG, first stop using data, deactivate, and then export the VG. For example:

```
# umount /srv/app
# vgchange -an appVG
vgchange -- volume group "appVG" successfully deactivated
# vgexport appVG
vgexport -- volume group "appVG" successfully exported
```

The PV(s) can then be moved to the new server. On the new server importing the VG, verify the PV(s) are seen, import, and then activate the VG. For example:

```
# pvscan
pvscan -- reading all physical volumes (this may take a while...)
pvscan -- inactive PV "/dev/sdb"  is in EXPORTED VG "appVG" [9.95 GB / 996 MB free]
pvscan -- total: 1 [9.95 GB] / in use: 1 [9.95 GB] / in no VG: 0 [0]
# vgimport appVG
vgexport -- volume group "appVG" successfully imported
# vgchange -ay appVG
vgchange -- volume group "appVG" successfully activated
```

Examining LVM Components

Verbose Display Commands

- **pvdisplay**
- **vgdisplay**
- **lvdisplay**

Succinct Display Commands

- **pvs**
- **vgs**
- **lvs**

Viewing Volume Information

The first version of LVM used the **{pv,vg,lv}display** commands. LVM2 introduced another set of commands that are more flexible in their output format: **pvs**, **vgs**, and **lvs**. Options for both sets of commands are well documented in their respective man pages. The new commands allow specifying field names for producing custom reports:

```
# lvs -o lv_name,lv_size,devices
  LV      LSize   Devices
  lv_root 688.25g /dev/sda2(0)
  lv_swap  9.89g /dev/sda2(176193)
  . . . snip . . .
```

Get a full list of valid field names with their descriptions by running:

```
# lvs -o help
Logical Volume Fields
-----
  lv_all    - All fields in this section.
  lv_uuid   - Unique identifier.
  lv_name   - Name.
  . . . snip . . .
```

The verbose commands provide multiple lines of output per component.

```
# vgdisplay
```

--- Volume group ---

VG Name	vg_loki
Format	lvm2
Metadata Areas	1
Metadata Sequence No	63
VG Access	read/write
VG Status	resizable

. . . snip . . .

VG Size	3.82 TiB
PE Size	4.00 MiB
Total PE	1000432
Alloc PE / Size	561362 / 2.14 TiB
Free PE / Size	439070 / 1.67 TiB

```
# lvdisplay /dev/vg_loki/code.gurulabs.com
```

--- Logical volume ---

LV Path	/dev/vg_loki/code.gurulabs.com
LV Name	code.gurulabs.com
VG Name	vg_loki
LV UUID	XeS0cr-keSH-pn6m-5C0K-UUTu-AeJT
LV Write Access	read/write
LV Creation host, time	loki.gurulabs.com, 2015-07-15
LV Status	available
# open	1
LV Size	50.00 GiB
Current LE	12800
Segments	1
Allocation	inherit

Changing LVM Components

Adding, Removing, Resizing PVs

- `vgreduce`, `vgextend`, `pvresize`

Move all data off a PV before a removal with `pvmove`

Growing, Shrinking, Resizing, Removing LVs

- `lvextend`, `lvreduce`, `lvresize`, `lvremove`

Resizing a filesystem

- `xfs_growfs`
- `resize2fs` (Ext2/3/4)

All-in-one command (resize FS and LV)

- `fsadm -l resize /dev/vg0/lv_var 75G`
- `lvresize --resizesfs -L +20G /dev/vg0/lv_var`

Resizing Operations

To grow or shrink a Logical Volume, the `lvextend` and `lvreduce` commands can be used. LVM2 introduces the `lvresize` and `fsadm` commands, as well as the `-r` option in the `lvresize` and `lvextend` commands. This option supports offline filesystem resizing, making use of the new `fsadm` command. Filesystems supported are ext2/3/4, ReiserFS, and XFS. Filesystem resizing programs include: `resize2fs`, `resize_reiserfs`, and `xfs_growfs`.

[R7] The following applies to RHEL7 only:

Red Hat Enterprise Linux does not support ReiserFS.

The extended filesystem tool `resize2fs` works on unmounted (shrink and grow) as well as mounted (grow) filesystems. To reduce the extended filesystem, it must not be mounted, and will require the `e2fsck` command to be run.

The order of operations when resizing is not enforced, but is critical. Before growing a filesystem, the Logical Volume needs to be grown. Similarly, the filesystem must be shrunk before shrinking the LV that contains it. Shrinking an LV smaller than the filesystem it contains will result in truncation of the filesystem and likely data loss and corruption.

The following example adds the `/dev/sdb` device to a Volume Group, then the Volume Group is extended, followed by one of the Logical Volumes, followed by the filesystem within the Logical Volume:

```
# pvcreate /dev/sdb
Physical volume "/dev/sdb" successfully created
# vgextend vg01 /dev/sdb
Volume group "vg01" successfully extended
# lvextend -L 500G /dev/vg01/databases
Extending logical volume databases to 500.00 GB
Logical volume databases successfully resized
# resize2fs /dev/vg01/databases
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/vg01/databases is mounted on /srv/databases; ↵
```

```
on-line resizing required
old desc_blocks = 2, new_desc_blocks = 4
Performing an on-line resize of /dev/vg01/databases to,
524288000 (1k) blocks.
The filesystem on /dev/vg01/databases is now 524288000 blocks long.
```

Migrating Off of a Physical Disk

Consider a hard drive whose firmware or S.M.A.R.T counters indicate that it is about to fail, or suppose that you want to migrate from a small physical drive to a newer, larger drive. LVM supports the on-line migration of LEs from the PEs on one drive to PEs elsewhere in the VG.

This example migrates the Logical Extents on `/dev/sda1` to wherever free Physical Extents are available in the Volume Group and then takes the Physical Volume offline:

```
# pvmmove /dev/sda1
```

To specify which destination Physical Volume is to receive the migrated PEs, specify the destination after the source. This example also uses the **-v** option which causes the **pvmmove** command to operate verbosely:

```
# pvmmove -v /dev/sda1 /dev/sdc1
```

Advanced LVM Overview

LVM Physical Extent Allocation Policies

- normal (default), cling, contiguous, anywhere — see `lvm(8)`

Types of Logical Volumes

- Snapshots
- Linear (default), Striped, Built-in RAID 1/5/6/10
- Thinly-provisioned and Cache

LVM in a Cluster

- CLVM allows a VG to simultaneously activated on multiple nodes
- GFS2, the Global FileSystem version 2
- OCFS2, the Oracle Cluster File System

Advanced LVM Concepts

Logical Volumes can be used to provide a few more advanced features. LVM has built-in support for snapshots, striping, mirroring, and now even includes built-in RAID. The health of volumes using these features can be monitored with `systemd lvm2-monitor` service. On servers in a SAN environment or with a hardware RAID controller, normally LVM is used for management and the SAN/HWRAID is used for redundancy. A LVM Cache Pool allows implementation of tiered storage so that a fast storage device can cache frequently accessed data. Thin provisioning allows over-commit of the physical storage with LVs that are larger than the available extents.

Extent Allocation Policies

The extent allocation policy is set on the VG and then inherited by the LVs by default. The allocation policy can be changed per LV if needed to optimize for specific I/O loads.

LVM Snapshots

Linux LVM provides the ability to create and use snapshots; instantaneous, mirrors of a Logical Volume. A snapshot of a LV can be taken by allocating free space within the Volume Group to create a Snapshot LV. Backups can then be made from the snapshot, rather than from the original LV (avoiding problems inherent to backing up an active filesystem), which remains online. It is worth reiterating that snapshots can only be taken on an existing LV.

Snapshots work by utilizing the copy-on-write (CoW) technique. When

a change is to be made to a file on a snapshotted filesystem, the original disk blocks are copied to the storage space allocated to the Snapshot Volume to preserve them, then the write operation takes place as it normally would on the original LV. Reading an unmodified part of the snapshot reads from the original LV. If the Snapshot LV fills, use `lvextend` to increase the snapshot size. Alternatively, the `lvm2-monitor` service can automatically extend a snapshot if the `snapshot_autoextend_threshold` setting in the `lvm.conf` is set to value less than 100.

Since snapshots are just another Logical Volume, they are created with the `lvcreate` command with the `-s` option:

```
# lvcreate -s -L 512M -n snapname /dev/volgrpname/lvname
```

This creates `/dev/volgrpname/snapname`, which is a read-only block device. Simply mount it and back it up using tools like `tar`, `cpio`, or `dump`. Afterwards, the snapshot can be removed with `lvremove`

Cluster Support for LVM

Use of LVM with a cluster allows the same Volume Group, and LVs inside of it, to be visible and active across multiple servers. This requires Cluster LVM (CLVM) running on top of a Pacemaker & Corosync HA cluster.

For Active/Active HA Clusters that require a POSIX filesystem to be mounted on multiple cluster nodes at the same time, the Global File System 2 (GFS2) filesystem and the Oracle Cluster File System (OCFS2) are filesystems that can be used inside of a CLVM LV.

Advanced LVM: Components & Object Tags

LVM metadata and internal subvolumes

- Normal LVM metadata too small for advanced features
- Hidden subvolumes used to store advance feature state

LVM Object Tags

- Grouping of PV, LV or VG objects for convenience
- Tags can be used in commands in place of PV, LV or VG arguments
 - Can shorten commands, reduce typos and misconfigurations
- tags are prefixed with a @

```
[CacheLV_cmeta] 2.00g
[lvol0_pmspare] 2.00g
root            831.94g
[root_corig]    831.94g
swap             4.00g
```

LVM Object Tags

LVM tags are a convenient way to reference multiple LVM objects that are related somehow. For example you could tag all the LVs related to a specific application with a designated tag, or use a tag to indicate which block devices are SSD drives. Then the tag can be used, prefixed with an @ in place of the objects. For example:

```
# pvs -o+tags
PV          VG      Fmt Attr PSize  PFree PV Tags
/dev/sda3   vg0     lvm2 a-- 835.94g  0
/dev/sdb    vg0     lvm2 a-- 893.75g  121.2
/dev/sdc    vg0     lvm2 a-- 893.75g  121.2
# pvchange --addtag ssd /dev/sdb /dev/sdc
Physical volume "/dev/sdb" changed
Physical volume "/dev/sdc" changed
2 physical volume changed / 0 physical volumes not changed
# pvs -o+tags
PV          VG      Fmt Attr PSize  PFree PV Tags
/dev/sda3   vg0     lvm2 a-- 835.94g  0
/dev/sdb    vg0     lvm2 a-- 893.75g  121g  ssd
/dev/sdc    vg0     lvm2 a-- 893.75g  121g  ssd
# lvcreate -L 50GB -n fastlv vg0 @ssd
```

LVM Metadata and Internal Subvolumes

Normally LVM metadata is stored in a special area(s) within each PV. The default size is 1020 KiB, but can be increased at PV creation time with `--metadatasize`. However, this metadata area is meant to keep track of the structure and state of LVM components and isn't up to the task of keeping track of all the state that the advanced features require.

LVM automatically creates and uses metadata and internal subvolumes when RAID volumes, Thinly-provisioned volumes or Cache Pools are in use. For example, LVM creates a metadata subvolume that is one extent in size for every data or parity subvolume in a RAID volume. A metadata subvolume can range in from one extent to 16 GiB in size.

Normally a system administrator need not worry about LVM subvolumes. The various LVM display commands don't show them unless the `--all` option is used. With that option, the names of the subvolumes are enclosed with square brackets. For example:

```
# lvs -o lv_name,lv_size
LV  LSize
root 831.94g
swap 4.00g
# lvs --all -o lv_name,lv_size
LV          LSize
[CacheLV]    889.75g
[CacheLV_cdata] 889.75g
```

Advanced LVM: Automated Storage Tiering

Automated Storage Tiering with LVM LVs

Allows fast PV(s) to cache slower PV(s)

Architecture/Terminology

- OriginLV — large slow LV
- CacheDataLV — smaller fast LV
- CacheMetaLV — smaller fast LV
 - Tracks which blocks are cached and dirty state
- CachePoolLV — CacheDataLV + CacheMetaLV
- CacheLV — OriginLV + CachePoolLV

The LV that gets used in the end, others hidden

Supports writethrough or writeback mode

Uses device mapper dm-cache target

Automated Storage Tiering with LVM Cache

Modern SSD drives or PCIe based flash storage provide significantly faster I/O performance than traditional hard disk drives. The difference is especially notable with random I/O where SSD and PCIe flash storage have no slow down in performance while hard disk drives can only muster 1/10th compared to their sequential I/O performance. That being said, traditional hard disks are much more economical on a price per gigabyte basis.

Because of this price and performance differential, automated tiered storage deployments that combine both slow+cheap with fast+expensive storage into a virtual disk with intelligent caching are very popular.

LVM Cache implements automated tiered storage for Linux by building on top of the device mapper dm-cache target. Technically the dm-cache target can be used to cache any block device in Linux, however, the LVM Cache approach is much easier without resorting to arcane `dmsetup` commands.

LVM Cache Prerequisites

First identify or create an LVM LV that is composed of slow+cheap PVs, it will be referred to as `Videos` in the example below. Secondly, identify the fast+expensive PVs that will be used for the cache. The best practice is to make use of LVM Object tags. The following examples assume the slow+cheap PVs are tagged with `hdd` and that the fast+expensive PVs are tagged with `ssd`. They must be part of

the same Volume Group — `vg0` in the example:

```
# pvs -o +tags
PV          VG      Fmt Attr PSize   PFree   PV Tags
/dev/sdb    vg0     lvm2 a-- 21.83t  3.77t   hdd
/dev/sdc    vg0     lvm2 a-- 893.75g 893.75g ssd
# lvcreate -n Videos -L 20TB vg0 @hdd
```

Caching a Large and Slow LV

To cache a slow LV it needs to be attached to a cache pool. A cache pool consists two LVs, one that will hold the cached data, often times called the cache data LV, and another LV that tracks the state of the cache, often called the cache metadata LV. The cache metadata LV should be 1000 times smaller than the cache data LV with a minimum size of 8 MiB. The `lvcreate` can do the math automatically and create a cache pool with a single command:

```
# lvcreate --type cache-pool -l 100%FREE -n CachePoolLV vg0 @ssd
```

Then, to attach the slow LV to the cache pool, use:

```
# lvconvert --type cache --cachepool vg0/CachePoolLV vg0/Videos
```

The original slow LV will be renamed to `Videos_corig` and be made a hidden subvolume. A cache LV will be created with the same name as the original slow LV so that configuration files such as `/etc/fstab` don't need to be updated.

Cache Mode and Statistics

By default, a cache pool operates in writethrough mode meaning that writes are not cached, only reads. This way if the fast+expensive PVs fail, no data is lost. If the fast+expensive PV(s) are redundant by way of either a hardware RAID controller or by building the cache pool using LVM's built-in RAID support as outlined in the man page, then cache pool can be reconfigured to operate in writeback mode so that writes are cached as well as reads and overall performance is increased. This can be configured when creating the cache pool; for example:

```
# lvcreate --type cache-pool --cachemode writeback -l 100%FREE -n CachePoolLV vg0 @ssd
```

To monitor the performance of the caching, use **lvs** and specify the relevant cache related field names; for example:

```
# lvs -o lv_name,cachemode,cache_total_blocks,cache_used_blocks,cache_policy vg0/Videos
  LV      Cachemode CacheTotalBlocks CacheUsedBlocks Cache Policy
  Videos writeback        14577600          84479    mq
# lvs -o lv_name,cache_read_hits,cache_read_misses,cache_write_hits,cache_write_misses vg0/Videos
  LV      CacheReadHits   CacheReadMisses CacheWriteHits   CacheWriteMisses
  Videos     4493383        15220685       16367108       8947597
```

Advanced LVM: Thin Provisioning

LVM Thin Provisioning — On-demand allocation of PEs

- Standard LVs allocate all PEs at creation time

Thin LVs have an apparent size

- Can be any size including larger than actual available storage
- Allocated extents come from ThinPool

ThinPool

- ThinPool is composed of two LVs
 - Data LV — Pool of available PEs, arbitrary size
 - Metadata LV — Tracking, 16 Gib max size

Snapshots of thin LVs are more efficient than normal snapshots

- Identical PEs shared amongst multiple snapshots

Thin Provisioning Architecture & Theory

Thin provisioning allows for over subscribing storage. The classic use case is with virtual machines, each virtual machine has a virtual disk that appears to be very large, but the actual size of the virtual machine's disk corresponds to the actual amount of data written to the disk. A thinly provisioned storage object will grow in actual size up until either the actual size hits the apparent size, or until the physical storage runs out of space. A thinly provisioned storage object is slightly less performant because of the extra work the storage layer must perform to expand the object. Thin provisioned objects are also more susceptible to fragmentation since they don't allocate all their storage up front.

Thin Snapshots

Traditional LVM snapshots uses a COW technique. Snapshots with thinly provisioned objects use a different technique made possible by the advanced tracking done within the metadata LV. Thin snapshots can be taken of thin LV or other thin snapshots. Data that is common to nested snapshots is shared and not duplicated. There is no limit to the number of snapshots that can be taken and no performance degradation to nested snapshots.

Because of benefits of LVM thin snapshots, they are leveraged by other software. For example, GlusterFS snapshots require the use of LVM thin provisioning.

The Thin Pool

Before any thin LVs can be created, a thin pool must be defined. The thin pool contains two LVs, a data LV that holds allocated storage, and a metadata LV that keeps track of what data belongs to which thin LV. The metadata LV should be stored on fast PV(s), or at least separate PVs from the data LV, if possible. They both start out as regular LVs which then get converted into a thin pool LV. The following example creates a thin pool with 20 TiB of allocatable space:

```
# lvcreate -n ThinPool -L 20T vg0 @hdd
# lvcreate -n ThinMetaLV -L 16G vg0 @ssd
# lvconvert --type thin-pool --poolmetadata vg0/ThinMetaLV,
            vg0/ThinPool
```

The thin pool will take name of the data LV (ThinPool above), and the data LV and the metadata LV are renamed and hidden.

Creating Thin LVs

After the thin pool is created, one or more thin LVs can be created referencing the thin pool, and using the **-V** to indicate the virtual apparent size:

```
# lvcreate -n thinlv01 -V 15T --thinpool vg0/ThinPool
# lvcreate -n thinlv02 -V 15T --thinpool vg0/ThinPool
# lvcreate -n thinlv03 -V 15T --thinpool vg0/ThinPool
```

In this example, there are 3 thin LVs, each with a virtual apparent size of 15, which totals 45 TiB — greater than the size of the thin pool.

Managing Utilization

It's possible for both the metadata LV and data LV portions of a thin pool to run out of space. To guard against that, it's best practice to make the metadata LV the maximum size — 16 GiB, and then enable the autoextend feature of the LVM monitoring daemon in the `lvm.conf`. This is done by setting the `thin_pool_autoextend_threshold` setting to a value less than 100. It's notable that, unlike other types of LVs, thin pool LVs can only be extended, not reduced in size. The following settings trigger a thin pool to be extended by 20% of its current size, once it hits 70% utilization:

File: /etc/lvm/lvm.conf

```
thin_pool_autoextend_threshold = 70
thin_pool_autoextend_percent = 20
```

The monitoring daemon will send warning messages to syslog when the thin pool utilization reaches 80%, 85%, 90%, and 95%. For the autoextend feature to work, there must be free space within the VG.

The `lvs` and `lvdisplay` commands will show the current utilization. For example:

```
# lvs
  LV        VG  Attr      LSize   Pool Origin Data%
ThinPool    vg0  twi-a-tz-  20T          26.46
thinlv01    vg0  Vwi-a-tz-  15T  ThinPool     9.57
thinlv02    vg0  Vwi-a-tz-  15T  ThinPool    20.50
thinlv03    vg0  Vwi-a-tz-  15T  ThinPool     5.21
```

Thin Pool Security vs Performance Trade-Offs

As thin logical volumes grow and consume data out of the thin pool, the newly provisioned chunks are zeroed to prevent old deleted data, once attached to other thin logical volumes, from becoming accessible at the block level of the thin volume that is growing. In IT security nomenclature, this is referred to as information leakage. In single tenant environments, or where direct block device access is not possible by untrusted users, then turning off the zeroing can result in improved performance. This can be done using `lvchange`, and the zeroing status is visible with the `z` attribute:

```
# lvs -o lv_name,lv_attr vg0/ThinPool
  LV          Attr
ThinPool    twi-a-tz--
# lvchange --zero n vg0/ThinPool
Logical volume "ThinPool" changed.
# lvs -o lv_name,lv_attr vg0/ThinPool
  LV          Attr
ThinPool    twi-a-t---
```

Advanced LVM: Striping & Mirroring

Striping and Legacy Mirroring

- For Striping specify number of stripes and strip size
`-i|--stripes` and `-I|--stripesize`
- For Mirrors specify number of mirrors
`-m|--mirrors`

Mirroring deprecated in favor of RAID1 Volumes

- Exception! CLVM works with Legacy Mirrors, not yet with RAID

Number of stripes or mirrors can't exceed number of PVs in VG

Striping

Striping can increase performance by distributing I/O operations over multiple PVs. When creating a striped LV you must specify two arguments. The number of stripes, specified with `-i|--stripes`, which corresponds with how many PVs the LV will be striped over. The number of stripes can't exceed the number of PVs in the VG.

There also must be enough free extents on each PV to hold its share of data. For example, creating 600GB striped LV with three stripes requires 200GB of free extents on three separate PVs.

The other required argument is the stripe size, specified with `-I|--stripesize`, sometimes called the strip width. Here is an example of creating a 600GB, three-legged stripe with a 256 KiB stripe size:

```
# lvcreate -L 600G -i 3 -I 256 -n stripedLV vg0
```

Legacy Mirroring

Mirroring provides redundancy. With legacy mirroring in LVM you specify number of copies of the data with the `-m|--mirrors` argument. Each LE will be copied to the specified number of PEs stored on different PVs. Behind the scenes, LVM builds the mirror using `mirror_regions` which defaults to 512 KiB in size. The regions are kept in sync with the help of small on disk mirror log. With that default region size, the maximum size of the mirror is 1.5T TiB. To increase the size of the mirror, increase the size of the region by powers of 2. To support a 5 TiB mirror size, use `-R 8M` to specify an 8 MiB region size. For example:

```
# lvcreate -L 500G --type mirror -m 1 -n mirroredLV vg0
```

Note that the new RAID1 volume type implements mirroring using a different method so that regions and a mirror log are not needed. However, RAID1 volume types don't yet work in a Clustered LVM VG.

Specifying the `-m` argument without `--type mirror` results in a new style RAID1 volume being created.

Advanced LVM: RAID Volumes

New RAID1/4/5/6/10 Volume

- Specify the RAID type with `--type`
- For raid5/raid6/raid10 specify number of stripes and strip size
`-i|--stripes` and `-I|--stripesize` (defaults to 64 KiB)
- For raid1 specify number of mirrors
`-m|--mirrors`

Uses device mapper dm-raid target

- Integrates Linux software RAID (MD) with Device Mapper

lvconvert — Change characteristics of a logical volume

- Convert legacy mirror to RAID1
- Convert regular LV to RAID LV and vice versa
- Replace failed RAID device

RAID5 Example

```
# lvcreate -L 500G --type raid5 -i 3 -n raid5LV vg0
```

Each stripe is mapped to a separate PV. With RAID5, one additional PV is needed to store the parity

RAID6 Example

```
# lvcreate -L 500G --type raid6 -i 3 -n raid6LV vg0
```

Each stripe is mapped to a separate PV. With RAID6, two additional PVs are needed to store the parity

RAID10 Example

```
# lvcreate -L 500G --type raid10 -i 4 -n raid10LV vg0
```

This would create a RAID10 LV with a stripe on top of 4 two-way mirrors and require 8 PVs.

lvconvert

The lvconvert(8) man page has many examples, but the most important operation to know about is how to replace a failed PV. In the following example, /dev/sdc is the failed device to remove, and the replacement is /dev/sdf.

```
# lvconvert --replace /dev/sdc vg0/raid6LV /dev/sdf
```

If the replacement is omitted, LVM will search the VG for a suitable replacement.

New LVM RAID Implementation

The traditional striping and mirroring support in LVM uses Linux device mapper targets, specifically the mirror and striped targets. Meanwhile Linux has robust and full featured support for RAID types via MD, the Linux software RAID subsystem. Historically, when going with a pure software RAID and LVM configuration, the RAID volumes were configured and setup first, and then the RAID volumes were then used as physical volumes in a LVM VG.

The new architecture integrates MD transparently with LVM using the `dm-raid` target so that there is a unified management interface for creating logical volumes that use, behind the scenes, a given RAID configuration. A major difference in architecture is that extents from different physical volumes are used as the building blocks for RAID LVs. Thus, RAID LVs are only feasible in a Volume Group that contains multiple PVs. When attempting to create a RAID LV without a sufficient number of PVs in a Volume Group, an error will be generated. For example, attempting to create a RAID6 LV with three stripes in a VG with only one PV results in this error message:

```
Number of stripes must not exceed number of physical volumes
```

RAID1 Example

```
# lvcreate -L 500G --type raid1 -m 1 -n raid1LV vg0
```

Simply specifying `-m 1` is sufficient to create a raid1 LV. The `--type raid1` is superfluous.

SLES Graphical Disk Tool

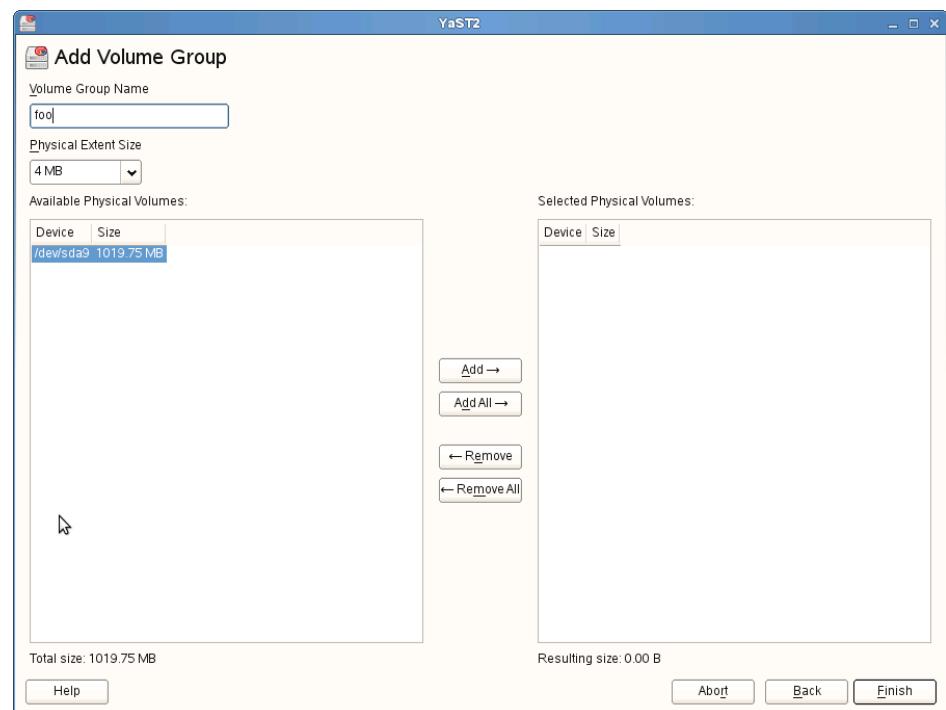
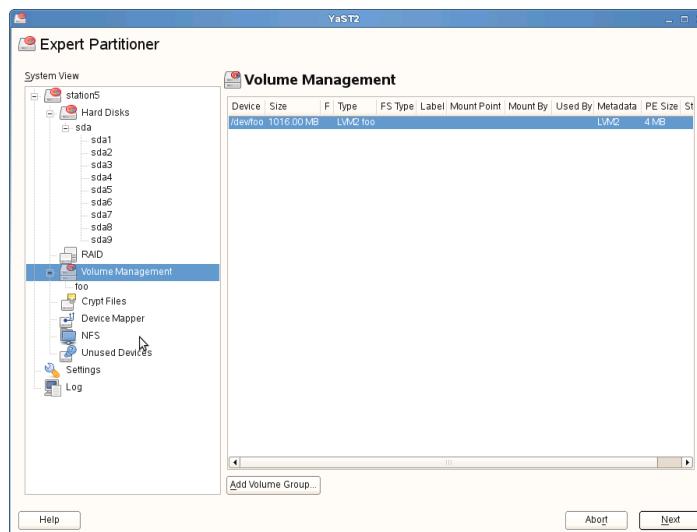
yast disk module

- Display VG, PV, LV properties
- Create/Remove VG
- Expand/Reduce VG
- Create/Remove PV
- Create/Expand/Remove LV

Expert Partitioner

Yast supports a graphical module for managing hard drive partitions, including managing logical volumes. It can be accessed by running **yast2 disk**. LVM features supported through the graphical utility include:

- ❖ Display information about volume groups, physical volumes and logical volumes
- ❖ Create/Remove volume groups
- ❖ Create/Remove physical volumes
- ❖ Create, expand, and remove logical volumes



RAID Concepts

What?

- Combining many physical devices into one logical device
- Implement in software or hardware

Why?

- Provide data redundancy
- Increase size
- Increase performance

Types

- Linear and RAID-0
- RAID-1
- RAID-5
- RAID-6

smaller, fragmented I/O operations that can be parallelized between the multiple physical disks.

RAID-1 ⇒ provides data redundancy by creating two physical devices which mirror each other; if one device fails, the system can be configured to switch to the mirror device automatically. Obviously, RAID-1 increases costs significantly, but it does make possible extremely high availability.

RAID-4 ⇒ uses block level striping, but unlike RAID-5, has a dedicated disk for parity. Requires 3 disks. Traditionally, performance problems can come from the block ordering, due to simultaneous access to blocks out of order.

RAID-5 ⇒ an attempt to get most of the benefits of both RAID-0 and RAID-1 by using both striping to increase performance and maximum capacity, and some data redundancy; rather than maintaining a live mirror of all data, RAID-5 maintains parity information from which data can be reconstructed if necessary.

RAID-6 ⇒ like RAID-5, however the amount of parity is doubled. This enables a RAID-6 volume to survive two disk failures.

Combinations of striping, mirroring, and parity are popular as disks are added, such as RAID 1+0 (striping across mirrors).

Linux supports many different hardware RAID controllers, as well as software RAID. Typically, software RAID is faster than hardware RAID, but it does place noticeable demands on the system CPU(s). Software and hardware RAID can even be combined in various ways, such as a RAID 1+0 configuration in which mirroring is done in hardware and striping is done in software.

Redundant Array of Independent Disks

RAID is a method of combining many physical disks to create one virtual disk. Virtual disks spanning multiple physical disks are useful for many reasons: they can be made to perform faster than a single physical disk, since they can allow read and write operations to be parallelized across many physical devices; they can be made larger than current physical limits for maximum drive size by appending multiple physical drives together to create one large virtual drive; and they can be made to provide data redundancy, by creating one virtual drive which stores all modifications made to it onto two or more physical drives. However, RAID cannot accomplish all these benefits simultaneously. An often-cited truism with RAID is "fast, safe, cheap—pick any two!" and that really is true to a large extent. RAID can be configured in a variety of different ways, depending upon the specific needs of the current situation.

Many different RAID configurations are possible, but the most popular RAID configurations include linear RAID, RAID-0, RAID-1, RAID-5 and RAID-6:

Linear RAID ⇒ one physical device is appended to another to create a large virtual device spanning the physical devices. Offers good performance for large contiguous read I/O operations that can leverage the A/V streaming modes of modern hard disks. Allows easily adding additional disks to extend the device.

RAID-0 ⇒ works much like linear RAID, but it stripes the virtual device across physical devices rather than first filling one device, then moving on to the next device. Offers good performance for

Array Creation with mdadm

`mdadm -h | --help`

Creation Mode

- An array can be created at install time
- Create partitions with type FD (using `parted`, or other partitioning software)
- `mdadm --create --help`
 - C | --create /dev/mdX
 - l | --level 0 | 1 | 4 | 5
 - n | --raid-devices # device [...]
 - x | --spare-devices # device [...]
- Create filesystem

mdadm Concepts

`mdadm` is an application that is used to create, manage, and monitor software RAID devices. `mdadm` is a single application, not a collection of separate programs. Unlike the older `raidtools` package, which depended on the `/etc/raidtab` configuration file, `mdadm` can perform most of its functions without the `/etc/mdadm.conf` configuration file.

There are several major modes of operation for the `mdadm` command, with a related option to access each mode. To view the list of correlated options for the different modes use the `--help` option or `--mode --help` for detailed help on a specific mode:

```
# mdadm --help
Usage: mdadm --create device options...
      mdadm --assemble device options...
. . . snip . . .
```

```
# mdadm --create --help
Usage: mdadm --create device -chunk=x --level=y -
      --raid-devices=z devices
. . . snip . . .
```

Creating the RAID Device

Linux Software RAID devices need to be created with partitions using types set to 0xFD. Use `mdadm`'s option `-C` to create a RAID device with these newly created partitions. There are several options required to create a RAID device, and additional options that modify the RAID device configuration. For example, to create a new RAID level 5 array that consists of four devices total, with one of them being configured as a spare, run the following:

```
# mdadm -C /dev/md0 -l 5 -n 3 /dev/sdd1 /dev/sde1,
      /dev/sdf1 -x 1 /dev/sdg1
mdadm: array /dev/md0 started.
```

Filesystem Creation

A filesystem can now be created on top of the RAID array. This is done with the `mkfs` wrapper, or filesystem specific formatting command, (e.g. `mke2fs`). For example, to create an Ext3 filesystem on the `/dev/md0` RAID device, run:

```
# mke2fs -j /dev/md0
```

Software RAID Monitoring

Monitoring Mode

- /etc/mdadm.conf
- **mdadm -F | --follow | --monitor**
- /proc/mdstat
- **mdmon**
- **systemd** mdmonitor.service

real-time information about RAID arrays and devices:

```
# cat /proc/mdstat
```

Personalities : [raid5]

```
md0 : active raid5 sdb1[0] sdc1[3] sdd1[2] sde1[1]
      224640 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]
      unused devices: <none>
```

The **systemd** mdmonitor service used by RHEL7/SLES12 calls **mdadm -F**, and the **mdmon@** service calls the external metadata array monitor provided by **mdmon**.

/etc/mdadm.conf

The /etc/mdadm.conf file is used to identify which devices are RAID devices and to which ARRAY a specific device belongs. The file supports comments by using the hash sign (#) at the beginning of a line. A line starting with white space is treated as a continuation of the previous line. The two main keywords are DEVICE and ARRAY.

An example /etc/mdadm.conf file:

```
DEVICE /dev/sda8 /dev/sda9 /dev/sda10 /dev/sda11
ARRAY /dev/md0 level=raid5 num-devices=3
      UUID=08807902:0233e1ea:e665f06c:e9c13002
      devices=/dev/sda8,/dev/sda9,/dev/sda10,/dev/sda11
```

Monitoring an Array

The systemd service **mdmonitor.service** is used to monitor RAID-1, RAID-5, or multipath arrays. By default, it uses the **mdadm** command with the **--monitor** option and runs **mdadm** as a daemon.

The /etc/mdadm.conf file is read for one of two variables that must be defined within the file. The two variables are MAILADDR and PROGRAM. MAILADDR is used to define an email address for alert notification. PROGRAM contains a path to a specific program that executes when specific events trigger the monitor. The **mdadm** monitors any arrays that are defined within the /etc/mdadm.conf file and any arrays found in /proc/mdstat.

/proc/mdstat is a virtual file maintained by the kernel which contains

Software RAID Control and Display

```
mdadm --manage
  • -f | --fail | --set-faulty
  • -r | --remove
  • -a | --add
mdadm --grow
mdadm --misc
  • -Q | --query
  • -D | --detail
  • -o | --readonly
  • -w | --readwrite
```

Managing Devices

To manage member devices of an array the `mdadm --manage` command can be used with several options. The `-f` (or `--fail` or `--set-faulty`) option can be used to fail a member device. This is useful for testing the reliability of a RAID level 1 or 5 device. The `-r` (or `--remove`) option is used to remove a member device from an array. The `-a` (or `--add`) option will add a member device and associate it to an array.

If a drive has failed, or is about to fail the following procedure should be used:

```
# mdadm --manage /dev/md0 -f /dev/sdc1
# mdadm --manage /dev/md0 -r /dev/sdc1
```

At this point the drive should be replaced with a working drive, and a partition defined with type fd. Then the drive can be added back to the RAID volume with:

```
# mdadm --manage /dev/md0 -a /dev/sdc1
```

Miscellaneous Mode

The `mdadm --misc` command is used to control and display information about a specific device. For example to determine which array the device `/dev/sde1` belongs to, use the `-Q` (or `--query`) option:

```
# mdadm --misc -Q /dev/sde1
. . . output omitted . . .
```

To display details about the `/dev/md0` RAID device, use the `-D` (or `--detail`) option:

```
# mdadm --misc -D /dev/md0
. . . output omitted . . .
```

`mdadm --misc` can also mark a RAID device as read-only or read-write with the `-o` (or `--readonly`) and `-w` (or `--readwrite`) options respectively.

Lab 6

Estimated Time:
S12: 75 minutes
R7: 75 minutes

Task 1: Creating and Managing LVM Volumes

Page: 6-24 Time: 20 minutes

Requirements:  (1 station)

Task 2: Creating LVM Thin Volumes

Page: 6-34 Time: 20 minutes

Requirements:  (1 station)

Task 3: Troubleshooting Practice: LVM

Page: 6-41 Time: 10 minutes

Requirements:  (1 station)

Task 4: Creating and Managing a RAID-5 Array

Page: 6-42 Time: 25 minutes

Requirements:  (1 station)

Objectives

- ❖ Create a new Volume Group and Logical Volume
- ❖ Create and use an LVM snapshot
- ❖ Extend a VG, LV and filesystem

Requirements

- ▀ (1 station)

Relevance

LVM makes the long term maintenance of both server and workstation storage significantly easier. Instead of predicting the probable usage of a machine, LVM makes it easy to respond to real needs as they are discovered.

Notices

- ❖ Normally, multiple physical volumes would require multiple physical disks (or SAN block devices). For convenience, this lab will use multiple partitions on a single disk.

- 1) Verify that the `lvm2` and `xfsprogs` packages are installed:

```
$ rpm -q lvm2 xfsprogs  
. . . output omitted . . .
```

If either package is missing, **install** it.

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 3) Create a record of the current partition table to make cleanup easier later:

```
# sfdisk -d /dev/sda > /root/part.table
```

- 4) Create three new partitions of type 8e, two 250M in size, and one 1250M in size. For example, if two primary partitions are already used, do the following with `fdisk`:

```
# fdisk /dev/sda
```

Lab 6

Task 1

Creating and Managing LVM Volumes

Estimated Time: 20 minutes

```
... output omitted (intro text) ...

Command (m for help): p
... output omitted (the current partition table) ...

Command (m for help): n
Partition type:
  p  primary (2 primary, 0 extended, 2 free)
  e  extended
Select (default p) p
Partition number (3,4, default 3): 3
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +250M
Partition 3 of type Linux and of size 250 MiB is set

Command (m for help): n
Command action
  p  primary (3 primary, 0 extended, 1 free)
  e  extended
Select (default e) e
Selected partition 4
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): 
Using default value y
Partition 4 of type Extended and of size 5.1 GiB is set

Command (m for help): n
All primary partitions are in use
Adding logical partition 5
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +250M
Partition 5 of type Linux and of size 250 MiB is set

Command (m for help): n
All primary partitions are in use
Adding logical partition 6
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +1250M
Partition 6 of type Linux and of size 1.2 GiB is set
```

- The size will likely be different, depending on your lab environment.

```
Command (m for help): p
. . . output omitted (the new partition table with the two new partitions) . . .

Command (m for help): t
Partition number (1-6, default 6): 3 • Enter the number of the first partition created.
Hex code (type L to list codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): t
Partition number (1-6, default 6): 5
Hex code (type L to list codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): t
Partition number (1-6, default 6): 6 • Enter the number of the last partition just created.
Hex code (type L to list codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

- 5) Record the numbers of the newly created partitions in the following table. The following is consistent with the previous step's example:

Device Name	Partition Number
/dev/sdaW (250MB)	3
/dev/sdax (250MB)	5
/dev/sdaY (1250MB)	6

- 6) Update the kernel's in-memory copy of the partition table:

```
# partprobe /dev/sda
```

It may be of value to verify in /proc/partitions that the new partitions are seen and are of the correct size. If necessary, reboot and let the kernel detect the partitions at startup.

7) [S12] *This step should only be performed on SLES12.*

Enable verbosity for various LVM commands, to ensure commands were successful:

```
# sed -i 's/silent = 0/' /etc/lvm/lvm.conf
```

8) Physical volumes must be created before a volume group can be created.

Remember, physical volumes can be created from whole disks, meta devices (RAID), loopback files, or partitions. You will be using the partitions you just created as the physical volumes. Be VERY CAREFUL that you are operating on the correct partitions; replace the devices below with the physical volume devices that you recorded earlier:

```
# pvcreate /dev/sda{W,X,Y}
Physical volume "/dev/sdaW" successfully created
Physical volume "/dev/sdaX" successfully created
Physical volume "/dev/sdaY" successfully created
```

- If you get a prompt about removing an existing filesystem signature, select **y** if you're sure that the correct partition numbers are used, and that those partitions are not in use. Check with your instructor if unsure.

9) Create a new volume group named vg01, define the physical extents to be 16MB in size and use only the two 250M physical volumes.

```
# vgcreate -s 16M vg01 /dev/sda{W,X}
Volume group "vg01" successfully created
```

10) Run the **vgs** command and note how many megabytes are available in the volume group.

```
# vgs vg01
VG #PV #LV #SN Attr   VSize   VFree
vg01   2    0    0 wz--n- 480.00m 480.00m
```

The combined size of all the logical volumes that can be created can not exceed the VSize value (unless additional physical volumes are added to this volume group).

- 11) Now that the volume group has been created, logical volumes can be created. In this scenario you would like to move all of your in-house databases onto a logical volume. This will enable you to take LVM snapshots and do dynamic re-sizing of the logical volume. Create a 352MB logical volume and name it databases:

```
# lvcreate -L 352M -n databases vg01
Logical volume "databases" created
```

- 12) With the databases logical volume created, an xfs filesystem can now be created on it:

```
# mkfs.xfs /dev/vg01/databases
meta-data=/dev/vg01/databases isize=256    agcount=4, agsize=22528 blks
          =                     sectsz=512  attr=2, projid32bit=1
          =                     crc=0     finobt=0
data      =                     bsize=4096   blocks=90112, imaxpct=25
          =                     sunit=0     swidth=0 blks
naming    =version 2           bsize=4096   ascii-ci=0 ftype=0
log       =internal log        bsize=4096   blocks=853, version=2
          =                     sectsz=512  sunit=0 blks, lazy-count=1
realtime  =none                extsz=4096   blocks=0, rtextents=0
```

- 13) Mount the logical volume to a temporary directory /mnt/ and then move the contents of the current /labfiles/databases directory to the /mnt/ directory and then create a mount point for the logical volume at /srv/databases/:

```
# mount /dev/vg01/databases /mnt/
# cp -a /labfiles/databases/* /mnt/
# ls -l /mnt/
total 2634
-rw-r--r-- 1 root root 821112 Jan  9  2014 finance
-rw-r--r-- 1 root root 907493 Jan  9  2014 hr
-rw-r--r-- 1 root root 936478 Jan  9  2014 marketing
# umount /mnt/
# mkdir /srv/databases/
```

- 14) Edit the /etc/fstab file so that the new /dev/vg01/databases LVM logical volume will be mounted at /srv/databases:

File: /etc/fstab
+ /dev/vg01/databases /srv/databases xfs defaults 1 2

- 15) Verify that the new filesystem mounts correctly:

```
# mount /srv/databases
# mount | grep databases
/dev/mapper/vg01-databases on /srv/databases type xfs (rw)
# ls -l /srv/databases/
total 2634
-rw-r--r-- 1 root root 821112 Jan  9 2014 finance
-rw-r--r-- 1 root root 907493 Jan  9 2014 hr
-rw-r--r-- 1 root root 936478 Jan  9 2014 marketing
```

- 16) A snapshot is a picture, or frozen image, of a filesystem at the time the snapshot was created. Snapshots make it easier to take consistent backups of a filesystem. Prepare to take a consistent backup of /srv/databases/ by taking a snapshot of it and mounting the snapshot to a temporary directory:

```
# lvcreate -s -L 100M -n snap1 /dev/vg01/databases
  Rounding up size to full physical extent 112.00 MB
  Logical volume "snap1" created
# mount -o nouuid /dev/vg01/snap1 /mnt
```

- Create the snapshot with a 100MB change buffer.
- The Linux kernel XFS filesystem doesn't allow two XFS filesystems to be mounted with the same UUID. Use the nouuid option to disable this check.

- 17) Use the tar command to create a consistent backup of the /dev/vg01/snap1 logical volume:

```
# cd /mnt/
# tar -cJvf /tmp/databases-backup.tar.xz .
. . . output omitted . . .
# cd
```

- 18) Verify that the snapshot logical volume stays the same even though new data is added to the /srv/databases directory. First do a quick visual comparison to see that the top level of /srv/databases/ is the same as the snapshot mounted on /mnt/:

```
# ls -l /srv/databases /mnt/
```

```
/mnt:  
total 2634  
-rw-r--r-- 1 root root 821112 Jan  9  2014 finance  
-rw-r--r-- 1 root root 907493 Jan  9  2014 hr  
-rw-r--r-- 1 root root 936478 Jan  9  2014 marketing  
  
/srv/databases:  
total 2634  
-rw-r--r-- 1 root root 821112 Jan  9  2014 finance  
-rw-r--r-- 1 root root 907493 Jan  9  2014 hr  
-rw-r--r-- 1 root root 936478 Jan  9  2014 marketing
```

Notice that the two directories are the same.

- 19) Examine the snapshot statistics to determine how much of the 100MB buffer is currently being used:

```
# lvs /dev/vg01/snap1  
  LV   VG Attr       LSize   Pool Origin  Data%  Move Log Cpy%Sync Convert  
snap1 vg01 swi-aos--- 112.00m      databases    0.02
```

The Data% field shows the change buffer statistics (snapshot allocation).

- 20) Create a new file on /srv/databases/ then verify that the snapshot does not change:

```
# cp /labfiles/guru-words /srv/databases/  
# ls -l /srv/databases/  
total 4360  
... snip ...  
-rw-r--r-- 1 root root 1793076 Aug 27 14:34 guru-words • The new file appears.  
# ls -l /mnt/  
total 2608  
-rw-r--r-- 1 root root 821112 Jan  9  2014 finance  
-rw-r--r-- 1 root root 907493 Jan  9  2014 hr  
-rw-r--r-- 1 root root 936478 Jan  9  2014 marketing
```

Note that the guru-words file is not listed on /mnt/. The snapshot is working as intended.

- 21) Flush the disk cache then check the snapshot statistics. Notice that the change buffer size has increased:

```
# sync  
# lvs /dev/vg01/snap1  
  LV   VG Attr       LSize   Pool Origin   Data%  Move Log Cpy%Sync  Convert  
snap1  vg01 swi-aos-- 112.00m      databases    0.39
```

The Data% field value should be larger than the last time **lvs** was run.

- 22) Having verified that snapshotting is working correctly, remove the snapshot:

```
# umount /mnt/  
# lvremove /dev/vg01/snap1  
Do you really want to remove active logical volume "snap1"? [y/n]: y  
Logical volume "snap1" successfully removed
```

- 23) Suppose your database content is scheduled for an update and will soon greatly increase in size so that it will exceed the 352MB you currently have allocated to the /dev/vg01/databases logical volume. You need the /dev/vg01/databases logical volume to be 1024MB in size. Determine how much free space exists in the volume group:

```
# vgs vg01  
  VG #PV #LV #SN Attr   VSize   VFree  
vg01   2   1   0 wz--n- 480.00m 128.00m
```

The vg01 volume group is 480MB in size, the logical volume is 352MB in size. There is only 128MB free in the volume group. Even if you extended the databases logical volume to fill the entire volume group, you won't reach the 1024MB needed.

- 24) Initially two 250MB and one 1250MB physical volumes were created but only the two 250MB physical volumes were added to the vg01 volume group. This can be easily shown using pvscan:

```
# pvscan  
PV /dev/sdaW   VG vg01           lvm2 [240.00 MiB / 0     free]  
PV /dev/sdaX   VG vg01           lvm2 [240.00 MiB / 128.00 MiB free]  
PV /dev/sda2   VG vg0             lvm2 [34.18 GiB / 22.21 GiB free]
```

```
PV /dev/sdaY          lvm2 [1.22 GiB]-----  
Total: 4 [35.87 GiB] / in use: 3 [34.65 GiB] / in no VG: 1 [1.22 GiB]
```

- Notice that this physical volume has no corresponding volume group.

25) Extend the volume group by adding the unused physical volume:

```
# vgextend vg01 /dev/sdaY  
Volume group "vg01" successfully extended
```

Although you added another partition on the same drive, it could just have easily been another drive or SAN LUN that was hot-added to the system.

26) Display the statistics from the volume group:

```
# vgs vg01  
VG #PV #LV #SN Attr VSize VFree  
vg01 3 1 0 wz--n- 1.69g 1.34g
```

- Notice that the VG has increased in size.

27) Now that the volume group has free space, grow the /dev/vg01/databases logical volume:

```
# lvextend -L 1024M /dev/vg01/databases  
Extending logical volume databases to 1.00 GB  
Logical volume databases successfully resized
```

28) Notice that although the logical volume has been extended, the size of the xfs filesystem has not changed:

```
# lvs /dev/vg01/databases  
LV      VG Attr    LSize   Pool Origin  Data%  Move Log Cpy%Sync Convert  
databases vg01 -wi-ao---- 1.00g-----  
# df -h /srv/databases/  
Filesystem      Size  Used  Avail  Use%  Mounted on  
/dev/mapper/vg01-databases        333M  5.0M  307M     2%  /srv/databases
```

- The LV is now 1.00 GB.

- The filesystem has not grown.

29) To take advantage of the new space, the filesystem must also be resized. The xfs_growfs command supports growing a filesystem while it is still mounted. When run without any size parameters, it will use all available space:

```
# xfs_growfs /dev/vg01/databases
```

. . . snip . . .

- 30) Check the size of the filesystem again. Notice that it is now larger:

```
# df -h /srv/databases/
Filesystem           Size   Used   Avail   Use%   Mounted on
/dev/mapper/vg01-databases
                     984M   5.7M   930M    2%   /srv/databases
```

Cleanup

- 31) To avoid potential problems with future labs, and to demonstrate tearing down a volume group, remove vg01 and its related PVs:

```
# umount /srv/databases/
# lvremove /dev/vg01/databases
Do you really want to remove active logical volume databases? [y/n]: y
Logical volume "databases" successfully removed
# vgremove vg01
Volume group "vg01" successfully removed
# pvremove /dev/sda{W,X,Y}
Labels on physical volume "/dev/sdaW" successfully wiped
Labels on physical volume "/dev/sdaX" successfully wiped
Labels on physical volume "/dev/sdaY" successfully wiped
```

- 32) Remove the /etc/fstab entry, and the directory to which the logical volume was mounted, and restore the original partition table that was backed up at the beginning of the lab:

```
# rmdir /srv/databases/
# sed -i '/databases/d' /etc/fstab
# sfdisk -f /dev/sda < /root/part.table
# partprobe /dev/sda
```

- 33) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Create an LVM thin pool and volume.
- ❖ Observe consequences of running out of data space in a thin pool.
- ❖ Expand a thin pool.

Requirements

- ▀ (1 station)

Relevance

Thin volumes allow you to create devices that do not take any space from the storage pool until data is actually written by an application, allowing storage to be fully utilized. Care, however, must be taken to ensure that the thin pools never run out of space.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Use the command vgs to determine how much free space exists in the volume group vg0:

```
# vgs  
VG #PV #LV #SN Attr VSize VFree  
vg0 1 4 0 wz--n- 34.18g 22.68g
```

Result: _____

- 3) Before a thin volume can be created, a thin pool must be created. The thin pool acts as the backing store for the used space in the thin volume.

Use the command lvcreate to create a thin pool which is backed by 1GiB of physical storage:

```
# lvcreate -L 1G --thinpool thinpool0 vg0  
[R7] Using default stripesize 64.00 KiB.  
Logical volume "thinpool0" created.
```

Lab 6

Task 2

Creating LVM Thin Volumes

Estimated Time: 20 minutes

- 4) Now that a thin pool exists, a thin volume can be created within the pool. Create a thin volume with the name data0 and a virtual size of 5GiB:

```
# lvcreate vg0/thinpool0 -V 5G -n data0
[R7] Using default stripesize 64.00 KiB.
[R7] WARNING: Sum of all thin volume sizes (5.00 GiB) exceeds → • This is warning should be heeded in the real world.
           the size of thin pool vg0/thinpool0 (1.00 GiB)!
[R7] For thin pool auto extension activation/thin_pool_autoextend_threshold should be below 100.
Logical volume "data0" created.
```

- 5) Examine the output of vgs. Take note of how much space is free in the volume group, and compare it to the previous run of the vgs command:

```
# vgs
VG #PV #LV #SN Attr   VSize  VFree
vg0   1   6   0 wz--n- 34.18g 21.67g
```

Result: _____

- 6) Use the command lvs to view the reported size of the thin pool thinpool0, and the logical volume data0:

```
# lvs | awk 'NR==1; /thin/'
  LV      VG Attr       LSize   Pool      Origin Data%  Meta%  Move Log Cpy%Sync Convert
data0    vg0 Vwi-a-tz--  5.00g thinpool0          0.00
thinpool0 vg0 twi-aotz--  1.00g                  0.00   1.07
```

By looking at the Attr column in the output, the thin pool can be identified as the first attribute is t. The thin volume can be identified by the leading attribute of V. See the NOTES section of the man page LVS(8) for a full explanation of the various attributes.

- 7) As a shortcut, the option -T or --thin when applied to lvcreate can be used to create both a thin pool and thin volume simultaneously.

Use the -T option to create a new thin volume which will be 8GiB of virtual storage backed by 1GiB of actual storage:

```
# lvcreate -L 1G -T vg0/thinpool1 -V 8G -n data1
[R7] Using default stripesize 64.00 KiB.
[R7] WARNING: Sum of all thin volume sizes (13.00 GiB) exceeds the size of thin pools (2.00 GiB)!
[R7] For thin pool auto extension activation/thin_pool_autoextend_threshold should be below 100.
Logical volume "data1" created.
```

- 8) Examine the output of vgs:

```
# vgs
VG #PV #LV #SN Attr   VSize  VFree
vg0   1   8   0 wz--n- 34.18g 20.66g
```

- 9) View the sizes of the thin pools and thin volumes, pay special attention to the Data% column:

```
# lvs | awk 'NR==1; /thin/'
  LV    VG Attr       LSize   Pool      Origin  Data%  Meta%  Move Log Cpy%Sync Convert
data0    vg0  Vwi-a-tz--  5.00g  thinpool0        0.00
data1    vg0  Vwi-a-tz--  8.00g  thinpool1        0.00
thinpool0 vg0  twi-aotz--  1.00g                0.00   1.07
thinpool1 vg0  twi-aotz--  1.00g                0.00   1.07
```

- 10) Create an XFS filesystem on the data0 and data1 thin volumes:

```
# mkfs -t xfs /dev/mapper/vg0-data0
. . . output omitted . .
# mkfs -t xfs /dev/mapper/vg0-data1
. . . output omitted . .
```

- 11) Create a mount point for both of the newly created filesystems then mount them:

```
# mkdir /mnt/data{0,1}
# mount /dev/mapper/vg0-data0 /mnt/data0/
# mount /dev/mapper/vg0-data1 /mnt/data1/
```

- 12) Verify both filesystems mount and take note of the used space on each:

```
# df -h /mnt/data?
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg0-data0  5.0G  33M  5.0G  1% /mnt/data0
/dev/mapper/vg0-data1  8.0G  33M  8.0G  1% /mnt/data1
```

- 13) Create a 500MiB file on thin volume data0:

```
# dd if=/dev/urandom of=/mnt/data0/testfileA bs=1M count=500
500+0 records in
500+0 records out
524288000 bytes (524 MB) copied, 1.00491 s, 522 MB/s
```

• This will take a moment.

- 14) Examine how much space is being used by both the filesystem and the thin volume:

```
# df -h /mnt/data0/
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg0-data0  5.0G  533M  4.5G  11% /mnt/data0
# lvs | awk 'NR==1; /thin/'
  LV      VG Attr   LSize   Pool   Origin Data%  Meta%
  data0   vg0  Vwi-aotz--  5.00g  thinpool0    9.98
  data1   vg0  Vwi-aotz--  8.00g  thinpool1    0.13
  thinpool0 vg0  twi-aotz--  1.00g
  thinpool1 vg0  twi-aotz--  1.00g
```

• 500MiB is 10% of 5GiB.
• 500MiB is 50% of 1GiB.

- 15) Create another file on data0, this time 700MiB in size:

```
# dd if=/dev/urandom of=/mnt/data0/testfileB bs=1M count=700
700+0 records in
700+0 records out
734003200 bytes (734 MB) copied, 102.167 s, 7.2 MB/s
```

• Don't panic. This will take a moment, and will silently fail.

- 16) It appears that the previous command succeeded, that the 5GiB thin volume backed by 1GiB of actual storage is magically storing 1.2GiB of data. However, it is lying.

Examine how much space is being used on the thin volume and the filesystem:

```
# ls -lh /mnt/data0/
total 1.2G
-rw-r--r--. 1 root root 500M Oct  3 08:48 testfileA
-rw-r--r--. 1 root root 700M Oct  3 09:42 testfileB
# df -h /mnt/data0/
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg0-data0 5.0G  1.3G  3.8G  25% /mnt/data0
# lvs | awk 'NR==1; /thin/'
LV      VG Attr       LSize   Pool      Origin  Data%  Meta%
data0    vg0 Vwi-aotz--  5.00g thinpool0          20.00
data1    vg0 Vwi-aotz--  8.00g thinpool1          0.13
thinpool0 vg0 twi-aotzD-  1.00g                  100.00 13.28
thinpool1 vg0 twi-aotz--  1.00g          1.05    1.07
```

- Remember that this file is 700MiB, it'll come up again in a few more steps.
- Neat! Storing 1.3GiB of data using only 1GiB of storage! If only that were possible.
- D in the attributes stands for Out of Data Space.

- 17) Look at dmesg to see what thin pool errors look like:

```
# dmesg -ePL
. . . snip . . .
[ 296.605946] device-mapper: thin: 253:6: reached low water mark for data device: sending event.
[ 296.664526] device-mapper: thin: 253:6: switching pool to out-of-data-space (queue IO) mode
[ 356.732176] device-mapper: thin: 253:6: switching pool to out-of-data-space (error IO) mode
[ 356.753529] Buffer I/O error on dev dm-8, logical block 327664, lost async page write
. . . snip . . .
```

- 18) Unmount and run xfs_repair on the data0 filesystem, since writing abruptly stopped the filesystem is likely damaged:

```
# umount /mnt/data0
# xfs_repair /dev/mapper/vg0-data0
. . . output omitted . . .
```

- 19) Remount the data0 filesystem and look at how much data is stored, note that testfileB is 200MiB smaller than in the previous step:

```
# mount /dev/mapper/vg0-data0 /mnt/data0/
# ls -lh /mnt/data0/
total 1012M
-rw-r--r--. 1 root root 500M Oct  3 08:48 testfileA
-rw-r--r--. 1 root root 512M Oct  3 09:42 testfileB
```

• The size of the file decreased to 512MiB.

Take note of the silent data loss. When using LVM Thin Provisioning care has to be taken to ensure that there is always enough free space in the thin pool to support all the thin volumes.

- 20) The thin pools can be expanded as easily as expanding a normal logical volume. Expand the thin pool thinpool0 to 5GiB:

```
# lvresize -L +4G vg0/thinpool0
[R7] WARNING: Sum of all thin volume sizes (13.00 GiB) exceeds the size of thin pools (6.00 GiB)!
[R7] For thin pool auto extension activation/thin_pool_autoextend_threshold should be below 100.
[R7] Size of logical volume vg0/thinpool0_tdata changed from 1.00 GiB (256 extents) to 5.00 GiB (1280 extents).
Logical volume vg0/thinpool0_tdata successfully resized.
```

- 21) thinpool0 now has 80% free space:

```
# lvs | awk 'NR==1; /thin/'
  LV      VG Attr       LSize   Pool      Origin  Data%  Meta%  Move Log Cpy%Sync Convert
  data0    vg0 Vwi-aotz--  5.00g  thinpool0            20.00
  data1    vg0 Vwi-aotz--  8.00g  thinpool1            0.13
  thinpool0 vg0 twi-aotz--  5.00g            20.00  13.67
  thinpool1 vg0 twi-aotz--  1.00g            1.05   1.07
```

Clean up

- 22) Unmount the filesystems:

```
# umount /mnt/data0 /mnt/data1
```

- 23) Remove the thin volumes:

```
# lvremove -f vg0/data0 vg0/data1
Logical volume "data0" successfully removed
Logical volume "data1" successfully removed
```

- 24) Remove the thin pools:

```
# lvremove -f vg0/thinpool0 vg0/thinpool1
Logical volume "thinpool0" successfully removed
Logical volume "thinpool1" successfully removed
```

- 25) Remove the mount directories:

```
# rmdir /mnt/data0/ /mnt/data1/
```

- 26) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Practice troubleshooting common LVM issues.

Requirements

- ☒ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You use these scripts to break your system in controlled ways, and then you troubleshoot the problem and fix the system.

- 1) Use tsmenu to complete the networking troubleshooting scenario:

Troubleshooting Group	Scenario Category	Scenario Name
Group 5	LVM	lvm-01.sh

Lab 6

Task 3

Troubleshooting Practice: LVM

Estimated Time: 10 minutes

Objectives

- ❖ Use fdisk to partition free space.
- ❖ Configure software RAID-5 with a hot spare.
- ❖ Fail a member device of the array then observe the automatic recovery using the hot spare.
- ❖ Fail another member device then observe the array continue to function despite the lack of a hot spare.
- ❖ Replace failed member devices then observe the recovery of the array.

Requirements

- ▀ (1 station)

Relevance

RAID provides robust protection against data loss without any down time. Linux software RAID is very high quality, and easy to manage. Because of a good price/performance ratio, RAID-5 is a popular solution.

Notices

- ❖ The commands and output used in this lab are based on the first SCSI drive of the system. If you are using another drive, or non-SATA IDE drives, replace /dev/sda with the appropriate value, (e.g. /dev/hda).
- ❖ Normally, creating a RAID array would require multiple physical disks. However, for convenience, this lab is designed to use multiple partitions on a single disk.

- 1) Verify that the mdadm package is installed:

```
$ rpm -q mdadm  
. . . output omitted . . .
```

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 3) Create a record of the current partition table to make cleanup easier later:

```
# sfdisk -d /dev/sda > /root/part.table
```

Lab 6

Task 4

Creating and Managing a RAID-5 Array

Estimated Time: 25 minutes

- 4) Create a total of four 250MB RAID (type fd) partitions. (Depending on your current hard drive configuration, you might need to create an extended partition.) The first three (`sdaW`, `sdaX`, `sdaY`) will be used to create a RAID-5 device. The fourth (`sdaZ`) will be used as a hot spare.

For example, with fdisk:

```
# fdisk /dev/sda
. . . output omitted (intro text) . . .

Command (m for help): p
. . . output omitted (the current partition table) . . .

Command (m for help): n
Partition type:
  p  primary (2 primary, 0 extended, 2 free)
  e  extended
Select (default p)


Partition number (3,4, default 3): 3
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +250M
Partition 3 of type Linux and of size 250 MiB is set

Command (m for help): n
Command action
  p  primary (3 primary, 0 extended, 1 free)
  e  extended
Select (default e)
e
Selected partition 4
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): 
Using default value y
Partition 4 of type Extended and of size 5.1 GiB is set

Command (m for help): n
All primary partitions are in use
Adding logical partition 5
First sector (x-y, default x):


```

- The size will likely be different, depending on your lab environment.

```
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +250M
Partition 5 of type Linux and of size 250 MiB is set
```

```
Command (m for help): n
All primary partitions are in use
Adding logical partition 6
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +250M
Partition 6 of type Linux and of size 250 MiB is set
```

```
Command (m for help): n
All primary partitions are in use
Adding logical partition 7
First sector (x-y, default x): 
Using default value x
Last sector or +sectors or +size{K,M,G} (x-y, default y): +250M
Partition 7 of type Linux and of size 250 MiB is set
```

```
Command (m for help): p
. . . output omitted (the new partition table with the two new partitions) . . .
```

```
Command (m for help): t
Partition number (1-6, default 7): 3
Hex code (type L to list codes): fd
Changed type of partition 'Linux' to 'Linux raid autodetect'
```

- Enter the number of the first partition created.

```
Command (m for help): t
Partition number (1-6, default 7): 5
Hex code (type L to list codes): fd
Changed type of partition 'Linux' to 'Linux raid autodetect'
```

```
Command (m for help): t
Partition number (1-6, default 7): 6
Hex code (type L to list codes): fd
Changed type of partition 'Linux' to 'Linux raid autodetect'
```

- Enter the number of the last partition just created.

```
Command (m for help): t
Partition number (1-6, default 7): 
Hex code (type L to list codes): fd
Changed type of partition 'Linux' to 'Linux raid autodetect'
```

- Enter the number of the last partition just created.

```
Command (m for help): w
```

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.

- 5) Record the new partition numbers in the following table:

Device Name	RAID Element	Partition Number
/dev/sdaW	raid-device 0	
/dev/sdaX	raid-device 1	
/dev/sdaY	raid-device 2	
/dev/sdaZ	hot spare	

- 6) Update the kernel's in-memory copy of the partition table:

```
# partprobe /dev/sda
```

- 7) If this lab has been run on this system previously, the old RAID data structures may cause mdadm to complain that a partition appears to be a part of an existing RAID array. Even if other installations have occurred since the RAID lab was previously performed, the RAID structures may not have been overwritten by the interim installs.

Just to be safe, erase any old RAID superblocks:

```
# mdadm --zero-superblock /dev/sda{W,X,Y,Z}
```

• Results may vary. If there are no old RAID superblocks, the result will be a message of mdadm: Unrecognised md component device - /dev/sdaW

- 8) Use the `mdadm` command to create a new RAID-5 device:
(Remember, the first 3 partitions are being used to create the device. The 4th partition is being used as a hot spare.)

```
# mdadm -C /dev/md0 -l 5 -n 3 /dev/sda{W,X,Y} -x 1 /dev/sdaZ  
mdadm: Defaulting to version 1.2 metadata  
mdadm: array /dev/md/0 started.
```

- 9) Although the `mdadm` command in step 8 appears to complete almost immediately, in reality, the RAID array is building in the background. Monitor the progress of the build until the new raid device reports "active" as shown here:

```
# cat /proc/mdstat  
Personalities : [raid6] [raid5] [raid4]  
md0 : active raid5 sdaY[3] sdaZ[4] sdaX[1] sdaW[0]  
      497792 blocks level 5, 64k chunk, algorithm 2 [3/2] [UU_]  
      [==>.....] recovery = 15.5% (39040/248896) finish=0.2min speed=13013K/sec  
  
unused devices: <none>
```

GUI messages might suggest the array is degraded. This is normal for how the array is created..

- 10) Create an XFS filesystem on the new RAID 5 array:

```
# mkfs.xfs /dev/md0  
... output omitted ...
```

- If a prompt for using `-f` is requested, this may be the result of residual filesystem metadata being present. `-f` is fine in this case.

- 11) Create a new `/etc/mdadm.conf` file, which can be used to identify the newly created RAID array and its associated member devices:

```
# echo 'DEVICE /dev/sdaW /dev/sdaX /dev/sdaY /dev/sdaZ' > /etc/mdadm.conf
```

- 12) Use the `mdadm` command to gather information about the newly created array and store this information in the `/etc/mdadm.conf` file. This information is useful in identifying which member devices belong to a specific RAID array:

```
# mdadm -Ds >> /etc/mdadm.conf
```

```
# cat /etc/mdadm.conf
DEVICE /dev/sdaW /dev/sdaX /dev/sdaY /dev/sdaZ
ARRAY /dev/md0 level=raid5 metadata=1.2 spares=1 name=stationX.example.com:0
      UUID=6539208b:b7fffa64:302daeae:696838f3
```

- 13) Edit the /etc/fstab file so that the new /dev/md0 RAID device is used for /srv/extra/:

File: /etc/fstab
+ /dev/md0 /srv/extra xfs defaults 1 2

- 14) Create a directory to house the RAID array and mount it:

```
# mkdir /srv/extra/
# mount /srv/extra/
```

- 15) Verify that the /srv/extra/ filesystem has been mounted:

```
# mount | grep md0
/dev/md0 on /srv/extra type xfs (rw)
```

- 16) Store some data on the /srv/extra/ filesystem and generate a checksum for future integrity checking:

```
# tar -c /etc/ > /srv/extra/etc-backup.tar
tar: Removing leading `/' from member names
# md5sum /srv/extra/etc-backup.tar > /srv/extra/etc-backup.md5
```

- 17) In Step 18, the failure of raid-device 1 will be simulated. Before doing so, verify that mdadm assigned raid-device 0 to /dev/sdaW:

```
# mdadm -D /dev/md0
/dev/md0:
      Version : 1.2
      Creation Time : Thu Oct 16 13:43:38 2014
      Raid Level : raid5
      Array Size : 510976 (499.08 MiB 523.24 MB)
      Used Dev Size : 255488 (249.54 MiB 261.62 MB)
      Raid Devices : 3
```

```
Total Devices : 4
Persistence : Superblock is persistent

Update Time : Thu Oct 16 13:52:18 2014
State : clean

Active Devices : 3
Working Devices : 4
Failed Devices : 0
Spare Devices : 1

Layout : left-symmetric
Chunk Size : 512K

Name : stationX.example.com:0 (local to host stationX.example.com)
UUID : 46b10be2:7e97ce63:12502b1e:06172b24
Events : 18

Number  Major  Minor  RaidDevice State
  0      8       3      /dev/sdaW active sync
  1      8       5      /dev/sdaX active sync
  4      8       6      /dev/sdaY active sync
  3      8       7      /dev/sdaZ spare
```

- 18) Use mdadm to simulate a failure, then use watch to monitor the rebuild process. When /dev/md0 was initially created, a hot spare was defined. Because of the hot spare, recovery will begin immediately. On some systems, the recovery process will complete very quickly. Combine the mdadm and watch commands into a single line to ensure that watch is launched before the rebuild finishes:

```
# mdadm /dev/md0 -f /dev/sdaW; watch -n 1 "cat /proc/mdstat"
Personalities : [raid5]
md0 : active raid5 sdaY[2] sdaZ[0] sdaX[1] sdaW[3](F)
        497792 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]
unused devices: <none>
```

• This output is what will show after recovery completion.

• Exit when complete.

The string "sdaW[3](F)" indicates that /dev/sdaW has failed and was moved to be the last member ([3]) of the raid array. The string "sdaZ[0]" indicates that the hot spare was inserted into the failed device's position ([0]) and was automatically used for recovery.

- 19) As an alternate technique, the status of the RAID device /dev/md0 can also be inspected with mdadm. Use it to see that the *sdaZ[0]* hot spare was inserted and that the failed device, *sdaW*, is marked faulty:

```
# mdadm -D /dev/md0
. . . snip . .
  Number  Major  Minor  RaidDevice State
      0        8       12          0    active sync   /dev/sdaZ
      1        8       11          1    active sync   /dev/sdaX
      2        8       10          2    active sync   /dev/sdaY
      3        8       9           -    faulty spare  /dev/sdaW
```

- 20) Check that the data stored on the RAID array is still intact:

```
# md5sum -c /srv/extra/etc-backup.md5
/srv/extra/etc-backup.tar: OK
```

- 21) Remove the failed member device from the array:

```
# mdadm /dev/md0 -r /dev/sdaW
mdadm: hot removed /dev/sdaW from /dev/md0
```

This command would allow a physical drive to be removed from the system for replacement or repair.

- 22) Examine the contents of /proc/mdstat to verify that the member device was removed:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdaY[4] sdaZ[3] sdaX[1] • Notice that /dev/sdaW is no longer included.
      510976 blocks level 5, 512k chunk, algorithm 2 [3/3] [UUU]
unused devices: <none>
```

As an alternate technique use the same mdadm command from Step 19.

- 23) Because there isn't a hot spare anymore, the RAID array will not be able to automatically recover after another failure. However, /dev/md0 was created using RAID-5. The RAID array will be able to continue running in a degraded (slower) state. Test this by simulating the failure of raid-device 1:

```
# mdadm /dev/md0 -f /dev/sdax  
mdadm: set /dev/sdax faulty in /dev/md0
```

- 24) When a member device fails, the kernel logs this failure. Examine the error messages in /var/log/messages:

```
# tail /var/log/messages  
[R7] . . . snip . . .  
[R7] Jan 12 15:34:39 stationX kernel: md/raid:md0: Disk failure on sdax, disabling device.  
[R7] md/raid:md0: Operation continuing on 2 devices.  
[R7] . . . snip . . .  
[S12] Sep 8 13:06:30 stationX kernel: raid5: Operation continuing on 2 devices.  
[S12] Sep 8 13:06:30 stationX kernel: RAID5 conf printout:  
[S12] Sep 8 13:06:30 stationX kernel: --- rd:3 wd:2  
[S12] Sep 8 13:06:30 stationX kernel: disk 0, o:1, dev:sdaZ  
[S12] Sep 8 13:06:30 stationX kernel: disk 1, o:0, dev:sdaX  
[S12] Sep 8 13:06:30 stationX kernel: disk 2, o:1, dev:sdaY  
[S12] Sep 8 13:06:30 stationX kernel: RAID5 conf printout:  
[S12] Sep 8 13:06:30 stationX kernel: --- rd:3 wd:2  
[S12] Sep 8 13:06:30 stationX kernel: disk 0, o:1, dev:sdaZ  
[S12] Sep 8 13:06:30 stationX kernel: disk 2, o:1, dev:sdaY
```

• Notice that /dev/sdax is no longer listed.

- 25) Verify that all files on /srv/extra/ are still accessible:

```
# ls -lR /srv/extra/  
. . . output omitted . . .  
# md5sum -c /srv/extra/etc-backup.md5  
/srv/extra/etc-backup.tar: OK
```

If you wish, you can even edit the files in /srv/extra/. However, the RAID array is operating in a degraded mode. All files which were stored on the failed disk must be regenerated on the fly from parity information. There is no way of telling (other than by access time) that some of the files are being read directly from, and written to, disk while other files—the ones which were physically stored on the failed partition—are being mathematically generated as needed.

- 26) Because of the risk of data loss, failed disks should be replaced as soon as possible. This can be done even while the filesystem is mounted. Simulate the replacement of the failed hardware by removing the faulty device, and re-adding /dev/sdaw:

```
# mdadm /dev/md0 -r /dev/sdax
mdadm: removed /dev/sdax
# mdadm /dev/md0 -a /dev/sdaw
mdadm: added /dev/sdaw
```

- 27) Although the mdadm command in Step 26 appears to complete almost immediately, in the background the RAID array is rebuilding the data that used to be stored on the failed device. Monitor the progress of the rebuild until it finishes:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdaw[1] sday[2] sdaz[0]
      497792 blocks level 5, 64k chunk, algorithm 2 [3/2] [UU_]
      [=====]>..... recovery = 27.8% (69644/248896) finish=0.2min speed=9949K/sec
unused devices: <none>
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdaw[1] sday[2] sdaz[0]
      497792 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]
unused devices: <none>
```

When RAID reconstruction finishes, the RAID array will no longer be operating in degraded mode; you will have successfully replaced one of the failed member devices.

- 28) As before, the rebuild process generates several message in /var/log/messages. Examine these messages:

```
# tail -n 20 /var/log/messages
. . . output omitted . . .
```

- 29) Replace the hot spare by re-adding /dev/sdax:

```
# mdadm /dev/md0 -a /dev/sdax
mdadm: added /dev/sdax
```

- 30) Examine the new state of the RAID array. Note which devices are active and which device has been configured as the spare:

```
# mdadm -D /dev/md0
/dev/md0:
      Version : 1.2
Creation Time : Thu Oct 16 13:43:38 2011
      Raid Level : raid5
      Array Size : 510976 (499.08 MiB 523.24 MB)
  Used Dev Size : 255488 (249.54 MiB 261.62 MB)
      Raid Devices : 3
    Total Devices : 4
      Persistence : Superblock is persistent

        Update Time : Thu Oct 16 14:08:25 2014
                  State : clean
      Active Devices : 3
Working Devices : 4
 Failed Devices : 0
  Spare Devices : 1

        Layout : left-symmetric
      Chunk Size : 512K

              Name : stationX.example.com:0  (local to host stationX.example.com)
              UUID : 46b10be2:7e97ce63:12502b1e:06172b24
            Events : 65

      Number  Major  Minor  RaidDevice State
          3      8       7        0  active sync   /dev/sdaZ
          5      8       3        1  active sync   /dev/sdaW
          4      8       6        2  active sync   /dev/sdaY
          6      8       5        -  spare     /dev/sdaX
```

Cleanup

- 31) Remove software RAID metadata for the array nodes used in this lab.

```
# umount /srv/extra  
# mdadm -S /dev/md0  
mdadm: stopped md0  
# mdadm --zero-superblock /dev/sda{W,X,Y,Z}  
# rm -f /etc/mdadm.conf
```

- 32) Remove the /etc/fstab entry, and the directory to which the RAID filesystem was mounted. Restore the original partition table that was backed up at the beginning of the lab:

```
# rmdir /srv/extra  
# sed -i '/extra/d' /etc/fstab  
# sfdisk -f /dev/sda < /root/part.table  
# partprobe /dev/sda
```

- 33) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```


Content

Remote Storage Overview	2
Remote Filesystem Protocols	4
Remote Block Device Protocols	5
File Sharing via NFS	7
NFSv4+	8
NFS Clients	9
NFS Server Configuration	10
YaST NFS Server Administration	12
Implementing NFSv4	13
AutoFS	15
AutoFS Configuration	16
Accessing Windows/Samba Shares from Linux	18
SAN Multipathing	19
Multipath Configuration	20
Multipathing Best Practices	22
iSCSI Architecture	24
Open-iSCSI Initiator Implementation	27
iSCSI Initiator Discovery	29
iSCSI Initiator Node Administration	31
Mounting iSCSI Targets at Boot	33
iSCSI Multipathing Considerations	34
Lab Tasks	36
1. Using autofs	37
2. NFS Server Configuration	42
3. iSCSI Initiator Configuration	47
4. Multipathing with iSCSI	55

Chapter

7

REMOTE STORAGE ADMINISTRATION

Remote Storage Overview

Remote Storage - Two different approaches

- Remote Filesystem Protocols
 - Makes remote filesystem appear as local filesystem
 - Protocol transmits high level filesystem operations
- Remote Block Device Protocols
 - Makes remote storage object appears as local block device
 - Protocol transmits low level read/write operations to blocks
 - Typically formatted with a filesystem by the client or used as an LVM Physical Volume

◎ Used on all sizes of networks from small home networks to large enterprise networks.

Remote Block Device Protocols

The basic premise of remote block device protocols is that raw block devices (or "disks" in more generic terminology) are shared to the clients. The client uses the remote block device as if it were a locally attached physical disk. Initially deployed in enterprise settings, remote block device protocols first made their appearance with the introduction of storage area networks (SANs). A major drawback of SAN is that it requires a lot of special, and typically expensive, hardware. The same approach of remote block device storage can be achieved without special hardware by encapsulating the raw storage protocols in network protocols. The distinguishing characteristics of such protocols are:

- ◎ On the server a storage object is shared to the clients. This could be a single large file, a disk partition, a whole disk, a RAID volume, or a LVM logical volume.
- ◎ The shared storage object appears to the client as a block device. Sometimes alternative terminology for what the client sees is used such as a disk, or LUN.
- ◎ Typically the server doesn't care or even monitor what the client writes to the storage object.
- ◎ Clients typically partition and format or incorporate the remote block device into their local storage pool before it can be used to store files and directories.
- ◎ It is not unusual for the client to re-share the directory using a

Remote Storage

Attaching to a network and accessing remote storage is one of the most commonly used benefits of network connectivity. Practically every operating system has the built-in ability to be a client, and often a server.

For decades this has been done using remote filesystem protocols, which is typically the simplest and easiest method for remote storage access. In recent years, remote block device protocols have arrived and offer an alternative approach. Remote block devices can offer more flexibility for the clients in how the storage is used, but are typically more complicated to configure both on the client and server sides.

Remote Filesystem Protocols

The original, and still very popular, approach to remote storage has been this use of remote filesystem protocols. The distinguishing characteristics of such protocols are:

- ◎ On the server a directory from the server filesystem is shared.
- ◎ The terminology for the shared directory is a "share" or "export".
- ◎ The client does not typically re-share the directory.
- ◎ The protocol transmits filesystem operations such as stat, delete, rename, and mkdir.
- ◎ Remote filesystem is made to appear as a local filesystem
- ◎ Multiple clients can access the shared directory and files in a read/write mode without any filesystem complications and file access is coordinated by the server.

remote filesystem protocol.

- ❖ In certain client applications, such as with some databases, the client uses the remote block device raw without using a filesystem.
- ❖ Having multiple clients use the same remote block device simultaneously is complicated. For read/write access, a cluster filesystem must be used and the clients must coordinate access amongst themselves.
- ❖ Commonly used in business data-centers, not in home networks.

Remote Filesystem Protocols

Most Commonly Used Remote Filesystem Protocols

- Network Filesystem (NFS)
 - Standard Unix File Sharing Protocol
 - Has evolved over several versions
- Server Message Block (SMB)
 - Standard Windows File Sharing Protocol
 - Can share printers and other communication resources
 - Also known as Common Internet File System (CIFS)

Other Remote Filesystem Protocols

- Apple Filing Protocol (AFP), Web-based Distributed Authoring and Versioning (WebDAV), Netware Core Protocol (NCP)

Remote Filesystem Protocols

The most distinguishing characteristic of remote filesystem protocols is that a portion of the filesystem on the server is shared to clients. The protocol transmits high level file and directory operations between the server and client. Over the years many different operating system specific protocols have been created. When a protocol gets popular enough, other client and/or server implementations are usually created so that other operating systems can interoperate. The following tables shows Linux support for various remote filesystem protocols:

Protocol	Server Status	Client Status
NFS	Kernel server supports versions 1-4	Kernel client supports versions 1-4. See mount.nfs(8)
SMB/CIFS	Userland Samba daemon	Kernel client. See mount.cifs(8)
WebDAV	Apache mod_dav/mod_dav_fs	Userland clients cadaver and davfs2
AFP	Userland Netatalk daemon	Userland client afpfs-ng
NCP	Novell Open Enterprise Server	Userland client ncpfs, Novell Client for Linux

NFS Support in Windows

Beginning with Windows Server 2003 R2, the components of the Windows Services for Unix (SUA) have been integrated into the operating system. Windows Enterprise and Ultimate Editions contain an NFS v4 client built-in. The Windows NFS client includes all three kerberos security modes for interoperability with Kerberized NFS servers. Windows Server includes NFS server support complete with Kerberos authentication. More information can be obtained at: <http://blogs.msdn.com/sfu/>.

C:\>mount server1:/export/tmp x:

C:\>mount

Local	Remote	Properties
x:	\server1\export\tmp	UID=50001, GID=50001 rsize=8192, wsize=8192 mount=soft, timeout=1.6 retry=1, locking=yes fileaccess=644, lang=ANSI casesensitive=no

Remote Block Device Protocols

Storage Area Network (SAN)

Fibre Channel SAN

- Fibre Channel Protocol (FCP)
 - fiber optic or twisted pair media
 - Normally transports SCSI commands
 - Fast speeds (2/4/8/16 Gbit/s) but expensive hardware

Commodity Protocol SAN

- Speed depends on Ethernet
 - SCSI over IP (iSCSI)
 - ATA over Ethernet (AoE)
 - Fibre Channel over Ethernet (FCoE) with DCB
 - Linux specific protocols nbd & DRBD

iSCSI SAN

Partly in response to the cost and hardware requirements of Fibre Channel, alternative SAN solutions were developed leveraging commodity components and protocols that already existed in the datacenter. The most popular and widely adopted commodity SAN protocol is iSCSI, with over 12 operating systems supporting iSCSI. The iSCSI protocols tunnel SCSI over TCP/IP using TCP ports 860 and 3260. In iSCSI terminology the client is an "Initiator" and the server is a "Target". The iSCSI solution is scalable in price and performance from a completely software driven architecture comprised of software initiators and software target storage arrays to hardware initiators and hardware storage arrays.

Most modern server class network cards include a built in iSCSI initiator that enables a server to boot off of an iSCSI target/disk and be completely disk-less. On Linux, with older NICs that lack an iSCSI initiator, a PXE boot can be used to load a Linux kernel with the iSCSI initiator in the initial RAM disk. This PXE boot approach adds some complexity during kernel upgrades as the upgrade needs to take place on the TFTP server. Some network cards with built in iSCSI initiators actually appear as a normal SCSI card to the operating system, thus lowering CPU utilization. Thus, a large appeal of iSCSI is the ability to start small and grow into larger higher performing deployments without disruption.

Remote Block Device Protocols

The most distinguishing characteristic of remote block device protocols is that they present a raw block device to the client. The client treats the device as if it were a local hard drive. This means that it is typically partitioned and formatted, or used as an LVM physical volume. Because of the remote network aspect to this technique the shared block device can be moved from one server to another very easily or even shared simultaneously among multiple servers for cluster applications. Several different technologies provide this same approach to remote storage.

Fibre Channel SAN

The original SAN technology is Fibre Channel which achieved standard approval in the early 1990s. Being the first SAN technology it was widely adopted in enterprise data centers. Fibre Channel is a layered protocol with 5 layers, similar to the OSI network model, named FC0 through FC4. The first layer is the physical layer, which can be fiber optic or twisted pair copper wire. The main drawback to a Fibre Channel SAN is the infrastructure including hardware arrays, switches, cabling, and Fibre Channel Host Bus Adapters (HBAs) that are installed in servers. In a datacenter using Fibre Channel, in addition to the standard CAT5/CAT6 cable plant for network traffic, a Fibre Channel cable plant must also be deployed. The primary advantage to Fibre Channel include a dedicated fabric for interference free communication, security, speed, and HBAs that provide a low CPU utilization server-to-SAN interface. Products are available that operate at 1 Gbit/s to 16 Gbit/s speeds.

AoE SAN

ATA over Ethernet is a SAN protocol that aims to be a simpler and lightweight competitor to iSCSI. The AoE specification is 12 pages compared to iSCSI's 257 page specification. It directly encapsulates ATA commands in Ethernet frames without using IP, and subsequently has lower CPU utilization than iSCSI software initiators. However, this means that AoE traffic cannot be routed. Currently only Linux, OpenBSD, and Plan 9 support AoE natively with bundled initiator support. Other operating systems are supported via installed software. Both software targets and hardware storage arrays are available.

FCoE SAN

Fibre Channel over Ethernet aims to reduce the complexity and hardware requirements of a Fibre Channel SAN by replacing the Fibre Channel switch fabric with Ethernet. It directly encapsulates the Fibre Channel protocols in Ethernet without using IP, thus it cannot be routed. It is very attractive to datacenters to only have a single cable plant (e.g. Cat6). Deployments of FCoE are expected to typically be done over 10Gbit Ethernet instead of 1Gbit Ethernet so that performance doesn't retreat compared to the 4 Gbit/s and 8 Gbit/s Fibre Channel SANs that are currently deployed.

Ethernet has always been a "best effort" protocol. To keep complexity and cost to a reasonable level, no reliability guarantees are made. This is in stark contrast to the Fibre Channel Protocol. When deploying FCoE, it is advantageous if the Ethernet layer supports Data Center Bridging (DCB). DCB is a collection of enhancements to Ethernet that provides features akin to the Fibre Channel Protocol, namely Congestion Notification (CN), Priority-based Flow Control (PFC), and Enhanced Transmission Selection (ETS). DCB is implemented in such a fashion that LAN and SAN traffic can operate on the same link.

Linux Network Block Devices

Linux specific SAN protocols have been part of Linux for a very long time. By their very nature these are software based and work over IP. The first such protocol is Network Block Device (NBD) which is very simple. The latest addition to Linux is Distributed Replicated Block Device (DRBD). DRBD is targeted specifically as the building block for two node software based high availability (HA) clusters.

File Sharing via NFS

Traditional Unix method of network file sharing

- Implemented as an RPC service (relies on `rpcbind`)

NFS v1/v2

- built on UDP/IP
- designed for LAN sharing; not well suited to WAN

NFS v3

- defaults to TCP/IP with UDP as fallback
TCP's flow and congestion control allows for WAN feasibility
- supports 64-bit file handles (supports files larger than 2Gb)

The Network File System

Created by Sun Microsystems in the '80s, the Network File System (NFS) is the standard Unix file sharing protocol. It is analogous to CIFS in the Windows networking world, and NCP in the Netware networking world. Unlike those protocols, NFS only deals with file sharing, leaving print sharing to the other protocols. NFS continues to evolve and adapt to the changing network environment.

The NFS versions supported by a Linux NFS server can be checked by running:

```
# cat /proc/fs/nfsd/versions
-2 +3 +4 +4.1 -4.2
```

Note that in order for the above command to work, the `nfsd` kernel module must be loaded. This happens automatically when a directory is being exported.

Alternatives to NFS

Although NFS is far and away the standard filesystem used across networks in Unix environments, other options are available. AFS, a recently open-sourced product with roots at IBM and Carnegie-Mellon University, is commonly used for extremely large, widely distributed shared filesystems. More recently, Carnegie-Mellon has also developed Coda, another popular network-based filesystem designed especially to handle transient connections (such as synchronizing files on laptops with servers).

NFSv4+

New NFS v4 Protocol Design

- Single TCP port for all traffic – 2049
- Optionally share a "pseudo filesystem"
- First Linux NFS protocol to support GSSAPI/Kerberos auth
Solves NFS security problems

New Additional Daemons

- Server – `rpc.idmapd` & Optionally `rpc.svcgssd`
- Client – `rpc.idmapd` & Optionally `rpc.gssd`

NFS v4.1 adds Parallel NFS (pNFS) for arbitrarily scalable speeds

NFS v4.2 adds Server Side Copy (SSC), Sparse Files, Space Reservation, IO_ADVISE, and Application Data Block (ADB) support

The NFSv4 Protocol Improvements

Some of the notable new features of NFSv4 include the operation of all RPC protocols over TCP port 2049. This simplifies the firewalling, tunneling, and network address translation of the NFS protocol.

NFSv4 introduces the concept of a single pseudo filesystem that the clients see as /. The system administrator may manually specify a directory on the server to be the top level of the NFSv4 pseudo filesystem. If directories outside of that directory tree need to be exported then they must be bind mounted into that directory tree. With `nfs-utils` v1.2.2+ and Linux kernel v2.6.33+ the pseudo filesystem will be dynamically created mapped to the actual / filesystem if one is not manually specified. The dynamically created pseudo filesystem will be secured so that only shared directories will be visible.

One problem that has caused trouble for NFS over the years is the fact that it communicates raw UID and GID numbers over the wire. If there are five users: bailey, dax, sydney, aria, jada and the accounts exist on both the NFS server and client but were created in a different order so that the UIDs and GIDs don't match up, then sharing files via NFS will be problematic. The files will show up as the wrong user and groups. The usual solution is to enforce UIDs and GIDs to match via manual validation or the use of a directory service such as LDAP. NFSv4 solves this problem by sending actual usernames and groups as strings over the wire instead of numerical UIDs and GIDs. Thus, as long as the same user and group exists, file sharing will work as expected.

The mapping of the strings to UIDs and GIDs is handled by a new daemon, `rpc.idmapd`. It must be running on both the client and server. It has a configuration file `/etc/idmapd.conf` where settings can be adjusted if the normally suitable defaults aren't sufficient.

Secure NFS

On Linux, NFSv4 is the first NFS version to support GSSAPI/Kerberos authentication. This provides strong host and user authentication keeping unauthorized computers and users away from NFS shared data. Only computers within a Kerberos realm can mount designated shares from the NFS server. Additionally GSSAPI enables integrity checked and encrypted NFS traffic.

When using NFSv4 with GSSAPI authentication additional daemons must be running on the client and server to link transactions to specific GSSAPI credentials. The server daemon is `rpc.svcgssd` and the client is `rpc.gssd`. These are only required if GSSAPI authentication is being used. The `rpc.gssd` is accessed over the network using a port registered with `rpcbind`.

Parallel NFS with NVSv4.1

Version v4.1 introduces an exciting feature: the ability to create a parallel NFS cluster for the purpose of speed. In a parallel NFS cluster, one or more meta-data servers are coupled with one to many data servers. Clients fetch file contents from all the data servers simultaneously. Speeds can be scaled as high as needed by adding additional data servers.

NFS Clients

NFS v3 uses RPC and so requires rpcbind

Requires client RPC daemons

- rpc.statd
- rpc.lockd

NFS mounts listed in /etc/fstab automatically mount at boot

- Via systemd remote-fs.target

showmount -e server

showmount -a server

nfsstat

Combine with Autofs to allow easy user initiated mounting

The showmount Command

The **showmount** utility is used most often to determine what NFS shares a server has available for mounting.

```
# showmount -e server1
Export list for server1:
/export/tmp          *
/export/home         *
/export/autoyast    *
/export/kickstart   *
/export/netinstall  *
```

The **showmount -a server** command can be used to see which exported directories are currently being used by which clients.

The nfsstat command

The **nfsstat** utility is used to list statistics kept about NFS client and server activity. When run without any arguments, it prints both client and server statistics.

Mounting NFS Shares

After determining which shares are available, any may be mounted on the local system in the following way:

```
# mount server1:/export/netinstall /mnt
```

NFS resources that will be accessed frequently can be added to the /etc/fstab file. For example:

File: /etc/fstab

```
+ server1:/exports/netinstall /srv/netinstall nfs intr 0 0
```

To see what NFS shares are currently mounted, view the /proc/mounts file. (/etc/mtab, **mount**, and **df** are not always aware of NFS mounts.)

The systemd **remote-fs.target** is pulled in via dependencies during boot to mount the NFS and other remote filesystems listed in the fstab file during boot.

NFS Server Configuration

Shared directories defined in /etc/exports

- **man exports** for details
- one line per exported filesystem
- can limit connections by IP address or FQDN
- shares default to "read only" and "root squashed"
- Requires that RPC (the **rpcbind** daemon) be running for v3 clients
- Behavior change with Kernel 2.6.33+ and nfs-utils 1.2.2+
Each line shared with protocol versions v3 AND v4

After editing /etc/exports, run

- **exportfs -vra**

Daemon configuration: /etc/sysconfig/nfs

NFS Exports File

All NFS shares are defined in the /etc/exports file. The following is an example of a simple exports file:

File: /etc/exports

```
/home          *.example.com(rw,sync)
/export/tmp    *(rw,sync)
/export/packages  10.100.100.0/24(ro)
```

[R7] The following applies to RHEL7 only:

The SELinux policy on Red Hat Enterprise Linux identifies the /export/ directory for use with NFS. If using another directory, such as /srv/export/, the **usr_t** type enforcement should be set, and **restorecon -R** run:

```
# semanage fcontext -at usr_t "/srv/export(/.*)?"
# restorecon -R /srv/export/
```

Using exportfs

The **exportfs** command is most commonly used to notify the kernel that the /etc/exports file has changed and to re-read and make the current sharing match what is in the file. To do this, run:

```
# exportfs -arv
```

root Squashing

NFSv3 and below base all access to files by UID. For security

purposes, the root user's UID on the client machine (0) is normally squashed to the UID of the **nfsnobody** user on the NFS server (typically 65534), preventing the root user on the client from gaining root-level privilege on the server. The option to turn this on is **root_squash**, and the option to turn it off is **no_root_squash**.

NFS Versions

When a client connects to an NFS server, the two machines negotiate and determine the maximum NFS version level which both ends support. Although this auto-negotiation usually works, some broken non-Linux clients, or servers, will require that the negotiated version be overridden. Version 2 of NFS can be forced, for example, by using the **nfsvers=2** option.

By default the NFS server will allow access to shares via protocol versions 4 and 3. This can be changed by editing the /etc/sysconfig/nfs file. Protocol 2 is not supported.

Packet Filter Security

Block unauthorized NFS clients with TCP Wrappers:

File: /etc/hosts.deny

```
lockd mounted portmap rquotad statd: ALL
```

File: /etc/hosts.allow

```
mountd: 10.100.0.0/255.255.255.0
```

[R7] *The following applies to RHEL7 only:*

By default, **lockd** and **statd** are dynamically assigned port numbers by **rpcbind** (portmapper). To simplify packet filtering rules, you can instead define static port assignments for **lockd** as shown in the following example:

File: /etc/sysconfig/nfs
+ LOCKD_TCP_PORT=32803
+ LOCKD_UDP_PORT=32769

NFS ports can then be added to Netfilter rules:

```
# iptables -I INPUT -m multiport -p tcp --dport 111,875,2049,20048,32803 -j ACCEPT  
# iptables -I INPUT -m multiport -p udp --dport 111,875,2049,20048,32769 -j ACCEPT
```

By default, port 875 is used for **rpc.rquotad** and port 20048 is used for **rpc.mountd**. See /etc/services.

[S12] *The following applies to SLES12 only:*

By default, **mountd** is dynamically assigned a port number by **rpcbind**. To simplify packet filtering rules, you can instead define a static port assignment as shown in the following example:

File: /etc/sysconfig/nfs
+ MOUNTD_PORT=20048

NFS Debugging

To aid in troubleshooting NFS problems, you may turn on full debugging with the following commands:

```
# echo 32767 > /proc/sys/sunrpc/nfsd_debug  
# echo 32767 > /proc/sys/sunrpc/nfs_debug
```

When you have finished troubleshooting, **echo 0** into the files.

YaST NFS Server Administration

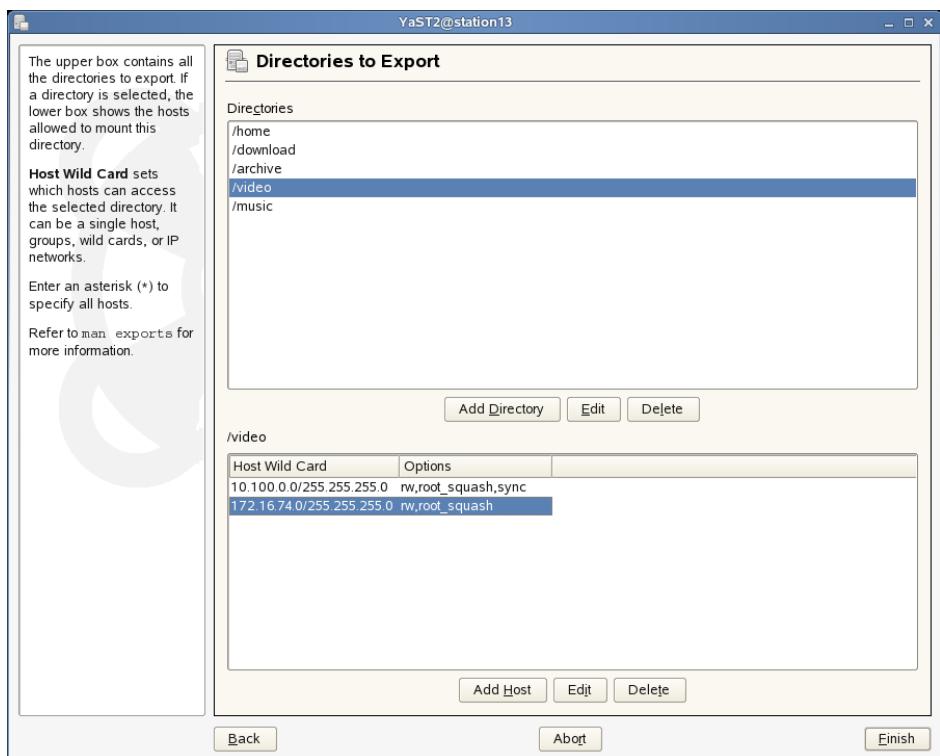
YaST module for configuring and administering NFS exports

- Takes care of the /etc/exports file
- Can properly configure the **SuSEfirewall2** to permit access to the NFS server
- Can configure NFS4 server support

YaST NFS Server Administration

YaST has modules for doing all of the common tasks related to many server services. The module for NFS server configuration can be used to do all NFS server configuration management. The traditional approach would be to manually edit the /etc/exports file and use the **/etc/init.d/nfsserver** SysV Init script or the **exportfs** command. The YaST NFS Server module combines all of these tools and capabilities into one place as well as convenient and automatic firewall configuration.

The **yast2 nfs_server** module can also be invoked from the command line. The ncurses text-based interface is very easy to use with a remote connection, (such as with **ssh**).



Implementing NFSv4

Everything in /etc/exports is shared via NFS v2/3/4 simultaneously

- **rpc.idmapd** running
- Requires kernel 2.6.33+ and nfs-utils v1.2.2+
- Older Linux versions require fsid=0 pseudo-root syntax

NFSv4 Client

- Must use **-t nfs4** option to **mount** command
Default with nfs-utils v1.2.2+
- Mount share or pseudo-root (/) from server
- **rpc.idmapd** running

(/etc/init.d/nfsserver) to control NFS server services. The **rpc.idmapd** daemon will be started by this script if the /etc/sysconfig/nfs file contains the NFS4_SUPPORT="yes" config statement (the default).

Historical NFSv4 Exports

A single NFSv4 export, the pseudo-root, is created by adding a line to the /etc/exports file in this form:

File: /etc/exports
+ /srv/export *(rw,fsid=0,no_subtree_check,async)

The key option is fsid=0, which designates the share as the top level of the NFSv4 pseudo filesystem.

Modern NFSv4 Exports

With modern NFSv4, all shares listed in /etc/exports will be shared with all three NFS protocol versions, 2, 3, and 4. A pseudo-root filesystem may also be manually defined as before, but this is no longer commonly done. When not manually defined, a pseudo-root filesystem will be automatically created that shares the actual / filesystem. Security is maintained as only the specific shared directories are visible in the pseudo-root filesystem.

Historical NFSv4 Mounting

From the client perspective, mounting an NFSv4 export is nearly the same as mounting any NFS export. The one difference is that the

NFSv4 Behavior Change

How to deploy and use NFSv4 on Linux has changed significantly. With Linux kernel 2.6.32+ and nfs-utils 1.2.2+ NFSv4 is now treated and administered just like NFSv2/v3. In this text, "Modern NFSv4" refers to this updated behavior, while "Historical NFSv4" refers to the previous behavior.

Deploying NFSv4

NFSv4 doesn't require the traditional RPC daemons such as **rpc.statd**, **rpc.mountd**, **rpc.rquotad**, and **rpc.lockd** to be running. However, it is helpful to have **rpc.mountd** running so that **showmount -e** works from the clients.

NFSv4 introduces a new requirement of **rpc.idmapd**. Make sure that it is started on both the client and server.

[R7] *The following applies to RHEL7 only:*

Under RHEL7, to start an NFS server with v4 support, run the following:

```
# systemctl enable nfs-server
# systemctl start nfs-server
# systemctl status nfs-idmap
```

Notice that **rpc.idmapd** started automatically as a dependency.

[S12] *The following applies to SLES12 only:*

SLES12 uses a single unified SysV init script

filesystem type must be specified as **nfs4** (including in /etc/fstab), otherwise an NFSv3 mount request will occur:

```
# mount -t nfs4 server:/ /mnt
```

Modern NFSv4 Mounting

From the client perspective, mounting an NFSv4 export is exactly the same as v3 or v2. The system will attempt a v4 mount first, and then fallback to a v3 attempt if failure occurs. With modern NFSv4 behavior, v4 shares can exist outside the pseudo-root filesystem and those are typically mounted directly:

```
# mount server:/srv/export /mnt
```

AutoFS

Automated mounting of filesystems on demand

- un-privileged users can trigger mount
- automatically unmounts when no longer in use

Kernel driver plus userspace daemon

Direct vs. indirect map behavior

AutoFS Design

The Linux AutoFS implementation consists of two components: a kernel driver and a userspace daemon.

autofs (kernel module) ⇒ monitors specific directories in the filesystem for any attempts to access those directories. This device driver signals the **automount** userspace daemon when those directories are accessed, so they can be mounted, or when directories are no longer being accessed, so they can be unmounted.

automount (userspace daemon) ⇒ when signaled by kernel driver, runs the **mount** command to mount whatever filesystem should be mounted at the indicated directory, or the **umount** command to unmount filesystems that are no longer in use.

Usage and Behavior

AutoFS allows filesystems to be mounted on demand, as they are needed, including by un-privileged users. Only the root user can directly mount a filesystem, but any user can try to access a directory under AutoFS control, causing the appropriate resource to be mounted on that directory. AutoFS can be used with local filesystems, such as CD-ROMs and other removable media. It can also be used with network filesystems, such as remote SMB or NFS shares.

The exact behavior of a mount point controlled by AutoFS will depend on the type of AutoFS map and the options used. With direct maps,

executing the **ls** command in the parent directory will show a placeholder directory entry for the map point. Be aware that the meta-data for this directory (permissions, owner, etc.) is not the meta-data for the target filesystem until the specified filesystem is mounted by **automount**.

With indirect maps (more common), a user accessing the directory controlled by AutoFS (such as with the **ls** command) will not see the defined mount points. The directory will instead initially appear to be empty. Only when a process attempts to switch to a sub-directory that is matched by the corresponding AutoFS map file will a directory entry be created and the configured filesystem be mounted. If the **BROWSE_MODE** variable is set to YES in the **/etc/sysconfig/autofs** file, then AutoFS will create placeholder directory entries for each defined mount point. A mount is still only triggered when a process attempts to change to the sub-directory.

AutoFS Configuration

Main configuration /etc/auto.master

- defines indirect and direct maps

Example file

- /etc/auto.misc

Using /home with AutoFS

Access any NFS share via the /net directory

- In RHEL7, enabled by default in /etc/auto.master
- In SLES12, edit /etc/auto.master to enable

AutoFS Configuration

The primary configuration file for AutoFS by default is /etc/auto.master (an alternate file can be specified in /etc/sysconfig/autofs). For scalability and centralized control, the AutoFS configuration files can also be stored in NIS or LDAP. With the exception of a few general options (see auto.master(5) for details), each line in the auto.master file is a map that has the following simplified form: *mount_point map*.

AutoFS supports two different kinds of maps: indirect and direct. When configuring the more common indirect map, the *mount_point* specified should be an empty directory (any contents will be inaccessible once AutoFS starts). Once AutoFS starts, the directory will be registered with the autofs kernel driver. Attempts to access a sub-directory within this directory cause the running automount daemon to lookup that keyname in the corresponding *map* file and trigger a mount. With direct maps, the *mount_point* will always be */-*, and the corresponding *map* file will contain a list of mountable filesystems including the full path where they should be mounted. The autofs kernel driver will then monitor each of these directories for access and signal the automount daemon to perform the mount when appropriate.

The default /etc/auto.master file contains the following indirect map entry:

File: /etc/auto.master

```
/misc    /etc/auto.misc
```

The specified map file must then contain keys and corresponding resources to mount (including mount options). For example, suppose that /etc/auto.misc contains the following:

File: /etc/auto.misc

```
kernel    -ro,soft,intr    ftp.kernel.org:/pub/linux
cd        -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom
```

In the above example, all attempts by users to access /misc/kernel will transparently result in ftp.kernel.org:/pub/linux being NFS-mounted at /misc/kernel (if it is not already mounted there). All attempts to access /misc/cd will signal **automount** to mount the local CD-ROM drive on /misc/cd (if it is not already mounted there).

If a filesystem type is not specified using the *fstype* option, AutoFS defaults to using NFS. Valid filesystem types which can be specified include smbfs, ext2, iso9660, nfs, auto, and autofs.

Mounting Home Directories

Commonly AutoFS is used to automatically mount home directories from a central file server. This coupled with a directory service for authentication such as NIS or LDAP allows for true roaming desktops.

In the master AutoFS configuration file (see the `auto.master(5)` manual EXAMPLES section), add the line:

```
File: /etc/auto.master
+ /home    /etc/auto.home      --timeout=60
```

Create a new file (matching the filename referenced in the `auto.master` file) and add the line:

```
File: /etc/auto.home
+ *      -rw,soft,intr      nfsserver:/home/&
```

The * wildcard will match whatever sub-directory name the system attempts to change to, usually the username (since the home directory name generally matches the username). The & is a variable that will take the value of whatever the * matched, (see the `autofs(5)` manual, under the Wildcard Key section).

Alternatively, the PAM `pam_mkhomedir` module can be used to create a home directory on first login. The disadvantage to this approach is having multiple unique per-server home directories for each user so any custom login scripts or content isn't the same on all the Linux boxes within an organization.

[R7] *The following applies to RHEL7 only:*

Use of NFS remote home directories managed by AutoFS require that the SELinux `use_nfs_home_dirs` boolean be enabled.

Using the /net AutoFS Directory

Long available on some Unix variants, the `/net` AutoFS directory enables access to any NFS server by changing to a directory named for the hostname. This is especially convenient for users, since normally only administrative users can mount NFS shares on their own. For example:

```
$ cd /net/server1/export/netinstall
$ cd /net/10.100.0.1/data/backupfiles
```

The `/net/` directory is enabled by default, in the `/etc/auto.master` file, with a stock install of AutoFS.

[S12] *The following applies to SLES12 only:*

On SLES12, the `/net/` and `/misc/` directories, as well as `/etc/auto.master.d/`, are not enabled by default in the `/etc/auto.master` file. To enable it, uncomment the appropriate line and restart AutoFS:

```
File: /etc/auto.master
- #/net -hosts
+ /net -hosts
```

Accessing Windows/Samba Shares from Linux

smbclient

- Provides an FTP-like interface through which SMB shares can be accessed

CIFS Filesystem

- Provides a filesystem interface to access SMB shares, allowing them to be mounted as a filesystem
- The cifs filesystem supports ACLs, symlinks, and actual uid/gid display when communicating with a server which supports the CIFS Unix extensions (such as Samba)

net

- Similar to the net command in Windows

Using smbclient

smbclient provides an FTP-like interface through which SMB shares can be accessed. To use **smbclient** interactively, provide a UNC path:

```
$ smbclient //server1/jkelson  
Password: password   
smb: \>
```

At the smb:> prompt, standard FTP-like commands are available: **get**, to download files; **put**, to upload files; **cd**, to change directories; **quit**, to exit.

smbclient can also be used to list available resources:

```
$ smbclient -L //mooru  
Password: password   
Sharename  Type      Comment  
-----  -----  
courseware Disk      Guru Labs Courseware  
marketing   Disk      Guru Labs Marketing  
  
Server      Comment  
-----  
MOORU       Mooru Samba Server  
  
Workgroup  
-----  
GURULABS    MOORU
```

Using the cifs Filesystem

The cifs filesystem is integrated into the Linux kernel and Linux **mount.cifs** command, allowing SMB shares to be mounted and accessed as filesystems under Linux. The cifs filesystem will attempt to negotiate the CIFS Unix Extensions which supports POSIX acls, locks, paths, symlinks, and retrieving UIDs, GIDs, and mode.

```
# mount -t cifs -o user=dkelson //mooru/dkelson /mnt/  
Password: password 
```

Normally a single user's credentials are used for accesses to all files and directories on the share. Instead, the multiuser option can be used in conjunction with Kerberos so each local user has unique security rights when accessing the share.

Using net

net is a tool used for the administration of CIFS servers. The following examples demonstrate a couple of uses of **net**.

Reboot a CIFS server:

```
$ net rpc shutdown -r -S hostname -U User%Pass
```

List, stop, and start available services on a CIFS server:

```
$ net rpc service list -S hostname  
$ net rpc service stop Service_Name -S hostname  
$ net rpc service start Service_Name -S hostname
```

The **-U** parameter can be used to specify a username and password.

SAN Multipathing

Aggregating multiple paths to storage is referred to as multipathing

Provides redundancy and increased throughput

device-mapper provides vendor-neutral multipathing configuration and consists of:

- dm-multipath (kernel module)
- multipath (command)
- multipathd (daemon)
- kpartx (command)

Multipathing creates /dev/mapper/*name* device file

- Name is the WWID, auto-created friendly name, or user defined friendly name

Once configured for a LUN, only use it's multipath device

Device-mapper Multipath Components

Device-mapper multipathing consists of several notable components:

dm-multipath ⇒ kernel module responsible for making routing decisions under normal operation and during path failure.

multipath ⇒ command used for initial configuration, listing, and viewing multipathed devices.

multipathd ⇒ daemon that monitors paths, marks failed paths, reactivates restored paths, adds and removes device files as needed, and can be used to monitor and manage individual paths.

kpartx ⇒ command used to create device-mapper entries for partitions on a multipathed LUN. It is invoked automatically when the **multipath** command is used.

Device-mapper Multipathing

Many different vendor-specific multipath implementations exist resulting in difficult configuration. Device-mapper multipathing exists in order to provide a consistent method of configuring multipathing under Linux. Regardless of the vendor hardware in use, device-mapper creates a block device under `/dev/mapper/` for each LUN attached to the system.

Multipath Configuration

/etc/multipath.conf

Configuration file sections

- defaults
- blacklist
- blacklist_exceptions
- devices
- multipaths

Define a blacklist if *not* using the find_multipaths setting

Sample config: multipath.conf.annotated
mpathconf tool generates multipath.conf

/etc/multipath.conf

Both the **multipath** command and **multipathd** are configured using the /etc/multipath.conf file. The configuration file is only used during configuration of device-mapper multipathing. If updated, the **multipath** command must be run in order to reconfigure the multipathed devices. The file consists of five sections:

defaults ⇒ System-level default configuration

blacklist ⇒ Black listed devices - a list of devices which should not be controlled by device-mapper multipathing

blacklist_exceptions ⇒ Exceptions to the blacklist - individual devices which should be managed by device-mapper multipathing even if they exist in the blacklist

devices ⇒ settings to be applied to individual storage controller devices

multipaths ⇒ fine-tune configuration of individual LUNs

Detailed explanations of possible configuration options and values may be found in the multipath.conf.annotated file under the /usr/share/doc/ directory tree. Options not configured use default values.

Multipath Blacklisting

Care should be taken to include a complete blacklist in /etc/multipath.conf of all the block devices which should not be controlled by the device-mapper multipathing if the find_multipaths setting is not used. A good starting point includes the following

configuration:

File: /etc/multipath.conf

```
blacklist {
    devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
    devnode "^(sd[a-z])[0-9]*"
}
```

Manual Configuration

In earlier versions of multipath, or in newer versions when find_multipaths is not used, multipath attempts to create multipath devices for every non-blacklisted device. This requires a proper blacklist definition. The procedure for first time setup is as follows:

```
# mpathconf --enable # creates /etc/multipath.conf
# vi /etc/multipath.conf # modify blacklist and configure
# mpathconf --enable --with_multipathd y
# multipath -F # flush unused multipath device maps
# multipath -v2 # scan non-blacklisted devices
create: mpatha (3600140574eff7bc56a747faaa52d508a) LIO-ORG
size=25G features='0' hwhandler='0' wp=undef
`-- policy='round-robin 0' prio=1 status=undef
  `-- 6:0:0:0 sdb 8:16 undef ready running
`-- policy='round-robin 0' prio=1 status=undef
  `-- 7:0:0:0 sdc 8:32 undef ready running
```

The **--with_multipathd y** parameter to **mpathconf** persistently enables and starts the daemon. After any changes to the

multipath.conf file, run **systemctl reload multipathd**.

Automatic Configuration

A new feature of multipath is to have it do the right thing, by default, for the typical scenarios without requiring any manual editing of the configuration file. This is done using the `find_multipaths` setting. With that setting, `blacklist` is *not* needed. Instead, whenever there are two or more devices with paths to the same LUN, it creates a multipath device. It will also create a multipath device for a LUN if there was a previously configured multipath device for that LUN. This logic makes configuration simpler. The procedure for first time setup is as follows:

```
# mpathconf --enable --find_multipaths y --with_multipathd y
Starting multipathd daemon: [ OK ]
# multipath -F # flush unused multipath device maps
# multipath -v2 # scans for LUNs with multiple paths
create: mpatha (3600140574eff7bc56a747faaa52d508a) LIO-ORG
size=25G features='0' hwhandler='0' wp=undef
`-- policy='round-robin 0' prio=1 status=undef
   `-- 6:0:0:0 sdb 8:16 undef ready running
`-- policy='round-robin 0' prio=1 status=undef
   `-- 7:0:0:0 sdc 8:32 undef ready running
```

The multipath device name to LUN mappings are stored in the /etc/multipath/bindings file.

Verifying Configuration

Once multipathing has been configured, the **multipath** command can be used to display information about multipathed devices. For example:

```
# multipath -ll
mpatha (3600140574eff7bc56a747faaa52d508a) dm-5 LIO-ORG,IBLOCK
size=25G features='0' hwhandler='0' wp=rw
`-- policy='round-robin 0' prio=1 status=active
   `-- 6:0:0:0 sdb 8:16 active ready running
`-- policy='round-robin 0' prio=1 status=enabled
   `-- 7:0:0:0 sdc 8:32 active ready running
```

If satisfied with the completed configuration, the **multipathd** should be started and turned on persistently. For example:

```
# systemctl start multipathd
```

systemctl enable multipathd

The **multipath** command can also be used to display a list of the currently-blacklisted devices. For example:

```
# multipath -v3 -ll | grep blacklist
. . . output omitted . . .
```

Controlling Device File Properties

The owner, UID, GID, and name of the multipath device file can be controlled using a UDEV rule. First bring up the multipath using `user_friendly_name` and determine what the `DM_UUID` UDEV ENV is set to for the multipath device:

```
# udevadm info /dev/mapper/mpatha | grep DM_UUID
E: DM_UUID=mpatha-36001405956a7ebcb7cccd8d
```

Then create an UDEV rule using that `DM_UUID` value, for example:

File: /etc/udev/rules.d/99-mpath.rules
+ ENV{DM_UUID}=="mpatha-36001405956a7ebcb7cccd8d", OWNER=="oracle", GROUP=="dba", MODE=="660", SYMLINK+="ORALun01"

This can be useful in situations where an unprivileged application, such as the Oracle Database, needs read/write access to the block device.

LVM on top of Multipath Devices

When using multipath devices as LVM physical volumes, you should configure LVM to ignore the individual paths, otherwise you will get spurious error messages. This is done by adjusting the `global_filter` in the `lvm.conf` file. For the example scenario where LVM should only examine /dev/sda (the local disk) and device mapper created devices use:

File: /etc/lvm/lvm.conf
+ global_filter = ["a /dev/sda.* ", "a /dev/disk/by-id/dm.* ", "r .* "]

Multipathing Best Practices

What naming method should be used?

- user_friendly_names — easier to type
- WWIDs — no ambiguity

Multipathing boot considerations

- Use _netdev fstab option for iSCSI or FCoE LUNs

Using LVM on top of multipath device?

- Filter out underlying devices in /etc/lvm/lvm.conf

Consistent multipath device names across multiple machines

- Use same Udev rules on all machines, or if possible
- RHEL7: Use same /etc/multipath/bindings file

Underlying paths should fail quickly

Use queue_if_no_path for critical LUNs

Multipathing Best Practices

In order to ease troubleshooting, device-mapper may be configured to create human-readable device files under /dev/mapper/ instead of files named after the WWID.

Device-mapper may be told to create device files with names such as /dev/mapper/mpatha, enabled with the user_friendly_names option in /etc/multipath.conf (enabled by default when using `mpathconf`). For example:

File: /etc/multipath.conf

```
defaults {  
+    user_friendly_names yes  
}
```

Instead, if more control over the name is desired for a particular LUN, use a Udev rule.

Boot Considerations

With some types of mounts, such as SMB or NFS, the first column of /etc/fstab makes it obvious that the mount point is network-based so no special mount options are required. Since multipathed devices are often network-based, yet appear as local device files in /dev/mapper/, a special mount option _netdev must be used to inform system start up to perform the mount operation for the device.

When configuring /etc/fstab to persistently mount multipathed LUNs, do not attempt to mount using filesystem labels as the system will attempt to read labels from the underlying block devices rather than the multipathed device. Instead, use either the LVM logical volume name or the multipath WWID, Udev alias, or user_friendly_names.

Partitioning Multipathed LUNs

While it's possible to partition multipathed LUNs, care must be taken to make certain they are made available to the host OS. If partitioned LUNs are desired, the partitions should be created prior to setting up multipathing, then `kpartx` should be run to detect the partitions and create device-mapper entries for the partitions. Rather than partitioning the LUN directly, it's usually better to use the LUN as a physical volume for LVM and partition out the available space using LVM if multiple partitions are required.

Consistent Device Names

When using the same LUN multipath device across multiple systems, as is commonly done in a cluster, it can ease administration work if the multipath device names are the same across all the systems. If all the systems have identical multipath configurations, having identical Udev rules files accomplishes this goal.

[R7] *The following applies to RHEL7 only:*

Having an identical /etc/multipath/bindings on all the systems is another way to accomplish this goal.

Failure Timeouts

When using multipath, it is desirable to have the underlying paths detect failures quickly, and once detected, fail quickly. This allows the multipath layer to move pending I/O requests to another path and resume normal operations quickly. How this is done is specific to the particular type of connection used, for example, with FC HBA cards this is usually done with kernel module parameters.

If a multipath device contains a critical filesystem (e.g. `/`), then use the `queue_if_no_path` option so that I/O will be queued at the multipath layer until a path returns. This is like a hard NFS mount.

Path selection

In the commented out defaults section example of `/etc/multipath.conf`, the `path_selector` is set to "round-robin 0". The round-robin selector uses each path in a path group equally. Recent dm-multipath implementations also provide the `queue-length` and `service-time` and use `service-time` as the default. These new algorithms allow for path selection based on the number of outstanding I/O requests, or observed service time.

LUN resizing

When a LUN has been resized, be sure to rescan for each underlying device (determined with `multipath -ll`).

```
# echo 1 > /sys/block/sdX/device/rescan
```

Once the underlying block devices have been rescanned, *then* resize the multipath device:

```
# multipathd -k "resize map multipath_device"
```

Only do this if all paths are active and there are no queued commands.

Resetting Multipath Configuration

To reset or remove the multipath configuration from the machine, first stop using any multipath devices, and then run the following commands:

```
# systemctl stop multipathd
# systemctl disable multipathd
# rm /etc/multipath.conf /etc/multipath/*
# multipath -F
```

iSCSI Architecture

Target = iSCSI terminology for Server

- A Target acts as a SCSI storage device
- A Target contains one or more LUNs
LUN = Typically a numbered read/writable disk object

Initiator = iSCSI terminology for Client

Addressing provides unique names for targets and initiators

- Typically an iSCSI Qualified Name (IQN)

Security at the Target and Discovery levels

- Username & password via CHAP
- Mutual authentication possible
- IPsec for confidentiality and integrity

Several performance enhancing techniques available

optional TCP port number (defaults to 3260 if undefined). Default portal is configured as 0.0.0.0 on port 3260. If iSCSI Extensions for RDMA are supported in hardware iSER can be enabled within the portal's node.

Storage Object ⇒ Refers to the mapping between Target LUNs and backstores.

backstore ⇒ When using a Linux server with iSCSI target software, the backstore for each LUN is usually an LVM logical volume that resides on hardware RAID. This gives great flexibility as the logical volume can be resized as needed. Other options for the backing store are disks, partitions, block devices, plain files, and even RAM Disks.

Addressing

In order to identify iSCSI targets and initiators (nodes) an addressing scheme is needed—an iSCSI name. An iSCSI name is a location independent, permanent identifier for an iSCSI node.

An iSCSI node has one or more iSCSI addresses that specifies a single path to an iSCSI node. An iSCSI address consists of two components: the iSCSI name plus the Network Portal.

For the iSCSI name, the standard defines three possible formats: iSCSI Qualified Name (IQN), Extended Unique Identifier (EUI), T11 Network Address Authority (NAA). The most commonly used format is IQN, examples follow:

iqn.2001-04.com.example:iscsibox-sn-a3a32309

iSCSI Terminology

Much of the iSCSI terminology is directly imported from SCSI terminology with additions for iSCSI specifics. Whereas most networking services use the common client/server nomenclature, iSCSI instead refers to clients as initiators and servers as targets. An iSCSI node can be either an initiator, a target, or both.

Targets, LUNs, and Portals

To understand targets and LUNs in an iSCSI context it is helpful to first examine targets and LUNs in the physical SCSI world. A single physical SCSI disk usually provides a single target, which in turn provides a single LUN. In another example, a SCSI RAID card with multiple volumes created from one or more RAID sets presents each volume to the system as a LUN. The LUNs would be inside of a SCSI target. A target is effectively a logical container for LUNs.

TPG ⇒ Target Portal Group. allows the iSCSI target to support multiple complete configurations within one target node. A single TPG is created by default, and almost all setups only need one.

LUN ⇒ iSCSI LUNs (Logical UNits) are used the same way physical disks are used, to hold data after some sort of initialization.

Generally you can think of a LUN and hard disk interchangeably as far as what you do with them.

Target ⇒ Targets are similar to NFS exports, they are destinations for initiators to connect to. An iSCSI Target node typically has multiple targets with one or more LUNs defined in each target.

Portal ⇒ The Network Portal is an IPv4 or IPv6 address with an

iqn.1999-07.com.gurulabs:virtual-tape-library
iqn.1999-07.com.gurulabs:users.dkelson:project13.target2
iqn.1999-07.com.gurulabs:81746e9edb1

- ❖ iqn indicates the IQN format
- ❖ 1999-07 the year and month the organization acquired the domain name used in the IQN. Ensures unique IQNs even when the domain name is sold or transferred
- ❖ com.gurulabs reversed DNS name of the organization
- ❖ virtual-tape-library a string of the organization's choosing, must be unique

The NAA format provides compatibility with SCSI/SAS and Fibre Channel ids. The EUI format provides compatibility with the SCSI RDMA Protocol (SRP) used on InfiniBand networks, a high speed alternative to Ethernet common in high performance computing environments. Examples of NAA and EUI iSCSI names:

naa.209C325CD809855D
eui.02010507AB37234C

Both targets and initiators can have an iSCSI Alias that is a simple descriptive string up to 255 characters in length. They are meant to be a descriptive label that software can display next to the iSCSI name for the benefit of humans. For example an initiator might have an alias of Web Server #4 and a target might have an alias of Database Server #2 Log Disk.

Security

When an initiator connects to a target, the target can optionally require authentication. This is done with a username and password defined by the target. The Challenge-Handshake Authentication Protocol (CHAP) allows for authentication without ever sending the password over the wire. This is important since there is no built in encryption as part of the iSCSI protocol. The username and password must be set on both the target and the initiator. Once authenticated to a target, the initiator will have access to all LUNs in the target unless the target supports LUN masking. Depending on security compartmentalization requirements it may be a good idea to have a separate target and separate username/password for each initiator.

Mutual authentication can be enabled and a separate username and password defined for the target to authenticate back to initiator. This way the initiator can detect a rogue hostile target.

Additional security can be provided via ACLs on the TPG; lun mappings must be created under the ACL that refers back to the TPG LUN. ACLs with the addition of firewall rules that filter initiator source addresses make for a more robust security profile. to the Network Portal.

Because of CHAP, passwords will not be exposed on the wire, but all iSCSI traffic operates in plain text. If an attacker can capture or modify packets on the network, a skilled attacker will be able to copy file contents or alter file contents. A first step to prevent these attacks is to use a dedicated LAN segment for iSCSI traffic. This can be done with a physically separate switch, or with a VLAN. For complete confidentiality and integrity verification, iSCSI can be deployed on top of IPsec, however depending on CPU utilization, performance can take a hit unless IPsec accelerator hardware is used.

By default iSCSI depends on error detection in the IP and TCP layers, however, at high speed the 16-bit TCP checksum will not necessarily detect all errors, resulting in possible, silent, data corruption. As an additional layer of protection, iSCSI has an optional header and data digest using a 32-bit CRC that can be enabled.

Most iSCSI target software allows IP whitelist ACLs to be defined for allowed clients in addition to, or instead of, CHAP authentication.

iSCSI Session Types

To make configuration easier on an initiator, the initiator can establish a discovery session with an iSCSI target node and issue the SendTargets command. The iSCSI target node responds to the command with a list of targets which the initiator may access as well as a list of iSCSI addresses for each target. This is somewhat analogous to the **showmount -e** discovery command with NFS. Some iSCSI targets can be configured to require CHAP authentication for discovery sessions.

Besides a discovery session, a normal operational session is the only other type of session that can be established.

Performance Considerations

In the iSCSI design, each Network Portal (an IP address and TCP port usable by iSCSI) is assigned to a Portal Group (a logical container). In the simplest (and most common) configuration, an iSCSI target node with a single NIC will have a single Portal Group with a single

Network Portal in it. However, multiple NICs in a Portal Group provide the groundwork for multi-pathing.

The iSCSI design supports two types of multi-pathing I/O (MPIO). With the historical MPIO method, two or more sessions are established between the initiator and target to the same LUN(s). Operating system specific MPIO software on the initiator, such as device-mapper multipath on Linux, manages the multiple sessions requiring complex configuration. The improved method uses multiple connections within a single session (MC/S) along with SCSI connection recovery known as Error Recovery Level 2 (ERL2). With MC/S and ERL2 the iSCSI layer itself provides multi-pathing features including full load balancing and failover, and is operating system independent.

The iSCSI Extensions for remote DMA (iSER) protocol provides zero-copy networking over TCP for improved latency and throughput especially over high speed networks such as 10GbE.

On an Ethernet network subject to congestion, putting all iSCSI traffic on its own VLAN and prioritizing that VLAN over regular traffic can ensure consistent performance. Alternatively, using IP QoS techniques to prioritize iSCSI traffic on TCP port 3260 can offer the same benefits.

Open-iSCSI Initiator Implementation

Open-iSCSI Initiator

- Linux has a single iSCSI Initiator Implementation
- Highly optimized kernel and userland components
- Persistent configuration database

Userland daemon - `iscsid`

Userland configuration utility - `iscsiadm`

Default configuration defined in `/etc/iscsi/`:

- `/etc/iscsi/iscsid.conf`
- `/etc/iscsi/initiatorname.iscsi`

Red Hat Enterprise Linux: Persistent configuration stored under
`/var/lib/iscsi/` in text files

SUSE Linux Enterprise Server: Persistent configuration stored under
`/etc/iscsi/` in text files

simultaneously modifies the persistent configuration databases. There are four main operating modes for `iscsiadm`:

discovery ⇒ In discovery mode an iSCSI target node is queried to discover what targets are available. The results are stored in the database and the values of all the settings are populated from the `/etc/iscsi/iscsid.conf`. The iSCSI target node may require CHAP authentication for the discovery to succeed. Three discovery methods are available: a direct query using the `sendtargets` operation to a specific IP, a multicast Service Location Protocol (SLP) query, or an Internet Storage Name Service (iSNS) query.

discoverydb ⇒ In discoverydb mode a record and settings for a `sendtargets` target in the persistent database can be populated without having to edit `/etc/iscsi/iscsid.conf`. If using `discoverydb` to define settings for a target, the actual discovery must also be made in this mode otherwise the `discoverydb` record in the database will be overwritten by defaults in `/etc/iscsi/iscsid.conf`.

node ⇒ In node mode, target login and logout for session establishment can be performed. Upon login a local block device is mapped to each LUN in the target. In node mode properties for the target can also be set.

session ⇒ In session mode, information about running sessions can be displayed.

iface ⇒ In iface mode, network portals can be defined. If iface mode isn't used, then the Linux network stack figures out the best path to use. This mode is primarily used when configuring multipathing.

Open-iSCSI Architecture

Linux is fortunate to have a single, mature, excellent iSCSI initiator that has been part of Linux since 2005. The architecture of the Open-iSCSI initiator has the kernel portion only handling high speed data transfer. All other control logic such as configuration, iSCSI discovery, log in/out, error processing, keep alive via Nop-In and Nop-Out, iSNS, SLP, and Radius are handled by the userspace daemon. This clean separation is what led to the quick acceptance of the code into the official kernel.

The Open-iSCSI userspace daemon is `iscsid`. Persistent configuration is created when a target is discovered for the first time, using defaults from the `/etc/iscsi/iscsid.conf` configuration file. Changes to the configuration file will not affect existing node settings in the database.

[R7] The following applies to RHEL7 only:

Persistent configuration is stored in files under the `/var/lib/iscsi/` directory tree. As a result, booting the system from an iSCSI target requires that `/var/lib/` be located within the root filesystem.

[S12] The following applies to SLES12 only:

Persistent configuration is stored in files under the `/etc/iscsi/` directory tree.

Persistent Configuration

Configuration is performed by the `iscsiadm` command which

The iSCSI Initiator Name

The iSCSI initiator name is stored in the /etc/iscsi/initiatorname.iscsi file in the InitiatorName variable. The value is an iSCSI Qualified Name (IQN) and is automatically generated when the package is installed. As a helper for humans, an alias can be defined in the same file with the InitiatorAlias variable. For example:

```
File: /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1999-07.com.gurulabs:81746e9edb1
InitiatorAlias=station3.example.com
```

iSCSI Initiator Discovery

Discovery method

- Unicast sendtargets operation (most commonly used)
- Multicast Service Location Protocol (SLP) query
- Unicast Internet Storage Name Service (iSNS) query

iSCSI Target node must have initiator in IP ACL whitelist

Discovery authentication may be enforced by iSCSI Target node

- Initiator to Target Auth
 - discovery.sendtargets.auth.username
 - discovery.sendtargets.auth.password
- Target to Initiator Auth
 - discovery.sendtargets.auth.username_in
 - discovery.sendtargets.auth.password_in

Update database entries purging targets that no longer exist and adding new targets for a specific iSCSI target:

```
# iscsiadm -m discovery -p 10.100.0.254 -o new -o delete
```

Enabling Discovery CHAP Authentication

The sendtargets settings for a target are stored in the file /var/lib/iscsi/send_targets/10.100.0.254_3260/st_config.

When using the discovery method, this file is created and populated using the values from the /etc/iscsi/iscsid.conf file when the discovery is performed.

If the iSCSI target node requires CHAP authentication (potentially mutual) in order to do target discovery, before issuing the sendtargets command, make the following edit:

File: /etc/iscsi/iscsid.conf

```
+ discovery.sendtargets.auth.authmethod = CHAP
+ # initiator -> target auth
+ discovery.sendtargets.auth.username = username
+ discovery.sendtargets.auth.password = password
+ # target -> initiator auth
+ discovery.sendtargets.auth.username_in = username
+ discovery.sendtargets.auth.password_in = password
```

However, this can be achieved without editing the /etc/iscsi/iscsid.conf by using the discoverydb mode.

iSCSI Target Discovery

Discover a list of available targets from an iSCSI target node by issuing a sendtargets command.

```
# iscsiadm -m discovery -t sendtargets -p 10.100.0.254
10.100.0.254:3260,1 iqn.1999-07.com.gurulabs:target01
10.100.0.254:3260,1 iqn.1999-07.com.gurulabs:target02
```

Get details from the database by setting the printlevel option to one:

```
# iscsiadm -m discovery -P 1
SENDTARGETS:
DiscoveryAddress: 10.100.0.254,3260
Target: iqn.1999-07.com.gurulabs:target01
    Portal: 10.100.0.254:3260,1
    Iface Name: default
Target: iqn.1999-07.com.gurulabs:target02
    Portal: 10.100.0.254:3260,1
    Iface Name: default
DiscoveryAddress: server1.example.com,3260
iSNS:
No targets found.
. . . snip . . .
```

Delete all database entries pertaining to a specific iSCSI target node with the delete operation:

```
# iscsiadm -m discovery -p 10.100.0.254 -o delete
```

When defining settings using discoverydb mode, it's important that the discover and subsequent operations are also made using the discoverydb method since a regular discovery will overwrite settings using the values in the /etc/iscsi/iscsid.conf.

For example:

```
# iscsiadm -m discoverydb -t st -p 10.100.0.254 -o new
# iscsiadm -m discoverydb -t st -p 10.100.0.254 -o update -n discovery.sendtargets.auth.authmethod -v CHAP
# iscsiadm -m discoverydb -t st -p 10.100.0.254 -o update -n discovery.sendtargets.auth.username -v jkelson
# iscsiadm -m discoverydb -t st -p 10.100.0.254 -o update -n discovery.sendtargets.auth.password -v smileykid
# iscsiadm -m discoverydb -t st -p 10.100.0.254 -o update -n discovery.sendtargets.auth.username_in -v LIOrules
# iscsiadm -m discoverydb -t st -p 10.100.0.254 -o update -n discovery.sendtargets.auth.password_in -v itsmeforsure
# iscsiadm -m discoverydb -t st -p 10.100.0.254 --discover
```

iSCSI Initiator Node Administration

Using iscsiadadm in node mode

- Primarily logging into a previously discovered target
During login LUNs in target become usable
- Properties for a target can be configured

Session CHAP authentication may be enforced by iSCSI Target node

- Initiator to Target Auth
 - node.session.auth.username
 - node.session.auth.password
- Target to Initiator Auth
 - node.session.auth.username_in
 - node.session.auth.password_in

```
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -u
```

Logging out of session [sid: 5, target: iqn.1999-07.com.gurulabs:target01, portal: 10.100.0.254,3260]
Logout of [sid: 5, target: iqn.1999-07.com.gurulabs:target01, portal: 10.100.0.254,3260]: successful

Viewing Current Session Information

To examine session information for a running session use the session mode. This is particularly useful to see what device file the remote LUNs have been assigned:

```
# iscsiadadm -m session -P 3
```

iSCSI Transport Class version 2.0-871
version 2.0-871

Target: iqn.1999-07.com.gurulabs:target01
Current Portal: 10.100.0.254:3260,1
Persistent Portal: 10.100.0.254:3260,1

Interface:

Iface Name: default
Iface Transport: tcp
Iface Initiatorname: iqn.1999-07.com.gurulabs:81746e9edb1
Iface IPaddress: 10.100.0.3
Iface HWaddress: <empty>
Iface Netdev: <empty>
SID: 6

iSCSI Session Authentication Config

Typically iSCSI is deployed with CHAP authentication for security. After the targets have been discovered, a username and password can be added to the database entries for both outgoing and incoming authentication using the update operation. For example:

```
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.session.auth.authmethod -v CHAP
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.session.auth.username -v guru
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.session.auth.password -v work
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.session.auth.username_in -v server1
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.session.auth.password_in -v itsreallyme
```

iSCSI Login/Logout and Automatic Start

To login to a specific target that is defined in the database including using defined authentication settings:

```
# iscsiadadm -m node -T iqn.1999-07.com.gurulabs:target01 -l
Logging in to [iface: default, target: iqn.1999-07.com.gurulabs:target01, portal: 10.100.0.254,3260]
Login to [iface: default, target: iqn.1999-07.com.gurulabs:target01, portal: 10.100.0.254,3260]: successful
```

To logout use the **-u** option:

```
iSCSI Connection State: LOGGED IN
iSCSI Session State: LOGGED_IN
Internal iscsid Session State: NO CHANGE
*****
Negotiated iSCSI params:
*****
HeaderDigest: None
DataDigest: None
MaxRecvDataSegmentLength: 262144
MaxXmitDataSegmentLength: 8192
FirstBurstLength: 65536
MaxBurstLength: 262144
ImmediateData: Yes
InitialR2T: Yes
MaxOutstandingR2T: 1
*****
Attached SCSI devices:
*****
Host Number: 6 State: running
scsi6 Channel 00 Id 0 Lun: 0
scsi6 Channel 00 Id 0 Lun: 1
Attached scsi disk sdb           State: running
```

LUN Rescanning

When a LUN on a SAN (Fiber Channel or iSCSI) is resized, issue a rescan to detect the changed size. For iSCSI the following will detect changed sizes (and new LUNs):

```
# iscsiadm -m node --targetname target_name -R
```

Behind the scenes this does:

```
# echo "----" > /sys/class/scsi_host/host/scan
# echo 1 > /sys/block/sdX/device/rescan
```

Mounting iSCSI Targets at Boot

`systemctl enable iscsid`
Enable automatic login to iSCSI targets
Special treatment in /etc/fstab

- Use `_netdev` option

Error handling

Enabling Automatic Start at Boot

The SysV init script for the open-iSCSI initiator is called `iscsid`. Be sure that it is set to start at boot.

To configure the initiator to automatically log in to the target on startup, modify the database record:

```
# iscsiadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.startup -v automatic
```

/etc/fstab and the `_netdev` Option

During boot, all locally attached filesystems in the `/etc/fstab` are automatically mounted. With filesystems on top of iSCSI the mount attempt will fail if the network has not yet been started and the initiator launched. Use the `_netdev` option for all filesystems that reside on top of an iSCSI target either directly or indirectly. For example:

File: `/etc/fstab`

```
+ /dev/vg0/wwwdata /srv/wwwdata ext3 _netdev 1 2
```

Error Handling

If the initiator loses connection to the iSCSI target node for any reason including network failure, hardware failure, reboot or crash of the iSCSI target node, the `node.session.timeout.replacement_timeout` setting determines how long the iSCSI initiator layer will queue and retry before giving up and returning I/O errors to applications performing I/O. This is a more flexible version of NFS's soft vs hard setting. The default setting is 120 seconds, you may want to increase it to a large value such as hours or days if you prefer the initiator to effectively enter suspended animation until the iSCSI target node become accessible again. Here is an example of setting the value to one day (aka 86400 seconds):

```
# iscsiadm -m node -T iqn.1999-07.com.gurulabs:target01 -o update -n node.session.timeout.replacement_timeout -v 86400
```

If you are using multipathing, the timeout should be reduced to 5 or 10 seconds to allow the multipath layer to queue the requests if all paths are down instead of the iSCSI layer.

iSCSI Multipathing Considerations

Use iface mode to define multiple network portals

Discovery and login via each network portal

Goal: dm-multipath should react quickly

- iSCSI should detect errors quickly
- On error detection, iSCSI should fail quickly

Defining multiple Network Portals

A redundant network topology will entail an iSCSI target node having multiple network cards. Each network card is connected to a different Ethernet switch. An iSCSI target node may even have two or more network cards bonded together with each bond group connected to a different switch. Effectively the iSCSI target node will have multiple IP addresses that it will be reachable on.

On the initiator distinct iSCSI interfaces can be defined using the iface mode of **iscsiadm**, and then discovery and login are done per interface. An example of an iSCSI target node that is accessible by IPs 10.100.0.254 and 10.5.4.1 and an initiator with two NICs:

```
# iscsiadm -m iface -I iface0 --op=new
# iscsiadm -m iface -I iface0 --op=update,
  -n iface.hwaddress -v 00:0C:29:6C:5B:80
# iscsiadm -m iface -I iface1 --op=new
# iscsiadm -m iface -I iface1 --op=update,
  -n iface.hwaddress -v 00:0C:29:6C:5B:8A
# iscsiadm -m iface
default tcp,default,unknown,default,unknown
iser iser,default,unknown,default,unknown
bnx2i bnx2i,default,unknown,default,unknown
iface1 tcp,00:0C:29:6C:5B:8A,default,default,unknown
iface0 tcp,00:0C:29:6C:5B:80,default,default,unknown
# iscsiadm -m discovery -t sendtargets -p 10.100.0.254 -I iface0
# iscsiadm -m discovery -t sendtargets -p 10.5.4.1 -I iface1
# iscsiadm -m node -l
```

Under some routing conditions, each interface's rp_filter strict mode configuration may cause packet loss. To compensate, set the net.ipv4.conf.interface.rp_filter boolean from 1 to 2 (and add to /etc/sysctl.conf for persistence):

```
# sysctl -w net.ipv4.conf.p3p1.rp_filter=2
```

Adjusting iSCSI timers

The iSCSI and SCSI stacks have error handling and timers to hide transient storage errors from applications. Normally that is a desired behavior, but when multipathing is being used the timers and error handling cause a delay before the multipathing layer sees the error and then fails over. The goal is to adjust the default iSCSI timers so that hard failures are propagated up the stack to the multipathing layer quickly. However, if the timers are lowered too much, then short lived errors or fluctuations in the Ethernet network can cause unnecessary failover.

iSCSI Ping

To detect network problems the iSCSI layer sends iSCSI pings to the target. These are also known as noop requests. By default they are sent every 5 seconds and each ping response has 5 seconds to return before an error condition is raised and session reestablishment is attempted. These defaults (as shown below) are usually acceptable but can be lowered if desired with the **iscsiadm** command:

```
node.conn[0].timeo.noop_out_interval = 5
node.conn[0].timeo.noop_out_timeout = 5
```

iSCSI Session Re-establishment Timer

By default iSCSI has a 120 second timer during which it queues I/O commands while trying to re-establish a connection. In a multipath configuration it is recommended to set value between 5 to 15 sec:

```
node.session.timeo.replacement_timeout = 5
```

Lab 7

Estimated Time:
S12: 65 minutes
R7: 65 minutes

Task 1: Using autofs

Page: 7-37 Time: 10 minutes

Requirements: (1 station) (classroom server)

Task 2: NFS Server Configuration

Page: 7-42 Time: 15 minutes

Requirements: (2 stations)

Task 3: iSCSI Initiator Configuration

Page: 7-47 Time: 20 minutes

Requirements: (1 station) (classroom server)

Task 4: Multipathing with iSCSI

Page: 7-55 Time: 20 minutes

Requirements: (1 station) (classroom server)

Objectives

- ❖ Configure autofs to access predefined NFS shares
- ❖ Configure autofs to access arbitrary NFS shares

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

Since the ability to mount NFS shares is normally restricted to the root user, the autofs system is very useful in allowing normal users to access NFS shared files on an as needed basis.

- 1) View the current contents of the /etc/auto.misc file:

```
$ cat /etc/auto.misc
. . . snip . . .
cd           -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom

# the following entries are samples to pique your imagination
#linux        -ro,soft,intr          ftp.example.org:/pub/linux
#boot         -fstype=ext2          :/dev/hda1
#floppy       -fstype=auto          :/dev/fd0
#floppy       -fstype=ext2          :/dev/fd0
#e2floppy     -fstype=ext2          :/dev/fd0
#jaz          -fstype=ext2          :/dev/sdc1
#removable    -fstype=ext2          :/dev/hdd
```

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso 
```

- 3) Modify /etc/auto.misc adding two new autofs targets: tmp and netinstall that point to NFS shares on server1:

File: /etc/auto.misc
+ tmp -rw,soft,intr server1.example.com:/export/tmp
+ netinstall -ro,soft,intr server1.example.com:/export/netinstall

Lab 7

Task 1

Using autofs

Estimated Time: 10 minutes

4) [S12] This step should only be performed on SLES12.

Edit the /etc/auto.master file and uncomment the entry that maps the /misc directory to the /etc/auto.misc file:

File: /etc/auto.master
- #/misc /etc/auto.misc
+ /misc /etc/auto.misc

5) Verify that the /misc automount point is listed (and not commented out) in the /etc/auto.master file:

```
# grep ^/misc /etc/auto.master
/misc    /etc/auto.misc
```

6) Restart autofs, and verify that it is monitoring the defined automount points:

```
# systemctl restart autofs
. . . output omitted . . .
```

7) Trigger the mounts to occur by accessing the targets:

```
# cd /misc/tmp
# ls -al
total 0
[R7] drwxrwxrwt. 2 nobody nobody 4096 Dec 20 09:32 .
[R7] drwxr-xr-x. 3 root      root     0 Dec 20 09:48 ..
[S12] drwxrwxrwt 2 root root 6 Dec 19 15:24 .
[S12] drwxr-xr-x 5 root root 0 Jan 13 12:06 ..
# >foo
# ls -al
total 0
[R7] drwxrwxrwt. 2 root      root     16 Jan 13 12:08 .
[R7] drwxr-xr-x. 3 root      root     0 Jan 13 12:12 ..
[R7] -rw-r--r--. 1 nfsnobody nfsnobody 0 Jan 13 12:08 foo
[S12] drwxrwxrwt 2 root      root     16 Jan 13 12:08 .
[S12] drwxr-xr-x 5 root      root     0 Jan 13 12:06 ..
[S12] -rw-r--r-- 1 nobody nogroup  0 Jan 13 12:08 foo
```

```
# cd /misc/netinstall
# ls -al
total 28
drwxr-xr-x 8 root root 4096 Dec 19 17:59 .
drwxr-xr-x 5 root root 0 Jan 13 12:06 ..
drwxr-xr-x 2 root root 4096 Dec 19 18:00 configs
drwxr-xr-x 10 root root 4096 Dec 19 17:51 OL7
drwxr-xr-x 4 root root 101 Dec 19 17:59 postinstalldata
-rw-r--r-- 1 root root 154 Dec 10 14:24 README.txt
drwxr-xr-x 10 root root 4096 Dec 19 17:54 RHEL7
drwxr-xr-x 7 root root 4096 Dec 19 17:56 SLES12
drwxr-xr-x 12 root root 4096 Dec 19 17:58 U1604
# mount | grep nfs
. . . output omitted . . .
```

After changing to the two directories and listing their contents, what filesystems now show as mounted?

- 8) [S12] This step should only be performed on SLES12.

Enable the /net autofs directory which enables the arbitrary accessing of any NFS share. Edit the /etc/auto.master file:

File: /etc/auto.master

-	#/net -hosts
+	/net -hosts

- 9) Verify that the /net automount point is listed (and not commented out) in the /etc/auto.master file:

```
# grep ^/net /etc/auto.master
/net -hosts
```

- 10) [S12] This step should only be performed on SLES12.

Restart autofs to activate the change:

```
# systemctl restart autofs
. . . output omitted . . .
```

- 11) Use the /net/ autofs directory to access a shared NFS directory on server1. As with all autofs use, this works for all users, not just root:

```
# cd /net
# ls -al
total 2
drwxr-xr-x 2 root root 1024 2006-12-27 20:35 .
drwxrwxrwt 8 root root 1024 2006-12-27 20:35 ..
[S12] dr-xr-xr-x 2 root root 0 Jan 13 12:20 localhost
# cd server1
# ls -al
total 0
dr-xr-xr-x 3 root root 0 2006-12-27 20:33 .
drwxr-xr-x 3 root root 0 2006-12-27 20:33 ..
dr-xr-xr-x 8 root root 0 2006-12-27 20:33 export
# cd export
# ls -al
total 48
dr-xr-xr-x 8 root root 0 2006-12-27 20:33 .
dr-xr-xr-x 3 root root 0 2006-12-27 20:33 ..
drwxr-xr-x 2 root root 4096 2006-12-21 16:40 autoyast
drwxr-xr-x 7 root root 4096 2006-12-26 14:53 courserepos
drwxr-xr-x 22 root root 4096 2006-06-13 10:25 home
. . . snip . .

# cd home
# ls -al
total 168
drwxr-xr-x 22 root root 4096 2006-06-13 10:25 .
dr-xr-xr-x 8 root root 0 2006-12-27 20:33 ..
drwx----- 4 2001 2001 4096 2006-06-13 10:25 dsuser1
. . . snip . .
```

- Notice that the directory is empty.
- You could have used the IP address or the FQDN of the machine instead.
- Note that the /export share appears. If the server had additional top level directories shared, they would appear also.

- 12) Change your current working directory out of the autofs controlled directory and examine all the directories automatically mounted by autofs:

```
# cd /
# mount | grep nfs
. . . output omitted . .
```

- 13) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Setup an NFS server and export directories
- ❖ Explore the root_squash and all_squash options

Requirements

█ █ (2 stations)

Relevance

NFS is the de-facto network file system standard for Unix/Linux. Learning how to manage filesystems across networks is a very powerful asset.

Notices

- ❖ Working in pairs, make sure to replace the reference to stationY with your lab partner's station number. If needed, stationY can refer to your own system.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Run the following commands to create some directories, set permissions, and add a file to one of them:

```
# cd /tmp; mkdir {dir,remote_dir}{1,2}  
# chmod 777 dir{1,2}  
# echo "your_name was here." > dir1/file
```

- 3) Create the NFS share /tmp/dir1 by editing /etc/exports:

File: /etc/exports
+ /tmp/dir1 *(sync)

- 4) The systemd unit for the NFS daemons only starts the server daemons if there are contents in the /etc/exports file. Restart the NFS daemons by running the following command:

```
[R7] # systemctl restart nfs  
[S12] # systemctl restart nfsserver
```

Lab 7

Task 2

NFS Server Configuration

Estimated Time: 15 minutes

5) Wait until both lab partners have completed Step 4 before proceeding.

6) Verify that the NFS share can be mounted:

```
# mount stationY:/tmp/dir1 /tmp/remote_dir1
# mount | grep dir1
stationY:/tmp/dir1 on /mnt type nfs4 (rw,relatime,vers=4.0,rsize=131072,wsize=131072,namlen=255,-
    hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=10.100.0.X,local_lock=none,addr=10.100.0.Y)
```

7) Attempt to create a file in the mounted directory as both the root and guru user:

```
# cd /tmp/remote_dir1
# touch test1
touch: cannot touch `test1': Read-only file system
# su guru
$ touch test2
touch: cannot touch `test2': Read-only file system
$ exit
# cd /tmp
```

- This will take more than a few seconds. Please, be patient.

Why were the permission denied errors generated?

8) Wait until both lab partners have completed Step 7 before proceeding.

9) Both lab partners modify the entry in the /etc/exports file so that they are sharing the directory read-write:

File: /etc/exports
- /tmp/dir1 *(sync)
+ /tmp/dir1 *(rw,sync)

10) Notify the NFS daemon that the /etc/exports file has been modified:

```
# exportfs -a
[S12] . . . output omitted . . .
```

- 11) Remount the directory and repeat the tests you performed earlier (in Step 7):

```
# mount -o remount stationY:/tmp/dir1 /tmp/remote_dir1
# cd /tmp/remote_dir1
# touch test1
# su guru
$ touch test2
```

- 12) Finally, execute a long directory listing in the mounted directory and look at the new file's owner and group information:

```
$ ls -l
total 1
[R7] -rw-r--r--. 1 root      root       16 Jun 9 12:35 file
[R7] -rw-r--r--. 1 nfsnobody nfsnobody   0 Jun 9 12:56 test1
[R7] -rw-rw-r--. 1 guru      guru       0 Jun 9 12:56 test2
[S12] -rw-r--r-- 1 root      root      16 Jun 9 12:35 file
[S12] -rw-r--r-- 1 nobody    nogroup    0 Jun 9 12:56 test1
[S12] -rw-rw-r-- 1 guru      users      0 Jun 9 12:56 test2
$ exit
```

Can you explain the ownership of the newly created files?

- 13) Wait until both lab partners have completed Step 12 before proceeding.

- 14) Temporarily export /tmp/dir2/ to your lab partner this time without changing the /etc/exports file and specifying read-write and no-root-squash options:

```
# exportfs -o rw,no_root_squash stationY:/tmp/dir2
[S12] . . . snip . . .
```

- 15) After your lab partner has completed the step above, verify that the directory has been successfully exported:

```
# showmount -e stationY
Export list for stationY:
/tmp/dir1 *
/tmp/dir2 stationX.example.com
```

• new share with the no_root_squash option

- 16) Mount the NFS share and verify that the no_root_squash option works as you would expect:

```
# mount stationY:/tmp/dir2 /tmp/remote_dir2
# cd /tmp/remote_dir2
# touch test3
# ls -l
total 1
-rw-r--r-- 1 root      root      0 Jun  9 12:58 test3
```

- The no_root_squash option allows the newly created file to be owned by the root user.

- 17) Wait until both lab partners have completed Step 16 before proceeding.

- 18) Add a new group that can be used to test the all_squash option with a specified GID:

```
# groupadd -g 5000 team1
```

- 19) Re-export the second directory with new options:

```
# exportfs -o rw,all_squash,anongid=5000 stationY:/tmp/dir2
```

- 20) Create a new file to test the operation of the all_squash option:

```
# cd /tmp/remote_dir2
# touch test4
# ls -l test4
[R7] -rw-r--r--. 1 nfsnobody team1 0 Sep 26 15:03 test4
[S12] -rw-r--r-- 1 nobody   team1 0 Sep 26 15:03 test4
```

Remounting the share was not needed as NFS is stateless. Note that the group ownership has been set to the specified GID.

Cleanup

- 21) Delete the group:

```
# groupdel team1
```

- 22) Unmount both of the mounted directories:

```
# cd /
# umount /tmp/remote_dir{1,2}
```

- 23) Remove the share, and stop the NFS service:

```
# sed -i '/dir1/d' /etc/exports
[R7] # systemctl stop nfs
[S12] # systemctl stop nfsserver
. . . snip . . .
```

- 24) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Install Open-iSCSI Initiator
- ❖ Connect to a LUN on server1
- ❖ Use LUN as PV for a new LVM Volume Group

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

iSCSI is becoming an increasingly common storage component in networks of all sizes. Being able to configure and use Linux as an initiator to existing iSCSI targets is an important skill for deploying Linux into modern infrastructure.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Ensure the Open-iSCSI initiator package is installed:

```
[R7] # yum install -y iscsi-initiator-utils lsscsi  
[S12] # zypper install -y open-iscsi lsscsi
```

- 3) [R7] This step should only be performed on RHEL7.

Change the default configuration so that iSCSI targets are not automatically logged into as it is best practice to enable automatic login on a per target basis.

File: /etc/iscsi/iscsid.conf

-	node.startup = automatic
+	node.startup = manual

- 4) Configure Open-iSCSI daemon, iscsid to start at boot.

```
# systemctl enable iscsid
```

Lab 7

Task 3

iSCSI Initiator Configuration

Estimated Time: 20 minutes

- 5) Use iscsiadadm and discover the pre-configured iSCSI targets available on server1.example.com:

```
# iscsiadadm -m discovery -t sendtargets -p server1.example.com
. . . snip . .
10.100.0.254:3260,1 iqn.1999-07.com.example:station3
10.100.0.254:4260,1 iqn.1999-07.com.example:station3-2ndpath
10.100.0.254:3260,1 iqn.1999-07.com.example:station2
10.100.0.254:4260,1 iqn.1999-07.com.example:station2-2ndpath
10.100.0.254:3260,1 iqn.1999-07.com.example:station1
10.100.0.254:4260,1 iqn.1999-07.com.example:station1-2ndpath
. . . snip . .
```

Notice how there is a target for each possible student machine. Ignore the 2ndpath targets, these are used for the multipath lab.

- 6) Perform the discovery populated the database with entries. Examine the contents of the database:

```
# iscsiadadm -m discovery -P 1
SENDTARGETS:
DiscoveryAddress: server1.example.com,3260
. . . snip . .
Target: iqn.1999-07.com.example:station3
    Portal: 10.100.0.254:3260,1
        Iface Name: default
Target: iqn.1999-07.com.example:station3-2ndpath
    Portal: 10.100.0.254:4260,1
        Iface Name: default
Target: iqn.1999-07.com.example:station2
    Portal: 10.100.0.254:3260,1
        Iface Name: default
Target: iqn.1999-07.com.example:station2-2ndpath
    Portal: 10.100.0.254:4260,1
        Iface Name: default
Target: iqn.1999-07.com.example:station1
    Portal: 10.100.0.254:3260,1
        Iface Name: default
Target: iqn.1999-07.com.example:station1-2ndpath
    Portal: 10.100.0.254:4260,1
```

```
Iface Name: default  
. . . snip . . .
```

- 7) The iSCSI targets on server1 are configured to require authentication. Update the database to enable CHAP authentication. Perform this ONLY for the target that corresponds with your assigned station number:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.authmethod -v CHAP
```

- 8) Update the database with an outgoing username and password that your initiator software will use to authenticate to iSCSI target software running on server1. The username and password has already been configured on server1. Perform this ONLY for the target that corresponds with your station number:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.username -v stationX  
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.password -v letmein
```

- 9) Update the database with an incoming username and password that the iSCSI target software running server will use to authenticate to your initiator software. The username and password has already been configured on server1. Perform this ONLY for the target that corresponds with your station number:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.username_in -v server1  
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.password_in -v itsreallyme
```

- 10) Inspect the database configuration for the target and confirm that the authentication settings have been stored:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o show -S | grep auth  
. . . snip . . .  
node.session.auth.authmethod = CHAP  
node.session.auth.username = stationX  
node.session.auth.password = letmein  
node.session.auth.username_in = server1  
node.session.auth.password_in = itsreallyme  
. . . snip . . .
```

- 11) Before logging in, examine the current state of the system to see what existing block devices it is aware of:

```
# cat /proc/partitions  
. . . output omitted . . .  
# lsscsi  
. . . output omitted . . .  
# lsscsi -t  
. . . output omitted . . .
```

After you login to the target this list will expand to include block devices that map to the iSCSI target LUNs.

- 12) Login to the target:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -l  
Logging in to [iface: default, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,3260]  
Login to [iface: default, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,3260]: successful
```

- 13) Configure the system to login to the target automatically at boot:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.startup -v automatic
```

- 14) Now that the connection is established, examine the session data in increasing amounts of verbosity:

```
# iscsiadadm -m session  
tcp: [14] 10.100.0.254:3260,1 iqn.1999-07.com.example:stationX  
# iscsiadadm -m session -P 1 • Interface and Portal data is displayed.  
Target: iqn.1999-07.com.example:stationX  
    Current Portal: 10.100.0.254:3260,1  
    Persistent Portal: 10.100.0.254:3260,1  
    *****  
    Interface:  
    *****  
    Iface Name: default  
    Iface Transport: tcp  
    Iface Initiatorname: iqn.1994-05.com.redhat:a5d21275df73  
    Iface IPaddress: 10.100.0.X  
    Iface HWaddress: <empty>
```

```
Iface Netdev: <empty>
SID: 14
iSCSI Connection State: LOGGED IN
iSCSI Session State: LOGGED_IN
Internal iscsid Session State: NO CHANGE
# iscsiadm -m session -P 2
. . . output omitted . . .
# iscsiadm -m session -P 3
. . . snip . . .
*****
Attached SCSI devices:
*****
Host Number: 14 State: running
scsil4 Channel 00 Id 0 Lun: 0
scsil4 Channel 00 Id 0 Lun: 1
Attached scsi disk sdb           State: running
```

- Negotiated iSCSI parameters are added to the display.
- Importantly, attached SCSI devices are added to the display.

Notice that with -P 3 each LUN is displayed along with the Linux device file name for the LUN.

- 15) Use the lsscsi command in various ways to see how the iSCSI LUN is displayed:

```
# lsscsi
[0:0:0:0]    cd/dvd  TSSTcorp CDRWDVD TS-H492C DE02  /dev/sr0
[2:0:0:0]    disk    ATA      ST3160318AS        CC37  /dev/sda
[3:0:0:0]    disk    LIO-ORG t5            4.0    /dev/sdb
# lsscsi -l
. . . output omitted . . .
# lsscsi -t
[0:0:0:0]    cd/dvd  ata:                  /dev/sr0
[2:0:0:0]    disk    ata:                  /dev/sda
[3:0:0:0]    disk    iqn.1999-07.com.example:stationX,t,0x1 /dev/sdb
```

- 16) From the output above, **record** the local Linux device file that maps to the LUN (/dev/sdb in the sample output shown):

Result: _____

- 17) Examine the current state of the system to see what existing block devices it is aware of:

```
# cat /proc/partitions  
. . . output omitted . . .
```

Since you logged into the target, this list includes the block device that maps to the iSCSI target LUN (the local Linux device file recorded in step 16).

- 18) Use the iSCSI LUN to create a LVM Logical Volume Group and logical volume. Be sure to use the local Linux device file recorded above:

```
# pvcreate /dev/sdb  
Physical volume "/dev/sdb" successfully created  
# vgcreate iSCSIVg /dev/sdb  
Volume group "iSCSIVg" successfully created  
# lvcreate iSCSIVg -L 225M -n sales  
Rounding up size to full physical extent 228.00 MB  
Logical volume "sales" created
```

- 19) Create an ext4 filesystem on top of the newly created LV /dev/iSCSIVg/sales:

```
# mkfs -t ext4 /dev/iSCSIVg/sales  
. . . output omitted . . .
```

- 20) Turn off automatic checking by disabling the maximal mount count and interval settings:

```
# tune2fs -c 0 -i 0 /dev/iSCSIVg/sales  
. . . output omitted . . .  
# tune2fs -l /dev/iSCSIVg/sales | grep options  
Default mount options: user_xattr acl
```

• Note the default mount options.

- 21) Create a mount point for the filesystem:

```
# mkdir /srv/sales
```

- 22) Edit the /etc/fstab file so that the new iSCSI based /dev/iSCSIvg/sales LVM logical volume will be mounted at /srv/sales/:

```
File: /etc/fstab
+ /dev/iSCSIvg/sales  /srv/sales  ext4  _netdev  0 0
```

The _netdev option causes the mount to occur after the network is activated and the iSCSI initiator software started.

- 23) Mount the filesystem to test the /etc/fstab entry:

```
# mount /srv/sales/
```

- 24) Copy a file to /srv/sales/ and create a SHA256 checksum to use for file integrity checking after a reboot:

```
# cp /etc/passwd /srv/sales/
# sha256sum /srv/sales/passwd > /srv/sales/SHA256
```

- 25) Reboot to test that everything is properly activated and the filesystem mounted at boot:

```
# reboot
```

- 26) Log in as the guru user. Verify the existence and integrity of /srv/sales/passwd:

```
$ sha256sum -c /srv/sales/SHA256
/srv/sales/passwd: OK
```

Cleanup

- 27) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso 
```

- 28) Return the system to its original state so that future lab tasks are not impacted:

```
# umount /srv/sales/  
# vgchange iSCSIvg -a n  
 0 logical volume(s) in volume group "iSCSIvg" now active  
# iscsadm -m node -T iqn.1999-07.com.example:stationX -u  
Logging out of session [sid: 4, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,3260]  
Logout of [sid: 4, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,3260]: successful  
# iscsadm -m discovery -p server1.example.com -o delete
```

- 29) Edit the /etc/fstab file and remove the added entry:

File: /etc/fstab
- /dev/iSCSIvg/sales /srv/sales ext4 _netdev 0 0

- 30) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Configure system for multipathing
- ❖ Setup multiple paths to LUN on server1
- ❖ Block path to server1 and watch multipath take place

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

Setting up multipathing in an enterprise environment is critical to ensure a constant connection to LUNs in the event of a path failure.

Notices

- ❖ Ethernet device name may vary. This lab assumes eth0.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Ensure the Open-iSCSI initiator package is installed:

```
[R7] # yum install -y iscsi-initiator-utils lsscsi  
[S12] # zypper install -y open-iscsi lsscsi
```

- 3) [R7] *This step should only be performed on RHEL7.*

Change the default configuration so that iSCSI targets are not automatically logged into as it is best practice to enable automatic login on a per target basis.

File: /etc/iscsi/iscsid.conf

-	node.startup = automatic
+	node.startup = manual

- 4) Configure Open-iSCSI daemon, iscsid to start at boot.

```
# systemctl enable iscsid
```

Lab 7

Task 4

Multipathing with iSCSI

Estimated Time: 20 minutes

- 5) [R7] This step should only be performed on RHEL7.

Modify the SELinux policy to allow iscsid to use TCP port 4260:

```
# semanage port -at iscsi_port_t -p tcp 4260
```

- 6) Perform a discovery to see what targets server1 has:

```
# iscsiadadm -m discovery -p 10.100.0.254 -t sendtargets
10.100.0.254:3260,1 iqn.1999-07.com.example:station1
10.100.0.254:4260,1 iqn.1999-07.com.example:station1-2ndpath
10.100.0.254:3260,1 iqn.1999-07.com.example:station2
10.100.0.254:4260,1 iqn.1999-07.com.example:station2-2ndpath
10.100.0.254:3260,1 iqn.1999-07.com.example:station3
10.100.0.254:4260,1 iqn.1999-07.com.example:station3-2ndpath
. . . snip . . .
```

Notice there are two targets for each classroom station. Each pair of stationX targets has the same LUN within it.

- 7) Configure authentication for the first stationX target:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.authmethod -v CHAP
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.username -v stationX
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.password -v letmein
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.username_in -v server1
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.auth.password_in -v itsreallyme
```

- 8) Configure authentication for the second stationX target:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.session.auth.authmethod -v CHAP
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.session.auth.username -v stationX
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.session.auth.password -v letmein
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.session.auth.username_in -v server1
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.session.auth.password_in -v itsreallyme
```

- 9) Inspect the database configuration for the targets and confirm that the authentication settings have been stored:

```
# iscsiadadm -m node -T iqn.1999-07.com.example:stationX -o show -S | grep auth
```

```
... snip ...
node.session.auth.authmethod = CHAP
node.session.auth.username = stationX
node.session.auth.password = letmein
node.session.auth.username_in = server1
node.session.auth.password_in = itsreallyme
... snip ...
# iscsiadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o show -S | grep auth
... snip ...
node.session.auth.authmethod = CHAP
node.session.auth.username = stationX
node.session.auth.password = letmein
node.session.auth.username_in = server1
node.session.auth.password_in = itsreallyme
... snip ...
```

- 10) By default, iSCSI will retry I/O requests for 120 seconds. When using multipathing, the time should be greatly reduced so that the multipath layer can do its job. The replacement_timeout setting controls the timeout. Set it to 5 seconds:

```
# iscsiadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.session.timeo.replacement_timeout -v 5
# iscsiadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.session.timeo.replacement_timeout -v 5
```

- 11) Configure iscsid to automatically log into this target at boot:

```
# iscsiadm -m node -T iqn.1999-07.com.example:stationX -o update -n node.startup -v automatic
# iscsiadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -o update -n node.startup -v automatic
```

- 12) Login to both targets:

```
# iscsiadm -m node -T iqn.1999-07.com.example:stationX -l
Logging in to [iface: default, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,3260] (multiple)
Login to [iface: default, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,3260] successful.
# lsscsi -t
[0:0:0:0]    disk    ata:                      /dev/sda
[1:0:0:0]    cd/dvd  ata:                      /dev/sr0
[4:0:0:0]    disk    iqn.1999-07.com.example:stationX,t,0x1  /dev/sdb --- • The iSCSI LUN
# iscsiadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -l
Logging in to [iface: default, target: iqn.1999-07.com.example:stationX-2ndpath, portal: 10.100.0.254,4260] (multiple)
Login to [iface: default, target: iqn.1999-07.com.example:stationX, portal: 10.100.0.254,4260] successful.
```

```
# lsscsi -t
[0:0:0:0]    disk   ata:                               /dev/sda
[1:0:0:0]    cd/dvd  ata:                               /dev/sr0
[4:0:0:0]    disk   iqn.1999-07.com.example:station5,t,0x1  /dev/sdb
[5:0:0:0]    disk   iqn.1999-07.com.example:station5-2ndpath,t,0x1  /dev/sdc
```

The iSCSI LUN on the second path

- 13) Compare the SCSI ID values reported by both LUNs:

```
# /lib/udev/scsi_id --whitelisted /dev/sdb
3600140527e296d83b1748f5951344e05
# /lib/udev/scsi_id --whitelisted /dev/sdc
3600140527e296d83b1748f5951344e05
```

Notice the IDs are the same.

- 14) Examine the iSCSI session information:

```
# iscsiadm -m session -P 3 | less
. . . snip . . .
```

• type q to quit.

- 15) Record information of the session and drives:

```
# iscsiadm -m session -P 3 | egrep 'Current|Attached scsi'
Current Portal: 10.100.0.254:3260,1
Attached scsi disk sdb      State: running
Current Portal: 10.100.0.254:4260,1
Attached scsi disk sdc      State: running
```

• Take note of the TCP port number.
• Take note of the attached disk.
• Take note of the TCP port number.
• Take note of the attached disk.

- 16) From the output of the previous command record which *Attached Disk* and *Port* is being used for each device. This information will be needed later in the lab so ports can be blocked to show the multipathing failover:

Device	Attached Disk	Port Number
/dev/sdb		3260
/dev/sdc		4260

- 17) [R7] This step should only be performed on RHEL7.

Use the `mpathconf` command to build a basic multipath configuration file:

```
# mpathconf --enable --with_multipathd y --find_multipaths y
```

- 18) [S12] This step should only be performed on SLES12.

Create a basic configuration file for multipath:

File: /etc/multipath.conf

```
+ defaults {  
+     user_friendly_names yes  
+     find_multipaths yes  
+ }
```

- 19) Flush all unused multipath maps, detect multiple paths to devices:

```
# multipath -F  
# multipath -v2  
create: mpatha (3600140527e296d83b1748f5951344e05) undef LIO-ORG ,t5  
size=250M features='0' hwhandler='0' wp=undef  
|--- policy='service-time 0' prio=1 status=undef  
|   `-- 4:0:0:0 sdb 8:16 undef ready running  
`--- policy='service-time 0' prio=1 status=undef  
    `-- 5:0:0:0 sdc 8:32 undef ready running
```

- 20) View the friendly device naming mapping file:

```
# cat /etc/multipath/bindings  
... snip ...  
mpatha 3600140527e296d83b1748f5951344e05
```

- 21) Enable multipathd and iscsid to start at boot:

```
# systemctl enable --now multipathd
```

- Starting this service, will also load the required kernel module: `dm-multipath`.

```
# systemctl enable iscsid
```

- The iscsid service does not need to be started by hand as it was previously started by the iscsiadm tool during discovery.

- 22) Make a note of the device name of the multipath device, and which block device is being used for the active path:

```
# multipath -ll
mpatha (3600140527e296d83b1748f5951344e05) dm-4 LIO-ORG ,t5
size=250M features='0' hwhandler='0' wp=rw
|--- policy='service-time 0' prio=1 status=active
|   `-- 4:0:0:0 sdb 8:16 active ready running
`--- policy='service-time 0' prio=1 status=enabled
    `-- 5:0:0:0 sdc 8:32 active ready running
```

Make note of the multipath device name.

Result: _____

- 23) Create a GPT partition table, and a single partition of maximum size on the multipath device:

```
# sgdisk -p -N1 /dev/mapper/mpatha
Creating new GPT entries.
Disk /dev/mapper/mpatha: 512000 sectors, 250.0 MiB
Logical sector size: 512 bytes
Disk identifier (GUID): B4226C8B-679A-42A7-B247-4CBC105D6A50
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 511966
Partitions will be aligned on 2048-sector boundaries
Total free space is 511933 sectors (250.0 MiB)
```

Number Start (sector) End (sector) Size Code Name

Warning: The kernel is still using the old partition table.

The new table will be used at the next reboot.

The operation has completed successfully.

- 24) Display the newly created partition table:

```
# sgdisk -p /dev/mapper/mpatha
Disk /dev/mapper/mpatha: 512000 sectors, 250.0 MiB
Logical sector size: 512 bytes
Disk identifier (GUID): B4226C8B-679A-42A7-B247-4CBC105D6A50
```

Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 511966
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	511966	249.0 MiB	8300	

- 25) Run kpartx on the multipath device to create a device node for the partition:

```
# kpartx -av /dev/mapper/mpatha
[R7] add map mpatha1 (253:5): 0 509919 linear /dev/mapper/mpatha 2048
[R7] add map mpatha_part1 (253:5): 0 509919 linear /dev/mapper/mpatha 2048
```

- 26) Create a XFS filesystem on the partition:

```
[R7] # mkfs.xfs /dev/mapper/mpatha1
[R7] . . . output omitted . .
[R7] # mount /dev/mapper/mpatha1 /mnt
[S12] # mkfs.xfs /dev/mapper/mpatha_part1
[S12] . . . output omitted . .
[S12] # mount /dev/mapper/mpatha_part1 /mnt
```

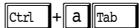
- 27) Install screen for multi-shell use in future steps:

```
[R7] # yum install -y screen
[S12] # zypper install -y screen
```

- 28) Write over and over again to the filesystem handled by multipathing devices:

```
# screen -U
+a +S
+a tab
+a c
# cd /mnt/
# touch foo
# while true; do
> cp foo fool; rm -f foo
> cp fool foo; rm -f fool
```

• This will interminably print date command output for each change to the filesystem.

```
> date  
> sleep .5  
> done  

```

- Print each time the copy process was run.

- 29) Use the port identified in the previous step for the firewall rule cause the first path to fail:

```
# iptables -A OUTPUT -d 10.100.0.254 -p tcp --dport 3260 -j REJECT
```

- Time stamps should stop updating after issuing this command. Use 4260 if that is the path being used.

- 30) On a remote connection, it might take a few moments for the file operations to show as stopped (as represented by the date command output). Run multipath -ll to see the enabled portal is now different:

```
# multipath -ll  
... snip ...  
| +- policy='service-time 0' prio=1 status=active  
|   `-- 4:0:0:0 sdb 8:16 failed faulty running  
`-- policy='service-time 0' prio=1 status=enabled  
   `-- 5:0:0:0 sdc 8:32 active ready running
```

- Note this is faulty.

- 31) Remove the firewall rule so that the first path can function again:

```
# iptables -D OUTPUT -d 10.100.0.254 -p tcp --dport 3260 -j REJECT
```

- 32) Repeatedly monitor the status until the first path returns to active ready running:

```
# multipath -ll  
... snip ...  
|   `-- 4:0:0:0 sdb 8:16 failed ready running  
... snip ...  
# multipath -ll  
... snip ...  
|   `-- 4:0:0:0 sdb 8:16 active ready running  
... snip ...
```

Cleanup

- 33) Remove the multipath device and iSCSI configuration so that it doesn't interfere with future labs:

```
# [Ctrl]+[a][Ctrl]+[\\]  
[screen is terminating]  
# umount /mnt  
# rm -f /etc/multipath.conf  
# rm /etc/multipath/bindings  
# multipath -F  
# iscsadm -m node -T iqn.1999-07.com.example:stationX -u  
# iscsadm -m node -T iqn.1999-07.com.example:stationX-2ndpath -u  
. . . output omitted . . .  
# iscsadm -m discovery -p 10.100.0.254 -o delete  
# systemctl disable --now iscsid multipathd
```

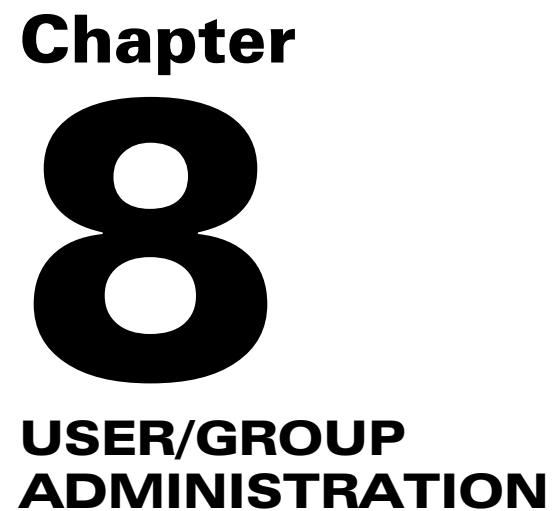
- If connecting through a remote or virtual interface, the key combination may not work. If not, each shell will need running programs stopped (if any; use `[Ctrl]+[C]`) before exiting. When all shells are closed, screen will terminate on its own.

- 34) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```


Content

Approaches to Storing User Accounts	2
User and Group Concepts	3
User Administration	4
Modifying Accounts	6
Group Administration	7
Password Aging	9
Default User Files	11
Controlling Login Sessions	12
RHEL DS Client Configuration	14
SLES DS Client Configuration	16
System Security Services Daemon (SSSD)	18
Lab Tasks	20
1. User and Group Administration	21
2. Using LDAP for Centralized User Accounts	24
3. Troubleshooting Practice: Account Management	30



Chapter

8

USER/GROUP ADMINISTRATION

Approaches to Storing User Accounts

Standalone server

- Uses /etc/ text files

Centralized accounts via a Directory Service

- LDAP Server with RFC2307 schema
With Optional Kerberos Authentication
- Network Information Service (NIS)
- Microsoft Active Directory

Abstraction layers enable switching backends

- Pluggable Authentication Modules (PAM)
- Name Service Switch (NSS)
- System Security Services (SSS)

User and Group Accounts in Linux

Linux can store the list of user and group accounts in a variety of locations. In a standalone server configuration, flat text files with one user or group per line in the /etc/ directory are used to store the accounts. Ad-hoc solutions can be created relatively easily to synchronize these text files among a group of servers, but in practice, actual full fledged directory services are typically used. The first UNIX directory service, Network Information Service (NIS), was released in 1985 by Sun Microsystems under the name Yellow Pages. Today Linux generally uses LDAP with an RFC2307 schema combined with Kerberos authentication, such as provided by FreeIPA. Linux can also integrate with an existing Microsoft Active Directory.

Account Database Abstraction

The Linux operating system has many applications that need to access the list of user and group accounts and perform hostname and IP address lookups ranging from the **ls** command to **sshd**. Instead of each application reinventing the wheel processing the files in the /etc/ directory directly, applications make use of various functions in the system C Library. The C functions **getpwnam(3)** and **getpwuid(3)** get the standard 7 fields of information (username, encrypted password, user id, primary group id, GECOS info, home directory, and login shell) about a user by name or UID, respectively. The C functions **getgrnam(3)** and **getrgid(3)** does the same thing from groups. The **getspnam(3)** function returns the shadow password fields including the encrypted/hashed password. The **getpwnam(3)** and **gethostbyname(3)** and **gethostbyaddr(3)** convert

hostnames to IP address and vice versa.

Account Database Abstraction

On a standalone system, these C functions typically and ultimately access and use the /etc/ files such as /etc/passwd, /etc/group, and /etc/hosts. However, the actual backend used by these functions is controlled by the Name Service Switch (NSS), the /etc/nsswitch.conf, and installed NSS modules. See the **nss(5)** and **nsswitch.conf(5)** man pages.

The **getent** command is a generic command that uses the standard C functions, which in turn follows NSS's /etc/nsswitch.conf. So while the command **grep jkelson /etc/passwd** will only work if the /etc files are the current backend in use, the following command will work no matter the backend:

```
# getent passwd jkelson
jkelson:x:50051:50051:Jada Kelson:/home/jkelson:/bin/bash
```

While NSS maps the traditional /etc files to different backends, Pluggable Authentication Modules (PAM) plays a complimentary role by allowing authentication, accounting, password changing and session setup to be performed by arbitrary backends.

System Security Services (SSS) and its daemon, **sssd**, acts as a backend for both NSS and PAM, whose main innovation is adding caching and offline support to an LDAP/Kerberos backend.

User and Group Concepts

```
/etc/passwd
• -rw-r--r-- root root
• account:password:UID:GID:GECOS:directory:shell
/etc/group
• -rw-r--r-- root root
• group:password:GID:user1,user2,user3
/etc/shadow
• r----- root root
```

User and Group Database Files

User account information in Unix is usually stored in two world-readable files: /etc/passwd and /etc/group. Each line of the passwd file defines a separate user account with the following fields:

dkelson:x:464:464:Nice Guy:/home/dkelson:/bin/bash

1. account name
2. encrypted password (or placeholder "x" if shadow passwords are in use)
3. numerical UID
4. numerical GID for user's primary group
5. GECOS entry (e.g. full name, contact information)
6. path to user's home directory
7. program to launch at login (usually shell)

Each line of the group file defines a separate group and consists of four colon-delimited fields:

staff:x:200:bob,joe,sally

1. group name
2. group password (if assigned; or placeholder "x" if shadow passwords are in use)
3. unique numerical GID
4. comma-separated list of users assigned to the group

The Shadow System

Most Linux distributions use a shadow password system to overcome security deficiencies inherent in the original Unix password system. Following the shadow system, no password data is stored in the world-readable passwd and group files. Encrypted passwords are instead stored in separate files which only the root user can read. User passwords are stored in the /etc/shadow file. The /etc/shadow file also controls advanced password parameters, such as password aging. Group passwords are stored in the /etc/gshadow file. When a group password is set, a user must enter the group's password to access files owned by that group, (instead of through group membership).

[S12] *The following applies to SLES12 only:*

SUSE Linux Enterprise Server no longer supports /etc/gshadow.

Allowed Characters

Usernames are case-sensitive. Numbers (and some other characters) are allowed. Mixed-case usernames are allowed but highly discouraged, (remember that email addresses are not case-sensitive and are typically constructed from user names).

For compatibility with legacy systems and software, limiting user and group names to 8 lowercase characters is recommended.

User Administration

Creating new users

- useradd
- newusers

Deleting existing users

- Clean-up user files before deleting user account
- userdel

Validate existing users

- pwck

GUI Tools

- RHEL7: system-config-users
- SLES12: yast2 users

Adding Users to the System

New users can be added to the system by the root user with the **useradd** command. Behavior of the **useradd** program is controlled by the following:

- ❖ command-line options
- ❖ /etc/login.defs configuration file
- ❖ /etc/default/useradd configuration file

Examples of Adding a New User Account

Create a new user account using the defaults for all parameters:

```
# useradd jdoe
# passwd jdoe
Changing password for user jdoe.
New password: secret 
Retype new password: secret 
passwd: authentication tokens updated successfully.
```

Set the GECOS full name, and force a specific UID:

```
# useradd -u 30001 -c "Bryan Croft" bcroft
```

Create a system account with no home directory, no interactive shell, and set initial group membership (to an existing group):

```
# useradd -r -s /sbin/nologin -g sys -G sup1 derek
```

Deleting User Accounts

The corresponding **userdel** command is used to delete existing accounts. **userdel** only supports a single option **-r**, which tells the command to delete the user's account, home directory and mail spool file. When using **userdel -r**, a common practice is to first note the user's UID before deleting their account. That UID can then be used to locate files owned by that user on the system outside the user's home directory. A command like the following might then be used to delete that user's other files:

```
# find / -uid 1701 -exec rm -rf {} +
```

[R7] *The following applies to RHEL7 only:*

The **useradd** command for RHEL7 automatically creates a private group for the user and creates the user's home directory (**useradd -m**). If you want to suppress the creation of the private group, use the **-n** option. If you want to suppress the creation of the home directory, use the **-M** option.

[S12] *The following applies to SLES12 only:*

The **useradd** command for SLES12 does not create the home directory automatically. Use the **-m** option to cause the home directory to be created.

Automating Account Creation

To create several accounts with initial passwords set to the value

"secret" you can use a simple loop from the shell:

[R7] The following applies to RHEL7 only:

```
# for newacct in bailey dax kim ross sophia; do  
>   useradd $newacct  
>   echo secret | passwd --stdin $newacct  
> done
```

[S12] The following applies to SLES12 only:

```
# for newacct in bailey dax kim ross sophia; do  
>   useradd $newacct  
>   echo -e "secret\nsecret" | passwd $newacct  
> done
```

The **newusers** command can be used to bulk import accounts from a file. Create a file, modeled after /etc/passwd, that lists the users and passwords. Use the file name as argument to **newusers**. For example:

```
# cat accounts  
bryan:secret:501:501:Bryan Croft:/home/bryan:/bin/bash  
dax:secret:502:502:Dax Kelson:/home/dax:/bin/bash  
. . . snip . . .  
# newusers accounts
```

By default, the **newusers** command will store the provided clear-text passwords in the crypt format when creating the entry in the /etc/shadow file if there is no ENCRYPT_METHOD setting in the /etc/login.defs to override that default. Check for the existence and value of that setting before using **newusers**.

Validating Accounts

The command **pwck** checks the validity of the /etc/passwd & /etc/shadow files. It does this by validating the each entry in /etc/passwd for:

- ❖ The correct number of fields
- ❖ A unique and valid user name
- ❖ A valid user and group identifier
- ❖ A valid primary group
- ❖ A valid home directory
- ❖ A valid login shell

pwck then validates each entry in /etc/shadow for:

- ❖ Every passwd entry has a matching shadow entry, and every shadow entry has a matching passwd entry
- ❖ Passwords are specified in the shadowed file
- ❖ Shadow entries have the correct number of fields
- ❖ Shadow entries are unique in shadow
- ❖ The last password changes are not in the future

If any errors are detected, **pwck** will prompt to fix them unless the **-r** option is passed:

```
# pwck  
user 'ftp': directory '/var/ftp' does not exist  
user 'pulse': directory '/var/run/pulse' does not exist  
user 'oprofile': directory '/var/lib/oprofile' does not exist  
user 'brokenuser': no group 1011  
user 'brokenuser': directory '/home/brokenuser' does not exist  
no matching password file entry in /etc/shadow  
add user 'brokenuser' in /etc/shadow? No [Enter]  
pwck: no changes
```

Modifying Accounts

Changing existing accounts

- **vipw/vigr**
- **usermod**
- **chsh**
- **chfn**
- **passwd**

Modifying User Accounts

Although /etc/passwd and /etc/group are flat text files and can be edited with normal text editors, the **vipw** and **vigr** commands should instead be used to edit these files. **vipw** by default will run whatever editor is defined by the \$EDITOR environmental variable on /etc/passwd after first locking it to prevent simultaneous edits. **vigr** by default does the same, except it opens the /etc/group file for editing. If changes are made to the passwd or group file, **vipw** or **vigr** will display a prompt to edit the corresponding shadow or gshadow file.

In addition to editing the passwd file directly, the **usermod** command can be used by root to modify existing accounts. The following examples show a few common uses of the **usermod** command:

Move a user to a new home directory (updating the passwd entry and moving existing files):

```
# usermod -d /home2/brandon -m brandon
```

Lock an account:

```
# usermod -L rebekah
```

Rename an account (also changing the name of the home directory to match):

```
# usermod -l julie -d /home/julie -m julianne
```

Editing Accounts as a Non-Privileged User

Some changes to the passwd file can also be made with other commands which can even be run by non-privileged users; **chsh** can be used to change the user's shell field to any shell listed in the /etc/shells file, while **chfn** can change the user's GECOS entry, and **passwd** can change the user's password in the shadow file.

An example of using the **chfn** and **finger** commands is shown here:

```
$ chfn
```

Changing finger information for guru.

Password:

```
Name []: Guru Lab User
```

```
Office []: Room1
```

```
Office Phone []: (801)298-5227
```

```
Home Phone []:
```

Finger information changed.

```
$ finger guru
```

```
Login: guru
```

```
Directory: /home/guru
```

```
Office: Room1, (801)298-5227
```

```
Name: Guru Lab User
```

```
Shell: /bin/bash
```

```
. . . snip . . .
```

Group Administration

Creating new groups

- groupadd

Deleting existing groups

- groupdel

Changing existing groups

- groupmod

RHEL7 GUI Tool

- system-config-users

SLES12 GUI Tool

- yast2 users

Group Administration

Most of the commands for manipulating user accounts have corresponding commands for managing groups. As it's important to use **vipw** to modify the /etc/passwd file, it is likewise important to use the **vigr** command to edit the /etc/group file so that multiple administrators do not try to simultaneously modify the file.

Assigning Secondary Groups to a User

Possibly counter-intuitive, to add secondary groups to a user, the **usermod** command is used. For example:

```
# usermod -G webguys,mis jdoe
```

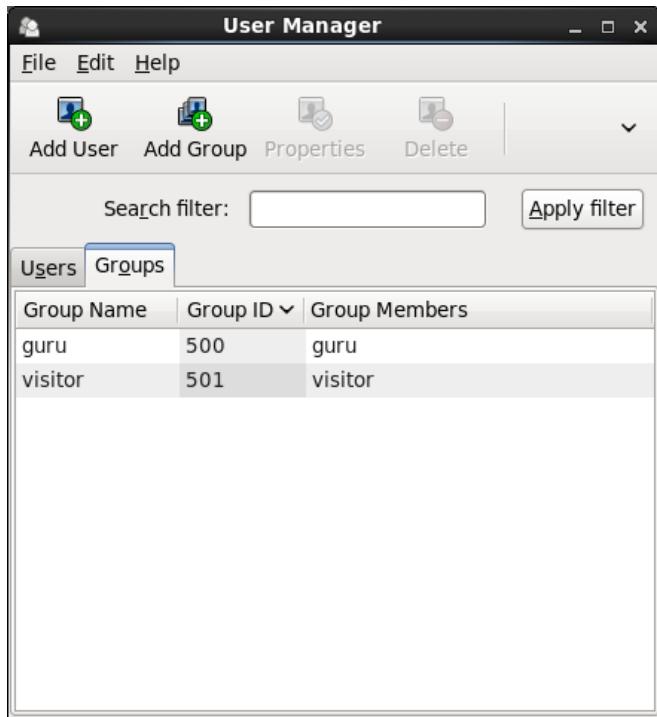
Note that all secondary groups that the user should be a member of need to be listed. The user will be removed from any secondary groups not listed. The most common way of making a relative change is to edit the /etc/group file directly.

The **-a** option can be used with the **usermod** command in RHEL7/SLES12 to make a relative change to a user's group memberships. The following example shows the usage of this option:

```
# id sarah
uid=5003(sarah) gid=5003(sarah) groups=5003(sarah),510(admin)
# usermod -a -G music sarah
# id sarah
uid=5003(sarah) gid=5003(sarah) groups=5003(sarah),-
      510(admin),520(music)
```

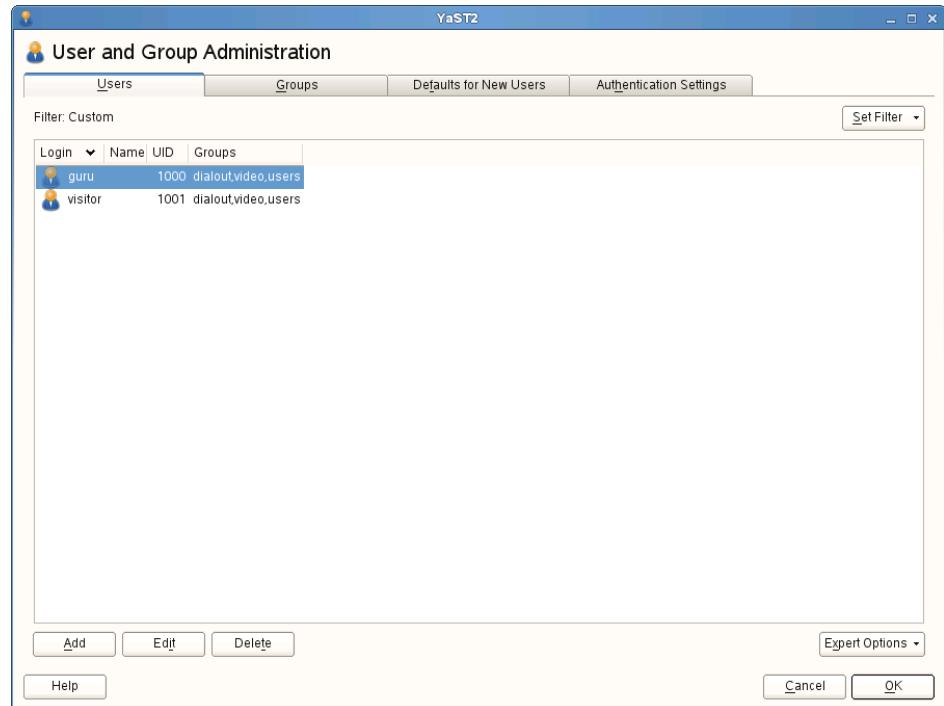
Graphical User and Group Administration on RHEL

The **system-config-users** graphical utility exists to help with the creation and management of both user and group accounts. By default it will only display non-system accounts and groups, but this feature can be toggled in the preferences tab:



Graphical User and Group Administration on SLES

The **yast2 users** module provides a graphical utility to help with the creation and management of both user and group accounts. The Expert Options button lets you configure the default values used when creating new users and set the password encryption method used.



Password Aging

/etc/shadow structure

- login name
- encrypted password
- date, in days since epoch, of last change
- number of days until change allowed
- number of days until change required
- number of days prior to expiration to begin warning
- number of days after expiration before account disabled
- date, in days since epoch, that password expires

System defaults set in /etc/login.defs

Manipulating individual user entries

- chage

chage Syntax and Examples

```
chage [-hl] [-d lastday] [-E expiredate] [-I inactive]
[-m mindays] [-M maxdays] [-R chrootdir ] [-W warndays] user
```

To set the account christec to have passwords that can be used for a maximum of 90 days, the following command can be run:

```
# chage -M 90 christec
```

Another common use of the **chage** commands is to force a user to set a new password by setting the Last Change field to Never. For example:

```
# chage -d 0 christec
```

Then the next time the user logs in, they are prompted to change their password as shown here:

```
station1 login: christec
```

```
Password: secret 
```

You are required to change your password immediately
(root enforced)

```
Changing password for christec
```

```
(current) UNIX password: secret 
```

```
New UNIX password: 2newsecret 
```

```
Retype new UNIX password: 2newsecret 
```

```
Password Changed
```

Configuring Password Aging

In addition to storing the encrypted user passwords, /etc/shadow also contains data regarding password security. These entries can be used to specify a minimum password age (or number of days the password must exist before it can be changed), a maximum password age (or number of days after which the password must be changed) as well as to enforce system policies regarding how much warning to give users. Although /etc/shadow can be directly edited by the system administrator, the **chage** command provides a convenient interface, supporting both an interactive and non-interactive method for modifying these settings.

The Epoch

Computers maintain internal system time in intervals since the epoch, the time which they recognize as zero. For the Unix family, the epoch is 00:00:00 GMT, January 1, 1970. The system time is measured in seconds since the epoch. This Unix custom is sometimes reflected in user programs, such as password aging (tracking password lifetimes in seconds since the epoch).

To set an account or password to never expire or to never become inactive a value of 0 or -1 can be supplied with the proper option. For example, to set the christec account to never expire, use the following command:

```
# chage -E 0 christec
```

System Wide Aging Defaults

Password aging can be configured with a global default by modifying the PASS_MAX_DAYS, PASS_MIN_DAYS, and PASS_WARN_AGE variables in the /etc/login.defs file.

Default User Files

/etc/skel/

- Bash configuration files
 /etc/skel/.bashrc
- GNOME configuration files
- KDE configuration files
- emacs configuration files

Bash System Files

- /etc/profile (general shell profile)
 /etc/profile.d/
- RHEL7: /etc/bashrc
- SLES12: /etc/bash.bashrc

The Template Home Directory, /etc/skel/

Configuration files which should be customized by each user are commonly placed in the /etc/skel/ directory. When new user accounts are created using **useradd**, the contents of this directory are copied over to the new user's home directory, providing the user with a default working configuration which can be customized further if desired. For example, the individual shell "run commands" file, executed each time a shell is run interactively, other than at log in, (e.g. ~/./kshrc).

The default path for the template directory used by the **useradd** command is specified in the /etc/default/useradd file. The template directory used by the **useradd** command can also be overridden with the **-k** option.

In addition to creating appropriate configuration files for /etc/skel/, the system administrator should also pay close attention to system-wide configuration files which the end user cannot modify. Examples of these include the system-wide shell configuration files, and other files such as the ~/vimrc and /etc/vimrc configuration files.

Controlling Login Sessions

systemd

- `systemd-logind.service` and `/etc/systemd/logind.conf`

Tracks users and sessions

Automatic spawning of text logins (gettys) on virtual terminals (TTYs)

`logindctl` — inspect and control user login sessions

login

- Provides text mode login prompt, respects:

- `/etc/nologin`
- `/etc/securetty`
- `~/.hushlogin`

Controlling Text Logins

`login`'s behavior on Linux can be manipulated in several different ways:

The file `/etc/nologin` can be created by the root user to prevent all logins to the system by non-root users; all users attempting to login will be refused access and shown the contents of the `nologin` file. The `/etc/nologin` file will automatically be deleted on reboot.

The `/etc/securetty` file can be used to restrict root access to the machine; root logins are permitted only on devices which are listed in this file. When using serial consoles it is important to have `/dev/ttyS0` through `/dev/ttyS3` in this file so that root may login.

`login` also checks for the presence of `.hushlogin` files in a user's home directory when they login. If `~/.hushlogin` is present, `login` will suppress display of most initial messages.

TTY pre-login Messages

The system provides pre-login messages for both logins on the virtual consoles and for telnet connections. Typically, these login messages say something like:

`Linux_Distro_Version_String`

Kernel 2.6.30-1 on an i686

login:

The pre-login message is stored in the `/etc/issue` and `/etc/issue.net` files. These files may contain escape sequences that

will be expanded into various values. For example, "`\s \n \m \v`" would be expanded into: the operating system name, the system's host name, the system architecture, and the OS version. Or in other words:

Linux server1 i386 #1 Tue Mar 19 21:54:09 MET 2011

A complete list of the usable tokens is found in the `mingetty` man page.

TTY post-login Messages

The contents of the `/etc/motd` file is displayed by the `login` program after a successful login, but before the user's shell is executed. This is often used as a way of communicating important messages to users on the system.

systemd-logind and logindctl

Systemd's `systemd-logind` daemon tracks and manages user logins. It keeps track of users and their sessions, including their idle state and processes. It also implements shutdown/sleep inhibition login for applications, responding to power/sleep hardware keys, device file access control for users and even multi-seat management.

The `logindctl` command allows the root user to inspect and manage user sessions and control `systemd-logind`. The `logindctl` supports a variety of commands, for example, after remotely SSHing into a box as root you can list login sessions, inspect a session, lock the session, and unlock a session with the following commands:

```
# logctl list-sessions
SESSION      UID USER          SEAT
 57          0  root
 71          2031 dkelson       seat0

2 sessions listed.
# logctl session-status 71
71 - dkelson (2031)
  Since: Mon 2015-02-02 16:30:09 MST; 11s ago
  Leader: 19120 (gdm-session-wor)
    Seat: seat0; vc1
  Display: :0
  Service: gdm-password; type x11; class user
    State: active
    Unit: session-71.scope
      |-19120 gdm-session-worker [pam/gdm-password]
      |-19129 /usr/bin/gnome-keyring-daemon --daemonize --login
. . . snip . . .
# logctl show-session 71
Id=71
Timestamp=Mon 2015-02-02 16:30:09 MST
TimestampMonotonic=21427493852
VTNr=1
Display=:0
Remote=no
Service=gdm-password
Scope=session-71.scope
Leader=19120
Audit=71
Type=x11
Class=user
Active=yes
State=active
IdleHint=no
IdleSinceHint=0
IdleSinceHintMonotonic=0
Name=dkelson
# logctl lock-session 71
# logctl unlock-session 71
```

Locking a GUI session will initiate the screensaver lock. The user can supply a password to the screensaver to unlock the session, or the root user can unlock the screensaver with **logctl**.

RHEL DS Client Configuration

system-config-authentication

- GUI menu driven automatic configuration file editing
- Can be run non-interactively with command line parameters
- NIS, LDAP, Kerberos, SMB, Hesiod

authconfig

- CLI/TUI equivalent utility

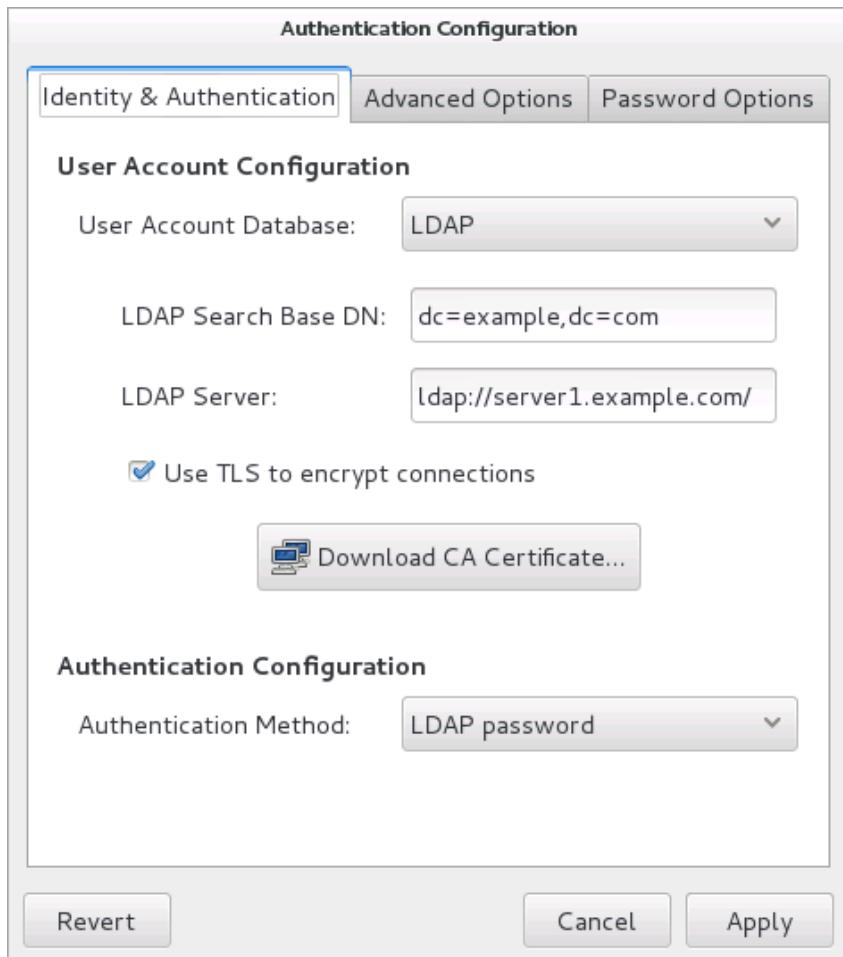
ipa-client-install

- Attach to a FreeIPA IdM Server

system-config-authentication

The **system-config-authentication** and **authconfig** tools configure the required files, depending on the chosen directory service and specified settings. This will help avoid introducing typos in the related configuration files that would prevent authentication, and reduce the complexity of the differing PAM and NSS files that may need to be modified.

Implementing more complex security policies may require that PAM and NSS files be edited directly. One of the problems that comes from manual configuration is that if **system-config-authentication** or **authconfig** are ever subsequently used, it makes file changes, such as in `/etc/pam.d/`, wiping out previous changes. The most common of these files is `/etc/pam.d/system-auth-ac`. To make changes that will be preserved, create a new file, such as `/etc/pam.d/system-auth-permanent`, and change the `/etc/pam.d/system-auth` symbolic link to point to it. Other relevant files include `password-auth-ac`, `smartcard-auth-ac`, and `fingerprint-auth-ac`. Other files that get modified include `/etc/nslcd.conf` and `/etc/openldap/ldap.conf`, which provides default configuration settings for the various LDAP command line tools and the location of the LDAP server's TLS certificate.



Local LDAP Name Service Daemon

The configuration for **nslcd** and **nss-pam-ldapd** might look like:

File: /etc/nslcd.conf

```
uid nslcd
gid ldap
uri ldap://server1.example.com
base dc=example,dc=com
ssl start_tls
tls_cacertdir /etc/openldap/cacerts
```

SLES DS Client Configuration

YaST2

- Menu driven automatic configuration file editing
- auth-client module

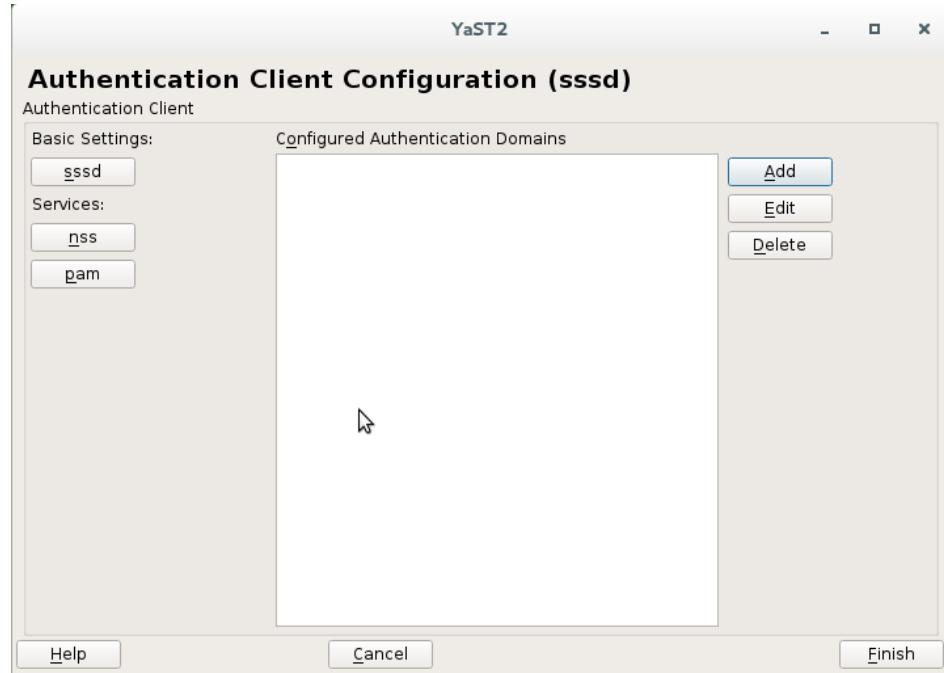
pam-config

- CLI equivalent utility

YaST

SUSE's YaST2 command has modules for both LDAP and NIS and will automatically configure the required files depending on the chosen directory service and specified settings. These modules will also configure TLS for LDAP, Kerberos, and the Linux Automounter. Using YaST2 for DS client configuration is recommended versus manual configuration to avoid typos and the complexity of configuring PAM and NSS.

Implementing more complex security policies may require that PAM and NSS files be edited directly. One of the problems that comes from manual configuration is that if YaST (or **pam-config**) are ever subsequently used, it makes file changes, such as in /etc/pam.d/, wiping out previous changes. The most common of these files is /etc/pam.d/common-auth-*pc*, but also include common-password-*pc*, common-account-*pc*, and common-session-*pc*. Other files that get modified include the NSS and PAM configuration /etc/ldap.conf file, and /etc/openldap/ldap.conf, which provides default configuration settings for the various LDAP command line tools and the location of the LDAP server's TLS certificate.



Example contents of /etc/ldap.conf:

File: /etc/ldap.conf

```
host ldap-srv.example.com
base dc=example,dc=com
ssl start_tls
pam_password md5
```

System Security Services Daemon (SSSD)

Maintains Compatibility with NSS and PAM

Caches LDAP-based Authentication

- Enabling Offline Authentication
- Reducing LDAP Server Connections

Supports Multiple Domains

- Transparent Merging
- Conflict-free Merging

System Security Services Daemon (SSSD)

The SSSD service provides client-side caching for LDAP-based user management. It also provides optional Kerberos integration. As it matures, it has the potential to provide additional security-related services.

By replacing nss_ldap and pam_ldap, SSSD provides new capabilities while maintaining backwards compatibility. Its most notable capability is caching user account information, thereby enabling offline authentication. SSSD also reduces load on the LDAP server by reusing a single existing connection instead of repeatedly creating new connections.

When configured for multiple domains (user databases), SSSD provides two alternative strategies for merging account names. By default, all domains will be transparently merged. Alternatively, individual domains can be configured to require the name of the domain as part of the user or group name. The use of fully qualified names prevents conflicts between two domains containing the same names. For example, if the system were configured to use two different LDAP directories, perhaps as a result of a corporate merger, and both directories contained accounts named "alice", SSSD could be configured to require one user to log in as "alice@COMPANY1" and the other user to log in as "alice@COMPANY2".

Documentation and Configuration

SSSD is documented in several man pages. The most important of which are `sssd(8)`, `sssd.conf(5)`, `sssd-ldap(5)`, `sssd-krb5(5)`, and `pam_sss(8)`.

Configuration of SSSD is done by editing the file `/etc/sssd/sssd.conf`. Like Samba, the file format is INI-style. In addition, to enable user management, NSS and PAM must be configured to use SSSD-aware libraries. For example:

[R7] *The following applies to RHEL7 only:*

File: `/etc/nsswitch.conf`

```
→ passwd:      files sss
→ shadow:      files sss
→ group:       files sss
```

File: `/etc/pam.d/system-auth`

```
+ auth        sufficient  pam_sss.so use_first_pass
+ account     sufficient  pam_sss.so
+ password    sufficient  pam_sss.so use_authok
+ session     optional    pam_sss.so
```

[S12] The following applies to SLES12 only:

File: /etc/nsswitch.conf

```
-> passwd: compat sss  
-> group: files sss
```

```
# grep pam_sss.so *-pc  
common-account-pc:account required pam_sss.so use_first_pass  
common-auth-pc:auth required pam_sss.so use_first_pass  
common-password-pc:password required pam_sss.so use_authok  
common-session-pc:session required pam_sss.so
```

Enumerating Accounts

Traditionally, it has been possible to list all users on a Linux system by running the command **gentent passwd**. In order to list all accounts in a domain, the option `enumerate = true` must be added to the domain's definition. For example:

File: /etc/sssd/sssd.conf

```
[domain/default]  
id_provider = ldap  
auth_provider = ldap  
ldap_uri = ldap://server1.example.com/  
... snip ...  
+ enumerate = true
```

Disabling SSSD

SSSD does not support unencrypted LDAP-based authentication. If unencrypted LDAP-based authentication is required, `nss_ldap` and `pam_ldap` must be used instead. The easiest way to disable SSSD is:

File: /etc/sysconfig/authconfig

```
-| FORCELEGACY=no  
+| FORCELEGACY=yes
```

```
# authconfig --updateall
```

Enabling SSSD

On SLES12, SSSD is not used by default, but can be enabled using the Yast2 `ldap-client` module. This requires that the `sssd` and `sssd-tools` packages are installed.

Troubleshooting SSSD

When troubleshooting with SSSD enabled, in addition to examining the standard log files, the logs under `/var/log/sssd/` should also be checked. To enable more verbose logging, change the `debug_level` in `/etc/sssd/sssd.conf`.

Enabling `sssd` and `nscd` at the same time is not recommended.

SSSD is implemented by multiple cooperating processes. For example, `sssd_nss` and `sssd_pam`. If these processes are missing, they may need to be enabled or restarted.

File: /etc/sssd/sssd.conf

```
[sssd]  
+ services = nss, pam
```

```
# systemctl restart sssd
```

Lab 8

Estimated Time:
S12: 35 minutes
R7: 35 minutes

Task 1: User and Group Administration

Page: 8-21 Time: 10 minutes

Requirements:  (1 station)

Task 2: Using LDAP for Centralized User Accounts

Page: 8-24 Time: 15 minutes

Requirements:  (1 station)  (classroom server)

Task 3: Troubleshooting Practice: Account Management

Page: 8-30 Time: 10 minutes

Requirements:  (1 station)

Objectives

- Learn to customize /etc/skel/.
- Learn to add new users and manage password aging.

Requirements

- (1 station)

Relevance

Controlling the environment created for new user accounts and setting things such as password aging parameters are a central part of most user/group security policies. This lab shows the tools and methods used to control these parameters.

- 1) To prepare to modify /etc/skel/ so that new users will have a more comfortable working environment upon log in, identify what files are there already.

```
$ ls -al /etc/skel/  
. . . output omitted . . .
```

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 3) Test the system's use of the /etc/skel/ contents when creating new accounts:

```
# useradd -m guru2  
# ls -al /home/guru2/  
. . . output omitted . . .  
# userdel -r guru2
```

Lab 8

Task 1

User and Group Administration

Estimated Time: 10 minutes

- Which files exist here?

- 4) [R7] *This step should only be performed on RHEL7.*

Users commonly create HTML pages by storing them in \$HOME/public_html/ and then accessing them via <http://localhost/~user/>. To make this easier for users, create a public_html/ directory in /etc/skel/:

```
# mkdir /etc/skel/public_html/
```

- 5) Sometimes storing files in the system /tmp/ directory is not desirable from a privacy standpoint. Instead users create a tmp/ directory inside their own home directory. Make this automatic for new users by creating a tmp/ directory inside of /etc/skel/:

```
# mkdir /etc/skel/tmp/
```

- 6) Add an account named guru2 and set a password for the account to work:

```
# useradd -m guru2  
# passwd guru2
```

[R7] Changing password for user guru2.

New Password: **work**

Bad Password: too short

Reenter New Password: **work**

[R7] passwd: all authentication tokens updated successfully.

[S12] passwd: password updated successfully

```
# ls -al ~guru2
```

. . . output omitted . . .

• The prompts you see may differ from those shown here.

• What files are in guru2's home directory now?

- 7) Set up password aging for the guru2 account to require passwords to be changed every ninety (90) days and configure the account password so that it must be changed at the first login:

```
# chage -M 90 -d 1 guru2
```

• Note the use of the -d parameter to set the last-changed date to one day after the Epoch to ensure a password change on login. Check the chage man page for details if you do not follow this usage.

- 8) Login as the user guru2. This can be done at a terminal accessed via **Ctrl**+**Alt**+**F2**, via a GUI login, or by using ssh to connect to localhost. For example:

```
# ssh guru2@localhost
```

guru2@localhost's password: **work**

You must change your password now and login again!

Changing password for guru2.

[R7] (current) UNIX password: **ctrl**+**d**

[S12] (current) UNIX password: **ctrl**+**c**

• The prompts you see may differ from those shown here.

No matter which method was used to login as guru2, a password change will be required.

Cleanup

- 9) Delete the guru2 user:

```
# userdel -r guru2  
# rmdir /etc/skel/tmp/  
[R7] # rmdir /etc/skel/public_html/
```

- 10) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Configure LDAP authentication
- ❖ Configure AutoFS to mount home directories

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

As environments grow, having account data stored locally on each machine becomes unmanageable. Setting up LDAP to authenticate users can provide flexibility and security that NIS cannot. Additionally, providing user home directories on one (or several) highly available centralized servers increases maintainability and reduces costs.

Notices

- ❖ The classroom server, `server1.example.com`, is functioning as a secure LDAP server.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Make sure that the OpenLDAP client tools are installed:

```
[R7] # yum install -y openldap-clients nss-pam-ldapd  
[S12] # zypper install -y sssd nss_ldap
```

- 3) The LDAP server running on `server1.example.com` is configured to only allow encrypted connections. The LDAP client tools will only make encrypted connections to LDAP clients whose identity can be verified. Download the SSL/TLS certificate:

```
# cd /etc/openldap  
# wget http://server1.example.com/server.crt  
. . . output omitted . . .
```

Lab 8

Task 2

Using LDAP for Centralized User Accounts

Estimated Time: 15 minutes

- 4) Add the following line to the /etc/openldap/ldap.conf file to so that the LDAP libraries load the server's certificate:

File: /etc/openldap/ldap.conf
+ TLS_CACERT /etc/openldap/server.crt

- 5) Verify connectivity with the LDAP server on server1:

```
# ldapsearch -x -ZZ -H ldap://server1.example.com -b "dc=example,dc=com" "(uid=*)" dn
```

- 6) [R7] This step should only be performed on RHEL7.

To allow it to see the NSS change, and be able to map UIDs to usernames, enable LDAP-based user accounts and restart the NFSv4 ID mapper:

```
# authconfig --updateall --enableldap --enableldapauth --ldapserver=server1.example.com --ldapbasedn='dc=example,dc=com' --enableldaptls --ldaploadcacert=http://server1.example.com/server.crt  
# systemctl restart nfs-idmapd
```

• rpcidmapd service for rpc.idmapd.

- 7) [S12] This step should only be performed on SLES12.

Resize the terminal window so that the YaST auth-client module will fit properly.

```
# resize -s 31 80
```

Enable LDAP-based user accounts:

```
# yast auth-client
```

This will open the Authentication Client Configuration (sssd) module in text-mode. (To run in graphical mode, use the command yast2 auth-client.) How to use YaST: Navigate using **Tab** and Arrow Keys. Select with **Enter**. Keycombos **Alt** + the highlighted letter can be used to navigate to an option faster.

Select [Change Settings]

Enable Allow Domain User Logon

Enable Create Home Directory

Select [Join Dom] to configure a new SSSD domain.

Under Name, enter **example.com**

Highlight LDPA for both identity data and user authentication options.

Select [OK]

Deselect Treat user and group names as case sensitive.

Navigate to LDAP schema type and use arrow keys to select rfc2307 from the dropdown list.

Navigate to Base DN for LDAP search and enter: dc=example,dc=com

Navigate to URIs of LDAP servers and enter: ldap://server1.example.com

Select [OK]

Select [Extended Options]

Highlight ldap_id_use_start_tls and select [Add]. Type to enable.

Select [Extended Options]

Highlight ldap_tls_cacert and select [Add] and enter: /etc/openldap/server.crt

Highlight the option enumerate and select [Edit].

Type to enable.

Select [OK] to exit the domain editor.

Select [Cancel] to exit yast

The auth-client module makes several useful changes to sssd.conf, as well as updating the /etc/nsswitch.conf to be aware of sssd, disabling the Name Service Caching Daemon (nsqd) in favor of sssd, and updating the /etc/pam.d/common-* files to be aware of sssd.

- 8) [S12] This step should only be performed on SLES12.

Restart sssd so the above manual edits take effect:

```
# systemctl restart sssd
```

- 9) Verify that the LDAP user accounts are now available on your computer:

```
# getent passwd dsuser1
dsuser1:*:1001:1001::/export/home/dsuser1:/bin/bash
```

- 10) Try logging in as dsuser1:

```
# ssh dsuser1@localhost
```

```
dsuser1@localhost's password: password   
Could not chdir to home directory /export/home/dsuser1: No such file or directory  
$ id -un  
dsuser1  
$ pwd  
/  
$ exit
```

If unable to log in, re-check the LDAP configuration settings and try again.

- 11) Using AutoFS, it's possible to make the missing home directories available whenever users log in to the system. Create the file /etc/auto.home with the special shortcut syntax to mount home directories:

File: /etc/auto.home
+ * -rw,soft,intr server1.example.com:/export/home/&

- 12) Edit the /etc/auto.master file to reference the auto.home file you just created:

File: /etc/auto.master
+auto.master +/export/home /etc/auto.home

- 13) Load the new AutoFS configuration:

```
# systemctl restart autofs
```

- 14) Try logging in as dsuser1:

```
# ssh dsuser1@localhost  
dsuser1@localhost's password: password   
$ id -un  
dsuser1  
$ pwd  
/export/home/dsuser1  
$ mount
```

```
... output omitted ...  
$ exit
```

- Authenticated via LDAP with dsuser1's home directory mounted from server1 via autofs.

If you were unable to login, re-check your settings and try again.

Cleanup

- 15) [R7] This step should only be performed on RHEL7.

Run system-config-authentication in non-interactive mode to restore the system back to a standalone client:

```
# authconfig --updateall --disableldap
```

- 16) [R7] This step should only be performed on RHEL7.

Stop sssd:

```
# systemctl stop sssd
```

- 17) [S12] This step should only be performed on SLES12.

Run the YaST program as the root user to restore the system back to a standalone client:

```
# yast auth-client
```

The Authentication Client Configuration (sssd) dialog appears.

Highlight example.com under Domain Options.

Select [Leave Dom]

Type to confirm domain deletion.

Select [OK]. If any pop-ups appear, select [Continue] or [OK] to exit.

- 18) Edit the autofs master file, /etc/auto.master removing reference to the file you just removed for targets under /export/home:

File: /etc/auto.master
- /export/home /etc/auto.home

- 19) Remove the file /etc/auto.home and the LDAP server certificate:

```
# rm /etc/auto.home  
# rm /etc/openldap/server.crt
```

- 20) Reload the AutoFS configuration:

```
# systemctl restart autofs
```

- 21) Remove the TLS entry to the OpenLDAP utility configuration file:

```
# sed -i '/^TLS_CACERT /d' /etc/openldap/ldap.conf
```

- 22) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

❖ Practice troubleshooting user account issues.

Requirements

☒ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You use these scripts to break your system in controlled ways, and then you troubleshoot the problem and fix the system.

- 1) Use tsmenu to complete the account management troubleshooting scenario:

Troubleshooting Group	Scenario Category	Scenario Name
Group 4	Users / Groups	usergroup-01.sh

Lab 8

Task 3

Troubleshooting Practice: Account Management

Estimated Time: 10 minutes

Content

PAM Overview	2
PAM Module Types	3
PAM Order of Processing	4
PAM Control Statements	6
PAM Modules	7
pam_unix	9
pam_nologin.so	10
pam_limits.so	11
pam_wheel.so	12
pam_xauth.so	13
Lab Tasks	
1. Restricting superuser access to wheel group membership	15
2. Using pam_nologin to Restrict Logins	17
3. Setting Limits with the pam_limits Modules	21
4. Using pam_limits to Restrict Simultaneous Logins ..	25

Chapter

9

PLUGGABLE AUTHENTICATION MODULES (PAM)

PAM Overview

Pluggable Authentication Modules

- Vary authentication methods without recompiling clients
- Implement additional levels of authentication verification and login restrictions

PAM modules are implemented as libraries and stored in

/lib64/security/ **or** /lib/x86_64-linux-gnu/security/

- Some modules have configuration files in /etc/security/

Configuration for PAM clients in /etc/pam.d/

Documentation

- Man pages
- /usr/share/doc/

PAM Linux System Administrator's Guide

Pluggable Authentication Modules

On most Linux distributions, all programs requiring access to authentication information are compiled against the PAM library. PAM is an abstraction layer which allows applications to use authentication data while being agnostic about the details of how the authentication is implemented. This innovation greatly increases system flexibility, as it makes modification of authentication schemes much simpler for system administrators. It also provides a centralized location, /etc/pam.d/, where the administrator can modify configuration files to adjust system-wide authentication policies. Older implementations of PAM used a single configuration file, /etc/pam.conf, for all services. If /etc/pam.d/ exists, /etc/pam.conf is ignored.

Historically, modifications to Unix authentication methods (e.g. Kerberos, MD5 encryption for passwords instead of a salted DES algorithm, biometric data such as voice or palm prints) required that all applications which accessed user account or authentication data be recompiled to support the new authentication method. Changing authentication sources is now as simple as using a different PAM module.

PAM Documentation

Many powerful and useful configurations are possible with PAM, it is highly suggested to read the documentation for the PAM modules packaged with the PAM RPM to learn about the available features the various modules provide. Documentation includes man pages and READMEs on a per module basis, and the Linux PAM System

Administrator's Guide (SAG). In some cases, module documentation is limited, only obtainable from the module's source code.

[R7] *The following applies to RHEL7 only:*

On Red Hat Enterprise Linux, the PAM documentation (including the SAG and READMEs) can be found in the /usr/share/doc/pam-*/* directory.

[S12] *The following applies to SLES12 only:*

On SUSE Linux Enterprise Server, the PAM documentation can be found in the /usr/share/doc/packages/pam/ directory.

Checking if a Program Uses PAM

PAM is used pervasively on modern Linux systems. However, it can occasionally be unclear from a program's documentation whether it supports PAM at all, or has been compiled with PAM support enabled. For dynamically linked binaries, use the **ldd** command to check if the program is linked against the libpam library. For example, notice in the following output that the **sshd** program shows that it is linked against libpam:

```
# ldd /usr/sbin/sshd | grep libpam  
libpam.so.0 => /lib64/libpam.so.0 (0x00007f388f136000)
```

Keep in mind that seeing the library listed in the **ldd** output is not sufficient to establish that a program is currently using PAM. It may still require specific configuration options to enable PAM or to set things such as the PAM service name used by the program.

PAM Module Types

auth (**authentication**)

- verifies identity, grants group memberships

account (**authorization**)

- implements authorization/security policy

password

- updates user's authentication token

session

- sets up user environment

PAM Services (Module Types)

PAM can be used to provide a variety of services:

auth ⇒ Authentication modules used to verify the identity of users and grant group memberships. An example would be a PAM module that checks a provided username and password against the values stored in a database.

account ⇒ Authorization modules determine a user's access to a service based on things other than the user's identity. An example would be a PAM module that permits or denies access based on the time of day. Other uses include preventing user login when system load is too high, triggering password updates, and reporting an account has been disabled.

password ⇒ Modules that update a user's authentication credentials. An example would be a PAM module that checks to see if a new password meets certain criteria and then updates the stored password value.

session ⇒ Modules that run code at the start and end of a particular session. Often used to configure the user's environment in some way. An example would be a PAM module that sets resource limits when a user logs in.

PAM modules can provide functions to fill the requirements of one or more of the four module types. For example, the `pam_cracklib.so` and `pam_pwquality.so` modules can only be invoked as a password module. However, the `pam_unix.so` module can be called as any of the four module types (and provides different functionality depending on how it was called).

PAM Order of Processing

Top down with control statements allowing for early exit

include control option allows branching to another file

- substack does the same, but will only skip the modules in the substack

R7: Most PAM client files include system-auth

- password-auth identical to system-auth minus biometrics or other auth methods that only make sense when the user is physically at the box
- Network daemons include password-auth instead of system-auth

PAM Service Name

The PAM modules called for a particular PAM client application are determined by the configuration file in /etc/pam.d/ matching the PAM service_name used by the application. If no matching service_name is found then the other file is used.

Most services use the same service name as the name of the program; for example **login**, **sshd**, and **gdm**. Some programs use a different service name. The only way to know for sure is to check the program's documentation, or watch for clues in the logs.

[R7] *The following applies to RHEL7 only:*

On RHEL7, PAM logs to /var/log/secure.

[S12] *The following applies to SLES12 only:*

On SLES12, PAM logs to /var/log/messages.

Order of Processing

Once a matching PAM configuration file is selected, modules are loaded and run in a simple top down line order. Processing will continue until either all modules have been run, or a module's return code and corresponding control statement cause processing to be terminated early.

For each module type, the order of the rules in the appropriate configuration file is used. For example, the /etc/pam.d/SERVICE configuration file may contain the following:

File: /etc/pam.d/SERVICE

```
auth    required    pam_env.so
auth    sufficient  pam_unix.so nullok try_first_pass
auth    requisite   pam_succeed_if.so uid >= 1000,
        quiet_success
account required   pam_unix.so
password sufficient pam_unix.so sha512 shadow nullok,
                   try_first_pass use_authtok
session required   pam_limits.so
```

Many PAM client applications have common security needs and require the same (or nearly the same) PAM modules called. Several different approaches (both PAM version specific and distro specific) have been taken to address this need.

Centralized PAM Configuration

PAM provides the include statement, use to load another PAM configuration file, which provides additional rules for processed.

[R7] *The following applies to RHEL7 only:*

Red Hat centralizes common PAM configuration into the system-auth, password-auth, fingerprint-auth, and smartcard-auth files, and then calls these files via the include control directive. These files are configured using the **system-config-authentication** or **authconfig** commands. In older versions of Red Hat Enterprise Linux, this centralization was done via a custom PAM module called **pam_stack.so**.

[S12] *The following applies to SLES12 only:*

SUSE centralizes common PAM configuration into common-auth, common-account, common-password, and common-session, and then calls these files via the include control directive. These files are configured using the **pam-config** command.

PAM Control Statements

Statements determine how individual module's return values affect the final value returned to the client.

- **required**
success necessary; entire stack processed on fail
- **requisite**
success necessary; stack not processed further on fail
- **sufficient**
success will authenticate without processing stack further
- **optional**
include
 - include at this point in the stack all tests of *module type* from the file specified.

include, this configuration file could also contain requisite and/or sufficient control statements.

The following four control values can be used (descriptions taken from the PAM documentation):

required ⇒ Indicates that the success of the module is required for the module-type facility to succeed. Failure of this module will not be apparent to the user until all of the remaining modules (of the same module-type) have been executed.

requisite ⇒ Like required, however, in the case that such a module returns a failure, control is immediately returned to the application with the return value set by the first required or requisite module to fail.

sufficient ⇒ Success of this module is deemed 'sufficient' to satisfy the Linux-PAM library that this module-type has succeeded in its purpose. In the event that no previous required module has failed, no more 'stacked' modules of this type are invoked. A failure of this module is not deemed as fatal to satisfying the application that this module-type has succeeded and processing of the stack continues.

optional ⇒ As its name suggests, this control statement marks the module as not being critical to the success or failure of the user's application for service. In general, Linux-PAM ignores such a module when determining if the module stack will succeed or fail.

PAM Control Statements

PAM works by passing whatever authentication is provided by the user attempting to access the system through a series of modules specified in the control file for the program being used. When PAM reads the control file, it builds a stack of modules to be processed for each of the four module types. As each stack of modules is processed, PAM control statements are consulted to determine the effect that module's return value will have on the overall summary value that is returned to the calling PAM client application. For example, the configuration file /etc/pam.d/sshd may look like:

File: /etc/pam.d/sshd		
auth	include	system-auth
account	required	pam_nologin.so
account	include	system-auth
password	include	system-auth
session	optional	pam_keyinit.so force revoke
session	include	system-auth
session	required	pam_loginuid.so

This configuration dictates that all attempts to login via SSH will require that the user's password first successfully traverse all PAM module checks within another pam config called system-auth. Attempts to login will also check pam_nologin (which will be discussed in detail later on) and then pass through system-auth to make sure that the user attempting to login has a valid, active account on the system. In addition to required, optional, and

PAM Modules

Enforce Policy

Record Events

Control Access

Many Modules Available

- Pre-configured for distribution
- Shipped with distribution but not enabled
- From third parties

PAM Modules

PAM modules are available for system administrators to use in a variety of different ways. Several PAM modules are available which can be used to establish system policies (such as acceptable password strengths, or the number of unsuccessful login attempts which are allowed from a specific client). Other PAM modules are available just to log events or display information to users. Most PAM modules, however, are used to control access to the system or to services on the system.

Linux ships with many PAM modules. While some are pre-configured, others are unused but can be enabled by an administrator needing their functionality. In addition, it is relatively easy to find third party modules or write custom modules to meet a specific need.

PAM modules are located in the /lib64/security/ directory. Some modules have accompanying utilities, (e.g. `pam_timestamp_check`). In general, these utilities are not intended to be used interactively, but are used by modules when called from a control statement. The exact modules included vary from Linux distribution to distribution. This is a typical list of PAM modules:

```
# ls /lib64/security/
pam_access.so      pam_filter.so      pam_localuser.so    pam_rootok.so      pam_tty_audit.so
pam_cap.so         pam_fprintd.so    pam_loginuid.so    pam_securetty.so   pam_umask.so
pam_chroot.so     pam_ftp.so        pam_mail.so       pam_selinux_permit.so pam_unix_acct.so
pam_console.so    pam_gnome_keyring.so pam_mkhomedir.so  pam_selinux.so     pam_unix_auth.so
pam_cracklib.so   pam_group.so      pam_motd.so       pam_sepermit.so    pam_unix_passwd.so
pam_debug.so       pam_issue.so      pam_namespace.so  pam_shells.so     pam_unix_session.so
pam_deny.so        pam_keyinit.so    pam_nologin.so    pam_sss.so        pam_unix.so
pam_echo.so        pam_krb5.so       pam_oddjob_mkhomedir.so pam_stress.so    pam_userdb.so
pam_env.so         pam_krb5afs.so   pam_permit.so     pam_succeed_if.so pam_warn.so
pam_exec.so        pam_krb5.so      pam_postgresok.so pam_systemd.so   pam_wheel.so
pam_faildelay.so   pam_lastlog.so   pam_pwhistory.so pam_tally2.so    pam_xauth.so
pam_faillock.so    pam_limits.so    pam_pwquality.so  pam_time.so      pam_timestamp.so
pam_filter          pam_listfile.so  pam_rhosts.so
```

pam_unix

Provides standard Unix style authentication

- /etc/passwd, /etc/shadow
- NIS

Documentation:

- pam_unix(8)
- SLES: pam_unix2(8)

The pam_unix.so PAM Module

Most distributions use pam_unix to provide standard Unix authentication through the Name Service Switch (NSS), including /etc/passwd, /etc/shadow, and NIS.

pam_unix.so can manage all four PAM management types. pam_unix provides token based user authentication (e.g. password) through the auth management type, and is able to affect user credentials once authenticated, but before the session is made available to the user, (such as with a Kerberos ticket).

The pam_unix account type affects account management, such as password aging, or other features typically implemented through a shadow file.

The password type implementation affects changes to the authentication token. It uses the unix_chpwd(8) command as a helper. A **unix_passwd** command also exists to help with SELinux.

The session type implementation provides session logging. This can be used with the debug and audit options.

Options

Options that can be passed to pam_unix include:

use_first_pass ⇒ Use a previous rule's stacked password instead of prompting for one

try_first_pass ⇒ Use a previous rule's stacked password first, before prompting for one

use_authok ⇒ Use stacked password for a password change. If PAM_AUTHTOK has not been set previously, pam_unix will fail

not_set_pass ⇒ Do not remember passwords in the stack (i.e. in PAM_items)

shadow ⇒ Use a shadow file

bigcrypt/md5/sha256/sha512 ⇒ Useful as a mechanism for updating the old DES and MD5 crypt() scheme, these option define the encryption algorithm used to store changed passwords

nodelay ⇒ Do not delay authentication (i.e. prevent failures due to delay)

nis ⇒ Use NIS for setting passwords

remember= ⇒ Deprecated method for recording old passwords in /etc/security/opasswd. Requires a number value. Use pam_pwhistory instead.

rounds= ⇒ Number of sha256/512 password hash rounds

broken_shadow ⇒ When using account management, ignore shadow file read errors.

pam_nologin.so

Type

- account (authorization)

Use to block all non-root login attempts

- /etc/nologin

Preventing Logins with the pam_nologin.so Module

The pam_nologin.so module can be used to block all attempts by non-root users to access the system. Traditionally, Unix has supported usage of an /etc/nologin file; if this file exists, all non-root users accessing the system see the contents of the file, and are then denied access to the system. This behavior is implemented using the pam_nologin.so module. To implement nologin support, /etc/pam.d/login must contain the line.

The file /etc/nologin, if it exists, is automatically deleted on reboot.

File: /etc/pam.d/login

account	required	pam_nologin.so
---------	----------	----------------

pam_limits.so

Type

- session

Use to apply resource limits to interactive users or groups

- /etc/security/limits.conf and /etc/security/limits.d/

The pam_limits.so Module

The pam_limits.so module is used to establish system policies for user system resource usage limits. By default limits are taken from the /etc/security/limits.conf configuration file if it exists and then individual *.conf files from the /etc/security/limits.d/ directory are read. The effect of the individual files is the same as if all the files were concatenated together. The pam_limits.so module is enabled by default from the /etc/pam.d/ configuration files:

[R7] File: /etc/pam.d/system-auth

[S12] File: /etc/pam.d/common-session

+ session required pam_limits.so

This tells PAM to read its configuration and apply the specified limits to the user; limits are implemented per-login, not globally. Limits are not imposed on UID 0. The /etc/security/limits.conf or *.conf files uses the following syntax:

domain type item value

domain ⇒ user to be limited (potential values include user names, group names indicated using an @group syntax, or a * wild card to match all users)

type ⇒ type of limit to be imposed (limits can be hard, meaning a user cannot exceed them; soft, meaning a user can exceed them if they modify their settings; or -, meaning both a soft and a hard limit)

item ⇒ resource to be limited, as shown in the table.

value ⇒ value to which to limit that item.

An example of a pam_limits.so configuration file which limits the guru user to two logins, and 50 total processes:

File: /etc/security/limits.conf

#domain	type	item	value
guru	hard	nproc	50
guru	-	maxlogins	2

Some useful items which can be limited include the following:

nofile ⇒ maximum number of open files

cpu ⇒ maximum CPU time (specified in minutes)

nproc ⇒ maximum number of processes

maxlogins ⇒ maximum number of logins for this user

maxsyslogins ⇒ maximum number of logins on the system

priority ⇒ priority with which to run this user's processes

locks ⇒ max number of file locks

sigpending ⇒ max number of pending signals

nice ⇒ nice priority

rtprio ⇒ realtime priority

chroot ⇒ chroot a user into this directory

See man limits.conf(5) for more details.

pam_wheel.so

Type

- authentication

Use to control access to capability to su to root

- restrict to wheel group
- make wheel group su root password-less

The pam_wheel.so Module

The pam_wheel.so module can be used to set system policies regarding the ability to use the **su** command. By default, on Linux any one can use the **su** command to become root temporarily (assuming they know the root password, of course). On many other systems, however, users must belong to the wheel group to be able to **su** to root; non-wheel members, even if they know the root password, are denied **su** access to root. On other systems, anyone can **su** to root with the root password, and members of wheel can **su** to root without having to type the root password. If desired, these alternate system policies can be configured using PAM.

Potential configuration statements in /etc/pam.d/su include:

File: /etc/pam.d/su
auth required pam_wheel.so use_uid

which would require membership in the wheel group to be allowed to **su** to root, or the far less secure:

File: /etc/pam.d/su
auth sufficient pam_wheel.so trust use_uid

which would allow anyone to **su** to root, and would allow members of wheel to **su** without having to type a password.

pam_xauth.so

Type

- session

Use to transfer xauth cookies

- `~/.xauth/import`
- `~/.xauth/export`

Forwarding xauth Cookies

X uses cookies stored in `~/.Xauthority` to control access to a user's X session. In order to display a window or read input from the keyboard or mouse, a program must first supply this cookie.

Whenever a user wants to run a program as a different user, for example after an `su`, it is necessary to transfer the cookie. This can be done manually by the user using `xauth`. Alternatively, PAM-enabled applications can do it automatically.

Users wishing to more tightly control the forwarding of cookies can do so by creating a couple of files in their home directory: `~/.xauth/import` and `~/.xauth/export`. The import file contains a list of users from which that user accepts cookies. Alternatively, the export file controls which users their own cookie will be forwarded to. A file not existing is the same as allowing all users, except in the case of the root user. If `/root/.xauth/export` doesn't exist, than root will not forward X cookies to any user. Of course, normal Unix permissions still apply.

Many programs are configured on RHEL7/SLES12 to use the `pam_xauth.so` module during session creation.

Lab 9

Estimated Time:
S12: 25 minutes
R7: 25 minutes

Task 1: Restricting superuser access to wheel group membership

Page: 9-15 Time: 5 minutes

Requirements:  (1 station)

Task 2: Using pam_nologin to Restrict Logins

Page: 9-17 Time: 5 minutes

Requirements:  (1 station)

Task 3: Setting Limits with the pam_limits Modules

Page: 9-21 Time: 10 minutes

Requirements:  (1 station)

Task 4: Using pam_limits to Restrict Simultaneous Logins

Page: 9-25 Time: 5 minutes

Requirements:  (1 station)

Objectives

❖ Practice setting up wheel group behavior for su

Requirements

☒ (1 station)

Relevance

Restricting access to switch to root using the su command can help protect from unwanted users accessing the root account.

- 1) Try to su to root. After doing so, return back to the guru user.

```
$ su -  
Password: makeitso   
# id -un  
root  
# exit  
$ id -un  
guru
```

- 2) [R7] This step should only be performed on RHEL7.

In another terminal, as the root user, restrict su to allow only members of the wheel group to switch to root by modifying /etc/pam.d/su:

File: /etc/pam.d/su		
-	#auth required pam_wheel.so use_uid	
+	auth required pam_wheel.so use_uid	

- 3) [S12] This step should only be performed on SLES12.

In another terminal, as the root user, restrict su to allow only members of the wheel group to switch to root by modifying /etc/pam.d/su.

File: /etc/pam.d/su		
	auth include common-auth	
+	auth required pam_wheel.so use_uid	

Lab 9

Task 1

Restricting superuser access to wheel group membership

Estimated Time: 5 minutes

- 4) Switch back to the console logged in as guru. Try to switch to the root account from the guru account. What happens?

```
$ su  
Password: makeitso   
su: Permission denied
```

This fails because guru is not a member of the wheel group.

- 5) Switch back to the terminal running as the root user and add guru to the wheel group:

```
# usermod -aG wheel guru
```

- 6) Switch back to the guru terminal and try to switch to root. What happens now?

```
$ su  
Password: makeitso   
# id -un  
root
```

This succeeds because guru is now a member of the wheel group.

Cleanup

- 7) [R7] This step should only be performed on RHEL7.

Revert to the normal system by editing the /etc/pam.d/su file:

File: /etc/pam.d/su		
-	auth	required pam_wheel.so use_uid
+	#auth	required pam_wheel.so use_uid

- 8) [S12] This step should only be performed on SLES12.

Revert to the normal system by editing the /etc/pam.d/su file:

File: /etc/pam.d/su		
-	auth	required pam_wheel.so use_uid

Objectives

- Use pam_nologin to restrict logins with the /etc/nologin file.

Requirements

- (1 station)

Relevance

On occasion it is necessary to restrict all logins to a system. For example, if maintenance is being done that requires files to not be in use or that makes the /home directory unavailable. Understanding and being able to use the /etc/nologin functionality effectively allows you to perform this task.

- 1) Spend a few minutes reading the module documentation to review the functionality provided by the pam_nologin module:

```
# man pam_nologin  
... output omitted ...
```

[R7] *The following applies to RHEL7 only:*

Compare with the /usr/share/doc/pam-1*/txts/README.pam_nologin README file.

[S12] *The following applies to SLES12 only:*

Compare with the /usr/share/doc/packages/pam/modules/README.pam_nologin README file.

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 3) Add the user nologinlab:

```
# useradd -m nologinlab  
# passwd nologinlab  
New password: pass  
BAD PASSWORD: it is too short
```

Lab 9

Task 2

Using pam_nologin to Restrict Logins

Estimated Time: 5 minutes

```
BAD PASSWORD: is too simple
Retype new password: passEnter
passwd: password updated successfully
```

- 4) Create a /etc/nologin file to prevent users from logging into the system:

File: /etc/nologin
+ Sorry, this system is undergoing maintenance. Please try again later.

- 5) Test the operation of the pam_nologin module by attempting to login as nologinlab with a SSH connection:

```
# ssh nologinlab@localhost
nologinlab@localhost's password: pass Enter
. . . snip . . .
Sorry, this system is undergoing maintenance. Please try again later.
. . . snip . . .
```

- 6) [R7] This step should only be performed on RHEL7.

Disable the SSH server's use of pam_nologin by editing the /etc/pam.d/sshd file and commenting out the single related line:

File: /etc/pam.d/sshd
- account required pam_nologin.so
+ #account required pam_nologin.so

- 7) [S12] This step should only be performed on SLES12.

Disable the SSH server's use of pam_nologin by editing the /etc/pam.d/sshd file and commenting out the single related line:

File: /etc/pam.d/sshd
- auth requisite pam_nologin.so
+ #auth requisite pam_nologin.so
- account requisite pam_nologin.so
+ #account requisite pam_nologin.so

- 8) Test the operation of the SSH server by attempting to login as nologinlab using SSH:

```
# ssh nologinlab@localhost
nologinlab@localhost's password: pass 
. . . snip . . .
$ whoami
nologinlab
$ exit
```

Cleanup

- 9) Remove the nologinlab account:

```
# userdel -r nologinlab
. . . output omitted . . .
```

- 10) Delete the /etc/nologin so that future lab tasks are not impacted:

```
# rm -f /etc/nologin
```

- 11) [R7] This step should only be performed on RHEL7.

Re-enable the SSH server's use of pam_nologin by editing the /etc/pam.d/sshd file:

File: /etc/pam.d/sshd
- #account required pam_nologin.so
+ account required pam_nologin.so

- 12) [S12] This step should only be performed on SLES12.

Re-enable the SSH server's use of pam_nologin by editing the /etc/pam.d/sshd file:

File: /etc/pam.d/sshd
- #auth requisite pam_nologin.so
+ auth requisite pam_nologin.so
- #account requisite pam_nologin.so
+ account requisite pam_nologin.so

- 13)** Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Enable and set process limits using pam_limits

Requirements

- (1 station)

Relevance

Setting reasonable resource limits can prevent abnormal behavior from degrading system performance or reliability.

- Create a file named /tmp/bomb.sh that contains:

```
File: /tmp/bomb.sh
+#!/bin/sh
+sleep 30
+cat /dev/urandom > /dev/null
```

When run, this simple script will wait for 30 seconds, then start the cat command running (eating up some CPU).

- Change permissions on the file to make it executable:

```
$ chmod a+x /tmp/bomb.sh
```

- In step 5 you will be launching the bomb.sh script. Before doing so, **save** anything of importance because the system may become unresponsive or unstable.

- The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso 
```

Lab 9

Task 3

Setting Limits with the pam_limits Modules

Estimated Time: 10 minutes

- 5) To illustrate the importance of imposing process limits, create a test account then set off the bombs:

```
# useradd -m madbomber
# su - madbomber
$ for i in $(seq 150)
> do /tmp/bomb.sh &
> done
[1] 4856
[2] 4857
[3] 4858
. . . snip . . .
```

- 6) Thirty seconds after executing the above command, the sleep timers will expire, causing 150 of your bombs to go off at the same time. The system should exhibit a definite noticeable slowdown. If possible, use ps, top, etc. to examine the state of the system. If the system is responsive enough, kill off the bombs to restore sanity; otherwise, reboot:

```
$ ps
. . . output omitted . . .
$ top
. . . output omitted . . .
$ pkill 'bomb|cat'
[1]  Terminated          /tmp/bomb.sh
. . . snip . . .
$ exit
# id -un
root
```

• Use the **q** command to exit top.

- 7) As root, set a process number limit for the `madbomber` user by adding the following line to the bottom of the `/etc/security/limits.conf` file:

File: `/etc/security/limits.conf`

#ftp	hard	nproc	0
#@student	-	maxlogins	4
+ madbomber	hard	nproc	30
# End of file			

- 8) View the new limit, try to change it, then try setting off the bombs again. This time, the kernel does not allow all 150 bombs to be launched and the damage done is limited:

```
# su - madbomber
$ ulimit -a | grep "max user processes"
max user processes          (-u) 30
$ ulimit -u
30
$ ulimit -u unlimited
-bash: ulimit: max user processes: cannot modify limit: Operation not permitted
$ for i in $(seq 150)
> do /tmp/bomb.sh &
> done
[1] 4957
[2] 4958
[3] 4959
. . . snip . . .
/tmp/bomb.sh: fork: Resource temporarily unavailable
[S12] /tmp/bomb.sh: fork: retry: No child processes
[S12] . . . snip . . .
```

- Directly view the process limit without using grep.
- Try, and fail, to change the limit.
- With the limit in place, try launching the bomb again.
- If this floods the screen, be patient.

- 9) Examine the system state and then kill off the bombs:

```
$ ps
. . . output omitted . . .
$ top
. . . output omitted . . .
$ pkill 'bomb|cat'
[1]  Terminated                 /tmp/bomb.sh
. . . snip . . .
$ exit
# id -un
root
```

- It may take a minute for the bombs to take over system resources. Use the **q** command to exit top.

Cleanup

- 10) Remove the madbomber account:

```
# userdel -r madbomber  
. . . output omitted . . .
```

- 11) Reset the /etc/security/limits.conf file:

File: /etc/security/limits.conf

#ftp	hard	nproc	0
#@student	-	maxlogins	4
- madbomber	hard	nproc	30
# End of file			

- 12) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Use pam_limits to restrict members of the onelogin group to a single login

Requirements

- (1 station)

Relevance

Restricting users to a set amount of simultaneous logins is a form of resource control. Being able to implement such a feature is useful in implementing a security policy that defines the maximum number of simultaneous logins.

- 1) Spend a few minutes reading the documentation to review the functionality provided by the pam_limits.so module:

```
# man pam_limits
```

```
... output omitted ...
```

• Type **q** to quit.

[R7] *The following applies to RHEL7 only:*

Compare with the /usr/share/doc/pam-*/txts/README.pam_limits README file.

[S12] *The following applies to SLES12 only:*

Compare with the /usr/share/doc/packages/pam/modules/README.pam_limits README file.

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
```

Password: **makeitso**

- 3) Add the user limitslab and the group onelogin:

```
# useradd -m limitslab
```

```
# passwd limitslab
```

New password: **pass**

BAD PASSWORD: it is too short

BAD PASSWORD: is too simple

Lab 9

Task 4

Using pam_limits to Restrict Simultaneous Logins

Estimated Time: 5 minutes

```
Retype new password: pass [Enter]
# groupadd onelogin
# gpasswd -a limitslab onelogin
Adding user limitslab to group onelogin
```

Any additional users that the security policy dictates should only have one simultaneous login could be added to the group at this time.

- 4) Modify the pam_limits configuration file and add an entry so that users in the onelogin group can only have one simultaneous login.

```
File: /etc/security/limits.conf
```

#@student - maxlogins 4
+ @onelogin - maxlogins 1
End of file

- 5) Test the new limit by trying to login twice concurrently.

```
# ssh limitslab@localhost
limitslab@localhost's password: pass [Enter]
. . . output omitted . .
limitslab@stationX ~]$ ssh limitslab@localhost
. . . snip . .
Are you sure you want to continue connecting (yes/no)? yes
limitslab@localhost's password: pass [Enter]
. . . snip . .
Too many logins for 'limitslab'.
. . . snip . .
```

• The pam_limits restriction is in effect.

- 6) Logout as limitslab:

```
$ exit
# id -un
root
```

Cleanup

- 7) Remove the limitslab account:

```
# userdel -r limitslab
```

```
# groupdel onelogin
```

- 8) Reset the /etc/security/limits.conf file:

```
File: /etc/security/limits.conf
```

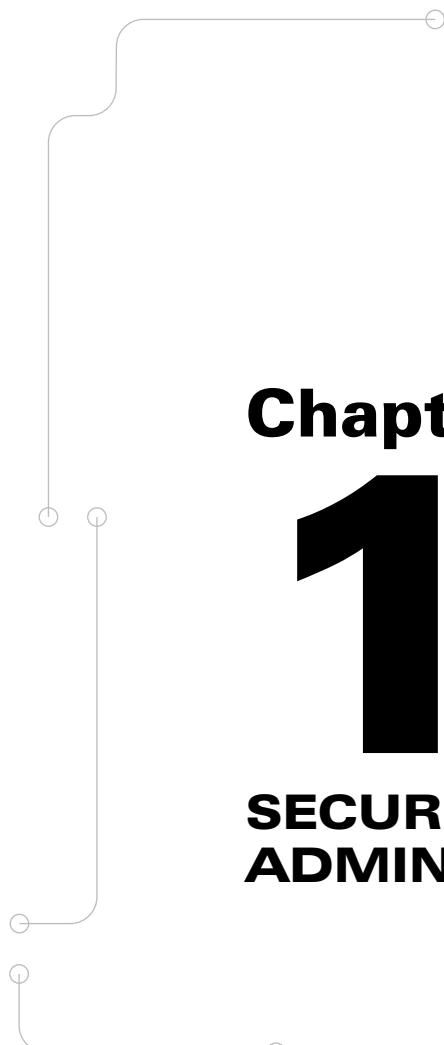
#@student - maxlogins 4
- @onelogin - maxlogins 1
End of file

- 9) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```


Content

Security Concepts	2
Tightening Default Security	4
SuSE Security Checker	6
Security Advisories	7
Fine Grained Authorizations with Polkit	8
File Access Control Lists	10
Manipulating FACLs	11
Viewing FACLs	12
Backing Up FACLs	13
File Creation Permissions with umask	14
User Private Group Scheme	16
Alternatives to UPG	18
AppArmor	19
SELinux Security Framework	20
SELinux Modes	22
SELinux Commands	24
Choosing an SELinux Policy	25
SELinux Booleans	27
Permissive Domains	28
SELinux Policy Tools	29
SUSE Basic Firewall Configuration	31
FirewallD	32
Lab Tasks	33
1. User Private Groups	34
2. Using Filesystem ACLs	39
3. Exploring AppArmor [S12]	48
4. Exploring SELinux Modes	53
5. SELinux File Contexts [R7]	59
6. SELinux Contexts in Action [R7]	61



Chapter

10

SECURITY ADMINISTRATION

Security Concepts

Principle of least privilege
Multiple layers of defense
Diversity of layers of defense
Only as secure as the weakest link
Simplicity is a virtue
Security through obscurity

Security Concepts

There are many tools useful for keeping a computer system secure. Which ones are used depends on how paranoid the administrator is, the type and value of data being protected, and similar variable factors. Regardless of the tools chosen, there are several security concepts which are commonly applied when configuring systems.

Principle of Least Privilege

In 1975, a paper entitled "The Protection of Information in Computer Systems" was published by Jerome Saltzer and Michael Schroeder. One of the many concepts discussed in this essay was the "principle of least privilege," which they described as "Every program and every user of the system should operate using the least set of privileges necessary to complete the job."

Amongst other things, this idea suggests that programs should not be run as root which do not need root privileges. Instead, programs should be granted the bare minimum of privileges they need to accomplish their task.

Multiple Layers of Defense & Diversity in Layers of Defense

Obviously, the more layers of defense that are created, the more chances there are of stopping an outside threat. Because of this, people typically use a so-called "belt and suspenders" approach to security in which multiple, redundant defenses are erected to ensure that there is no single point of failure. When creating layered defenses, different kinds of layers should be used whenever possible

to minimize overlap between common failures. A common application of this concept would be limiting relay access through an SMTP server to specific IP addresses in its configuration files, and limiting SMTP-based traffic using libwrap and a firewall.

Only as Secure as the Weakest Link

The basic security principle of "You're only as secure as your weakest link," means that an attacker will seek out the point of least resistance. They will look for the place where they can get in without being noticed. Therefore, it is important to identify the relative weakness of every link in the chain and try to improve the worst ones first.

Simplicity is a Virtue

Many administrators have a tendency to over-complicate their security arrangements. Usually, this is because of the false impression that a system can be so complicated as to be impossible for an outsider to navigate without being noticed and therefore will be more secure. In fact, the opposite is true. Complex systems actually afford an attacker a lot more places to hide. Also, a complex system (including systems with hundreds or thousands of apps) is more likely to have at least one app with known security vulnerabilities at any given time. A simple system (or one with as few apps installed as possible) is much easier to administer, monitor and audit. There are also far fewer links that could be weak.

Security by Obscurity

A common mantra in security circles is that "security through obscurity is not security." However, this truism is slightly misleading. While security based solely on obscurity (such as assuming a machine will not be cracked only because its IP address is not widely known) is ineffectual, obfuscation does play a valid role in security (such as not needlessly revealing the user names which have accounts on a machine).

Tightening Default Security

Disable all unneeded services

- standalone daemons
- socket activated daemons (including **xinetd**)

Disable unneeded accounts

Remove all unnecessary SUID/SGID executables

Restrict access to privileged accounts

Restrict remote access to the machine

Restrict physical access to the machine

Disabling Unneeded Services

No daemon can be remotely attacked which is not running. Because of this, a basic first step used to harden systems is to disable unneeded services. These can be configured easily using **systemctl** to enable or disable service startup at system boot. Make sure to stop currently running daemons as well. For example, these commands could be used together to ensure that Samba is not configured to start up automatically and is not currently running:

```
# systemctl disable smb  
# systemctl stop smb  
. . . output omitted . . .
```

In addition, remember that many services are started by an Internet super-daemon like **xinetd**. Most of these services will need to be disabled as well. **xinetd** is configured by the `/etc/xinetd.conf` file. The various files in the `/etc/xinetd.d/` directory should be edited to disable individual services (see `disable =` in the respective configuration file). Remember to restart or reload **xinetd** after modifying its configuration files:

```
# systemctl reload xinetd  
. . . output omitted . . .
```

Disable Unneeded Accounts

Though modern Linux distributions create system accounts for each service as part of the installation of the service's package(s), there are a few system accounts that are created at install time, whether they

are "needed" or not. In some cases, the removal of packages do not delete accounts that were added during installation.

Remove All Unnecessary SUID/SGID Executables

Executables with either the SUID, or SGID bits (or both), set can be a security risk. Remember, the SUID/SGID bits allow anyone (with execute permission) to launch the program and have it run as the user/group that owns the program file.

It is especially important to pay attention to SUID root executables. In some cases, there are ways to reconfigure a program (perhaps changing permissions and ownership on certain files and directories) such that it no longer needs the SUID bit set. If this is possible, do it. If not, evaluate whether or not the program in question is needed.

Restrict Access to Privileged Accounts

Restrict access to the root account. This includes controlling access to accounts that have access to the root account. For example, if the PAM configuration for **su** is altered so that only members of the wheel group can use it, then those user accounts which are members of the wheel group can be better controlled. This could include implementing extra requirements for passwords for those users beyond what is done for others.

Restrict Remote Access to Machines

A very common attack is to try logging into the root account via **ssh** remotely, trying common and random passwords. This attack is more

effective when attackers get large numbers of other machines to try thousands of possibilities against millions of IP addresses. The easy defense is to disallow root access via **ssh**. This can be done by setting PermitRootLogin no within the /etc/ssh/sshd_config file.

Be certain to consider any user accounts who can login over the network. One important change may be to require SSH secure key access only, disabling remote password access to the system. This can be done by changing PasswordAuthentication to no.

Restrict Physical Access to Machines

It's easy to get caught up in all the great security work needed to protect systems and networks. It's also just as easy to completely forget about/neglect the physical security of the systems being secured. Remember, if an attacker can gain physical access to the box, then all that wonderfully configured network security is all for naught; the game will end and the attacker will win. For example, with physical access to a target box, the attacker could just reboot and cause the bootloader to pass init=/bin/bash to the kernel, or boot to a CD or USB thumb drive, or take the hard drive, or the whole box, with them.

Don't fall into the trap of implementing a good security framework on the network and hosts, then forget about the physical security of the machines.

SuSE Security Checker

Also known as seccheck

Shell scripts that perform routine checks

- daily
- weekly
- monthly

Report email to the root user

- Can be configured by editing /etc/sysconfig/seccheck

The seccheck Scripts

seccheck is a collection of shell scripts that check different aspects of system security. The scripts are run by **cron** on a daily, weekly, and monthly basis. The scripts don't change any settings or do any reconfiguration, but only produce a report. Exact times of execution can be changed in /etc/cron.d/seccheck. The seccheck scripts are located in /usr/lib/secchk/. Each script (**daily**, **weekly**, **monthly**) checks for different security settings and system state. For example, the **daily** script checks:

- ❖ Sanity checks on /etc/passwd, /etc/shadow and /etc/group
- ❖ Checks for accounts with uid/gid of 0 or 1 besides root and bin
- ❖ \$PATH and **umask** check for the root account
- ❖ Verify that system accounts are listed in /etc/ftpusers
- ❖ Check /etc/aliases for addresses that deliver to a program
- ❖ Check .rhosts files for + signs
- ❖ Verify home directory ownership and permissions
- ❖ Check various dot files in home directories to verify they are not writable by others
- ❖ Verify user mailboxes have proper permissions
- ❖ Check for safe use of NFS exports and mounts
- ❖ Inspect network cards to see if any are in promiscuous mode
- ❖ List loaded kernel modules and open network ports

The **weekly** script checks:

- ❖ Runs John the Ripper on the /etc/passwd and /etc/shadow files to look for weak passwords

- ❖ Uses the RPM validate feature to check for any mis-matched MD5 signatures
- ❖ Lists all SUID and SGID files
- ❖ Lists all executables which are group/world writable
- ❖ Lists all files which are world writable
- ❖ Lists all devices

When the **daily** and **weekly** checks are run, the reports that are generated only show changes from the previous run.

The **monthly** check performs all the same checks as the daily and weekly, however, it returns full results, not just changes from the previous run.

More documentation can be found in the file /usr/share/doc/packages/seccheck/README file.

The results of the checks are sent by email to the root user. This can be changed in the file /etc/sysconfig/seccheck.

Security Advisories

Errata

- Red Hat Network <https://rhn.redhat.com/errata>
- SUSE Customer Center <http://scc.suse.com>

Mailing Lists

- US-CERT <https://www.us-cert.gov/mailing-lists-and-feeds>
- CERT <http://www.kb.cert.org/>
- SANS <http://www.sans.org/>
- BUGTRAQ <http://www.securityfocus.com/archive/1>
- Red Hat's enterprise-watch mailing list
- SUSE Security Announcements

Red Hat's enterprise-watch Mailing List

For administrators of Red Hat Enterprise Linux boxes, it is highly recommended that they subscribe to the low-volume enterprise-watch mailing list,
<http://www.redhat.com/mailman/listinfo/enterprise-watch-list/>. This list will send them emails regarding all Red Hat security updates.

SUSE's sle-security-announce Mailing List

For administrators of SUSE Linux Enterprise Server and Desktop, it is highly recommended to monitor the SUSE Security Announcements at <http://www.suse.com/support/security/>.

General Security Related Mailing Lists

It is difficult at best to thwart attempts to exploit vulnerabilities in software without being informed of those vulnerabilities. Several Internet mailing lists are available to which administrators can subscribe in order to receive notification of security vulnerabilities found in software applications they use.

In addition, the (in)famous BUGTRAQ mailing list is well worth reading. It has a wider scope than enterprise-watch and covers network security advisories and holes in all computer products. It is a full-disclosure mailing list, meaning that security holes are often announced even if no fix is yet available for them, and that exploit code is often posted as proof that a security hole exists. Although some vendors argue that full disclosure allows "bad guys" access to security holes, the simple truth is that the "bad guys" have typically found security holes long before they are announced or patched by the vendor; full disclosure serves as an early warning to the "good guys" that particular software should be avoided (if possible) or more closely monitored, until it is fixed.

In addition to mailing lists, several organizations publish lists of common security problems of which to be cognizant. Of these, the most prominent are CERT and SANS.

Fine Grained Authorizations with Polkit

Polkit provides Authorization API for application developers

- Enables Centralized Authorization Policy for privileged actions
Actions can be fine grained, no longer all or nothing
- Stock policy files with local override in /etc/polkit-1/
/var/lib/polkit-1/localauthority/
/usr/share/polkit-1/actions|rules.d/

Authorization rules can be written via two methods

- INI-like .pkla localauthority files
- JavaScript .rules files in polkit v0.106+
Provides flexibility with conditional logic
- Can distinguish console versus remote login session

Formerly known as PolicyKit

A GUI or CLI agent can prompt for passwords.

The **polkitd** daemon is started by the polkit systemd service. It monitors the directories and configuration files and automatically reloads them for any changes, additions, or deletions.

polkit Stock Policy and Actions

Applications that use polkit ship XML policy files which are installed into /usr/share/polkit-1/actions/ that contain a list of actions for that application as well the default authorization rules. The defaults be overridden using files in /etc/polkit-1/. The following example gets a list installed actions, and then further inspects a specific action:

```
# pkaction
org.freedesktop.hostname1.set-hostname
org.freedesktop.hostname1.set-machine-info
. . . snip . . .
# pkaction -v -a org.libvirt.unix.manage
org.libvirt.unix.manage:
description:      Manage local virtualized systems
message:          System policy prevents management of VMs
vendor:
vendor_url:
icon:
implicit any:    auth_admin_keep
implicit inactive: auth_admin_keep
implicit active:   auth_admin_keep
```

polkit Architecture

Linux inherits the UNIX security model with the all powerful root user that can do anything and regular unprivileged users who can't perform any privileged action. The **sudo** command is the traditional approach to allowing selected users or groups to run defined commands with root privileges.

The polkit authorization manager has been adopted in Linux to allow more a fine grained authorization model so that specific actions within a command or application can have authorization checks applied to that action. Not every application uses polkit, just the ones where the developers have implemented the polkit API.

Using polkit, a specific action can have one of several default authorization rules which can have different values depending on the state of user's login session. Available states are:

allow_any ⇒ Applies to any client

allow_inactive ⇒ Applies to inactive local sessions

allow_active ⇒ Applies to active local sessions

Each session state uses one of the following authorization values:

no ⇒ Denied

yes ⇒ Allowed

auth_self ⇒ Allowed after prompting for user's own password

auth_admin ⇒ Allowed after prompting for root or defined admin's password

auth_(self|admin)_keep ⇒ Keep authorization for 5 mins

Customizing Who is Considered an Admin

With polkit version v0.106 and higher, the recommendation is to use the new expressive JavaScript syntax for rules. By using JavaScript the full power of conditional processing can be used to match whatever security policy that can be imagined.

When the auth_admin/auth_admin_keep authorization value is used, it will prompt for root's unless the addAdminRule method is modified. For example, consider this rule:

File: /etc/polkit-1/rules.d/50-default.rules

```
polkit.addAdminRule(function(action, subject) {
    return ["unix-group:wheel"];
});
```

With this rule in place, members of the wheel group are treated as admins for polkit. This means if the user is a member of the wheel group, when a auth_admin/auth_admin_keep authorization triggers, the user will be prompted for their own password. If they aren't a member they will be prompted for the root password instead. To treat members of the wheel group and the kkelson user as admins use this code:

File: /etc/polkit-1/rules.d/49-default.rules

```
polkit.addAdminRule(function(action, subject) {
    return ["unix-group:wheel", "unix-user:kkelson"];
});
```

The rules are processed in a "first match" wins, so lowered numbered files override higher numbered files.

Custom polkit .rules Files

In most cases, you write the code for the `addRule` method of the `polkit` object to implement your authorization policy. Examples:

```
/* virtadmins group manage libvirt with no password prompt */
polkit.addRule(function(action, subject) {
    if (action.id == "org.libvirt.unix.manage" &&
        subject.isInGroup("virtadmins")) {
        return polkit.Result.YES;
    }
});
```

Allow user akelson to be able to mount filesystems using the **udisksctl** CLI or **gnome-disks** commands without authenticating.

```
polkit.addRule(function(action, subject) {
    if (action.id == "org.freedesktop.udisks2.filesystem-mount" ||
        && subject.user == "akelson") {
        return polkit.Result.YES;
    }
});
```

Prohibit reboot, shutdown, suspend and hibernate for all users

```
polkit.addRule(function(action, subject) {
    if (action.id == "org.freedesktop.login1.suspend" ||
        action.id == "org.freedesktop.login1.reboot" ||
        action.id == "org.freedesktop.login1.power-off" ||
        action.id == "org.freedesktop.login1.hibernate")
    {
        return polkit.Result.NO;
    }
});
```

Legacy Custom .pkla Files

Prior to the new JavaScript based rules files, rules were written in an INI-like syntax with .pkla files. This legacy format is still supported; the .pkla files should be installed into the /etc/polkit-1/localauthority/50-local.d/ directory. An example:

```
File: /etc/polkit-1/localauthority/50-local.d/50-custom.pkla
[Allow libvirt management permissions]
Identity=unix-user:dkelson;unix-user:kkelson
Action=org.libvirt.unix.manage
ResultAny=yes
ResultInactive=yes
ResultActive=yes
```

File Access Control Lists

FACLs

- Assign different permissions to additional users and additional groups on a given file or directory
- FACLs are implemented using xattr (extended Attributes)

Masks

- Effective permissions are derived from the FACL permissions for the user or group after applying the mask
- This allows an administrator to temporarily deny access without actually changing the FACLs themselves
- Affects long listing of group mode

`chmod` command changes FACL mask instead of group mode

File Access Control Lists

Every file on a Unix or Linux system has one owner and one group for which specific permissions (read, write, and execute) are assigned. If multiple, but not all, users need access to the same file, they must be members of the group with permissions for that file.

This arrangement, while simple to understand and administer, is not very flexible. What if there is more than one group of users who each need to have different permissions for the same file? With standard Unix file security, multiple copies of the same file would need to be created and kept synchronized. This is an administrative burden and too complex to be realistic.

What is needed to provide this flexibility is the ability to assign permissions for multiple users, or groups (or both) to a single file. File Access Control Lists, or FACLs, are lists of additional users (and groups), and their respective permissions, attached to a single file. If FACLs are present, they are matched in the same order as regular Unix permissions (i.e. user, group, then other; see ACCESS CHECK ALGORITHM of `acl(5)` for details.)

Filesystem Prerequisites

Historically, With the ext4 filesystem, the `acl` mount option had to be used. On the latest Linux distributions the ext4 and XFS filesystem has ACL support unconditionally enabled. The `acl` mount option, if used explicitly or in the default mount options field in the super block, is superfluous.

FACL Masks

Once a FACL is set on a file or directory, the group mode shown with `ls -l` reflects the current state of the FACL mask, i.e. the group mode listing is re-purposed to display the FACL mask. A change to the group mode with `chmod` (e.g. `chmod g=r file`) changes the mask, not the group mode.

If a FACL mask is set to `rwx`, then the read, write, and execute mode on the file will be honored. If the FACL mask is set to `r-x`, then write access will not be granted to additional users or groups, even if set in the FACL. The mask never affects the standard Unix user or group owner, or other, file modes. For normal day-to-day operation, there is no need for the FACL mask to be set to anything other than `rwx`.

The most common use for some other mask setting is when there is a need to block everyone temporarily from a number of files and directories. With one `setfacl` command, every file on the system can have a FACL mask of `---`, denying all access for every additional user and group. Once the problems are fixed, another single `setfacl` command gives everyone access again.

Manipulating FACLs

Creating and Modifying FACLs

- `setfacl -m u:bcroft:r-x /opt/plan9/bin/acme`

Removing FACLs

- `setfacl -x u:sinuhe /usr/bin/vim`
- `setfacl -b /srv/ftp/`

Directories and default FACLs

- `setfacl -m d:g:webdev:rw /srv/www/`

Creating and Modifying FACLs

Linux must have acl support built into the kernel module for each filesystem which should support access control lists. RHEL7/SLES12 support FACLs when using ext2/ext3, and additional configuration can provide support for enabling the xfs, jfs, and other filesystems as well as provide FACL support for each. FACLs can only be used on filesystems which are mounted with the acl mount option. Place this option in /etc/fstab for filesystems on which you want to use FACLs.

The **setfacl** command is used to create a FACL on a given file.

```
# setfacl -m u:charlotte:rw /depts/recv/ship-log.txt  
# setfacl -m u:kkelson:r /depts/recv/ship-log.txt
```

The **-m** option tells **setfacl** to modify the FACLs on the file(s) specified. Following the modify option is the *facl_spec* for the new entry. This could be a change to an existing FACL or create an entirely new one. The **-M** option causes **setfacl** to read one or more FACLs from a file or from STDIN, instead of from the command line.

The first part of the *facl_spec* specifies what type of FACL entry this is. Valid values are `[u[ser]:]uid[:perms]`, `g[roup]:gid[:perms]`, `m[ask][:][:perms]` or `o[ther][:]`. For the *uid* or *gid* parameters, **setfacl** will accept either a name or number value.

The last part of the *facl_spec* is the permissions to store for the FACL entry. This is a combination of letters representing the permissions: *r* for read, *w* for write, *x* for execute, and *X* for execute,

"only if the file is a directory or already has execute permission for some [other] user." This part of the *facl_spec* can also be left blank to indicate that no permissions should be granted for this FACL entry.

Default FACLs on Directories

The **setfacl** command can also set a `default` FACL on a directory. When such default FACLs exist, any file or directory created within that directory will have those default FACLs assigned automatically.

Default FACLs can be created using **setfacl** like this:

```
# setfacl -m default:g:devteam:rw /depts/dev/
```

The `/depts/dev/` directory now has a default group FACL giving members of the devteam group read and write permissions. This FACL will be copied to all files and directories created within the `/depts/dev/` directory.

Removing FACLs

To remove a FACL, use the **setfacl** command with the **-x** option, like this:

```
# setfacl -x u:charlotte /depts/payroll/payrates.txt
```

This command removes the FACL for the user charlotte from the `payrates.txt` file. The **-X** option causes **setfacl** to read the FACL from a file or from STDIN, instead of from the command line.

Use the **-b** option with **setfacl** to completely remove all FACLs from a given file. The Unix permissions will not be affected.

Viewing FACLS

Recognizing files which have FACLS

- `ls -l`

Listing FACLS

- `getfacl`

File Listings

When using FACLS on filesystems, it is helpful to know when a file has a FACL attached to it. The `ls -l` command will produce output such as this when there is a FACL present:

```
# ls -l
total 1
-rw-rw---+ root root 3738 Dec 29 17:37 ship-log.txt
-rw----- root root 251 Nov 3 9:01 command-codes.txt
```

Note the plus sign attached to the end of the permissions of the first file. This indicates that there are FACLS attached to that file. Files without FACLS attached have no plus sign.

Viewing FACLS

The `getfacl` command displays all FACLS for one or more files.

```
# getfacl /depts/recv/ship-log.txt
# file: ship-log.txt
# owner: root
# group: root
user::rw-
user:charlotte:r-
user:kkelson:r--
group::r--
mask::rwx
other::---
```

Notice the three different user: lines. The first lists the standard Unix permissions for the owner of the file. The other two are FACLS for additional users.

If `getfacl` is used on a file without a FACL set, the output will be similar to the example, but missing certain lines. The `getfacl` command only shows the standard UNIX permissions of the file in such cases. There will also be no mask:: line

Backing Up FACLS

Ideally the backup software natively supports FACLS

Manually creating FACL backups

- **getfacl**
 - R
 - L
 - absolute-names

Restoring FACL backups

- **setfacl**
 - restore

Restoring FACLS and Masks

Backing Up FACLS

Most backup software does not yet have the capability to backup FACLS along with other file metadata.

Using **getfacl**, FACLS can be backed up. This is done by creating a file listing the FACLS of files that are going to be backed up. Do this by redirecting the output of **getfacl** to a file, such as:

```
# getfacl -R * > FACLS.txt
```

The **FACLS.txt** file contains the FACLS of all files in the current directory, including sub-directories. This file can then be backed up like any other file. It should be backed up along with the files whose FACLS it contains.

The **-L** option may also be useful. This option causes **getfacl** to follow symbolic links and include them in its processing.

Although it may be tempting to use the **--absolute-names** option, which tells **getfacl** to not strip off the leading / of absolute pathnames, usually you should not. This is because most archiving formats do not record absolute paths. If this switch is used, then the files must be restored to the exact same path in order to be able to use the file **getfacl** produced with **setfacl** to restore the FACLS to the files.

Restoring FACL Backups

Restore or extract the file that contains the FACLS along with the files they are for.

With the file **FACLS.txt** which was created above, the **setfacl** command can be used to set the FACLS specified on their respective files.

```
# setfacl --restore=FACLS.txt
```

The **FACLS.txt** file can be deleted after **setfacl** is done with it.

FACL Masks During FACL Backup/Restore Operations

When you restore FACL backups using **setfacl** as described above, it is important to remember that the masks will be reset along with the other FACLS on each file. If you have (temporarily) blocked access to files using masks, this point must be kept in mind when doing such restores.

It is even more important to think about what masks you have set when creating your backups, as that value is what will be set during FACL restores.

File Creation Permissions with umask

Default permissions for newly created filesystem objects

- files: 666
- directories: 777

umask

- defines what permissions to withhold from the default permissions
- used to display or change your umask
- usually set in the user or system shell dot files
- used to provide the user private group (UPG) scheme

Controlling Initial File and Directory Permissions

When new files and directories are created in Linux, default permissions are initially set. These permissions are calculated by taking the default permissions of the files/directories created and subtracting the umask value from it. The umask is a four digit octal number that represents the value of permissions that will be masked out. In other words, permissions specified in the umask represent the permissions that will be automatically withheld when you create a new file.

Files and directories have different default permissions when they are created. The default permissions applied to files is 0666. For directories, the default permissions are 0777. The following example illustrates the process of how initial file permissions are calculated:

File	Directory
r w - r w - 666 Default mode	rwxrwxrwx 777 Default mode
----w-rwx 027 Umask value	----w-rwx 027 Umask value
r w - r ----- 640 Initial mode	rwxr-x--- 750 Initial mode

Viewing and Setting the umask Value

The **umask** command is the utility that is provided to view or change the current umask. The umask comes preset in configuration files and to view the current umask issue the command without any options.

The umask may be changed at any time by typing **umask** followed by the new desired value. Notice that the leading digit is not required if it is zero, (and is zero by default):

```
$ umask 022; umask  
0022
```

Security Implications

The root account has a default umask of 022. All files created by the root user have default permissions of 644 (rw-r--r--), allowing only read access to anyone other than root.

Note that a default umask of 002 gives away write permission to all group members. In the User Private Group (UPG) scheme, the default group is a private group with the same group name as the username. The result is that newly created files are only writable by that user, readable by everyone. This includes files downloaded from the Internet: it's impossible to download an executable to a Linux system and accidentally run it.

An even more secure umask configuration would be 007, restricting access only to the file owner (i.e. 660 for files). Remember that not all Unix systems use UPG, or maintain the same default **umask**. Care should be taken when running commands such as **scp -rp foo stationY**: to make sure that the resulting group ownership and its permissions reflect the same local access. It is important to avoid the common inclination of users to grant 777 permissions.

[R7] *The following applies to RHEL7 only:*

The default umask for an unprivileged user in Red Hat Enterprise Linux is 002 (defined in /etc/profile). The users' default primary group is their private group. This means all files will have permissions of 664 at creation time: read and write for user and group, and read for others.

[S12] *The following applies to SLES12 only:*

In SUSE Linux Enterprise Server, the default umask for all users is set to 022 (defined by pam_umask(8)). SUSE Linux Enterprise Server makes all users' default group the users group. When creating files, write access will only be granted to the user who created the file and not to anyone in the users group.

User Private Group Scheme

UPG provides a convenient way to share files when working in a group project directory

UPG scheme implemented by:

1. placing each user in their own private group
2. setting the umask to 0002
3. setting the group ownership of the project directory to a commonly shared GID
4. setting the project directory SGID

Enabling UPG on SUSE Linux Enterprise Server

- set file-creation mask to 002
- create a wrapper shell script that creates/uses private groups

used by default. If you want to disable it and use the traditional Unix approach then ensure that the **useradd** command is invoked with the **-N** option. This is a specific option that suppresses the creation and use of the private group.

[R7] *The following applies to RHEL7 only:*

The /etc/profile script tests if the user is using a User Private Group scheme. If not the /etc/profile script will set the file-creation mask to 022. The /etc/bashrc performs the same task for non-login shells.

Configuring UPG on SUSE Linux Enterprise Server

SUSE has chosen to follow the traditional Unix approach and place all users in the same default group. Setting up the system to use the UPG scheme requires diligence of the system administrator. For instance, the first user added to the system (UID 1000) will be assigned to the users group (GID 100) by the YaST installer. The **useradd** command will still follow the default behavior after the changes below are made, which means all members of the users group will have write access to a user's files if the **UPGuseradd** command below is not used.

Set the default file-creation mask to 002 by editing the file /etc/profile and changing the following line:

[R7] *The following applies to RHEL7 only:*

On Red Hat Enterprise Linux the User Private Group Scheme (UPG) is

File: /etc/profile

```
- umask 022
+ if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
+   umask 002
+ else
+   umask 022
+ fi
```

Create a wrapper script (with 700 permissions) which handles the task of creating a new group for each new user, and overriding the file /etc/default/useradd. An example of such a script is shown below.

File: /usr/local/bin/UPGuseradd

```
+#!/bin/bash
+ # Create group based on last parameter (e.g. username)
+ GROUP="${!#}"
+ groupadd "$GROUP"
+ useradd -m -g "$GROUP" "$@"
```

Alternatives to UPG

Goal: Enable Easy Collaboration like UPG

FACL directory defaults

Network filesystems

- NFS all squash
- Samba: forcing permissions and ownership

FACLS

Instead of setting the SGID permission on a directory, setting a default FACL on a directory will ensure that all files and subdirectories created within the directory are automatically assigned permissions for the specified user or group. For example, the following would create a new directory accessible only to members of the red and blue groups (and of course root). New files and directories created within the directory would also be read/write to both groups (with subdirectories additionally inheriting the execute permission). Finally, users will not be able to delete files they do not own (although the FACL granting write would allow them to truncate the file's contents):

```
# mkdir -m 1700 data
# setfacl -m g:blue:rwx data
# setfacl -m g:red:rwx data
# setfacl -m default:g:blue:rwx data
# setfacl -m default:g:red:rwx data
```

Note that the value of the group and other portion of the umask has no effect on this scheme and can be set to any value.

NFS all_squash Share Option

Another alternative to UPG that provides a convenient collaborative working directory where users can easily work on common files is the use of the NFS all_squash share option. For example, consider the following NFS share definition which would allow all users on the station1 host to create files in the shared directory with the NFS server automatically changing the owner UID and GID to the specified

values:

File: /etc/exports

```
+ /data station1(rw,all_squash,anonuid=520,anongid=100)
```

The obvious disadvantage of this scheme is that the UID/GID of the true creator of files is lost as all files will be owned by the specified anon{uid,gid}. However, this does permit users to easily collaborate without being forced to manually change permissions or group ownership of the files, and without compromising the security of their files created within other directories.

Samba force group and force mode Share Options

Similar to the NFS all_squash option Samba allows for all files and directories created within a share to have a specified mode and owning GID automatically set (effectively providing the equivalent of an SGID directory). For example, all writes made to this share by the two allowed users will result in files with an owning group of daxfam:

File: /etc/samba/smb.conf

```
[kelson-backup]
path = /data/kelson-backup
writable = yes
write list = dkelson kkelson
force directory mode = 2770
force create mode = 0660
force group = daxfam
```

AppArmor

AppArmor Concepts

- LSM and Name vs Label based enforcement

AppArmor Profiles

- /etc/apparmor.d/

Bootup Script

- /etc/init.d/boot.apparmor

Commands

- genprof/autodep, logprof, complain, enforce, unconfined

AppArmor Concepts

AppArmor, previously SubDomain, is an LSM (Linux Security Modules) module, providing Mandatory Access Control using name based enforcement. Unlike SELinux, which provides a label based Role and Type Enforcement, AppArmor provides a simpler alternative. AppArmor was first introduced by Immunix, Inc. for its GNU/Linux distribution. It was later integrated with SUSE Linux Enterprise Server 10, after Novell acquired Immunix. In 2010, Apparmor was included as part of the 2.6.36 release of the Linux kernel.

Further information about AppArmor can be found at <http://www.suse.com/documentation/apparmor/>, <http://wiki.apparmor.net/index.php/FAQ>, and <http://en.opensuse.org/SDB:AppArmor>.

Profiles

AppArmor policy targets applications through profiles, enforcing strict file and resource access. Enforcement is governed by the existing Discretionary Access Control mechanisms (file ownership and read, write, and execute permissions) and POSIX capabilities.

Profiles can be configured through individual commands, such as **genprof** and **enforce**, or directly through configuration files found in the /etc/apparmor.d/ directory. After modifying a profile, restart AppArmor:

```
# service apparmor restart
```

In general, each profile contains lines referencing files to be governed, network resources to be accessed, or includes of other profiles for files shared with another program. Each line is white space delimited, first defining the file, or files (through globbing), to be governed, and then the rule to be applied:

```
File: /etc/apparmor.d/foo  
/usr/bin/foo px
```

This allows access to the /usr/bin/foo command. Rules include:

- a ⇒ Append, exclusive to w
- ix ⇒ When executed, inherent parent's environment
- k ⇒ File locking
- l ⇒ Follow symbolic links
- m ⇒ Allow mmap(2) PROT_EXEC
- px | Px ⇒ Allow program execution, based on a profile, scrubbing the environment (P preserves environment)
- r ⇒ Allow read access
- w ⇒ Allow write access

The YaST AppArmor module is available for graphical configuration of AppArmor, including profile configuration.

SELinux Security Framework

Allows administrator to specify security policy

Policy defines the "correct operation" of the system

- interaction of processes and files
- use of POSIX capabilities
- use of system resources (shared memory, etc.)
- network sockets
- interaction of processes (signals, pipes, IPC, etc.)

Uses labels called "Security Context"s

- *identity:role:type:security_level*

Occasional relabeling of the filesystem may be required

Security Administrator Specifies Policy

The SELinux extensions allow an administrator to define a security policy. This policy can be very simple providing only basic limitation, or it can be very detailed defining which of the many complex interactions of system components are to be allowed. One of the advantages of the SELinux security framework is its flexibility in allowing security administrators to create a policy that meets their TCB goals. The policy is then loaded and enabled so that the Linux kernel can enforce compliance with the policy.

[S12] *The following applies to SLES12 only:*

SUSE Linux Enterprise Server enables SELinux so SELinux may be used without having to replace parts of the system. However, SUSE may support adding SELinux policies on a case by case basis.

Policy Defines the "Correct Operation" of the System

When a security administrator creates a policy, he or she is essentially detailing the interactions that are expected for correct operation of the system and its services. For example, a program running under a specific security context should either be allowed to interact with a given file, or it should not. The policy defines the allowed level of interaction between the program and then the kernel enforces it. If the program later tries to interact with the file in some non-permitted way, or perhaps interact with some completely different file, then the kernel will deny the attempt.

Security Context

The security policy determines the permissible interactions between objects on the system. To determine if a specific interaction is permitted, the system compares the security context of the interacting objects with the security policy.

The security context (sometimes called the label) is a string and consists of the following components:

identity ⇒ Name of the 'owner' of the object. System objects have special identities. Unix user accounts can be mapped to specific identities. A default identity (*user_u*) exists for Unix accounts not explicitly mapped to an identity. Identity controls the available roles.

role ⇒ For processes, lists the domain of the process. For files, has a placeholder value (*object_r*).

type ⇒ Classifies the object as to its specific security needs; that is each object that has unique (with respect to the other objects) security needs will be assigned a unique type. Conversely, objects with common security needs can share the same type.

security_level ⇒ Compound value in the form *sX-sX:cY-cY* where *sX* is the sensitivity level (valid values from *s0-s15*) and *cY* is the category (*c0-c255*). Used by the MCS/MLS policies.

Relabeling Files

If files on the filesystem have the wrong (or no) security context label, then applications can fail. The most common reasons that security labels become incorrect is either from copying files, or running the system with SELinux disabled. You can relabel files or directories using the **setfiles**, **restorecon**, or **chcon** commands. You can relabel the entire filesystem using the **fixfiles relabel** command.

SELinux Modes

Enforcing and Permissive modes

- **getenforce**
- **setenforce**

Disabling SELinux

- /etc/selinux/config
- Boot Loader
 selinux=0
 enforcing=0

Enforcing vs Permissive

When loaded, SELinux can operate in two different modes: enforcing and permissive. In both modes, SELinux LSM kernel hooks are active. The difference lies in how the hooks affect running processes.

enforcing ⇒ actions contrary to policy are blocked and event is logged
permissive ⇒ actions contrary to policy are only logged

Note that even when SELinux is operating in permissive mode, actions may be blocked by a standard security check (such as regular Unix file permissions) or by another stacked LSM security module. In other words, just because SELinux policy permits an action, it is not guaranteed that the action will succeed.

Toggling SELinux Modes

Because changing the mode SELinux is operating in is a key component of managing and troubleshooting, SELinux commands exist for toggling between modes. The **setenforce** command is used to set the current running mode of SELinux, and the **getenforce** command displays the current mode:

```
# setenforce 0
# getenforce
Permissive
# setenforce 1
# getenforce
Enforcing
```

Changing the SELinux mode with **setenforce** is non-persistent and the default mode set in /etc/selinux/config will become the active mode at the time of the next reboot.

Disabling SELinux

While permissive mode has many benefits, such as logging security problems, and maintaining system file contexts, disabling SELinux entirely is an option. Disabling SELinux will stop all SELinux functionality inside the Linux kernel (avoiding both the overhead and protection). In most scenarios disabling SELinux is completely unnecessary, and running in permissive mode is preferred.

[R7] *The following applies to RHEL7 only:*

To disable SELinux edit the file /etc/selinux/config and set SELINUX=disabled, then reboot the system. A system reboot is necessary to start the kernel without SELinux support. If you want to reactivate SELinux at a later date this can be done by editing the same file, changing the mode to permissive or enforcing, and rebooting the system.

[S12] *The following applies to SLES12 only:*

On SUSE Linux Enterprise Server, SELinux is disabled in favor of Apparmor. To enable SELinux, Apparmor must be disabled, and the SELinux Reference Policy from Tresys, downloaded from <http://software.opensuse.org>. This enables the minimum policy. Enabling targets will likely need customization. Help can be obtained through SUSE's Customer Support.

Re-Enabling SELinux Enforcing Mode

In addition to editing the `/etc/selinux/config` file, you must ensure that the file security labels are correct. Files created when SELinux was not running will have no security label, and other files may have incorrect labels. So, when changing the mode, particularly setting permissive or enforcing modes, you should strongly consider relabeling the entire filesystem. The recommended practice is to run `touch /.autorelabel` and then reboot.

SELinux Commands

sestatus

- display current SELinux settings
- **-v** displays contexts for files and processes listed in /etc/sestatus.conf

chcon – set the security context of a file or files

Many core commands have new options to support Security Context labels

New Commands Included with SELinux

Several new commands were created to provide an interface to the new SELinux functionality. These commands are provided in the policycoreutils package. Descriptions and example invocations of these new commands follow.

The sestatus Command

The **sestatus** command can be used to display the current state of an SELinux enabled system. If the **-v** option is used then the report will include the security contexts of all the files and processes listed in the /etc/sestatus.conf file. This allows an administrator to quickly verify the context of key objects. This can be helpful in spotting incorrect context on critical files that may commonly have their context clobbered due to improper handling.

```
# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
SELinux root directory:          /etc/selinux
Loaded policy name:              targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

The chcon Command

Security contexts on system objects can be changed with the **chcon** command. It has similar functionality and syntax to the **chmod** command, but changes contexts instead of traditional Unix permissions. The three options that are used for changing an object's context are **-u** for user, **-r** for role, and **-t** for type. Like many Unix commands, the **-R** option performs recursive file modification.

```
# ls -Z
-rw-r--r--. guru guru system_u:object_r:user_home_t file.txt
# chcon -t staff_home_t file.txt
# ls -Z
-rw-r--r--. guru guru system_u:object_r:staff_home_t file.txt
```

Supporting Security Context Labels

Many of the core commands that work with files have options to support security context labels. When possible, these commands use the **-Z** option (the meaning may vary based on the purpose of the command). Examples include: **login**, **su**, **id**, **ls**, **ps**, **cp**, **mv**, **stat**, and **find**. For example:

```
$ ps -ez | egrep (master|gpm)
system_u:system_r:postfix_master_t 4184 ? 00:00:00 master
system_u:system_r:gpm_t           4200 ? 00:00:00 gpm
```

Notably, newer versions of the **mv** command, use **-Z** which causes the file to inherit the security context label from the target directory instead of retaining the label, a common source of SELinux troubles.

Choosing an SELinux Policy

Targeted

- separate types for most commands and services
- everything else runs under the unconfined_t, kernel_t, or initrc_t domains

MLS (Multi-Level Security)

- implements sensitivity and category security labels
- primarily used in military and government

Minimum

- A modification of the targeted policy
- Almost everything runs unconfined_t
- All targeted modules are available if desired

Selected via /etc/selinux/config

example policy was updated and rewritten to create the reference policy. From the reference policy, Red Hat derived the variants targeted, strict, and mls. The strict policy was dropped, and much of the functionality it contained is now in the targeted policy. Since that time, the minimum policy was added as an option.

[S12] *The following applies to SLES12 only:*

SLES12 defaults to the Minimum policy. Custom configuration is necessary. The Tresys SELinux Reference Policy can be downloaded for this purpose, either directly or from the openSUSE Build Service.

Switching Policies

It is simple to switch between the targeted and MLS policies (or minimum):

1. Verify that the policy files for the desired policy exist. They are contained in the following RPMS: selinux-policy (configuration), selinux-policy-minimum, selinux-policy-targeted, and selinux-policy-mls.
2. Set the active policy in /etc/selinux/config to SELINUXTYPE=mls, SELINUXTYPE=minimum, or SELINUXTYPE=targeted.
3. Reboot the machine for the new policy to take effect.

SELinux Kernel Options

SELinux functionality can also be controlled by passing parameters to the kernel on boot. To disable SELinux a single time at boot, use the

SELinux Policies

Three different SELinux policies are included with the SELinux reference policy: targeted, MLS, and minimum.

targeted policy ⇒ Originally focused on the network services most likely to be the source of a security breach (e.g. Apache, BIND). With the targeted policy, any applications that do not have a policy defined will run under the unconfined_t, kernel_t, or initrc_t domain.

MLS policy ⇒ Multi Level Security provides a policy based around security levels and categories. The goal is to get LSPP, RBAC, and CAPP certification at EAL 4+. The security model provided by this policy is most commonly used in military or government deployments and not appropriate for typical corporate systems, with the possible exception of high-profile, sensitive servers.

Minimum Policy ⇒ The minimum policy was introduced in Fedora 10 as a variant of the targeted policy, preserving the unconfined_t target as the default: all services run as unconfined_t, kernel_t, or initrc_t, unless configured to be confined by the administrator.

Initial development versions of SELinux contained a single policy that was similar to the original strict policy. Due to conflicts generated by the nascent policy code, SELinux was disabled by default in distributions that first started shipping SELinux. In subsequent versions of Linux, SELinux was enabled and a targeted policy was implemented. The targeted was originally derived from the first policy developed for SELinux known as the example policy. Later, the

interactive GRUB interface to add one of two options to the kernel's arguments. To boot into permissive mode, add `enforcing=0`. To boot without loading SELinux at all, add `selinux=0`.

SELinux Booleans

Easy way of activating certain policy rules

Displaying available SELinux Booleans

- `semanage boolean -l`
- `sestatus -b`
- `system-config-selinux` (GUI)

Temporary Change

- `setsebool boolean_name value`

Persistent Change

- `setsebool -P boolean_name value`

Booleans

To make SELinux more flexible and easy-to-use, categories of policy have been added that can be turned on or off. These boolean values can be toggled in real time and immediately become active. Most are intuitively named such as `spamassassin_can_network`, `user_rw_usb`, and `named_write_master_zones`.

A complete list of possible boolean values for the current running policy can be found with the `sestatus -b`, `getsebool -a`, or the `semanage boolean -l` commands:

```
# sestatus -b
. . . snip . . .
Policy booleans:
httpd_can_network_connect_db          off
httpd_can_network_memcache             off
httpd_tty_comm                         off
. . . snip . . .
# semanage boolean -l
SELinux boolean                         State  Default Description
. . . snip . . .
httpd_can_network_connect_db           (off , off) Allow HTTPD scripts and modules to connect to databases over the network.
httpd_can_network_memcache            (off , off) Allow httpd to connect to memcache server
httpd_tty_comm                        (off , off) Allow entering of passphrase for SSL certificates at the terminal.
. . . snip . . .
```

Toggling Booleans

The most common tool for activating or disabling a boolean value is the `setsebool` command. Basic usage is as follows:

```
# setsebool [-P] boolean_name value
```

The value field may be *true* or *1* to enable the boolean; *false* or *0* to disable it. The `setsebool` command immediately makes changes active in the running policy.

Booleans set using the `-P` (persistent) option will be written to the policy file on disk and become permanent (i.e. survive a reboot). For example:

```
# setsebool -P samba_enable_home_dirs on
```

Permissive Domains

Fine grained policy control

Replaces disable_trans **booleans**
`semanage permissive -a httpd_t`
`semanage permissive -d httpd_t`
`semodule -l | grep permissive`

The `disable_trans` Booleans

Releases 4 and 5 of Red Hat Enterprise Linux and SUSE Linux Enterprise Server included booleans to selectively disable protection of specific services. Unfortunately, these booleans were an imperfect solution. Instead of completely disabling SELinux policy enforcement, changing the boolean would merely run the service in a different domain. As a result, files created by the service would be mislabeled, and related confined services would be unable to interact with the unconfined service.

For example, when protected by SELinux, Apache is launched on boot by an init script running in the `initrc_t` domain, but the process automatically transitions to the `httpd_t` domain. With `httpd_disable_trans` enabled, Apache would instead remain in the `initrc_t` domain and `httpd_t` specific policy rules would not protect it.

Permissive Domains

As with running SELinux in permissive mode, when a service's domain is made permissive, SELinux does not enforce policy for that domain. Log messages are still sent, easing the troubleshooting process. Files created by the service are labeled correctly. Other confined services are still able to interact with the permissive service. Compared to permissive mode, however, permissive domains are generally superior because the rest of the system is still protected by SELinux.

The `semanage` command can be used to control whether a domain is permissive or not. For example the following command would make vsftpd's `ftpd_t` domain permissive:

```
# semanage permissive -a ftpd_t
```

This can be verified by running the following command:

```
# semodule -l | grep permissive
permissive_ftpd_t 1.0
permissivedomains 20
```

To restore vsftpd to confined status:

```
# semanage permissive -d ftpd_t
```

The `permissivedomains.pp` module contains all the permissive domain declarations on the system. To disable all permissive domains, run the command:

```
# semodule -d permissivedomains
```

SELinux Policy Tools

semanage

- Non-interactive command line driven SELinux administration

system-config-selinux

- status, booleans, file-labeling, SELinux user creation and mapping, translations for MLS, network ports, and control of loaded policy modules

Documentation

- **man -k selinux** (almost 900 man pages available; many programmatically generated from policy source.)

The semanage Command

This command allows for comprehensive management of SELinux. In the following example, web content is stored in a non-standard directory location and the SELinux database is persistently modified to properly label the new directory:

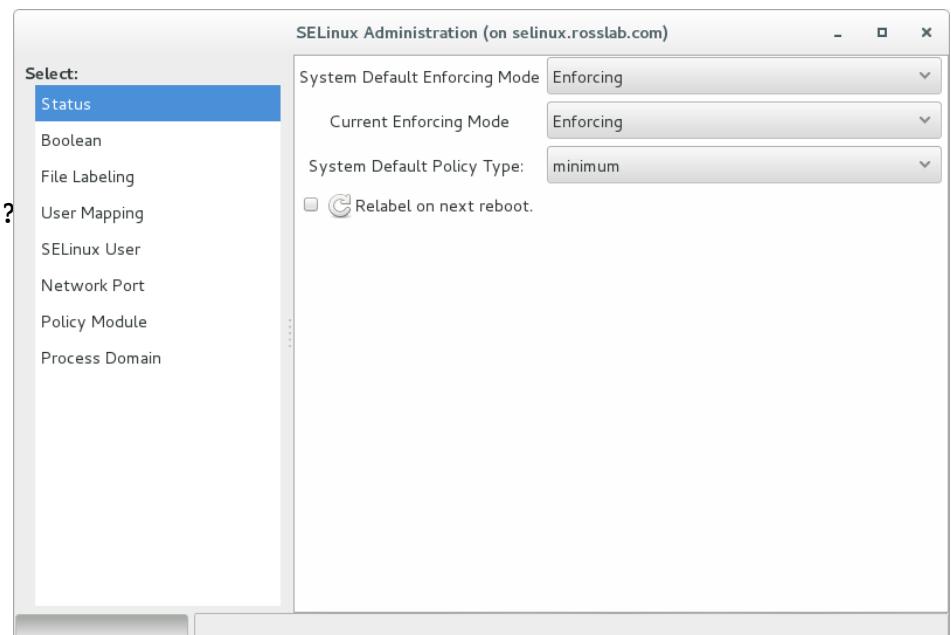
```
# mkdir -p /srv/websites/production/  
# semanage fcontext -at httpd_sys_content_t "/srv/websites(/.*)?"  
# restorecon -R /srv/websites/
```

The system-config-selinux Command

This tool provides a unified graphical interface for most of the system administration tasks associated with SELinux:

- ❖ selection of SELinux mode
- ❖ force relabel of filesystem on next boot
- ❖ view and persistently modify SELinux booleans
- ❖ view and modify file context labeling expressions
- ❖ create SELinux users and map them to Linux user accounts and roles
- ❖ define translations for MLS security-levels/categories
- ❖ define network ports accessible by SELinux types
- ❖ add and remove policy modules

[R7] The following applies to RHEL7 only:



SELinux Administration (on selinux.rosslab.com)

Select: Boolean

Status
Boolean
File Labeling
User Mapping
SELinux User
Network Port
Policy Module
Process Domain

Revert Customized

Filter

Active	Module	Description
<input type="checkbox"/>	apache	Allow httpd daemon to change its resource limits
<input type="checkbox"/>	apache	Allow httpd to act as a FTP server by listening on the f
<input type="checkbox"/>	apache	Allow httpd to access openstack ports
<input type="checkbox"/>	apache	Allow HTTPD to run SSI executables in the same dom
<input type="checkbox"/>	apache	Allow HTTPD scripts and modules to server cobbler fil
<input type="checkbox"/>	apache	Allow Apache to modify public files used for public file
<input type="checkbox"/>	apache	Allow HTTPD scripts and modules to connect to cobbl
<input type="checkbox"/>	apache	Allow httpd to connect to memcache server
<input checked="" type="checkbox"/>	apache	Allow http daemon to check spam
<input checked="" type="checkbox"/>	apache	Allow httpd to use built in scripting (usually php)

SELinux Administration (on selinux.rosslab.com)

Select: File Labeling

Add Properties Delete Customized

Filter

File Specification	Selinux File Type	File Type
/	root_t:s0	directory
/*	default_t:s0	all files
/[^/]+	etc_runtime_t:s0	regular
/afs	mnt_t:s0	directory
/a?quota\.(user group)	quota_db_t:s0	regular
\.autofsck	etc_runtime_t:s0	regular
\.autorelabel	etc_runtime_t:s0	regular
/bin	bin_t:s0	all files
/bin/*	bin_t:s0	all files
/bin/bash	shell_exec_t:s0	regular

SUSE Basic Firewall Configuration

SLES12: During Install

- Enabled/Disabled
- Several services can be allowed

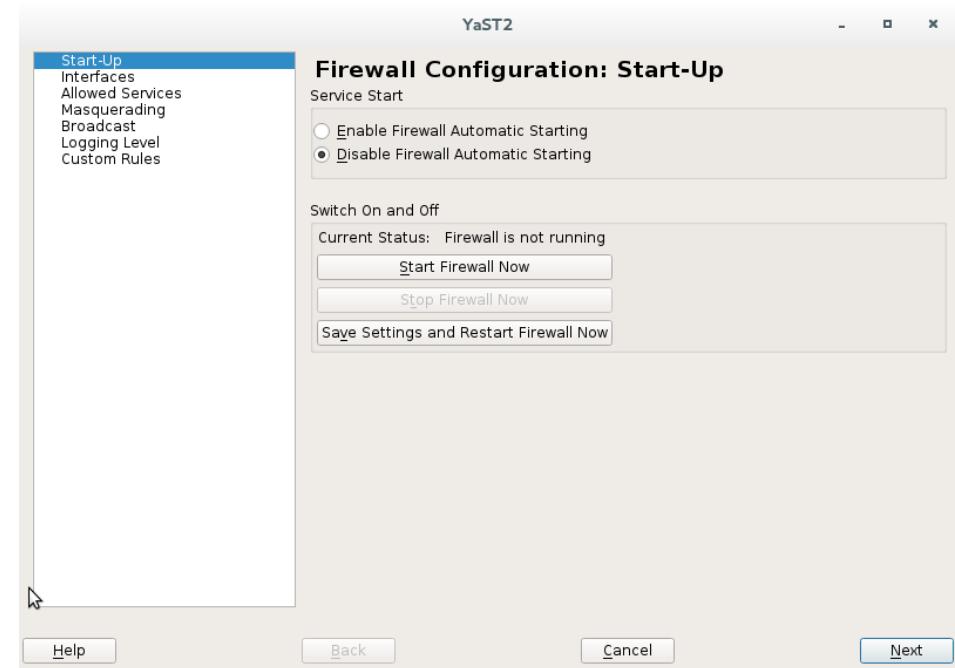
Builds a stateful packet-filter firewall using Netfilter

- <http://netfilter.kernelnotes.org/>

Firewall configuration tool

- YaST firewall module
- **SuSEfirewall2** control script
 - /etc/sysconfig/SuSEfirewall2
 - /etc/sysconfig/SuSEfirewall2.d/services/
- Network interfaces, IP networks mapped to three zones: external, internal, and DMZ.

Arguments to **SuSEfirewall2** include start, stop, and status. The default behavior is start. Run **SuSEfirewall2 --help** for more information.



SuSEfirewall2, Netfilter, & iptables

Compared to other firewalls that only support end-host configuration, the SUSE firewall is more sophisticated. It supports firewall configuration where multiple network interfaces are installed.

Firewalls can be created either with **SuSEfirewall2** (via the **yast firewall** module) or with **iptables** directly. The caveat to modifying firewall rules by hand with an **iptables** command is that the changes are likely to be lost if/when **SuSEfirewall2** is executed later.

By default no traffic is allowed inbound from the external network and any desired protocols and ports must be explicitly allowed. Each network interface on the firewall needs to be assigned to one of three zones:

External Network ⇒ Network interface(s) facing the Internet.

Internal Network ⇒ Network interface(s) facing internal hosts that may or may not be using RFC1918 IP addresses and require NAT.

DMZ Network ⇒ Network interface(s) containing hosts that can be reached by hosts on the external and internal networks, but cannot initiate connections into the internal networks.

Configuration changes made through YaST and **SuSEfirewall2** are stored in `/etc/sysconfig/SuSEfirewall2` and files under `/etc/sysconfig/SuSEfirewall2.d/services/`.

Examples of such a configuration could include defining ports that will be opened for particular services or Netfilter kernel modules required for that service.

FirewallID

Dynamically managed daemon

- GUI - `firewall-config`
- Command Line - `firewall-cmd`
- D-BUS interface for integration with other programs
GNOME Desktop, libvirt, etc

Zone based design

- trusted, external, and 7 pre-defined zones
- Interfaces, services, ports and protocols can be assigned to zones

It is possible to switch back to the traditional

/etc/sysconfig/iptables based firewall

FirewallID Features

FirewallID allows administrators and authorized programs to make changes to the running firewall config through D-BUS messages sent to the Python based `firewalld` daemon. Both graphical (`firewall-config`) and command line (`firewall-cmd`) interfaces are provided. FirewallID comes with many predefined services and zones to make it easier to create complex firewall configs more easily. It also supports a "direct" option and "rich language" rules that expose the full functionality of the underlying Netfilter system if required.

FirewallID uses the Netfilter connection tracking for stateful rules which pass traffic associated with established connections. Once established, these tracked connections will continue to be allowed even when rules are changed and/or the firewall rules reloaded. To drop established connections (by unloading the Netfilter `nf_conntrack_*` kernel modules) use:

```
# firewall-cmd --complete-reload
```

Pre-Defined Zones

FirewallID expects each network interface to be added to a zone, and this assignment will determine which firewall rules are applied to the interface. Pre-defined zones exist for common connection types and trust levels. XML files with descriptions of the expected use and specific firewall rules associated with each zone are found in `/usr/lib/firewalld/zones/*.xml`. Similarly, service definitions can be found in `/usr/lib/firewalld/services/*.xml`. Persistent configuration is stored under `/etc/firewalld/`.

firewall-cmd Syntax

The following examples show basic usage of the `firewall-cmd` utility to examine the current firewall state, determine interface and service to zone mappings, and permanently add a service to a particular zone. The Bash completion functionality (generally invoked with the `Tab` key) can be used to quickly type or explore the available options:

```
# firewall-cmd --state
running
# firewall-cmd --list-all-zones | grep -A1 active
external (active)
  interfaces: em1
--
trusted (active)
  interfaces: em2
# firewall-cmd --list-services --zone internal
dhcpcv6-client ipp-client mdns samba-client ssh
# firewall-cmd --list-services --zone external
ssh
# firewall-cmd --permanent --add-service=http --zone=external
success
# firewall-cmd --reload
success
```

The Procedure for Re-enabling the Traditional Firewall

Use `systemctl` to `mask` the `firewalld` service and `enable` the `iptables` and `ip6tables` services.

Lab 10

Estimated Time:
S12: 60 minutes
R7: 60 minutes

Task 1: User Private Groups

Page: 10-34 Time: 10 minutes

Requirements:  (1 station)  (classroom server)

Task 2: Using Filesystem ACLs

Page: 10-39 Time: 20 minutes

Requirements:  (1 station)

Task 3: Exploring AppArmor [S12]

Page: 10-48 Time: 20 minutes

Requirements:  (1 station)

Task 4: Exploring SELinux Modes

Page: 10-53 Time: 10 minutes

Requirements:  (1 station)

Task 5: SELinux File Contexts [R7]

Page: 10-59 Time: 5 minutes

Requirements:  (1 station)

Task 6: SELinux Contexts in Action [R7]

Page: 10-61 Time: 15 minutes

Requirements:  (1 station)

Objectives

- Configure a project directory to take advantage of the user private group scheme.

Requirements

- (1 station) (classroom server)

Relevance

User Private Group scheme is a very powerful security mechanism when used correctly. This lab will give basic skills to implement such a scheme.

- The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- Create two new groups and add them as secondary groups for the guru user:

```
# groupadd red  
# groupadd blue  
# usermod -aG red,blue guru
```

- Create a top level directory, /srv/projects/ to hold all project sub-directories. Then create two sub-directories for the red and blue project. Set the permissions on the new directories to be open to the group and to include the SGID bit:

```
# mkdir -p /srv/projects/{red,blue}  
# chmod -R 2770 /srv/projects/*
```

- Set the group ownership of the new directories and verify:

```
# cd /srv/projects/  
# chgrp blue blue/  
# chgrp red red/  
# ls -l  
total 8  
drwxrws--- 2 root blue 4096 May 26 20:54 blue  
drwxrws--- 2 root red 4096 May 26 20:54 red
```

Lab 10

Task 1

User Private Groups

Estimated Time: 10 minutes

If a . is present after the mode output, this signifies the presence of an SELinux security context.

- 5) Switch to the guru account and verify that it is a member of the new red and blue groups:

```
# su - guru  
$ id -Gn  
. . . output omitted . . .
```

The id command can also be used to view group memberships.

- 6) Check the current file-creation mask value to see if it is set to the collaboration friendly value of 0002:

```
$ umask  
[R7] 0002  
[S12] 0022
```

- 7) [S12] This step should only be performed on SLES12.

Since the file-creation mask was not set to 0002, change it:

```
$ umask 0002
```

This is a non-persistent change that will be lost when the shell exits.

- 8) Create some empty test files in each of the project directories and verify that the new files are owned by the project groups and writable to members of the groups:

```
$ cd /srv/projects/  
$ touch {red,blue}/testfile  
$ ls -lR *  
blue:  
total 0  
-rw-rw-r-- 1 guru blue 0 May 26 21:10 testfile  
  
red:  
total 0  
-rw-rw-r-- 1 guru red 0 May 26 21:10 testfile
```

A file-creation mask of 002 makes working in project and departmental directories straightforward and convenient.

9) [S12] *This step should only be performed on SLES12.*

Make the file-creation mask change permanent by editing the `~/.profile` file and adding the following to the end of the file:

File: /home/guru/.profile

+ umask 002

Of course, this only effects the guru user. The change could be done system-wide by modifying the `/etc/profile` file.

10) [S12] *This step should only be performed on SLES12.*

Create a file in the `/tmp/` directory to demonstrate the problems of the collaborative friendly file-creation mask value of 002 and the traditional shared common primary group users:

```
$ touch /tmp/upgtestfile  
$ ls -l /tmp/upgtestfile  
-rw-rw-r-- 1 guru users 0 Dec 26 20:14 /tmp/upgtestfile
```

While the file-creation mask value of 002 makes working in SGID project and departmental directories convenient, outside of those directories files created are writable by all members of the users group (nearly all users on the system).

- What problems exist with the permissions of the newly created file? How could these problems be overcome?

11) [S12] *This step should only be performed on SLES12.*

To fix this problem give the guru user its own private, primary group of the same name while still retaining the user management benefit of a shared common group. As the root user:

```
$ exit  
# id -Gn guru  
users red blue  
# groupadd guru  
# usermod -g guru guru  
# usermod -G users,dialout,video,red,blue guru
```

- Get a list of the current groups that user guru is a member of.
- Add a new group called guru.
- Make the guru user's primary group the guru group.
- Make the guru user a member of several secondary groups.

12) [S12] This step should only be performed on SLES12.

Become the guru user. Verify the group memberships and file-creation mask reflect changes from previous steps:

```
# su - guru
$ id -a
uid=1000(guru) gid=1002(guru) groups=1002(guru),16(dialout),33(video),100(users),1000(red),1001(blue)
$ umask
0002
```

13) [S12] This step should only be performed on SLES12.

Create another file in the /tmp/ directory to see how the UPG scheme provides secure file permission even with a file-creation mask value of 0002:

```
$ touch /tmp/upgtest2
$ ls -l /tmp/upgtest2
-rw-rw-r-- 1 guru guru 0 Dec 26 20:16 /tmp/upgtest2
```

Since the file's group is guru, the group writable permission is still safe.

Cleanup

14) [S12] This step should only be performed on SLES12.

Revert the change you made earlier by editing the ~/.profile file.

File: /home/guru/.profile
- —umask 0002

15) [S12] This step should only be performed on SLES12.

Exit the guru user's shell and revert group membership changes made earlier to the guru user.

```
$ exit
# usermod -G users guru
# groupdel guru
```

16) Delete project groups and directories created earlier:

```
[R7] $ exit  
# groupdel red  
# groupdel blue  
# rm -rf /srv/projects/
```

- Remember that as the root user the su command was used to switch to the guru user.

17) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Verify the filesystem is mounted with acl option
- ❖ Create, View, and Remove ACLs on files
- ❖ Create, View, and Remove Default ACLs on directories
- ❖ Backup and restore FACLs

Requirements

█ (1 station)

Relevance

The traditional UNIX file permission model has been in operation for decades and has stood the test of time. However, implementing certain security policies can be cumbersome. Knowing how to deploy and use POSIX file access control lists (facs) on Linux will enable simple solutions for those otherwise cumbersome security policy requirements.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) In order to use FACLs on a filesystem, the filesystem must be mounted with the acl option. This lab task will be performing FACL operations on the / filesystem. Determine which block device the root filesystem is being mounted from and what mount options were used to mount it:

```
# mount | grep '/'  
/dev/mapper/vg0-root on / type xfs (rw,relatime,attr2,inode64,noquota)
```

- Notice the options listed in parenthesis. The acl mount option is not listed, however it may still be present in the superblock. acl is enabled by default in the XFS filesystem.

Lab 10

Task 2

Using Filesystem ACLs

Estimated Time: 20 minutes

- 3) For this lab task, three groups need to be created that correspond to three missions: users, managers, and auditors as detailed in the following table. More groups could be created to organize managers and auditors as a traditional approach, however FACLs will be used instead. Use the `create_accts.sh` script to create the necessary users and groups automatically:

```
# /labfiles/create_accts.sh  
. . . output omitted . . .
```

	Mercury	Gemini	Apollo
Sub-Directory under <code>/srv/missiondata/</code>	mercury	gemini	apollo
Users	jglenn, vgrissom, lcooper	jlovell, ealdrin, cconrad	narmstng, mcollins, ashepard
Managers	kklein (read-only), wbland	wbland (read-only), glow	glow (read-only), jwebb
Auditors (read-only access)	marriott, carvalho	marriott, carvalho	marriott, carvalho

- 4) Now that all the groups and users are created, the directories that will be used to store data can be created. Create a `/srv/missiondata/` directory and several sub-directories and observe the default permissions:

```
# mkdir -p /srv/missiondata/{mercury,gemini,apollo}  
# cd /srv/missiondata/  
# ls -al  
total 40  
drwxr-xr-x  5 root root 4096 Apr 25 22:36 .  
drwxr-xr-x 26 root root 4096 Apr 25 22:36 ..  
drwxr-xr-x  2 root root 4096 Apr 25 22:36 apollo  
drwxr-xr-x  2 root root 4096 Apr 25 22:36 gemini  
drwxr-xr-x  2 root root 4096 Apr 25 22:36 mercury
```

- If a `.` is present after the mode output, this signifies the presence of an SELinux security context.

- 5) Before using FACLs, it's a good practice to accomplish as much as reasonably possible using the traditional file security model. First change the permissions so that only root and group members have access to their respective directories. The groups should have full read and write access. View the new permissions when done:

```
# chgrp mercury mercury/
# chgrp gemini gemini/
# chgrp apollo apollo/
# chmod 1770 *
# ls -al
total 40
drwxr-xr-x 5 root root 4096 Apr 25 22:36 .
drwxr-xr-x 26 root root 4096 Apr 25 22:36 ..
drwxrwx--T 2 root apollo 4096 Apr 25 22:36 apollo
drwxrwx--T 2 root gemini 4096 Apr 25 22:36 gemini
drwxrwx--T 2 root mercury 4096 Apr 25 22:36 mercury
```

- 6) Using setfacl, add a default directory FACL so that in each mission directory, any newly created files are automatically writable by the group members.

```
# pwd
/srv/missiondata
# for i in *
> do setfacl -m d:g:$i:rwx $i
> done
```

This could also be accomplished using the User Private Group scheme and the SGID bit turned on each directory.

- 7) With FACLs applied to the directories, the FACL indicator, +, is displayed in long directory listings. Run ls to see them:

```
# ls -l
total 24
drwxrwx--T+ 2 root apollo 4096 Apr 25 22:36 apollo
drwxrwx--T+ 2 root gemini 4096 Apr 25 22:36 gemini
drwxrwx--T+ 2 root mercury 4096 Apr 26 00:05 mercury
```

- 8) View one of the FACLs you just applied using getfacl:

```
# getfacl mercury/
# file: mercury
# owner: root
# group: mercury
user::rwx
group::rwx
other::---
default:user::rwx
default:group::rwx
default:group:mercury:rwx
default:mask::rwx
default:other::---
```

• This is the FACL that was added.

- 9) Add another default FACL to each mission directory so that the managers with full read/write access will have that permission on newly created files and sub-directories:

```
# setfacl -m u:wbland:rwx mercury/
# setfacl -m d:u:wbland:rwx mercury/
# setfacl -m u:glow:rwx gemini/
# setfacl -m d:u:glow:rwx gemini/
# setfacl -m u:jwebb:rwx apollo/
# setfacl -m d:u:jwebb:rwx apollo/
```

- 10) View one of the FACLs you just applied using getfacl:

```
# getfacl mercury/
# file: mercury
# owner: root
# group: mercury
user::rwx
user:wbland:rwx
group::rwx
mask::rwx
other::---
default:user::rwx
default:user:wbland:rwx
default:group::rwx
```

• This FACL was added.

• This FACL was added.

```
default:group:mercury:rwx
default:mask::rwx
default:other::---
```

- 11) Add another default FACL to each mission directory so that the managers with read-only access will have that permission on newly created files and sub-directories:

```
# setfacl -m u:kklein:r-x mercury/
# setfacl -m d:u:kklein:r-x mercury/
# setfacl -m u:wbland:r-x gemini/
# setfacl -m d:u:wbland:r-x gemini/
# setfacl -m u:glow:r-x apollo/
# setfacl -m d:u:glow:r-x apollo/
```

- 12) View one of the FACLs you just applied using getfacl:

```
# getfacl mercury/
# file: mercury
# owner: root
# group: mercury
user::rwx
user:kklein:r-x
user:wbland:rwx
group::rwx
mask::rwx
other::---
default:user::rwx
default:user:kklein:r-x
default:user:wbland:rwx
default:group::rwx
default:group:mercury:rwx
default:mask::rwx
default:other::---
```

- The two managers different permissions on the directory.

- 13) As the final default FACL on the mission directories, give the auditors the ability to enter the directories and list their contents without having write access:

```
# setfacl -m d:u:marriott:r-x *
```

```
# setfacl -m u:marriott:r-x *
# setfacl -m d:u:carvalho:r-x *
# setfacl -m u:carvalho:r-x *
```

- 14) View the FACLS for all three mission directories:

```
# getfacl *
# file: apollo
# owner: root
# group: apollo
user::rwx
user:glow:r-x
user:jwebb:rwx
user:marriott:r-x
user:carvalho:r-x
group::rwx
other::---
default:user::rwx
default:user:glow:r-x
default:user:jwebb:rwx
default:user:marriott:r-x
default:user:carvalho:r-x
default:group::rwx
default:group:apollo:rwx
default:mask::rwx
default:other:---
. . . snip . . .
```

- Note how the file or directory that this FACL block applies too is listed as a comment.

- 15) With the default FACLS on the mission directories, any files or sub-directories created inside will inherit the same FACLS. Create some data files in each of the directories:

```
# mkdir -p {mercury,gemini,apollo}/{training/{zero,high}G,engineering}
# touch {mercury,gemini,apollo}/payroll.dat
```

- 16) When a sub-directory is created inside of a directory containing default FACLS two things occur. First the FACLS get applied to the directory. Then second, the default FACLS are applied as well. The later is important for when files or directories get created inside of it.

Use getfacl to see the FACLs on one of the engineering/ sub-directories:

```
# getfacl mercury/engineering
# file: mercury/engineering
# owner: root
# group: root
user::rwx
user:klein:r-x
user:bland:rwx
user:marriott:r-x
user:carvalho:r-x
group::rwx
group:mercury:rwx
mask::rwx
other::---
default:user::rwx
default:user:klein:r-x
default:user:bland:rwx
default:user:marriott:r-x
default:user:carvalho:r-x
default:group::rwx
default:group:mercury:rwx
default:mask::rwx
default:other::---
```

- The FACLs are applied here at the top.
- The default FACLs get applied as well.

17) Inspect the FACLs on one of the payroll.dat files:

```
# getfacl mercury/payroll.dat
# file: mercury/payroll.dat
# owner: root
# group: root
user::rw-
user:klein:r-x          #effective:r--
user:bland:rwx           #effective:rw-
user:marriott:r-x        #effective:r--
user:carvalho:r-x        #effective:r--
group::rwx               #effective:rw-
group:mercury:rwx        #effective:rw-
mask::rw-
other::---
```

Remember that default FACLs are only applicable to directories.

- 18) Modify the FACLs on the payroll.dat file so that both managers for a mission have read/write access and that the users of the mission have no access at all:

```
# setfacl -m u:kklein:rwx mercury/payroll.dat
# setfacl -m u:wblanc:rwx gemini/payroll.dat
# setfacl -m u:glow:rwx apollo/payroll.dat
# setfacl -x g:mercury mercury/payroll.dat
# setfacl -x g:gemini gemini/payroll.dat
# setfacl -x g:apollo apollo/payroll.dat
```

- 19) Inspect the FACLs on one of the payroll.dat files:

```
# getfacl mercury/payroll.dat
# file: mercury/payroll.dat
# owner: root
# group: root
user::rw-
user:kklein:rwx
user:wblanc:rwx
user:marriott:r-x
user:carvalho:r-x
group::rwx
mask::rwx
other::---
```

Notice the mask changed to rwx so it is no longer impacting the individual FACLs, nor causing effective actual permissions.

- 20) Some backup programs do not save FACLs, in that case you must create a text file with the FACLs in them and backup that text. Create such a text file, then follow up by changing the permissions on a file (this will work for standard permissions or FACLs):

```
# getfacl -R * > FACLS.txt
# chmod 400 FACLS.txt
# chmod 777 apollo
# ls -ld apollo
drwxrwxrwx+ 4 root apollo 4096 Feb 11 15:08 apollo
```

- Set secure permissions otherwise users can determine what files exists inside the sub-directories.

- 21) Simulate a recovery of FACLs after restoration from a FACL unaware backup application:

```
# setfacl --restore=FACLs.txt  
# ls -ld apollo  
drwxrwx--T+ 4 root apollo 4096 Feb 11 15:08 apollo
```

- 22) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- >Create a basic AppArmor profile

Requirements

- (1 station)

Relevance

Creating and managing an AppArmor profile by hand is very reasonable to implement and can help increase security of a system.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Verify that the AppArmor related packages are installed:

```
# zypper install -t pattern apparmor  
... output omitted ...
```

- 3) Check the current status of AppArmor:

```
# aa-status  
apparmor module is loaded.  
XX profiles are loaded.  
XX profiles are in enforce mode.  
... snip ...  
X profiles are in complain mode.  
X processes have profiles defined.  
X processes are in enforce mode.  
... snip ...  
X processes are in complain mode.  
X processes are unconfined but have a profile defined.  
... snip ...
```

- 4) In order to avoid possible conflicts with existing policies create a hardlink of the cat command:

```
# ln /usr/bin/cat /bin/aademo
```

Lab 10

Task 3

Exploring AppArmor [S12]

Estimated Time: 20 minutes

Unlike SELinux, AppArmor works on the process name and not labels in the file inode.

- 5) Create a new file with the following content:

File: /tmp/ctf
+ This is the Blue Teams Flag

- 6) Verify that the cat command and aademo are able to read both a control file and the file /tmp/ctf:

```
# cat /etc/issue  
Welcome to SUSE Linux enterprise Server 12 (x86_64) - Kernel \r (\l).  
# cat /tmp/ctf  
This is the Blue Teams Flag  
# aademo /etc/issue  
Welcome to SUSE Linux enterprise Server 12 (x86_64) - kernel \r (\l).  
# aademo /tmp/ctf  
This is the Blue Teams Flag
```

- 7) Create a profile file for the /bin/aademo command:

File: /etc/apparmor.d/bin.aademo
+ /bin/aademo { + /lib/** rm, + /lib64/** rm, + /tmp/ctf r, + }

As a reminder the letters rm are a combination of two flags: r (Read) and m (Memory mapping).

- 8) Insert the new policy:

```
# apparmor_parser -a /etc/apparmor.d/bin.aademo
```

- 9) Check the current status of AppArmor for the new profile:

```
# aa-status | less +/bin/aademo
apparmor module is loaded.
XY profiles are loaded.
XY profiles are in enforce mode.
. . . snip . . .
/bin/aademo
. . . snip . . .
X profiles are in complain mode.
X processes have profiles defined.
X processes are in enforce mode.
X processes are in complain mode.
X processes are unconfined but have a profile defined.
. . . snip . . .
```

- 10) See if /bin/aademo can read the two files:

```
# aademo /etc/issue
aademo: /etc/issue: Permission denied
# aademo /tmp/ctf
This is the Blue Teams Flag
```

- 11) Modify the profile for /bin/aademo so that access to the /tmp/ctf file is explicitly denied:

File: /etc/apparmor.d/bin.aademo

```
/bin/aademo {
    /lib/** rm,
    /lib64/** rm,
-   /tmp/ctf r,
+   deny /tmp/ctf r,
}
```

- 12) Replace the existing policy:

```
# apparmor_parser -r /etc/apparmor.d/bin.aademo
. . . output omitted . . .
```

- 13) Verify that /bin/aademo is denied access and that cat is still read the two files:

```
# aademo /etc/issue  
aademo: /etc/issue: Permission denied  
# aademo /tmp/ctf  
aademo: /tmp/ctf: Permission denied  
# cat /etc/issue  
Welcome to SUSE Linux enterprise Server 12 (x86_64) - kernel \r (\l).  
# cat /tmp/ctf  
This is the Blue Teams Flag
```

- 14) To allow a process to run and log profile violations, but not stop them, a process may be placed into complain mode:

```
# aa-complain /bin/aademo  
Setting /bin/aademo to complain mode.
```

- 15) Check the current status of AppArmor for the profile:

```
# aa-status | less  
apparmor module is loaded.  
XY profiles are loaded.  
XX profiles are in enforce mode.  
. . . snip . . .  
Y profiles are in complain mode.  
    /bin/aademo  
X processes have profiles defined.  
X processes are in enforce mode.  
X processes are in complain mode.  
X processes are unconfined but have a profile defined.  
. . . snip . . .
```

- 16) Verify /bin/aademo can read the two files:

```
# aademo /etc/issue  
Welcome to SUSE Linux enterprise Server 12 (x86_64) - kernel \r (\l).  
# aademo /tmp/ctf  
This is the Blue Teams Flag
```

- 17) Disable the profile from being automatically parsed next time AppArmor is reloaded:

```
# ln -s /etc/apparmor.d/bin.aademo /etc/apparmor.d/disable/
```

- 18) Restart the AppArmor service:

```
# systemctl restart apparmor
```

- 19) Verify that the aademo profile is no longer listed in the output of aa-status:

```
# aa-status | grep aademo
```

Clean up

- 20) Remove the demo files:

```
# rm /bin/aademo  
# rm /tmp/ctf
```

- 21) Remove the profile and its symbolic link:

```
# rm /etc/apparmor.d/bin.aademo  
# rm /etc/apparmor.d/disable/bin.aademo
```

- 22) Restart the AppArmor service:

```
# systemctl restart apparmor
```

- 23) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

Toggle between SELinux modes

Requirements

1 station

Relevance

Since SELinux can operate in different modes, understanding how to toggle between them is a critical component of SELinux management.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Verify that the SELinux related packages are installed:

```
# rpm -qa | grep -E 'policy|selinux' | sort  
checkpolicy-2.1.12-6.el7.x86_64  
libselinux-2.2.2-6.el7.x86_64  
libselinux-python-2.2.2-6.el7.x86_64  
libselinux-utils-2.2.2-6.el7.x86_64  
policycoreutils-2.2.5-20.el7.x86_64  
policycoreutils-devel-2.2.5-20.el7.x86_64  
policycoreutils-python-2.2.5-20.el7.x86_64  
selinux-policy-3.13.1-60.el7.noarch  
selinux-policy-devel-3.13.1-60.el7.noarch  
selinux-policy-targeted-3.13.1-60.el7.noarch
```

- 3) [S12] This step should only be performed on SLES12.

Install the SELinux software necessary for this lab and disable AppArmor:

```
# systemctl stop apparmor  
# systemctl disable apparmor  
# zypper install -y selinux-policy selinux-policy-minimum selinux-tools  
. . . output omitted . . .  
# systemctl enable restorecond
```

Lab 10

Task 4

Exploring SELinux Modes

Estimated Time: 10 minutes

4) [R7] This step should only be performed on RHEL7.

Check the current status of SELinux:

```
# sestatus  
SELinux status:          enabled  
. . . snip . . .
```

5) [R7] This step should only be performed on RHEL7.

Turn off SELinux by changing the SELinux boot time parameters in /etc/selinux/config:

File: /etc/selinux/config
- SELINUX=enforcing
+ SELINUX=disabled

The SELINUXTYPE variable defaults to the targeted policy.

6) [R7] This step should only be performed on RHEL7.

It is possible that the edit (disabling SELinux) may already have been performed on your system in a previous lab. If SELinux was already disabled, then you can skip the reboot, otherwise reboot to ensure that SELinux is disabled:

```
# systemctl reboot
```

7) Log in as root when the system has rebooted.

8) Verify that SELinux is off:

```
# sestatus  
SELinux status:          disabled
```

9) Attempt to set SELinux to enforcing:

```
# setenforce 1  
Setenforce: SELinux is disabled
```

10) [R7] This step should only be performed on RHEL7.

Re-enable SELinux and configure it to operate in permissive mode:

File: /etc/sysconfig/selinux

-	SELINUX=disabled
+	SELINUX=permissive

It is also possible to pass kernel parameters on boot to manipulate SELinux. A few examples are: `selinux=0` or `selinux=1`; `enforce=0` or `enforce=1`.

11) [S12] This step should only be performed on SLES12.

SLES12 uses AppArmor by default. After installing SELinux, it is set to permissive mode in the `/etc/selinux/config`. The `SELINUXTYPE` variable defaults to the minimum policy. Instead, add kernel parameters to GRUB2 to enable SELinux:

File: /etc/default/grub

→	<code>GRUB_CMDLINE_LINUX_DEFAULT=" ...snip... security=selinux selinux=1 enforcing=0"</code>
---	--

12) [S12] This step should only be performed on SLES12.

It is necessary to configure PAM to enable SELinux, and to update GRUB with the previous step's changes:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg  
# pam-config -a --selinux
```

The `/usr/sbin/selinux-ready` command can be used to check the system for basic configurations needed for enabling SELinux.

13) Create the file /.autorelabel:

```
# touch /.autorelabel
```

The existence of this file ensures a complete relabel at the next boot. In some cases, the system might not recognize the need for a relabel; adding the line `-F` to the file forces a relabel action. When moving from disabled enforcement, or installing a new policy, a relabel should be done.

14) Reboot the system. While the system is starting a complete relabel will be run:

```
# systemctl reboot  
*** Warning -- SELinux targeted policy relabel is required  
*** Relabeling could take a very long time, depending on file  
*** system size and speed of hard drives.  
. . . snip . . .
```

- Depending on your environment, you may not see this message. After the system reboots, relabeling has finished.

Because the system is running in permissive mode, it may print out deny messages, but will not actually block access to resources.

15) Verify that SELinux is enabled:

```
# sestatus  
SELinux status:                 enabled  
SELinuxfs mount:               /sys/fs/selinux  
SELinux root directory:         /etc/selinux  
Loaded policy name:             targeted  
Current mode:                  permissive  
Mode from config file:         permissive  
Policy MLS status:             enabled  
Policy deny_unknown status:    allowed  
Max kernel policy version:    28  
. . . snip . . .
```

16) Switch to enforcing mode:

```
# setenforce 1  
# getenforce  
Enforcing
```

- SELinux starts enforcing the loaded policy immediately.

The setenforce command changes the current SELinux mode for the running system. This change will not be retained across reboots.

17) See that the change is active:

```
# sestatus  
. . . snip . . .  
Current mode:                  enforcing  
Mode from config file:        permissive
```

- SELinux starts enforcing the loaded policy immediately.

. . . output omitted . . .

Clean Up

- 18) [R7] This step should only be performed on RHEL7.

Reconfigure the system to run SELinux in enforcing mode on system boot:

File: /etc/sysconfig/selinux
- SELINUX=permissive
+ SELINUX=enforcing

- 19) [S12] This step should only be performed on SLES12.

Reconfigure the system to disable SELinux:

File: /etc/sysconfig/selinux
- SELINUX=permissive
+ SELINUX=disabled

- 20) [S12] This step should only be performed on SLES12.

Remove the kernel parameters to GRUB2 to disable SELinux:

File: /etc/default/grub
→ GRUB_CMDLINE_LINUX_DEFAULT=" ...snip... security=selinux selinux=1 enforcing=0"

- 21) [S12] This step should only be performed on SLES12.

It is necessary to configure PAM to disable SELinux, and to update GRUB with the previous step's changes:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg  
# pam-config -d --selinux
```

- 22) [S12] This step should only be performed on SLES12.

Re-enable AppArmor and disable restorecond:

```
# systemctl enable apparmor
```

```
# systemctl disable restorecond
```

- 23)** [R7] *This step should only be performed on RHEL7.*

Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

- 24)** [S12] *This step should only be performed on SLES12.*

Reboot the system so changes go into effect:

```
# systemctl reboot
```

Objectives

- ❖ Explore the rules governing the default context assigned to a newly created file
- ❖ Examine the effect of the cp and mv commands on SELinux file contexts

Requirements

▀ (1 station)

Relevance

When copying and moving files, the SELinux security label (file context) attached to a file can be modified. Understanding the effect of commands on these file context labels will allow you to fix and avoid file context problems.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Create new files and view the default SELinux file context assigned:

```
# cd; touch file{1,2,3}  
# ls -Z file*  
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file1  
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file2  
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file3
```

- 3) List the relevant rule from the file context database that causes files created in /root to get the admin_home_t type (if no more specific rule matches):

```
# semanage fcontext -l | grep 'admin_home_t'  
/root(/.*)? all files system_u:object_r:admin_home_t:s0
```

- 4) List the other rules that apply files created in the /root directory:

```
# semanage fcontext -l | grep '^/root'  
/root(/.*)? all files system_u:object_r:admin_home_t:s0  
/root/.ppprc regular file system_u:object_r:pppd_etc_t:s0
```

Lab 10

Task 5

SELinux File Contexts [R7]

Estimated Time: 5 minutes

```
/root/.manpath      regular file      system_u:object_r:mandb_home_t:s0
. . . snip . . .
```

- 5) Create another file to show that more specific rules can override the default context in a directory:

```
# touch .hushlogin
# ls -Z .hushlogin
-rw-r--r--. root root unconfined_u:object_r:local_login_home_t:s0 .hushlogin
```

- 6) Copy and move the file into the /etc directory and view the file contexts assigned to the new files:

```
# cp file1 /etc
# mv file2 /etc
# mv -Z file3 /etc
# ls -Z /etc/file?
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0      /etc/file1
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 /etc/file2
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0      /etc/file3
```

Notice that the file moved *without* the -Z option retained its original context (which is rarely desirable—likely wrong for the target directory).

- 7) Set the file context on the moved file to the correct value for the new location:

```
# chcon -t etc_t /etc/file2
# ls -Z /etc/file2
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0      /etc/file2
```

Cleanup

- 8) Remove the files:

```
# rm -f /etc/file? file? .hushlogin
```

- 9) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Work with SELinux context and information commands.

Requirements

- (1 station)

Relevance

SELinux contexts are at the core of SELinux decision making, understanding how those contexts function, as well as how to manage them is central to SELinux management.

- The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- Install the Apache HTTP server, seinfo and related commands:

```
# yum install -y httpd setools-console audit
```

- Examine the current SELinux security context for the Apache server's document root:

```
# ls -Z /var/www  
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin  
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
```

This indicates that the current SELinux type for these files is `httpd_sys_content_t` (web content) and `httpd_sys_script_exec_t` (executable web scripts).

- Start Apache, if it is not already running:

```
# systemctl start httpd
```

- `systemctl start` will have no affect if the service is already running.

- Connecting to the HTTP server at this point will result in an error since there is no content to serve:

```
# HEAD -d http://localhost/  
403 Forbidden
```

Lab 10

Task 6

SELinux Contexts in Action

[R7]

Estimated Time: 15 minutes

- 6) Make a basic web page and connect to the HTTP server again:

```
# echo 'Hello, World!' > /var/www/html/index.html  
# HEAD -d http://localhost/  
200 OK
```

The Apache server is now fully functional.

- 7) Make a new web page in the /tmp/ directory, and look at the SELinux type it was automatically assigned:

```
# echo 'The cow says moo.' > /tmp/cow.html  
# ls -lZ /tmp/cow.html  
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /tmp/cow.html • The type is user_tmp_t.
```

- 8) Move the cow.html file to a directory where Apache is configured to serve content from, and test retrieving the file:

```
# mv /tmp/cow.html /var/www/html/  
# HEAD -d http://localhost/cow.html  
403 Forbidden
```

- 9) Check in the Apache error log for any hints as to why we are unable to access the file:

```
# tail /var/log/httpd/error_log  
... snip ...  
[Wed Mar 15 11:07:42.386384 2017] [core:error] [pid 4996] →  
(13)Permission denied: [client 127.0.0.1:39016] AH00132: →  
file permissions deny server access: /var/www/html/cow.html
```

- Notice the misleadingly phrased error: "file permissions deny..."

- 10) The output from the previous step suggests a filesystem permissions problem. Check the filesystem permissions:

```
# ps -C httpd -o pid,user,comm  
PID USER COMMAND  
... snip ...  
4937 apache httpd  
4938 apache httpd
```

- Notice httpd runs as the user apache. Any file we want Apache to be able to read, will need to be accessible to the apache user.

```
4996 apache httpd

# namei -l /var/www/html/cow.html
f: /var/www/html/cow.html
dr-xr-xr-x root root /
drwxr-xr-x root root var
drwxr-xr-x root root www
drwxr-xr-x root root html
-rw-r--r-- root root cow.html
```

The user apache has read access to the file cow.html, and read and execute access through all subdirectories. The problem is not filesystem permissions. Whenever a permission denied error is encountered, and the most likely cause has been ruled out. SELinux is a likely culprit.

- 11) SELinux logs errors to the file /var/log/audit/audit.log, while this file can be read by hand, it is much quicker and easier to search through it with ausearch. Examine the SELinux log file, show all AVC events that happened today:

```
# ausearch -m AVC --start today
. . . snip . . .
type=AVC msg=audit(1489597662.385:13397): avc: denied { open } for,
    pid=4996 comm="httpd" path="/var/www/html/cow.html" dev="dm-3",
    ino=2136991 scontext=system_u:system_r:httpd_t:s0,
    tcontext=unconfined_u:object_r:user_tmp_t:s0 tclass=file
```

Reading this error message by hand can seem overwhelming at first, but if the message is broken up. It is easy to understand:

msg=audit(1489597662.385:13397) ⇒ The timestamp when the denial occurred and event id.
denied { open } ⇒ Which system call(s) where made and blocked.
pid=4996 comm="httpd" ⇒ The process ID and command which made the blocked system call(s).
path="/var/www/html/cow.html" ⇒ Which file the process attempted to make the blocked system call on.
scontext=system_u:system_r:httpd_t:s0 ⇒ The label which the process is running under.
tcontext=unconfined_u:object_r:user_tmp_t:s0 ⇒ The label which the files has.

Piecing it all back together, we can conclude: SELinux blocked httpd from making the open system call on the file /var/www/html/cow.html because a process running as the type httpd_t cannot open files with the type user_tmp_t.

- AVC stands for Access Vector Cache. All SELinux rules are stored in the AVC and any log events related to SELinux denials will appear under the AVC type in the audit.log

- 12) The command sesearch can be used to determine how two context types can interact. For example, to determine what context types Apache is allowed to read:

```
# sesearch --allow -s httpd_t  
. . . output omitted . . .  
# sesearch --allow -s httpd_t -t httpd_sys_content_t -c file  
. . . snip . . . allow httpd_t httpd_sys_content_t : file { ioctl read setattr lock open } ;. . . snip . . .
```

This rule allows anything running as the httpd_t type to make the system calls ioctl, read, setattr, lock, and open on a file with the type httpd_sys_content_t.

- 13) Most of the time, it is not necessary to read SELinux error messages by hand. The tool sealert can be used to translate AVC messages into something easily understandable. Before sealert can be used, first get the event ID for the AVC denial:

```
# ausearch -m AVC -c httpd --start today  
. . . snip . . . type=AVC msg=audit(1489597662.385:13397):. . . snip . . .
```

- Show all SELinux denials caused httpd since today.
- Make note of the SELinux AVC denial ID for the event currently being troubleshooted. In this case, event ID 13397.

Take note of the discovered event ID:

Result: _____

```
# ausearch -a 13397  
. . . output omitted . . .
```

- Replace 13397 with the discovered event ID.

- 14) Take the error message and have sealert parse it, replace 13387 with the event ID discovered in the previous step:

```
# sealert -a <(ausearch -a 13397)  
found 1 alerts in /dev/fd/63  
-----
```

- sealert doesn't accept STDIN, instead fancy bash redirection can be used. See the BASH(1) man page, section Process Substitution for more details.

SELinux is preventing /usr/sbin/httpd from open access on the file
/var/www/html/cow.html.

***** Plugin restorecon (92.2 confidence) suggests *****

If you want to fix the label.

/var/www/html/cow.html default label should be httpd_sys_content_t.
Then you can run restorecon.

Do

```
# /sbin/restorecon -v /var/www/html/cow.html
. . . snip . . .
```

Notice how sealert both identified the problem, and offered possible solutions.

The problem has been identified in several steps now. The context type on the file /var/www/html/cow.html is set to user_tmp_t which httpd cannot use. The type got set when the file was created in /tmp/. Moving files does not, by default, change their context type.

- 15) Use the restorecon command to set the context type correctly, according to the SELinux policy, for files in the /var/www/html/ directory:

```
# ls -lZ /var/www/html/cow.html
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /var/www/html/cow.html
# restorecon -vR /var/www/html/
restorecon reset /var/www/html/cow.html context unconfined_u:object_r:user_tmp_t:s0 -->
      ->unconfined_u:object_r:httpd_sys_content_t:s0
# ls -lZ /var/www/html/cow.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/cow.html
```

- 16) Test retrieving the file, again:

```
# HEAD -d http://localhost/cow.html
200 OK
```

Cleanup

- 17) Turn off the web server and remove the files created in this exercise:

```
# systemctl stop httpd
# rm -f /var/www/html/{index,cow}.html
```

- 18) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```


Content

IPv4 Fundamentals	2
TCP/UDP Fundamentals	3
Linux Network Interfaces	4
Ethernet Hardware Tools	6
Network Configuration with ip Command	8
Configuring Routing Tables	9
IP to MAC Address Mapping with ARP	12
Starting and Stopping Interfaces	13
NetworkManager	15
DNS Clients	17
DHCP Clients	19
SUSE YaST Network Configuration Tool	21
Network Diagnostics	22
Information from ss and netstat	25
Hardware and System Clock	27
Managing Network-Wide Time	29
Continual Time Sync with NTP	31
Configuring NTP Clients	32
Useful NTP Commands	34
Lab Tasks	
1. Network Discovery	36
2. Basic Client Networking	39
3. NTP Client Configuration	43

Chapter

11

BASIC NETWORKING

IPv4 Fundamentals

IPv4 addresses

- 32bit
- Contain 2 pieces
network/host

Special addresses

- network address
- broadcast address

Address Classes

- Classed addresses
- Classless addresses – CIDR

Reserved addresses

- RFC 1918

Understanding IP Addresses

The Internet Protocol (IP) uses an address to identify individual devices on a network. IP addresses consist of a 32 bit number that is commonly expressed in dotted quad notation (break the bits into octets, convert each octet to decimal, and separate the four octets by periods). Like most network protocols, IP groups hosts that reside within a common broadcast domain into a logical network. Each IP address within a given IP network will have the same prefix.

Unlike most other network protocols, the IP protocol allows for variable network prefix lengths. This means that to properly interpret an IP address, you must also know what subnet mask is associated with the address. Subnet masks indicate the division between the network bits, and the host bits. Subnet masks are also 32 bits in length and each set bit (binary one) indicates that the bit in the same relative position in the associated IP address is part of the network prefix. Subnet masks are commonly represented in either dotted quad notation (for example 255.255.0.0), or the more elegant /prefix_length format popularized by CIDR/VLSM (for example /16).

The host bits of a normal node IP address always consist of a mixture of binary ones and zeros. The case where all host bits in an IP address are set to binary one is used as the broadcast address (all hosts) for that network prefix. Conversely, the case where all host bits are set to binary zero is used to represent the network itself.

Address Allocation

Historically, all allocation of IP addresses was done by assigning an organization one or more class A, B, or C networks as defined in the following table. (Note that "Class D multicast" and "Class E experimental" are shown in the table for completeness, but their use is not described here.)

Class	First Octet value	Default Netmask
A	1-126	255.0.0.0
B	128-191	255.255.0.0
C	192-223	255.255.255.0
D	224-239	255.255.255.255
E	240-254	

The ad-hoc allocation of IP addresses strictly by classes often led to great inefficiencies (particularly in route aggregation). Today, address allocation is done in a much more controlled and flexible fashion known as CIDR and described in RFC 1518 and RFC 4632.

Private Addresses

To ease deployments of private IP networks, and to prevent address collisions, the following IP netblocks (which are not globally routed) are reserved for private use: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16. RFC 1918 describes the use of these netblocks.

TCP/UDP Fundamentals

TCP basics

- Connection oriented
- Reliable
- 20 byte header

UDP basics

- Connection-less
- Stateless
- Lightweight
- 8 byte header

System Services

- /etc/services

Transport Protocols

Very few applications transmit data in bare IP packets. Instead, most applications rely on one of the transport layer protocols. Many transport layer protocols have been implemented to run over the IP. The two most common are TCP and UDP.

TCP provides a connection-oriented reliable connection. TCP is useful when data integrity is important, because packets are acknowledged and any dropped or missing packets are re-transmitted. Several popular application layer protocols such as FTP, HTTP, and SMTP use TCP as their transport.

UDP is a lightweight, connection-less protocol useful for situations where data integrity isn't required at the transport layer. Application layer protocols may still provide reliable data delivery over UDP, or may not require reliable delivery at all for their operation. Application layer protocols that make use of UDP include TFTP, NTP, and DHCP.

Multiple TCP/IP Connections

To allow multiplexing more than one TCP/UDP connection per source/destination IP address pair, both TCP and UDP use port numbers to provide a transport layer address that defines the endpoint for each connection. Ports are 16 bit numbers usually expressed in decimal.

Certain ports are reserved for specific application protocols. Linux systems have the /etc/services file that will help identify most of the commonly used ports. A Unix tradition is that these reserved

ports are only usable by the root or super-user.

The following table shows a few of the reserved ports and their corresponding services:

Protocol/Port	Service
TCP 20, 21	FTP-Data, FTP-Control
TCP 22	ssh
TCP 23	Telnet
TCP 25, 465	SMTP, SSMTP (email)
UDP 53	DNS (name resolution)
TCP 80, 443	HTTP, HTTPS (web)
TCP 110, 995	POP3, POP3S (email)
TCP 119	NNTP (news)
TCP/UDP 123	NTP
UDP 139	NetBios
TCP 143, 993	IMAP, IMAPS (email)
TCP/UDP 161, 162	SNMP, SNMPTRAP
UDP 514	Remote syslog
TCP/UDP 636	LDAPS

Linux Network Interfaces

Historical and Virtual Machine Network Interface names, e.g. ethX

- X corresponds to the instance number starting at 0

Modern Interface Names based on firmware, topology and location information

- Goal: Provide consistent, predictable interface names
- systemd-udev Interface Naming Scheme used by default
 - Uses five separate, ordered schemes
- Alternate biosdevname Interface Naming Scheme
 - Used if requested **biosdevname=1** or by default on Dell servers

Listing network interfaces via /sys

- `ls /sys/class/net`

Linux Network Interfaces

The Linux operating system handles networking through virtual devices called interfaces. For most practical purposes, an interface is a network connection, such as a connection to an Ethernet network. Keep in mind that several IPs can be bound to any single physical networking device through the creation of virtual interfaces or aliases. The system can be configured with several virtual interfaces all accepting traffic destined for different IP addresses. This can be accomplished with only one physical network connection and several network interfaces bound to the Ethernet connection.

Local Loopback Interface

The loopback interface (`lo`) is a special network interface that points to the local machine. This interface is useful for testing basic networking. It is also useful when testing client-server IP applications (such as a web server), since it means the service will always have an IP address which can be used.

Some applications regularly use the localhost address (on the loopback interface) to talk to other applications on the local system. For example, a daemon that wants to send email and connects to the MTA through the loopback interface.

The loopback interface is assigned all IP addresses in the `127.0.0.0/8` netblock, though typically it is represented as having the host address `127.0.0.1`.

Linux Network Interface Naming

Traditionally interface devices are assigned names based on the type of network connection with which they are associated. Ethernet interface names begin with `eth` followed by a number, starting from zero, which represents the instance of that device in the machine. Thus, `eth0` is the name of the first Ethernet interface, `eth1` is the name of the second Ethernet interface, `ppp3` is the name of the fourth PPP interface, `slip0` is the first SLIP interface, `isdn1` is the second ISDN interface, `tr0` is the first Token Ring adapter, and `fddi0` is the first FDDI adapter.

Today, new naming schemes are used by default such as `systemd-udev` or `BIOSDEVNAME`. These new schemes have a common goal of providing consistent, predictable interface names tied to physical topology information exposed by the BIOS/firmware.

The `systemd-udev` Interface Naming Scheme

By default, the `systemd-udev` interface naming scheme is now used. With this scheme, all interface names have a two character prefix depending on the type of hardware.

`en` ⇒ Ethernet

`wl` ⇒ Wireless LAN (WLAN)

`ww` ⇒ Wireless Wide Area Network (WWAN)

After the two character prefix, the rest of the interface name is set by following 5 ordered schemes based on what information is exposed by the BIOS/firmware. In order, the first scheme able to be applied is

used.

oX ⇒ On-board device index number (e.g., eno1)

sX ⇒ PCI Express hotplug slot index number (e.g., ens1)

pXsX ⇒ PCI geographical location (e.g., enp3p1)

xMAC_ADDRESS ⇒ MAC address, however not used by default. (e.g., enxf0def14f36e6)

Traditional name (e.g., ethX) ⇒ Only used if all previous methods failed

To view the information available to UDEV for a network interface, run the following command:

```
# udevadm info /sys/class/net/eno1
P: /devices/pci0000:00/00:00:01.0/000:01:00.0/net/eno1
E: DEVPATH=/devices/pci0000:00/00:00:01.0/000:01:00.0/net/eno1
E: ID_BUS=pci
E: ID_MODEL_FROM_DATABASE=Ethernet 10G 4P X540/I350 rNDC
E: ID_MODEL_ID=0x1528
E: ID_NET_LABEL_ONBOARD=enNIC1
E: ID_NET_NAME_MAC=enxbc305befbf10
E: ID_NET_NAME_ONBOARD=eno1
E: ID_NET_NAME_PATH=enpl1s0f0
E: ID_OUI_FROM_DATABASE=Intel Corporation
E: ID_PCI_CLASS_FROM_DATABASE=Network controller
E: ID_PCI_SUBCLASS_FROM_DATABASE=Ethernet controller
E: ID_VENDOR_FROM_DATABASE=Intel Corporation
E: ID_VENDOR_ID=0x8086
E: IFINDEX=48
E: INTERFACE=eno1
E: SUBSYSTEM=net
E: SYSTEMD_ALIAS=/sys/subsystem/net/devices/eno1
E: TAGS=:systemd:
E: USEC_INITIALIZED=351223
```

To disable the systemd-udev interface naming scheme, add **net.ifnames=0** to the GRUB_CMDLINE_LINUX variable in the /etc/default/grub file.

BIOSDEVNAME Interface Naming Scheme

The BIOSDEVNAME scheme was created by Dell to provide a consistent, deterministic naming scheme where network interface names correspond to physical labeling for embedded NICs, and PCI slot and port numbers for addon cards. This happens automatically

for Dell Servers and systems that have specifically requested this scheme using **biodevname=1** during installation and also have SMBIOS type 41 records and type 9 records in the BIOS SMI tables (see dmidecode(8), i.e. **dmidecode -t 41**). Virtual machines still use the historical ethX scheme.

To disable the BIOSDEVNAME scheme, add **biodevname=0** to the GRUB_CMDLINE_LINUX variable in the /etc/default/grub file.

UDEV rules call the **biosdevname** command to convert kernel names to the new BIOSDEVNAME scheme. In debug mode, this command will show all detected network interfaces with both their BIOS and kernel names:

```
# biosdevname -d
BIOS device: p3p1
Kernel name: p3p1
Permanent MAC: D4:BE:D9:90:DE:0F
Assigned MAC : D4:BE:D9:90:DE:0F
Driver: tg3
Driver version: 3.119
Firmware version: sb
Bus Info: 0000:02:00.0
PCI name      : 0000:02:00.0
PCI Slot      : 3
Index in slot: 1
```

Manually Controlling the Interface Name

[R7] The following applies to RHEL7 only:

By convention the interface configuration file should have a suffix that matches desired interface name, (e.g., /etc/sysconfig/network-scripts/ifcfg-eth0), however the filename itself doesn't control anything. The HWADDR variable inside the file is used to associate the file to a network interface, and the DEVICE variable is used to manually set the name. The DEVICE variable is normally initialized using the systemd-udev scheme during installation. However, changing the DEVICE variable afterwards will cause the interface name to be changed on the next reboot. Any software configuration files, such as firewall rules, that reference the old name must be updated to the new name.

Ethernet Hardware Tools

ethtool

- Replaces legacy **mii-tool**
- Enhanced functionality specific to Ethernet hardware
- Display and configure Ethernet interface settings
- Display Available Options: **ethtool -h**
- Display Settings On Interface: **ethtool ethX**
- Configure Basic Settings: **ethtool -s ethX option value**
- Display Offload Settings: **ethtool -k ethX**
- Configure Offload Settings: **ethtool -K ethX option value**

Ethtool

The main Ethernet configuration tool, **ethtool**, displays and configures network card settings. Commonly configured options include speed, duplex, and flow control. **ethtool** also supports many advanced settings such as auto-negotiation, TCP Segmentation Offloading, and Wake-on-LAN. A list of available options is available by running **ethtool** without any parameters.

To display current settings on an interface, execute **ethtool** with the name of the interface:

```
# ethtool eth0
Settings for eth0:
Supported ports: [ TP MII ]
Supported link modes: 10baseT/Half 10baseT/Full
                      100baseT/Half 100baseT/Full
Supports auto-negotiation: Yes
Advertised link modes: 10baseT/Half 10baseT/Full
                      100baseT/Half 100baseT/Full
Advertised auto-negotiation: Yes
Speed: 100Mb/s
Duplex: Full
. . . snip . . .

Wake-on: g
Current message level: 0x00000007 (7)
Link detected: yes
```

Changing Settings

Basic changes are set using the **-s** option. Other settings that can be changed include offload parameters:

```
# ethtool -k eth0
Offload parameters for eth0:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp segmentation offload: on
udp fragmentation offload: off
generic segmentation offload: on
large receive offload: off
# ethtool -K eth0 ufo on
# ethtool -k eth0 | grep ^udp
udp fragmentation offload: on
```

Settings will be lost when the interface is reloaded.

[R7] The following applies to RHEL7 only:

ethtool options can be made persistent by using the legacy **ETHTOOL_OPTS** variable in the interface configuration file. For example:

```
File: /etc/sysconfig/network-scripts/ifcfg-ethX
+ ETHTOOL_OPTS="-K ethX gso on; -G X rx 512 tx 256"
```

Also, **ethtool** options can be made persistent with udev:

```
File: /etc/udev/rules.d/70-ethtool.rules
+ SUBSYSTEM=="net", ACTION=="add", NAME=="ethX", -->
    RUN+="ethtool -K ethX gso on; -G X rx 512 tx 256"
```

[S12] The following applies to SLES12 only:

ethtool options can be made persistent by using the **ETHTOOL_OPTIONS=''** setting in the interface configuration file.

Driver Information

Another interesting **ethtool** option is **-i**. This is used to show which driver is handling an interface.

```
# ethtool -i eth0
driver: e1000
version: 7.3.21-k6-NAPI
firmware-version: N/A
bus-info: 0000:03:01.0
```

Physically locating a NIC

On multihomed systems, it can be difficult to map Ethernet devices to their physical ports. The **ethtool** command has the **-p** option, which will cause the link light of the Ethernet device to blink.

However, use caution with this option: it may cause the NIC to disconnect from the network while **ethtool** is running.

Examining and Controlling Ethernet Hardware Properties

The more powerful **ethtool** replaces the older, and unmaintained, **mii-tool**. Some older network cards have better support in **mii-tool**, but most drivers have now been ported to **ethtool**.

```
# mii-tool
eth0: negotiated 100baseTx-FD, link ok
eth1: negotiated 100baseTx-FD, link ok
```

eth2: negotiated 1000baseTx-FD flow-control, link ok

Network Configuration with ip Command

ip

- Layer 2 management
`ip link`
- Layer 3 management (replaces `ifconfig`)
`ip addr`
- Routing table management (replaces `route`)
`ip route`
Can configure routing policies as well
- ARP table management (replaces `arp`)
`ip neigh`
- Can create GRE IP tunnels and configure multicasting

The ip Command

The iproute2 package provides utilities to facilitate advanced network configuration. The `ip` command is the primary tool, allowing administrators to configure network links, network interfaces, routing tables, routing policies, ARP tables, and network tunnels.

Comprehensive documentation on using this powerful tool can be found at <http://www.policyrouting.org/iproute2-toc.html>. Also, the Linux Advanced Routing HOWTO shows many cookbook examples, available at <http://lartc.org/>.

`ifconfig`, `route`, and other networking commands provided by the legacy net-tools package are being replaced by the `ip` command in most network management scripts, because `ip` is far more powerful and flexible.

The IP Command Reference (ip-cref) can be found at:

<http://linux-ip.net/gl/ip-cref/>

Configuring Network Interfaces

To view current state and basic configuration of all interfaces, run:

```
$ ip addr
```

This is similar to `ifconfig -a`. This command can also be written as `ip a`. Add the interface name as a final argument to get a specific interface's information. If only the link state of eth0 is desired, run:

```
$ ip link show eth0
```

Traditionally, configuring an interface with the `ifconfig` command required the device as the first argument, followed by the IP address, and optionally the netmask. The following configures the interface, then brings it down again:

```
# ifconfig eth0 10.200.2.4 netmask 255.255.255.0 up
# ifconfig eth0 down
```

The current way to define IP properties and activate the network interface is to use the `ip` command. For example:

```
# ip link set eth0 up
# ip addr add dev eth0 10.200.2.4/255.255.255.0
```

If the netmask is not specified, the natural mask is used.

Configuring Routing Tables

ip route – tool for runtime management of network routing tables

- Adding routes
 - # ip route add default via 192.168.1.1
 - # ip route add 10.100.0.0/24 via 192.168.1.25
- Deleting routes
 - # ip route del default
- List routes
 - # ip route
 - # netstat -rn

Configuring Routing Tables

The kernel uses routing table entries to determine the next hop and corresponding network interface to use when sending packets to a network it is not directly attached to. A typical configuration is for a default route to be set up for the active network interface so that all outgoing network traffic destined for remote hosts would be handled by a specified gateway system on the local Ethernet network. A default route can be added with either the **ip** or **route** commands:

```
# ip route add default via 192.168.1.1 #new (preferred)
# route add default gw 192.168.1.1      #legacy method
```

All entries in the routing table can be listed by running either: **ip route**, **route**, or **netstat -r**.

```
$ netstat -rn
Kernel IP routing table
Destination      Gateway        Genmask        Flags    MSS Window irtt Iface
10.50.1.0        0.0.0.0      255.255.255.0  U        0 0          0 eth1
10.100.0.0       0.0.0.0      255.255.255.0  U        0 0          0 eth0
127.0.0.0        0.0.0.0      255.0.0.0     U        0 0          0 lo
0.0.0.0          10.50.1.254  0.0.0.0      UG       0 0          0 eth1
$ ip route
10.50.1.0/24 dev eth1  proto kernel  scope link  src 10.50.1.3
10.100.0.0/24 dev eth0  proto kernel  scope link  src 10.100.0.254
127.0.0.0/8 dev lo   scope link
default via 10.50.1.254 dev eth1
```

The previous example output shows that the kernel will route traffic

destined for 10.100.0.0/24 network through interface eth0, traffic for the 10.50.1.0/24 network through eth1, and all other unmatched traffic will be sent through interface eth1 to the gateway at 10.50.1.254.

Setting the Default Gateway on Red Hat Enterprise Linux

The file for specifying a persistent default gateway is the /etc/sysconfig/network file. The default gateway can be specified by adding a line to the file:

```
File: /etc/sysconfig/network
+ GATEWAY=W.X.Y.Z
```

Replace *W.X.Y.Z* with the IP address of the default gateway.

Setting the Default Gateway on SUSE Linux Enterprise Server

The file for specifying a persistent default gateway is the /etc/sysconfig/network/routes file. The syntax for the file is:

destination gateway netmask interface

The default gateway can be specified by adding a line to the file, replacing each variable with the correct value. Adding a dash (-) omits the value as shown in the following example:

```
File: /etc/sysconfig/network/routes
+ default 10.100.0.254 - -
```

Defining Persistent Static Routes on Red Hat Enterprise Linux

To define static routes that persist across a reboot, create a separate file for each network interface:

/etc/sysconfig/network-scripts/route-*interfacename*

One of two alternative syntaxes can be used in this file. The preferred syntax uses the format ADDRESS*n*=*network* where *n* starts at zero and is incremented for each additional route. For example, the following edit will create a route to the 192.168.2.0/24 network via the next hop 10.2.3.200, reachable via the eth0 interface:

```
File: /etc/sysconfig/network-scripts/route-eth0
+ ADDRESS0=192.168.2.0
+ NETMASK0=255.255.255.0
+ GATEWAY0=10.2.3.200
```

Alternatively, a deprecated syntax can be used where each line should be the arguments that are passed to the **ip route add** command. To setup a static route equivalent to the previous example,

the deprecated syntax would be:

```
File: /etc/sysconfig/network-scripts/route-eth0
+ 192.168.2.0/24 via 10.2.3.200
```

For more information, see the file /usr/share/doc/initscripts-*/sysconfig.txt.

Defining Persistent Static Routes on SUSE Linux Enterprise Server

Static routes can be defined in the global routes file. For example, the following edit will create a route to the 192.168.2.0/24 network via the next hop 10.2.3.200 reachable via the eth0 interface:

```
File: /etc/sysconfig/network/routes
+ 192.168.2.0 10.2.3.200 255.255.255.0 eth0
```

Routing information in the global routes file will be ignored when DHCP is used to configure an interface. To define static routes that persist across a reboot, and are activated even when using DHCP, create a separate file for each network interface:

/etc/sysconfig/network/ifroute-*interfacename*

Each of these files use the same syntax as the main routes file. To setup a static route equivalent to the previous example, the syntax would be:

```
File: /etc/sysconfig/network/ifroute-eth0
+ 192.168.2.0 10.2.3.200 255.255.255.0 -
```

For more information, see **man routes**.

Resolving Individual Route

The **ip route get** command will display the next hop, outgoing interface, and source IP address used when sending packets to a particular destination.

```
# ip route
default via 173.9.99.1 dev eth1 proto static metric 100
10.100.0.0/24 dev eth0 proto kernel scope link src 10.100.0.254
173.9.99.0/24 dev eth1 proto kernel scope link src 173.9.99.175
# ip route get 10.100.0.2
10.100.0.2 dev eth0 src 10.100.0.254
# ip route get 8.8.8.8
8.8.8.8 via 173.9.99.1 dev eth1 src 173.9.99.175
```

Dynamic Routing

In larger more complex networks it is common to use a dynamic routing protocol that can automatically adjust routing tables based on changing conditions such as link states. Quagga is a fork of the GNU Zebra route server and a popular option which supports BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng. There is also OpenBGPD, an alternative to GNU Zebra and Quagga.

IP to MAC Address Mapping with ARP

IP packets are routed to destinations based on IP network address
Delivery of Ethernet frames is done by data-link or media access control (MAC) address

ARP maps network addresses to hardware addresses

- IP addresses → Ethernet MAC

RARP maps hardware addresses to network addresses

- Ethernet MAC → IP addresses

ARP Normal Operation:

- hostA sends a network broadcast ARP request for hostB
- hostB sends a unicast ARP reply answer back to hostA
- Both hosts then maintain local ARP caches with learned mappings

Examining and Manipulating the ARP Cache

Nearly all IP-enabled operating systems have methods for examining the ARP table. On Windows and Linux, one method is to use the **arp** command at a shell prompt. To view the contents of a host's ARP table, use the **-a** option. Only hosts contacted recently will be in the table. By default, an entry that has not been contacted will expire soon.

```
$ arp -a
station10.example.com (10.100.0.10) at 00:1b:21:24:f9:35 -
    [ether] on eth0
server1.example.com (10.100.0.254) at 00:1b:21:5a:ea:ee -
    [ether] on eth0
$ ip neigh
10.100.0.10 dev eth0 lladdr 00:1b:21:24:f9:35 STALE
10.100.0.254 dev eth0 lladdr 00:1b:21:5a:ea:ee REACHABLE
```

With the **ip** command, if name resolution is desired, use the **-r** option, otherwise results are unresolved. This shows manually removing 10.100.0.3 and adding 10.100.0.7:

```
# arp -d 10.100.0.3
# arp -s 10.100.0.7 00:01:03:DE:57:C9
# arp -a
station7.example.com (10.100.0.7) at 00:01:03:DE:57:C9 [ether] on eth0
station4.example.com (10.100.0.4) at 00:01:03:DE:57:38 [ether] on eth0
station8.example.com (10.100.0.8) at 00:01:03:DE:56:A4 [ether] on eth0
station9.example.com (10.100.0.9) at 00:01:03:DE:57:AF [ether] on eth0
```

```
# ip neigh del 10.100.0.3 dev eth0
# ip neigh add 10.100.0.7 lladdr 00:01:03:DE:57:C9 dev eth0
$ ip neigh
10.100.0.7 dev eth0 lladdr 00:01:03:DE:57:C9 PERMANENT
10.100.0.4 dev eth0 lladdr 00:01:03:DE:57:38 STALE
10.100.0.8 dev eth0 lladdr 00:01:03:DE:56:A4 REACHABLE
10.100.0.9 dev eth0 lladdr 00:01:03:DE:57:AF REACHABLE
```

ARP mappings can also be stored in a file and loaded using the **-f** option to the **arp** command. The default file used is **/etc/ethers**. After populating the file, enable and start the **arp-ethers.service** and start the service:

```
# systemctl enable arp-ethers
# systemctl start arp-ethers
```

Now the **/etc/ethers** will be processed at boot time.

Starting and Stopping Interfaces

Manually using ip

- # ip addr add 10.100.0.4/24 dev eth0
- # ip link set eth0 up
- # ip link set eth0 down

Using persistent configuration files

- **ifup**: Brings interface up
ifup eth0
- **ifdown**: Takes interface down
ifdown eth0

RHEL7 interface settings file

/etc/sysconfig/network-scripts/ifcfg-ethX

SLES12 interface settings file

/etc/sysconfig/network/ifcfg-ethX

An interface (in this example, eth0) under DHCP control will have a configuration file with the following contents:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:13:20:76:9E:BA
ONBOOT=yes
```

An example interface that is statically configured and turning off Ethernet auto-negotiation would have the following contents:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
IPADDR=10.100.0.14
NETMASK=255.255.255.0
HWADDR=00:13:20:79:2C:96
ETHTOOL_OPTS="speed 100 duplex full autoneg off"
```

With the configuration file created, the interface can be activated using **ifup interface-name** and deactivated with **ifdown interface-name**.

The HWADDR variable is what binds that configuration to the NIC. If the HWADDR variable doesn't exist, then the configuration will be applied to DEVICE unconditionally.

Controlling Linux Network Interfaces

Taking network interfaces up and down is a common system administration task. As described previously, the **ip** (or **ifconfig** command, where available) can be used to do this in Linux. One drawback to using **ip** is that it requires knowledge of all the specific parameters, such as the IP address and subnet mask, with which that interface was first configured.

Modern Linux distributions include **ifup** and **ifdown** scripts which automate most of the initial configuration required before bringing an interface up. These scripts read the system configuration files in /etc/sysconfig/ to determine preset parameters such as the interface's IP address(es). Once these configuration files exist, interfaces can be enabled by **ifup interface-name**, and disabled by **ifdown interface-name**. **ifup** and **ifdown** are shell scripts which look at the system configuration files and carry out appropriate **ip** commands based on the contents of those configuration files.

User Control of Network Interfaces on RHEL7 Systems

If the ifcfg-XXX file contains the entry USERCTL=yes then normal users will be able to use the **ifup** and **ifdown** commands on that interface.

RHEL7 Interface Configuration Files

The /etc/sysconfig/network-scripts directory contains network interface configuration files named after their corresponding interfaces such as: ifcfg-eth0, ifcfg-eth1, ifcfg-ppp0.

SLES12 Interface Configuration Files

The NIC configuration files are found in the /etc/sysconfig/network/ directory. For DHCP controlled interfaces, these files only need these entries:

```
File: /etc/sysconfig/network/ifcfg-eth0
```

```
BOOTPROTO='dhcp'  
STARTMODE='onboot'
```

For statically configured interfaces, these lines would be used:

```
File: /etc/sysconfig/network/ifcfg-eth0
```

```
BOOTPROTO='static'  
STARTMODE='auto'  
IPADDR='10.100.0.14'  
NETMASK='255.255.255.0'
```

There are dozens of additional configuration items that can be specified, though it is not normally needed. With the configuration file created, the interface can be activated using **ifup interface-name** and deactivated with **ifdown interface-name**.

RHEL7/SLES12 Ethernet Device Naming

The Udev file /etc/udev/rules.d/60-net.rules is used to rename the network interface. It looks to the DEVICE variable in an /etc/sysconfig/network-scripts/ifcfg-* file, assuming the MAC address in HWADDR matches an available ethernet interface. Otherwise, the /etc/udev/rules.d/71-biosdevname.rules file determines the interface name, which uses the **biosdevname** command to rename the interface, (BIOSDEVNAME is enabled with the biosdevname=1 kernel parameter). Udev defaults to the /usr/lib/udev/rules.d/80-net-name-slot.rules file, which instructs Udev to rename the interface based on the values provided by the /usr/lib/udev/rules.d/75-net-description.rules file. These can be overridden from /etc/udev/rules.d/. Also, see **systemd-udev(8)**.

NetworkManager

Tools

- **nmcli**
 nm-settings(5)
- GUI: **nm-connection-editor**
- TUI: **nmtui**

Manual ifcfg changes require nmcli c reload
systemd services

- NetworkManager.service
- NetworkManager-dispatcher.service

R7: NetworkManager-config-server RPM

- Disables DHCP on newly added NICs by default
- Configures NM to ignore link state on statically configured NICs

On Red Hat Enterprise Linux the **NetworkManager** plugin /usr/lib64/NetworkManager/libnm-settings-plugin-ifcfg-rh.so translates the traditional network configuration files found under /etc/sysconfig/network-scripts/ifcfg-* into **NetworkManager** profiles.

NetworkManager connections may be manually managed by editing the configuration files found under /etc/sysconfig/network-scripts/ifcfg-*. After editing, the command **nmcli c reload** must be run to make **NetworkManager** aware of any changes.

See man page **nm-settings-ifcfg-rh(5)** for additional details, including a full list of allowed variables within the ifcfg-* files.

NetworkManager Dispatcher

NetworkManager supports an interface that allows a script to be placed in /etc/NetworkManager/dispatcher.d/ to configure environmental settings when an interface is brought up, or down.

Two positional arguments are passed to the script: (\$1) name of the network interface and (\$2) the action, either up or down. The following will signal Postfix to attempt immediate redelivery of mail when any interface comes up:

```
File: /etc/NetworkManager/dispatcher.d/fire-off-mail.sh
+#!/bin/bash
+[ "$2" = "up" ] && postfix flush &> /dev/null
```

nmcli

Red Hat Enterprise Linux defaults to using Network Manager for its network configuration. The **nmcli** command is a utility for persistent configuration of network settings through **NetworkManager**. Also, it is useful for scripting configuration of networking devices. For example, a script could contain the following:

```
# nmcli connection show eth0 | grep method
ipv4.method              auto
ipv6.method                link-local
# nmcli connection modify eth0 ipv4.address 10.100.0.25/24
# nmcli connection modify eth0 ipv4.gateway 10.100.0.254
# nmcli connection modify eth0 ipv4.dns "10.100.0.254,8.8.8.8"
# nmcli connection modify eth0 +ipv4.dns 8.8.4.4
# nmcli connection modify eth0 connection.autoconnect yes
# nmcli connection modify eth0 ipv4.method manual
# nmcli connection reload
# nmcli connection show eth0 | grep method
ipv4.method              manual
ipv6.method                link-local
```

Associating New Devices with Connections

Network devices (most commonly physical interfaces) are not the same as connections which are logical profiles that use one or more devices. When additional network interfaces are added to a system, they will not initially be associated with any connection. Add them as shown in the following example:

```
# nmcli device show
DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  connected   eth0
eth1    ethernet  disconnected  --
lo     loopback  unmanaged   --
# nmcli connection show
NAME    UUID           TYPE      DEVICE
eth0   31113e7a-08b0-4cfc-85eb-ddf6405007ad  802-3-ethernet  eth0
# nmcli connection add type ethernet ifname eth1 con-name eth1
Connection 'eth1' (65882cfe-a13c-4ba7-8a16-867d360e3f64) successfully added.
# nmcli connection show
NAME    UUID           TYPE      DEVICE
eth0   31113e7a-08b0-4cfc-85eb-ddf6405007ad  802-3-ethernet  eth0
eth1   65882cfe-a13c-4ba7-8a16-867d360e3f64  802-3-ethernet  eth1
```

DNS Clients

/etc/resolv.conf

- Identifies name servers and name resolution options
search gurulabs.com example.com example.net
nameserver 192.168.2.12
nameserver 192.168.2.13

/etc/hosts

- Identifies hostnames and aliases with IP addresses
- Is not propagated to other machines
192.168.2.5 fun.gurulabs.com fun mail
192.168.2.6 picard.gurulabs.com

Configuring the DNS Resolver

In an IP networked environment, IP addresses (such as 192.168.5.55) can be used to connect with other systems. However, people generally prefer text names (easier to remember and type.) The most common method of resolving system names to IP addresses is the Domain Name System protocol.

The DNS is a client/server protocol and requires each participating client system to be configured with the IP address of at least one DNS server. This is done in the /etc/resolv.conf file.

Local Resolution via /etc/hosts File

A simpler, but less scalable, way of resolving system names to their network address is via a locally stored file (/etc/hosts). Keeping this file synchronized and current becomes difficult as the network grows.

The /etc/hosts file is a listing of IP addresses and their associated hostnames and aliases and typically looks something like:

File: /etc/hosts

```
127.0.0.1 localhost.localhost localhost
10.100.0.4 station4.example.com
```

Effects of DHCP on Name Resolution Configuration

Normally, when using DHCP on a network interface, the /etc/resolv.conf file is rewritten using the name servers specified in the DHCP response.

[R7] *The following applies to RHEL7 only:*

To prevent the DHCP server from updating the /etc/resolv.conf, add the line PEERDNS=no to the interface configuration file as shown in this example file:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
+ PEERDNS=no

However, this does not prevent NetworkManager from updating the /etc/resolv.conf file when NM_CONTROLLED=yes is set. Use DNS1, DNS2, and DNS3 for static resolver configuration with NetworkManager. If NetworkManager is running, changes should be immediate.

File: /etc/sysconfig/network-scripts/ifcfg-eth0

+ DNS1="127.0.0.1"
+ DNS2="10.100.0.1"
+ DNS3="10.100.0.254"

It is also possible to prevent the system's NTP servers and NIS servers & domains from being changed by DHCP:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
+ PEERNTP=no  
+ PEERNIS=no
```

[S12] *The following applies to SLES12 only:*

To prevent the DHCP server from updating the resolv.conf, add the line DHCLIENT_MODIFY_RESOLV_CONF=no to the /etc/sysconfig/network/dhcp file or add the line MODIFY_RESOLV_CONF_DYNAMICALLY=no to the /etc/sysconfig/network/config file.

DHCP Clients

Command line DHCP Client — dhclient

- Internet Software Consortium reference DHCP client
- Supports Dynamic DNS updates
- Runs as a daemon in order to manage DHCP leases
- Normally invoked automatically based on interface settings
Can be invoked manually for one-off situations
dhclient eth0

[S12] *The following applies to SLES12 only:*

On SUSE Linux Enterprise Server, the lease file is stored in /var/lib/dhcp/dhcpd-ethX.info

The **dhclient** daemon also provides the ability to script actions to be executed when a lease has been obtained. This is used to do things like automatic POP email retrieval whenever a network connection becomes available, or to launch VPN establishment scripts.

For example, the following script ensures that gurulabs.com is always in the DNS search domain.

[R7] File: /etc/dhcp/dhclient-exit-hooks

```
#!/bin/sh
grep search /etc/resolv.conf | grep -q gurulabs.com
if [ $? -ne 0 ]; then
    sed -i 's/search /search gurulabs.com /g' \
    /etc/resolv.conf
fi
```

[R7] *The following applies to RHEL7 only:*

On Red Hat Enterprise Linux, the files /etc/dhcp/dhclient-enter-hooks and /etc/dhcp/dhclient-exit-hooks can be used to run scripts before and after address lease and renewal for extra processing.

ISC DHCP Client

Historically, DHCP has been a slightly non-standard protocol; although DHCP is thoroughly specified in relevant RFCs, many vendor implementations of DHCP have been quite buggy, and have been spotty regarding the client cards which they support. Furthermore, different vendor DHCP servers have failed to work with different client NICs and client software, making deployment of DHCP in the enterprise much trickier than necessary. Initially, the only open-source DHCP Unix client available was **dhcpcd**, produced by the Internet Software Consortium.

Internet Software Consortium rewrote their DHCP client to support more hardware and offer more features. The **dhclient** command is the new DHCP reference client.

The **dhclient** utility can optionally use a configuration file. In the configuration file, timers can be adjusted, and other aspects of the DHCP handshake, including Dynamic DNS settings.

Lease Files

Once a lease has been obtained, the dhcp client stores the lease information in a file on the filesystem.

[R7] *The following applies to RHEL7 only:*

On Red Hat Enterprise Linux, the lease file is stored in /var/lib/dhclient/dhclient-ethX.leases

[S12] *The following applies to SLES12 only:*

On SUSE Linux Enterprise Server, there is no support for dhcp-specific network configuration scripts. If you want additional configuration done after your device is brought up, use the generic network config script directory, /etc/sysconfig/network/if-up.d/ which will be checked for scripts to run after address lease and renewal for extra processing.

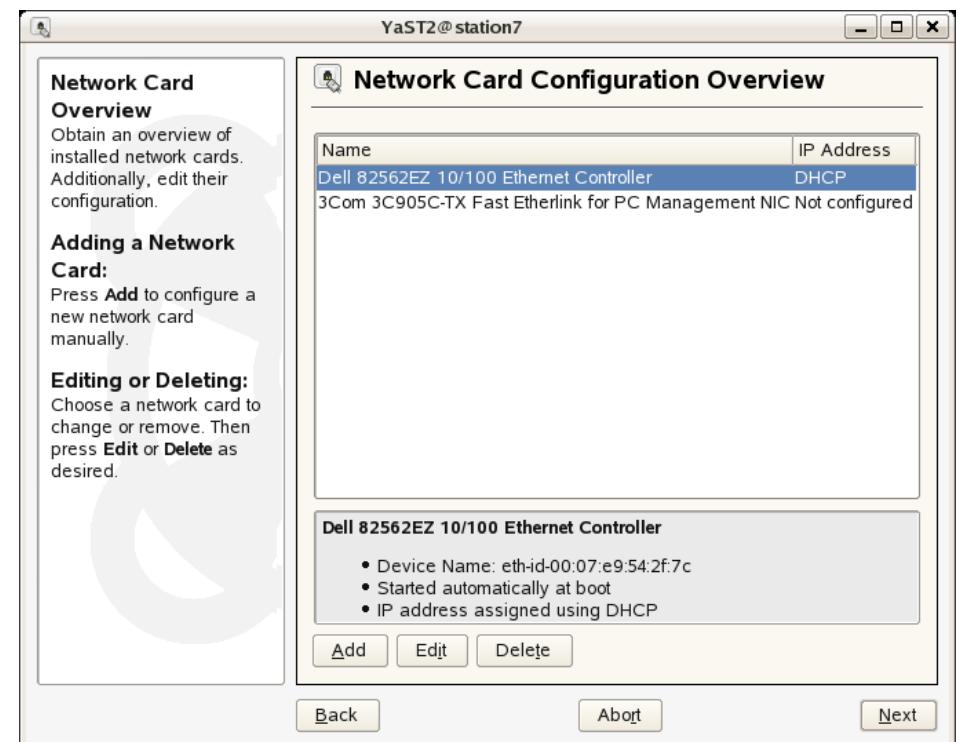
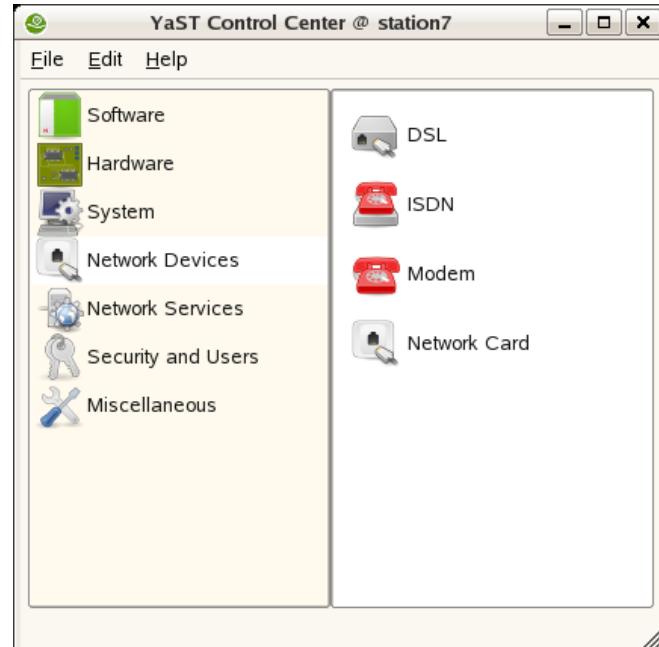
SUSE YaST Network Configuration Tool

YaST2

- Configures basic networking
- Console-based
- GUI-based

Network Configuration

SUSE's YaST offers a variety of different modules which can be used to maintain many of these network configuration files. Some of these applets include DSL and ISDN configurations. Historically, configuring dial-up networking on Linux had been quite difficult; YaST's modem module makes dial-up networking extremely straightforward.



Network Diagnostics

IPv4 connectivity testing

- ping, traceroute, tracepath, mtr

IPv6

- ping6, traceroute6, tracepath6

Hostname and DNS testing

- dig, host, nslookup

Probing for free/used addresses

- arping -D
- nmap -sP

IPv4 Connectivity Testing

The **ping** utility is useful for sending ICMP echo request packets to hosts. Any host which receives an ICMP echo request will normally reply with an ICMP echo response. This fact makes **ping** useful for testing connectivity between hosts. In addition, the time elapsed between echo request, and echo reply, can be used to gauge the speed of the connection between the two hosts.

The **traceroute** utility generates UDP probe packets, to hopefully unused ports, with increasingly larger TTL values. The goal is that each probe packet will be discarded by intermediate routers as the TTL decrements to zero. The resulting "ICMP time exceeded in-transit" error datagrams will identify the IP of each intermediate router in turn. The output show the name, IP, and amount of time it took each reply message arrive. The times can help identify where network problems are occurring by revealing route hops with excessive latency. The following example shows typical **traceroute** output:

```
$ traceroute www.google.com
traceroute to www.google.com (74.125.19.147), 30 hops max, 40 byte packets
 1 fw-devnet-dmz.gurulabs.com (10.2.13.1)  4.278 ms  4.264 ms  4.273 ms
 2 fw-int.gurulabs.com (10.1.0.1)    0.251 ms  0.256 ms  0.243 ms
 3 rtr.gurulabs.com (64.245.157.1)  0.879 ms  1.039 ms  1.013 ms
. . . snip . . .
10 209.85.249.30 (209.85.249.30)  44.734 ms  62.953 ms  46.366 ms
11 nuq04s01-in-f147.1e100.net (74.125.19.147)  47.394 ms  41.763 ms  50.814 ms
```

tracepath is a similar, but simpler, utility available on some hosts that also displays the path MTU.

mtr is a specialized **traceroute** like utility that sends ICMP Echo messages encapsulated in IP packets with increasing TTLs. As with traceroute, the ICMP time exceeded messages sent as packets are dropped are used to identify and measure latency to all intermediate routers. **mtr** can output results in many formats including a real-time updating display (default). The following example shows generating a simple report:

```
# mtr --report -c 5 google.com
```

	Snt: 5	Loss%	Last	Avg	Best	Wrst	StDev
station15.example.com		0.0%	0.5	0.2	0.2	0.5	0.1
server1.example.com		0.0%	2.4	3.7	2.1	9.2	3.1
classes-fw.gurulabs.com		0.0%	0.3	0.3	0.3	0.3	0.0
fw-int.gurulabs.com		0.0%	25.8	27.0	24.9	32.7	3.2
... snip ...		0.0%	22.8	23.6	21.9	26.9	2.1
209.85.251.94		0.0%					
nuq04s01-in-f99.1e100.net		0.0%					

Hostname and DNS Testing

The host and DNS domain name associated with the system can be seen with the **hostname** command. Common options include:

- f ⇒ Fully Qualified Domain Name (FQDN); for example loki.gurulabs.com
- s ⇒ Short name only; for example loki
- d ⇒ DNS domain name only; for example gurulabs.com

Use **dig** and **host** to query domain name servers about hostnames (and IP addresses). **dig** returns information in BIND zone file syntax by default making it more friendly to BIND DNS administrators. It uses internal resolver routines and does not honor any domain or search options that may be configured in the /etc/resolv.conf file. The end of the output shows additional info such as what server responded to the query, when the query was made (date/time), and how long it took for the response to arrive.

The **host** command is designed to be more end-user friendly. By default it can make several queries in an attempt to return all information that may be related or relevant to the actual query made. It does honor setting found in the /etc/resolv.conf file and returns results in a much simpler and more terse format by default.

nslookup is deprecated in favor of the other two, more accurate, commands, and may not be included in the future.

```
# dig www.google.com
; <>> DiG 9.3.6-P1-RedHat-9.3.6-4.P1.el5 <>> www.google.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<< opcode: QUERY, status: NOERROR, id: 18180
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 4, ADDITIONAL: 4
;; QUESTION SECTION:
;www.google.com.           IN      A
;; ANSWER SECTION:
www.google.com.      5500    IN      CNAME   www.l.google.com.
www.l.google.com.     287     IN      A       74.125.19.104
www.l.google.com.     287     IN      A       74.125.19.147
www.l.google.com.     287     IN      A       74.125.19.99
www.l.google.com.     287     IN      A       74.125.19.103
;; AUTHORITY SECTION:
```

```
google.com.      104825    IN     NS     ns4.google.com.  
google.com.      104825    IN     NS     ns1.google.com.  
google.com.      104825    IN     NS     ns2.google.com.  
google.com.      104825    IN     NS     ns3.google.com.
```

;; ADDITIONAL SECTION:

```
ns1.google.com.  97922    IN     A      216.239.32.10  
ns2.google.com.  97922    IN     A      216.239.34.10  
ns3.google.com.  97922    IN     A      216.239.36.10  
ns4.google.com.  97922    IN     A      216.239.38.10
```

;; Query time: 1 msec

;; SERVER: 10.100.0.254#53(10.100.0.254)

;; WHEN: Mon Apr 19 16:22:35 2010

;; MSG SIZE rcvd: 252

host www.google.com

www.google.com is an alias for www.l.google.com.

www.l.google.com has address 74.125.19.147

www.l.google.com has address 74.125.19.99

www.l.google.com has address 74.125.19.103

www.l.google.com has address 74.125.19.104

Probing for Free Addresses

A variety of commands exist that can probe a network segment so that you can determine what addresses are currently in use. This can be helpful if you have to temporarily bring up a host on a network segment that doesn't use DHCP, and where you don't have easy access to the current IP allocation information. One procedure that could be used is as follows:

- ⊗ To start, bring up your network interface without any address.
- ⊗ Select an address that you think is free and use.
- ⊗ **arping -D w.x.y.z** to verify that no other hosts are currently using the address. Repeat until a free address is found. Initialize your network interface with the discovered address.
- ⊗ Use **nmap -sP** to send ICMP echo-request probes and discover the full list of addresses in use. (Remember that some hosts may not respond to ping probes).

Information from ss and netstat

ss and netstat can be used to print network connections, routing tables, interface statistics, and masqueraded connections

- **-s** view protocol statistics
- **-ta** list all TCP connections and listening ports
- **-ua** list all UDP connections and listening ports
- **-ape** list TCP, UDP, and UNIX socket connections and their associated state, user, and program PID / name
- **-n** do not resolve (numeric)

netstat -r view the routing table

ss is an alternative from the iproute2 package

- Syntax is discoverable to **netstat** users
- **-r** resolve names

Example netstat Usage

```
# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask        Flags    MSS Window irtt Iface
208.177.141.0   0.0.0.0         255.255.255.0 U        40 0        0      eth0
208.177.141.0   0.0.0.0         255.255.255.0 U        40 0        0      ipsec0
10.2.0.0        0.0.0.0         255.255.255.0 U        40 0        0      eth1
10.200.1.0      208.177.141.1  255.255.255.0 UG       40 0        0      ipsec0
192.168.32.0    10.1.0.200     255.255.255.0 UG       40 0        0      eth2
. . . snip . . .

# netstat -ape
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address      State   User   Inode PID/Program name
tcp    0      0      *:32768           *:*                LISTEN  rpcuser 1292  791/rpc.statd
tcp    0      0      *:32769           *:*                LISTEN  root    1584  1087/rpc.mountd
tcp    0      0      *:838             *:*                LISTEN  root    1568  1082/rpc.rquotad
tcp    0      0      *:mysql           *:*                LISTEN  root    1664  1161/mysqld
. . . snip . . .

# netstat -s
Ip:
 313000712 total packets received
 279 incoming packets discarded
 278392503 incoming packets delivered
 392232156 requests sent out
. . . snip . . .
```

View network interfaces with **netstat**:

```
# netstat -i
Kernel Interface table
Iface      MTU Met     RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg
lo        16436   0       4       0       0       0       4       0       0       0      LRU
p3p1     1500    0    6996     29       0       0    2854       0       0       0      BMRU
```

Even better, use the **ip** command:

```
# ip -s link
. . . snip . .
2: p3p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether d4:be:d9:90:fc:b1 brd ff:ff:ff:ff:ff:ff
    RX: bytes packets errors dropped overrun mcast
        4280362    7011     29       0       0      675
    TX: bytes packets errors dropped carrier collsns
        569315    2867       0       0       0       0
```

Example ss Usage

```
# ss -ape
State  Recv-Q Send-Q Local Add:Port  Peer Add:Port
LISTEN  0    128      :::ssh      ::::* users:(("sshd",25101,4))  ino:129870 sk:f478e100
LISTEN  0    128      *:ssh      *::* users:(("sshd",25101,3))  ino:129868 sk:f4b36080
LISTEN  0    128  127.0.0.1:ipp  *::* users:(("cupsd",3575,3))  ino:8247 sk:f4b4c580
LISTEN  0    128      ::1:ipp    ::::* users:(("cupsd",3575,1))  ino:8246 sk:f4750b80
LISTEN  0    100      ::1:smtmp  ::::* users:(("master",3679,13)) ino:8765 sk:f478f700
LISTEN  0    100  127.0.0.1:smtmp  *::* users:(("master",3679,12)) ino:8763 sk:f4b36a80
. . . snip . . .
```

```
# ss -s
Total: 487 (kernel 488)
TCP:   11 (estab 1, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 5
```

Transport	Total	IP	IPv6
*	488	-	-
RAW	0	0	0
UDP	12	7	5
TCP	11	6	5
INET	23	13	10
FRAG	0	0	0

Hardware and System Clock

Hardware clock vs. system clock

- **hwclock**
 /etc/adjtime
- **date**

systemd's timedatectl

Manipulating the Hardware and System Clocks

The hardware clock is usually connected to a small battery that allows it to continue to function even when the rest of the system is turned off or disconnected from external power sources.

Functions within the Linux kernel rely on the system clock that is provided by the kernel instead of the hardware clock. Examples of operations that make use of the system clock include: file creation and modification timestamps and log file timestamps.

When the system boots, the hardware clock can be used to seed the system clock with its initial value. During extended operation, the values of the hardware and system clock can drift apart. In this case, it may be necessary to re-seed the system clock value from the hardware clock or vice versa (depending on which of the two clocks is diverging from the true time).

The **hwclock** command is the primary way of interacting with the hardware clock. The following examples illustrate its usage:

```
# hwclock --systohc  
# hwclock --set --date "12/30/2006 16:52 MST"  
# hwclock --show  
Sat 30 Dec 2006 12:52:40 AM MST -0.534333 seconds
```

The /etc/adjtime file is created when **hwclock** is executed using either the **--set** or **--systohc** option. /etc/adjtime is a plain ASCII text file containing three lines:

File: /etc/adjtime

```
3.000000 1377133973 0.000000  
1377133973  
UTC
```

Line 1 contains three space separated numeric values. The first value, from left to right, is a floating point value containing the seconds per day that the hardware clock is adjusted for drift. The second value is the Unix time stamp of the last time the hardware clock was set. The third value is always zero and is only present for compatibility.

The second line is again the Unix time when the clock was last set. The third line is the word UTC or LOCAL to indicate what was used by the **hwclock** command last time it set or adjusted the clock. If the **hwclock** command is invoked in the future without specifying the **--UTC** or **--local** options, the value of the third line will be used.

The **--adjust** option can be used to adjust the hardware clock for drift using the value from /etc/adjtime. The **hwclock** command will calculate the number of days that have passed since the last time the hardware clock was set through the use of **--set** or **--systohc** and add the number of seconds, determined by using the first value of the first line multiplied by the number of days passed, to the current hardware clock time.

The **date** command is the primary way of interacting with the system clock. In addition to being able to set the system clock, the **date** command has extreme configurability in the output format it uses to display the current system date and time. The following examples illustrate its usage:

```
# date --set "Tue May 10 20:03:30 MDT 2005"
Tue May 10 20:03:30 MDT 2005
# date +%x
05/10/05
# date "+%A, %B the %eth"
Tuesday, May the 10th
# echo "$(date +%) seconds have passed since the Linux Epoch"
1115773238 seconds have passed since the Linux Epoch
```

On newer systemd based systems, the **timedatectl** command can be used to view and set the system clock. It can also be used to enable or disable the use of an NTP daemon for time synchronization. Sample output is as follows:

```
# timedatectl
    Local time: Thu 2014-10-30 16:53:25 MDT
    Universal time: Thu 2014-10-30 22:53:25 UTC
        RTC time: Thu 2014-10-30 16:53:26
        Timezone: America/Boise (MDT, -0600)
    NTP enabled: yes
NTP synchronized: no
RTC in local TZ: yes
    DST active: yes
Last DST change: DST began at
                  Sun 2014-03-09 01:59:59 MST
                  Sun 2014-03-09 03:00:00 MDT
Next DST change: DST ends (the clock jumps one hour backwards) at
                  Sun 2014-11-02 01:59:59 MDT
                  Sun 2014-11-02 01:00:00 MST
# timedatectl list-timezones
Africa/Abidjan
Africa/Accra
... snip ...
# timedatectl set-timezone Africa/Abidjan
# timedatectl set-local-rtc 0
```

Managing Network-Wide Time

Importance of managing network-wide time

NTP Protocol Client

- `ntpdate`

Time Protocol Client

- `rdate`
- SLES12 legacy client `netdate`

Network Time

Whenever computers are networked together, their time clocks should be synchronized to agree with each other. Many network protocols are inherently time-based. For example, the Remote Procedure Call (RPC) protocol uses timestamps to authenticate client requests. If the time clocks on the server and the client are more than a few seconds out-of-sync with each other, the client is never authenticated by the RPC server. Having system clocks synchronized also allows correlation of log events across multiple systems.

Syncing via the NTP Protocol

For more accurate synchronization of networked systems, the NTP protocol can be used. NTP takes into account, and compensates for, things such as network latency and jitter. The `ntpdate` command can be used to synchronize the local system clock to a remote system running an NTP daemon. Although the `ntpdate` command supports many options, it is not uncommon to invoke it specifying only the address of the time server as shown in this example:

```
# ntpdate tick.usno.navy.mil
```

The `ntpdate` command will update the local system clock in one of the following two ways:

- ❖ If the local and remote time differ by more than .5 seconds then the local time is changed to match the remote clock.
- ❖ If the clocks differ by less than .5 seconds then the local clock is gradually slewed to match the remote time. This gradual

adjustment is less disruptive and more accurate than a one time change to the new value.

Syncing via the Time Protocol

The Time protocol is a simple protocol for synchronizing a system's clock with a remote system. It is defined in RFC 868 and can operate over UDP or TCP. Several commands implement the client portion of the Time protocol and must connect to a remote system running a time server daemon. The time server was historically implemented as an `inetd` service, however today `xinetd` serves that role.

Enabling a Time Server on RHEL7

Use the following commands to enable a Time server using TCP and UDP:

```
# chkconfig time-dgram on  
# chkconfig time-stream on
```

The rdate Command

When using `rdate`, the default operation is to show the remote time. It uses TCP, unless the `-u` option is used to specify UDP. The `-s` option must be used to specify the remote system to sync to:

```
# rdate -s timeserver1  
rdate: [timeserver1] Wed May 11 15:10:41 2005
```

Enabling a Time Server on SLES12

Use the following commands to enable a Time server using TCP and

UDP:

```
# chkconfig time on  
# chkconfig time-udp on  
# systemctl restart xinetd
```

The netdate Command

When using the **netdate** command, the local clock will be synchronized to match the remote system's clock. It uses UDP by default, though **tcp** can be specified as an argument, before the hostname (remote system) argument, to modify this behavior.

```
# netdate timeserver1  
Trying timeserver1...  
timeserver1 -0.619      Wed May 11 15:15:22.000
```

Continual Time Sync with NTP

NTP Operations

- unicast
- broadcast
- multicast

Ongoing synchronization daemon NTP

- `ntpd`

compensate for the system's hardware frequency error.

As a safety precaution the daemon compares the local and remote times before adjusting the local time. If the local and remote times differ by more than approximately 17 minutes the daemon assumes one of the two is misconfigured and does not try to synchronize the local time. One approach to ensuring accurate time on the client is to:

- ❶ Run the `ntpdate` application at system boot time to ensure that the system clock is initially accurate.
- ❷ Run the `ntpd` daemon on the running system to ensure that the system clock remains reliable.

[R7] *The following applies to RHEL7 only:*

On Red Hat Enterprise Linux, if the file `/etc/ntp/step-tickers` contains a list of NTP servers, the `ntpdate` command will run using those servers and set the clock before the `ntpd` daemon is launched.

File: `/etc/ntp/step-tickers`

```
0.pool.ntp.org
1.pool.ntp.org
2.pool.ntp.org
```

[S12] *The following applies to SLES12 only:*

On SLES12, the `ntpd` startup script extracts the of NTP servers from the main NTP configuration file and runs `ntpdate` using those servers and set the clock before the `ntpd` daemon is launched.

Configuring NTP Clients

ntpd

- /etc/ntp.conf
server or broadcast/multicast client
- RHEL: /etc/ntp/step-tickers
- driftfile

NTP Server Hierarchy

NTP uses stratum levels that define the distance from the reference clock, usually connected with a GPS radio receiver to an atomic clock. The reference clock is stratum0. A stratum1 server is directly connected to the reference clock; a stratum2 server is connected over the network to a stratum1 server, and so on. There are 15 definable stratum levels. A good reference is "The Rules of Engagement," which can be found at <http://support.ntp.org/bin/view/Servers/RulesOfEngagement>.

ntp Configuration

The daemon **/usr/sbin/ntpd** provides both NTP server and client functions. This daemon reads the configuration file **/etc/ntp.conf**, and configures it to operate as a server, or a client, or both, based on configuration parameters listed in the file. For basic client workstation configuration, only one line containing a remote NTP server name or IP address is required. It is preferable to use names rather than addresses, since over time the addresses can change, while the names seldom change. A simple **/etc/ntp.conf** should look like:

```
File: /etc/ntp.conf
+ server 0.pool.ntp.org iburst
```

The NTP server listed should be a stratum 2 or stratum 3 server. Up to three servers can be listed (one per line) to provide for redundancy.

Instead of explicitly configuring NTP servers to bind, the **ntpd** daemon can be configured to listen for broadcast or multicast time data. This can simplify long-term maintenance. To configure a system as a broadcast client, the following configuration statement is used (do not explicitly list server statements):

```
File: /etc/ntp.conf
+ broadcastclient
```

To configure a system as a multicast client, use the following configuration statement instead of listing servers to bind:

```
File: /etc/ntp.conf
+ multicastclient
```

Using an NTP driftfile

The NTP client can measure, and record to a file, the frequency error inherent to the hardware clock in the localhost and compensate for this built-in error. According to NTP documentation the default file is **/etc/ntp.drift** but this file may vary across different distributions. To activate this behavior, include the following configuration statement:

```
File: /etc/ntp.conf
+ driftfile /var/lib/ntp/drift
```

Panic Threshold

The NTP daemon will refuse to sync if the offset time is more than 1,000 seconds, (approximately 17 minutes). The configuration directive `tinker panic 0` instructs NTP not to give up if it sees a large jump in time. This is important for coping with large time drifts and also resuming machines (possibly virtual) from their suspended state. The directive `tinker panic 0` must be at the top of the `ntp.conf` file. For example:

```
File: /etc/ntp.conf
tinker panic 0
restrict 127.0.0.1
restrict default ignore
server 0.pool.ntp.org iburst
server 1.pool.ntp.org iburst
server 2.pool.ntp.org iburst
driftfile /var/lib/ntp/drift
```

The panic value is set in seconds by the `tinker` directive. Any value other than `0` will set the allowable offset in seconds. The default value is equivalent to `tinker panic 1000`.

The `iburst` modifier tells `ntpd` to retry queries that receive no response every 2 seconds for the first 8 requests instead of the default 64 second query interval. This allows NTP to synchronize and form a peer relationship quicker.

[R7] *The following applies to RHEL7 only:*

See the `ntp_misc(5)` and `ntp_acc(5)` manuals for details. Additionally, it is best practice to use the `/etc/ntp/step-tickers` file to list servers that are used to perform an initial time sync before the `ntpd` daemon is started.

[S12] *The following applies to SLES12 only:*

See the `/usr/share/doc/packages/ntp-doc/miscont.html` and `/usr/share/doc/packages/ntp-doc/accept.html` HTML files, part of the `ntp-doc` package, for details. On SUSE, the `ntpd` start up script automatically extracts the defined servers from the `ntp.conf` and performs an initial sync before launching the daemon.

Useful NTP Commands

NTP Support Commands

- **ntpstat**
- **ntpq**
- **ntpdc**
- **ntptrace**

ntpstat

This simple tool prints the NTP server to which the local system is synchronized, and basic time accuracy statistics.

ntpq

The **ntpq** command is the standard NTP query program. It can be used to query NTP servers (which implement the recommended NTP mode 6 control message format) about its current state and to request changes in that state. The program may be run in interactive mode or controlled mode using command line arguments.

To obtain a list of current peers of the server, along with details of each peer state, the following commands could be used:

```
# ntpq -c peers
```

remote	refid	st	t	when	poll	reach	delay	offset	jitter
+clock.team-cymr	204.123.2.72	2	u	351	1024	377	64.307	2.888	2.992
*ponderosa.piney	209.51.161.238	2	u	200	1024	377	71.025	-3.494	0.780
c-76-17-220-129	204.9.54.119	2	u	82h	1024	0	75.923	3.657	0.000

```
# ntpq -c associations
```

ind	assid	status	conf	reach	auth	condition	last_event	cnt
1	32585	941a	yes	yes	none	candidate	sys_peer	1
2	32586	961a	yes	yes	none	sys.peer	sys_peer	1
3	32587	8023	yes	no	none	reject	unreachable	2

ntpdc

The **ntpdc** command is a special NTP query program. It also can be used to query NTP servers about its current state and to request changes in that state. The program may be run in interactive mode or controlled mode using command line arguments. Extensive state and statistics information is available through the **ntpdc** interface. **ntpdc** uses mode 7 packets to communicate with the NTP server.

ntptrace

The **ntptrace** command is a program to trace a chain of NTP servers back to the primary source. This program determines where a given Network Time Protocol server gets its time from. It follows the chain of NTP servers back to their master time source.

Lab 11

Estimated Time:
S12: 25 minutes
R7: 25 minutes

Task 1: Network Discovery

Page: 11-36 Time: 5 minutes

Requirements:  (1 station)

Task 2: Basic Client Networking

Page: 11-39 Time: 5 minutes

Requirements:  (1 station)  (classroom server)

Task 3: NTP Client Configuration

Page: 11-43 Time: 15 minutes

Requirements:   (2 stations)  (classroom server)

Objectives

- Discover network configuration

Requirements

- (1 station)

Relevance

Determining the current network settings in use on a host is commonly done before making changes, or when troubleshooting network related problems.

Notices

- Run ip link. Verify the interface name of the local ethernet or wireless interface, (e.g. p3p1, wlan0). The lab task assumes the device eth0.

- The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- Speed and duplex settings:

```
# ethtool eth0 | grep -E '(Speed|Duplex)'  
[R7] # mii-tool eth0
```

Lab 11

Task 1

Network Discovery

Estimated Time: 5 minutes

- ethtool will work for most network cards.
- mii-tool sometimes has older drivers not found in ethtool.

Result: _____

When run within a virtual machine, sometimes only the link status will be detected.

- MAC address:

```
# ip addr list eth0 | grep ether  
# ifconfig eth0 | grep -i ether
```

Result: _____

4) IP address and subnet mask:

```
# ip addr list eth0 | grep inet  
# ifconfig eth0 | grep inet  
# hostname -i
```

Result: _____

5) Default gateway:

```
# ip route list match 0.0.0.0  
# route | grep ^def
```

Result: _____

6) DNS name server IP(s):

```
# grep ^name /etc/resolv.conf
```

Result: _____

7) If a DHCP client is running, it's likely that you're using DHCP:

```
[R7] # cat /var/run/dhclient-eth0.pid  
[R7] # pidof dhclient  
[S12] # pgrep dhcp
```

Result: _____

8) DHCP Info: if using DHCP, check the lease file:

```
[R7] # cat /var/lib/NetworkManager/dhclient-*-eth0.lease  
[S12] # cat /var/lib/wicked/lease-eth0-dhcp-ipv4.xml
```

- If NetworkManager is disabled, the lease file will appear as /var/lib/dhclient/dhclient-*-interface.lease.

9) Hostname:

```
# uname -n  
# hostname  
# hostname -s  
# hostname -d  
# cat /etc/hostname
```

- On SLES12, YaST can override the contents of this file if DHCP is being used to set the hostname.

Result: _____

10) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Enable static configuration of ethernet interface
- ❖ Demonstrate persistent network configuration

Requirements

- ▀ (1 station) ▀ (classroom server)

Relevance

While DHCP is commonly used to provide IP addresses for workstations, servers and other infrastructure machines are sometimes configured with static addresses. Being able to persistently configure core network settings when installing or moving a system is an essential systems administration skill.

Notices

- ❖ If your systems are already using static IP addresses, then you should skip this lab task. Check with your instructor if you are unsure.
- ❖ Run `ip link`. Verify the interface name of the local ethernet or wireless interface, (e.g. `p3p1`, `em1`). The lab task assumes the device `eth0`.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) So far, your system has been using DHCP for IP address and network information. Examine and record the IP address, subnet-mask and router address (default gateway) that have been obtained via DHCP:

```
[R7] # cat /var/lib/NetworkManager/dhclient-*-eth0.lease  
[S12] # cat /var/lib/wicked/lease-eth0-dhcp-ipv4.xml
```

Lab 11

Task 2

Basic Client Networking

Estimated Time: 5 minutes

- If NetworkManager is disabled, the lease file will appear as `/var/lib/dhclient/dhclient-*-interface.lease`.

Result: _____

- 3) Use the `systemctl` command to stop networking on your system:

```
# systemctl stop network
```

4) [R7] This step should only be performed on RHEL7.

Use the information recorded in Step 2, and switch your networking configuration over to static IP addressing instead of using DHCP:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
- BOOTPROTO=dhcp
+ BOOTPROTO="none"
+ IPADDR="IP_address"
+ NETMASK="netmask"
+ GATEWAY="IP_gateway"
+ DNS1="IP_name_server"
+ DOMAIN="example.com"
+ SEARCH="example.com"
```

Be sure you used the actual values for *IP_address*, *netmask*, *IP_name_server* and *IP_gateway* as obtained in Step 2

5) [S12] This step should only be performed on SLES12.

Switch to static IP addressing instead of DHCP. Edit the /etc/sysconfig/network/ifcfg-eth0 file. These lines are the ones that need to be edited:

File: /etc/sysconfig/network/ifcfg-eth0

```
- BOOTPROTO='dhcp'
+ BOOTPROTO='static'
+ IPADDR='IP_address'
+ NETMASK='netmask'
```

Be sure you used the actual values for *IP_address* and *netmask* as obtained in Step 2

6) [S12] This step should only be performed on SLES12.

Configure the default gateway. By adding the following line to the bottom of the file:

File: /etc/sysconfig/network/routes

```
+ default IP_gateway
```

Be sure you used the actual value of the *IP_gateway* as obtained in Step 2

- 7) Remove extraneous lines from the /etc/resolv.conf file:

```
[R7] # >/etc/resolv.conf  
[S12] # echo 'search example.com' > /etc/resolv.conf  
[S12] # echo 'nameserver IP_name_server' >> /etc/resolv.conf
```

- 8) Start networking:

```
[R7] # systemctl restart NetworkManager  
[S12] # systemctl start network
```

- 9) Verify that your DNS resolver information in /etc/resolv.conf matches the NetworkManager controlled settings defined in the previous step:

```
# cat /etc/resolv.conf  
search example.com  
nameserver IP_name_server
```

Be sure the value of *IP_name_server* matches the value obtained in Step 2

- 10) Verify connectivity back to the classroom server:

```
# ping -c 3 server1  
PING server1.example.com (10.100.0.254) from 10.100.0.X : 56(84) bytes of data.  
64 bytes from server1.example.com (10.100.0.254): icmp_seq=1 ttl=64 time=0.183 ms  
64 bytes from server1.example.com (10.100.0.254): icmp_seq=2 ttl=64 time=0.207 ms  
64 bytes from server1.example.com (10.100.0.254): icmp_seq=3 ttl=64 time=0.217 ms
```

If this ping is not successful then review the changes that you made in the previous steps and test again.

Cleanup

- 11) [R7] This step should only be performed on RHEL7.

Reverse the changes made at the start of the lab, returning your networking configuration to using DHCP:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
- BOOTPROTO=none
+ BOOTPROTO=dhcp
- IPADDR=IP_address
- NETMASK=netmask
- GATEWAY=IP_gateway
- DNS1=IP_name_server
- DOMAIN=example.com
```

12) [S12] This step should only be performed on SLES12.

Return to dynamic networking with DHCP. Edit the /etc/sysconfig/network/ifcfg-eth0 file, and edit the following lines:

File: /etc/sysconfig/network/ifcfg-eth0

```
- BOOTPROTO='static'
+ BOOTPROTO='dhcp'
- IPADDR='IP_address'
- NETMASK='netmask'
```

13) [S12] This step should only be performed on SLES12.

Remove the default gateway configuration by removing the following line:

File: /etc/sysconfig/network/routes

```
- default IP_gateway
```

14) Restart the networking on your system:

```
[R7] # systemctl restart NetworkManager
# systemctl restart network
```

15) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Configure NTP client manually

Requirements

- (2 stations)
- (classroom server)

Relevance

Having the correct time set across the network is an important factor of managing networks. The Network Time Protocol (NTP), implemented in the ntpd, is used to help keep clocks synchronized on systems across the network.

Notices

- The classroom server, server1.example.com, is configured to act as NTP time server for the classroom network.
- The DHCP daemon on server1 informs clients that 10.100.0.254 is a NTP server and the Linux DHCP client daemon, dhclient uses that information to automatically configure the system as a NTP client. Because of this, at the start of this lab you will unconfigure your currently functional NTP client. Do not reboot or restart the network on your system during this lab as it will revert your system back to the auto-configured NTP client state.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Stop any running NTP daemon and backup the current configuration files.

```
# systemctl stop ntpd  
# mv /etc/ntp.conf /etc/ntp.conf.orig  
[R7] # mv /etc/ntp/step-tickers /etc/ntp/step-tickers.orig
```

Lab 11

Task 3

NTP Client Configuration

Estimated Time: 15 minutes

- 3) Create the simplest possible NTP configuration file that uses 10.100.0.254 as the network time source:

File: /etc/ntp.conf
+ server 10.100.0.254

There are more settings that can be entered, but this illustrates that the configuration does not have to be complex.

- 4) Start the NTP daemon:

```
# systemctl start ntpd
```

- 5) Using the **peers** and **assoc** ntpq command check that the time source(s) that the daemon is using matches what you entered in the configuration file as well as the association status with the time source:

```
# ntpq -c peers
      remote          refid      st t when poll reach   delay   offset   jitter
=====
server1.example 59.167.252.133    3 u    37   64     3    0.174  -14.590   3.584
# ntpq -c assoc
      ind assID status  conf reach auth condition  last_event cnt
=====
      1    5042    9014    yes   yes   none    reject    reachable   1
```

- 6) Try using the **-n** switch to ntpq so that the IP is displayed instead of the hostname:

```
# ntpq -n -c peers
      remote          refid      st t when poll reach   delay   offset   jitter
=====
 10.100.0.254 59.167.252.133 3 u    57   64     7    0.172  18.046   0.780
```

- 7) Wait several minutes for the daemon to associate and run the ntpq commands again:

```
# ntpq -c peers
      remote          refid      st t when poll reach   delay   offset   jitter
=====
*server1.example 59.167.252.133    3 u    49    64   377    0.201   -0.702   9.436
# ntpq -c assoc
ind assID status  conf reach auth condition  last_event cnt
=====
  1 43150  9614    yes   yes  none  sys.peer  reachable  1
```

Notice in the output from the **assoc** command that the condition changed from reject to sys.peer and that a * appeared before the name in the output of the **peers** command.

- This is a good spot to take a break. If sys.peer is not acknowledged after several minutes, verify your configuration. If needed, check with your instructor.

- 8) The default NTP daemon configuration is quite trusting and open. It allows remote system to get very detailed information and the current status. Run ntpq commands against another computer in the classroom:

```
# ntpq -c peers stationY.example.com
. . . output omitted . . .
# ntpq -c assoc stationY.example.com
. . . output omitted . . .
# ntpq -c rv stationY.example.com
assocID=0 status=0644 leap_none, sync_ntp, 4 events, clock_sync,
version="ntpd 4.2.6p5@1.1612-o Wed Oct 14 14:41:40 UTC 2014 (1)",
processor="x86_64", system="Linux/$(uname -r)", leap=00, stratum=3,
precision=-20, rootdelay=58.774, rootdispersion=504.192, peer=19,
refid=10.100.0.254,
reftime=d0e2efd8.bb10c63b Thu, Feb 19 2015 10:31:04.730
clock=d0e2f062.102c6ad1 Thu, Feb 19 2015 10:33:22.063, peer=43397,
tc=10, mintc=3, offset=8.363, frequency=0.000, sys_jitter=0.000,
clk_jitter=1.181, clk_wander=0.247
```

- system= should match kernel version.

- 9) Some consider such information disclosure to be a security risk. Create a more secure configuration by using the restrict setting within the configuration file to permit time synchronization no querying to modification of settings:

File: /etc/ntp.conf

```
+ # Allow ourselves to act as a server but do not permit querying or modification
+ restrict default kod nomodify notrap nopeer noquery
+ # This line covers IPv6 access
+ restrict -6 default kod nomodify notrap nopeer noquery
+
+ # Allow full querying and modification from ourselves (localhost) both
+ # IPv4 and IPv6
+ restrict 127.0.0.1
+ restrict -6 ::1
+
server 10.100.0.254
```

- 10) Restart the NTP daemon:

```
# systemctl restart ntpd
```

- 11) Wait for stationY to reach this step before continuing.

- 12) Verify that you can query yourself for information using ntpq, then try against another computer in the classroom that has gotten to Step 10:

```
# ntpq -c peers
      remote          refid        st t when poll reach   delay    offset  jitter
=====
server1.example 204.9.54.119      2 u  23   64    1    0.121  -65.938   0.001
# ntpq -c peers stationY.example.com
stationY.example.com: timed out, nothing received • It will take a moment for this to appear.
***Request timed out
```

- 13) With proper access control now in place, implement another best practice wherein the NTP daemon will fallback to using the motherboard hardware clock as a time source if the network time source is unavailable:

File: /etc/ntp.conf

```
+ server 10.100.0.254  
+ server 127.127.1.0  
+ fudge 127.127.1.0 stratum 10
```

- 14) Restart the NTP daemon:

```
# systemctl restart ntpd
```

- 15) Now with two time sources defined, use the ntpq command and see how the output of **peers** and **assoc** now lists both time sources:

```
# ntpq -c peers  
      remote          refid        st t when poll reach    delay    offset    jitter  
=====  
server1.example 59.167.252.133  3 u   55   64    7    0.172   18.046   0.780  
LOCAL(0)       .LOCL.        10 l   58   64    7    0.000   0.000   0.001  
# ntpq -n -c peers  
      remote          refid        st t when poll reach    delay    offset    jitter  
=====  
10.100.0.254   59.167.252.133  3 u   57   64    7    0.172   18.046   0.780  
127.127.1.0    .LOCL.        10 l   60   64    7    0.000   0.000   0.001  
# ntpq -c assoc  
ind assID status  conf reach auth condition  last_event cnt  
=====  
 1 40636 9014  yes   yes  none   reject   reachable  2  
 2 40637 9014  yes   yes  none sys.peer   sys_peer  3
```

- 16) As a final best practice step, define a drift file so that the NTP daemon can maintain information across restarts about how the hardware clock drifts from the actual time.

File: /etc/ntp.conf

```
+ fudge 127.127.1.0 stratum 10  
+ driftfile /var/lib/ntp/ntp.drift
```

- 17) Restart the NTP daemon:

```
# systemctl restart ntpd
```

- 18) [R7] *This step should only be performed on RHEL7.*

On Red Hat Enterprise Linux, if the /etc/ntp/step-tickers contains a list of NTP servers (one per line), it is used to hard sync the system clock before the NTP daemon is started. This is done because if the system clock differs too much, then the NTP daemon will exit.

File: /etc/ntp/step-tickers

```
+ 10.100.0.254
```

- 19) [R7] *This step should only be performed on RHEL7.*

Set the proper SELinux context on the step-tickers file.

```
# chcon --reference=/etc/ntp/step-tickers.orig /etc/ntp/step-tickers
```

- 20) [R7] *This step should only be performed on RHEL7.*

Stop the NTP daemon, run ntpdate, then restart the NTP daemon.

```
# systemctl stop ntpd  
# ntpdate server1.example.com  
ntpdate[26925]: adjust time server 10.100.0.254 offset -52.093957 sec -- The local clock was 52.09 seconds ahead of the remote  
# systemctl start ntpd
```

- 21)** Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```


Content

Multiple IP Addresses	2
Configuring a DHCP server	4
IPv6	6
Interface Aggregation	8
Interface Bonding	9
Network Teaming	11
Interface Bridging	15
802.1q VLANs	17
Tuning Kernel Network Settings	19
Lab Tasks	20
1. Multiple IP Addresses Per Network Interface	21
2. Configuring IPv6	26
3. Troubleshooting Practice: Networking	30

Chapter

12

ADVANCED NETWORKING

Multiple IP Addresses

Why bind multiple IP addresses to a single interface?

- Virtual hosting SSL websites
- Router/Firewall on-a-stick

Two techniques

- Virtual Interfaces eth0:0, eth0:1, etc
- IP Secondary Aliases - Multiple IPs on same interface

Activating a Virtual Interface temporarily with ip

```
# ip addr add 192.168.2.4/24 label eth0:0 dev eth0
```

Persistent virtual interfaces on RHEL7 with ifcfg-ethX:Y files

Persistent IP aliases on RHEL7 with IPADDRX variables

Persistent IP aliases on SLES12 with IPADDR_XX variables

Reactivate persistent configurations using the ifdown and ifup

The Need for Multiple IPs Per Network Interface

The first need for binding multiple IP address on a single network interface was the requirement of hosting multiple websites on a single computer. This need was reduced with the release of HTTP v1.1 which introduced "name based hosting" allowing multiple websites to be hosted on a single IP address, however, hosting SSL websites still requires a unique IP address per site. Other uses for multiple IPs per NIC include hosting multiple FTP sites and other protocols which have no provision for the client to specify what server they want to access other than by the connection to the specific IP address. Finally, multiple IPs per NIC can be used when multiple IP networks are being used on the same physical network segment.

Linux Virtual Interfaces

Virtual interfaces are defined by appending a colon and a virtual interface number to an existing interface (e.g. eth1:0, eth1:1). To create, and then later delete, a virtual interface in a non-persistent manner, use either **ifconfig** or **ip**:

```
# ifconfig eth0:0 192.168.2.5 up  
# ifconfig eth0:0 down
```

or to do the same with **ip**, run:

```
# ip addr add 192.168.2.5/24 label eth0:0 dev eth0  
# ip addr del 192.168.2.5/24 dev eth0
```

Linux IP Aliases

IP aliases are multiple addresses on the same interface, but are only viewable and able to be managed by **ip**. To create an IP alias in a non-persistent manner, use the **ip** command:

```
# ip addr add 192.168.2.5/24 dev eth0
```

The IP alias can be seen with the command **ip addr list**, but will not be viewable by **ifconfig**.

```
# ip addr list  
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500  
    link/ether 00:1b:21:04:f6:a5 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.2.4/24 brd 10.200.1.255 scope global eth0  
        inet 192.168.2.5/24 scope global secondary eth0  
            inet6 fe80::21b:21ff:fe04:f6a5/64 scope link  
                valid_lft forever preferred_lft forever
```

To delete the IP alias, run:

```
# ip addr del 192.168.2.5/24 dev eth0
```

Persistently Defining Virtual Interfaces on RHEL7

To enable virtual interfaces to persist after a reboot, create an interface configuration file as normal, except referencing the virtual interface name. For example:

File: /etc/sysconfig/network-scripts/ifcfg-eth0:0

```
+ DEVICE=eth0:0
+ BOOTPROTO=static
+ ONPARENT=yes # Bring up with parent interface
+ IPADDR=10.100.0.14
+ NETMASK=255.255.255.0
```

Persistently Defining IP Aliases on RHEL7

To enable multiple IP addresses on a single interfaces use the new syntax available in an ifcfg file. For example:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
+ DEVICE=eth0
+ BOOTPROTO=static
+ IPADDR0=10.100.0.14
+ PREFIX0=24
+ IPADDR1=10.100.0.15
+ PREFIX1=24
+ IPADDR2=10.100.0.16
+ PREFIX2=24
```

Creating Many Virtual Interfaces Easily on RHEL7

If you have the need to create many virtual interfaces where the only change is the incrementing of the IP address and the virtual interface number (clone number), you can create a range file(s) . Each range file can define 254 virtual interfaces. For example, create the file:

File: /etc/sysconfig/network-scripts/ifcfg-eth0-range0

```
+ IPADDR_START=192.168.31.1
+ IPADDR_END=192.168.31.254
+ CLONENUM_START=0
```

Multiple range files per interface are allowed.

Persistently Defining IP Aliases on SLES12

SLES12 uses IP aliasing to bind multiple IP addresses to a single interface. To enable IP aliases to persist after a reboot, add pairs of lines to the ifcfg-ethX file with the following syntax:

File: ifcfg-ethX

```
+ IPADDR_YY=192.168.0.14
+ NETMASK_YY=255.255.255.0
```

The YY can be a unique identifier of your own choosing.

Configuring a DHCP server

ISC (reference) DHCP server

- <http://www.isc.org/products/DHCP>
- Includes support for DHCP failover, conditional behavior, access control, and dynamic DNS support
- Syntax check: `dhcpd -t`
- Main configuration file: `dhcpd.conf`

Contains a subnet definition for each network subnet it serves

Logs to the daemon facility

- log-facility daemon;
`omshell(1)`

Installation and Basics

ISC DHCP was written by Ted Lemon as a reference implementation of RFC 2131 and 2132. The server is configured with the `dhcpd.conf` file. The daemon can be queried and modified using the `omshell` command.

[R7] *The following applies to RHEL7 only:*

The ISC DHCP server is installed with the `dhcp` package.

[S12] *The following applies to SLES12 only:*

The ISC DHCP server is installed with the `dhcp-server` package.

The DHCP daemon is controlled with `dhcpd` `systemd` service.

Syntax checking is provided with the `dhcpd -t` command. For example:

```
# dhcpd -t
. . . output omitted . . .
# echo $?
0
```

Dynamic Configuration

Configure a dynamically leased range of addresses, with defaults for DNS, gateway, and other options, with the following `dhcpd.conf` file:

[R7] File: /etc/dhcp/dhcpd.conf

[S12] File: /etc/dhcpd.conf

```
ddns-update-style none;
option domain-name-servers 10.0.1.3;
subnet 192.0.2.0 netmask 255.255.255.0 {
    option routers 192.0.2.254;
    option subnet-mask 255.255.255.0;
    option domain-name "example.com";
    default-lease-time 21600;
    use-host-decl-names on;
    range 192.0.2.3 192.0.2.20;
}
```

Notice that some options, such as `domain-name-servers`, can be made a DHCP wide default, while others can be specific to certain clients.

The `/etc/sysconfig/dhcpd` controls daemon options, such as what interface ISC DHCP listens on.

Static DHCP Leases

Use the `host` statement to provide DHCP and BOOTP clients with the same IP address leases every time the interface is configured. The following configures `station1` and `station2` to receive static leases:

[R7] File: /etc/dhcp/dhcpd.conf

[S12] File: /etc/dhcpd.conf

```
+ host station1 {  
+   hardware ethernet 00:a0:cc:37:38:88;  
+   fixed-address 192.0.2.1;  
+ }  
+ host station2 {  
+   hardware ethernet 00:a0:cc:37:38:81;  
+   fixed-address 192.0.2.2;  
+ }  
}
```

[S12] The following applies to SLES12 only:

On SUSE Linux Enterprise Server, the dhcpd.leases file is stored in the /var/lib/dhcp/db/ directory.

[R7] The following applies to RHEL7 only:

On Red Hat Enterprise Linux the dhcpd.leases file is stored in the /var/lib/dhcp/ directory.

Logging

The ISC DHCP daemon logs to the daemon facility. This can be changed using the log-facility statement in the dhcpd.conf(5) configuration file.

Red Hat Enterprise Linux and SUSE Linux Enterprise Server log most facilities to the /var/log/messages file, including the daemon facility.

DDNS Updates

For DDNS configuration, set the ddns-update-style to interim (update-static-leases on is needed for fixed-address entries), and add the zone and RNDC key to the /etc/dhcpd.conf file:

[R7] File: /etc/dhcp/dhcpd.conf

[S12] File: /etc/dhcpd.conf

```
+ key dkelson.example.com {  
+   algorithm hmac-md5;  
+   secret oNKYLILr0eqisKxcDBjMVLRD;  
+ };  
+ zone example.com {  
+   primary 10.100.0.254;  
+   key dkelson.example.com;  
+ }  
+ zone 0.100.10.in-addr.arpa. {  
+   primary 10.100.0.254;  
+   key dkelson.example.com;  
+ }
```

IPv6

New Internet networking standard to replace aging IPv4

IPv4 has 32bit addressing

- 4294967296 addresses

IPv6 has 128bit addressing

- 340282366920938463463374607431768211456 addresses

Linux IPv6 support first appeared in 1996 with the 2.1.8 kernel

USAGI Project – UniverSAI playGround for Ipv6

- Current developers of Linux IPv6 code

Example IPv6 address

- 2001:db8::1/32
- 2001:0db8:0000:0000:0000:0000:0001:32

The Promise of IPv6

When the current Internet protocol standard, IPv4, was developed, the growth and size of the current Internet was never envisioned. IPv6 was developed to address current shortcomings of IPv4. Some of the major features of IPv6 include:

- ❖ Header simplification for improved performance
- ❖ Improved end-to-end security with integrated IPSec
- ❖ Efficient and hierarchical routing
- ❖ Stateless auto-configuration
- ❖ New protocol for interaction with LAN neighbors
- ❖ Comprehensive multicast support
- ❖ Greatly expanded IP address space

Unix systems (including Linux) have supported IPv6 for many years. Although vendors such as Cisco and Microsoft took longer to add IPv6 support, it is finally becoming standard. However, the momentum, and will, to move the current Internet to IPv6 has not yet materialized.

Running an IPv6 Linux Router

Most intranet IPv6 routers will run a router advertisement daemon. Linux comes with the **radvd** program that serves this purpose. The **radvd** configuration file is `/etc/radvd.conf`. IPv6 prefixes should be listed in the file associated with the desired network segments connected to the router. On Linux, this requires enabling IPv6 forwarding.

IPv6 on Red Hat Enterprise Linux Systems

For static IPv6 configuration of the default gateway, add the line:

```
File: /etc/sysconfig/network
+ IPV6_DEFAULTGW="IP_of_gateway"
```

A static IPv6 address can be defined on an interface with the **IPV6ADDR** variable in the interface configuration file. For example:

```
File: /etc/sysconfig/network-scripts/ifcfg-eth0
+ IPV6ADDR=IP_address
```

When running an IPv6 Linux router, enable IPv6 forwarding with the following configuration change:

```
File: /etc/sysconfig/network
+ IPV6FORWARDING=yes
```

Disabling IPv6

Support for IPv6 can be disabled by writing 1 to the `/proc/sys/net/ipv6/conf/all/disable_ipv6` file. Though suppressing the `ipv6` module through the **modprobe** configuration has been common (such as with the `ipv6.disable` kernel parameter), it is not recommended. Instead, best practice is to use the **sysctl** command and restart the network:

```
# sysctl -w net.ipv6.conf.all.disable_ipv6=1
```

```
# systemctl restart network
```

Make this persistent by adding an entry in the
`/etc/sysctl.d/ipv6.conf` file:

File: /etc/sysctl.d/ipv6.conf

net.ipv6.conf.all.disable_ipv6 = 1

The kernel parameter `ipv6.disable_ipv6=1` could be passed through
the boot loader to a similar effect.

[S12] *The following applies to SLES12 only:*

The YaST2 network module is able to suppress the IPv6 module,
though this is less desirable. Under Network Card, Global Options,
uncheck Enable IPv6. This does so using the `50-ipv6.conf` file in
`/etc/modprobe.d/`. This will also change the
`/etc/sysconfig/windowmanager` file, toggling the `KDE_USE_IPV6` value
to no.

Interface Aggregation

Why aggregate network links

- Increased bandwidth
- Increased reliability

Link aggregation methods

- bonding → uses bond kernel module
- teaming → uses the team kernel module and `teamd` daemon
Newer method which supports many additional features.

Aggregating Multiple Network Links

As bandwidth requirements on the network continue to grow, traffic requirements can exceed the bandwidth available on a single network interface. Channel bonding can be used to aggregate links together and create a higher capacity logical interface. For example, 4 different 10Gb Ethernet links might be combined together to single logical 40Gb channel bonded interface.

The other primary reason for aggregating links is to increase reliability. A logical channel bonded interface may have individual links fail and still operate (passing traffic across the surviving links). Depending on the topology of the links, and the aggregation method in use, the bonded interface can be resilient to failures of individual network cables, interface cards, or even switches.

Some aggregation methods require protocol support, and configuration on the switch. Popular link aggregation protocols include Cisco Etherchannel (adopted by non-Cisco switches as well), and the IEEE standardized 802.3ad protocol. Other aggregation methods require no special support on the switch and implement all necessary aggregation logic within the bond kernel module. Details about the supported modes can be found in the kernel documentation:

<https://www.kernel.org/doc/Documentation/networking/bonding.txt>

Teaming vs. Bonding

Both the bonding and teaming methods have equal support for the

following features: broadcast TX policy, round-robin TX policy, active-backup TX policy, hash-based TX policy, TX load-balancing support (TLB), VLAN support, LACP hash port select, Ehtool link monitoring, ARP link monitoring, ports up/down delays, configurable via Network Manager (gui, tui, and cli), multiple device stacking.

The teaming method supports the following additional features not supported by bonding: highly customizable hash function setup, D-Bus interface, ØMQ interface, port priorities and stickiness ("primary" option enhancement), separate per-port link monitoring setup, logic in user-space, modular design, NS/NA (IPV6) link monitoring, load-balancing for LACP support, lockless TX/RX path.

The remaining differences are summarized in the following table:

Feature	Bonding	Team
LACP (802.3ad) support	Yes (passive only)	Yes
user-space runtime control	Limited	Full
multiple link monitoring setup	Limited	Yes
extensibility	Hard	Easy
performance overhead	Low	Very Low
RX load-balancing support (ALB)	Yes	Planned
RX load-balancing support (ALB) in bridge or OVS	No	Planned

Interface Bonding

Requires creation of a bond (master) network interface

- individual links are slaves to this master logical interface

Can be created via: nmtui, nmcli, NetworkManager GUI applet, or editing ifcfg-* files directly.

Requires loading and configuring bond kernel module

Example Bonding Configuration on RHEL7

Once the switch has been configured, the standard RHEL7 network configuration files support bonding configurations. The official documentation can be found in the file:

/usr/share/doc/kernel-doc*/Documentation/networking/bonding.txt

To begin, create the master bond0 interface configuration file. Create /etc/sysconfig/network-scripts/ifcfg-bond0 with the contents:

File: /etc/sysconfig/network-scripts/ifcfg-bond0

```
+ DEVICE=bond0
+ BOOTPROTO=static
+ ONBOOT=yes
+ IPADDR=10.1.0.5
+ NETMASK=255.255.0.0
+ BONDING_OPTS="mode=4 miimon=100 xmit_hash_policy=layer2+3"
```

The BONDING_OPTS="" variable may be used in the ifcfg-bond0 file for module parameters. For instance, the miimon value defines the link polling interval for fault detection. Consult the Linux kernel documentation and the Deployment Guide for additional useful options.

For each physical interface that will make up the aggregate bond0 interface, create an interface configuration file using the bonding configuration lines MASTER and SLAVE. No IP configuration lines should be provided. For example assuming that bond0 uses eth0 and eth1,

the following files would need to be created:

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
+ DEVICE=eth0
+ BOOTPROTO=none
+ ONBOOT=yes
+ MASTER=bond0
+ SLAVE=yes
```

File: /etc/sysconfig/network-scripts/ifcfg-eth1

```
+ DEVICE=eth1
+ BOOTPROTO=none
+ ONBOOT=yes
+ MASTER=bond0
+ SLAVE=yes
```

Finally, add the following to a file ending in .conf in the modprobe.d/ directory:

File: /etc/modprobe.d/bond.conf

```
+ alias bond0 bonding
```

Converting Bond Configurations to Teaming

Red Hat Enterprise Linux has an alternative implementation of interface aggregation called network teaming. To convert an existing bonding configuration to a new style network teaming configuration,

run:

```
# /usr/bin/bond2team --master bond0
```

Example Bonding Configuration on SLES12

Once the switch has been configured, the standard SUSE Linux Enterprise Server network configuration files support bonding configurations. The official documentation can be found in the /usr/src/linux/Documentation/networking/bonding.txt file.

To begin, collect the bus id of each card that will participate in the bonding interface. This can be done by creating a temporary configuration file for each card using YaST and extracting the _nm_name from the file. If temporary configuration files were created using YaST, be sure to disable or delete them. Then, create the master bond0 interface configuration file /etc/sysconfig/network/ifcfg-bond0 with this content:

File: /etc/sysconfig/network/ifcfg-bond0

```
BOOTPROTO='static'
STARTMODE='onboot'
IPADDR='10.100.0.25'
NETWORK='10.100.0.0'
NETMASK='255.255.255.0'
BROADCAST='10.100.0.255'
BONDING_MASTER='yes'
BONDING_SLAVE_0='bus-pci-0000:02:02.0'
BONDING_SLAVE_1='bus-pci-0000:02:08.0'
BONDING_MODULE_OPTS='miimon=100'
```

The miimon value defines the link polling interval for fault detection. Consult the official documentation for additional useful settings.

Unfortunately, at this time the SUSE Linux Enterprise Server network scripts do not work well when combining bonding with DHCP.

Network Teaming

Major components

- **team** → kernel module
- **teamd** → userspace daemon
 - compiled with a runner which provides control logic
- **teamdctl** → to monitor and configure

Configuration via: `nmtui`, `nmcli`, `teamd`, `ifcfg-* files`, **NetworkManager GUI**

Teaming Architecture

Red Hat Enterprise Linux offers an alternative implementation of interface aggregation called Network Teaming. In the end it accomplishes the same thing, but has several improvements over the traditional Linux bonding implementation. With Network Teaming all of the logic is pushed into userspace which gives it more flexibility, however, it still has a kernel component, the `team` module that handles the performance sensitive portion. The `team` kernel module uses a lockless Tx/Rx path for much lower overhead compared to the bonding kernel module.

team ⇒ Small kernel module that implements fast handling of packet flows. Listens for communications from userspace applications via Netlink API.

teamd ⇒ Userspace daemon built on the Team lib library, and contains the common logic for interface aggregation.

runner ⇒ Userspace code compiled into a `teamd` instance which implements the specific load-balancing and active-backup logic.

teamdctl ⇒ Command to examine configuration, change state of ports (active/backup), add/remove ports, etc.

The following are a list of the currently available runners which can be used with `teamd`. Note that these are basically equivalent to the various modes implemented in the traditional bonding module:

broadcast ⇒ data is transmitted over all ports

round-robin ⇒ data is transmitted over all ports in turn

active-backup ⇒ one port or link is used while others are kept as a backup

loadbalance ⇒ with active Tx load balancing and BPF-based Tx port selectors

lacp ⇒ implements the 802.3ad Link Aggregation Control Protocol

Configuration via Direct Editing ifcfg-* Files

First define the team interface:

File: /etc/sysconfig/network-scripts/ifcfg-team0

```
+ DEVICE=team0
+ DEVICETYPE=Team
+ ONBOOT=yes
+ BOOTPROTO=none
+ IPADDR=10.100.0.3
+ PREFIX=24
+ TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool"}}'
```

Then create configs for each slave interface substituting the MAC address and device name as appropriate:

File: /etc/sysconfig/network-scripts/ifcfg-team0-port0

```
+ DEVICE=eth0
+ HWADDR=52:54:00:04:00:04
+ DEVICETYPE=TeamPort
+ ONBOOT=yes
+ TEAM_MASTER=team0
+ TEAM_PORT_CONFIG='{"prio": 100}'
```

File: /etc/sysconfig/network-scripts/ifcfg-team0-port1

```
+ DEVICE=eth1
+ HWADDR=52:54:00:04:00:05
+ DEVICETYPE=TeamPort
+ ONBOOT=yes
+ TEAM_MASTER=team0
+ TEAM_PORT_CONFIG='{"prio": 100}'
```

Configuration via nmcli

First create a new teaming connection and add ports to it:

```
# nmcli con add type team ifname team0 con-name team0
Connection 'team0' (6991240c-c9b5-4591-93ff-2f7a5ecf0f40) successfully added.
# nmcli con show
NAME      UUID              TYPE      DEVICE
team0    6991240c-c9b5-4591-93ff-2f7a5ecf0f40  team        team0
# nmcli con add type team-slave con-name team0-port0 ifname eth0 master team0
Connection 'team0-port0' (54c66033-da90-49d4-8bf0-42e62413ba5b) successfully added.
# nmcli con add type team-slave con-name team0-port1 ifname eth1 master team0
Connection 'team0-port1' (35c4ce56-5c5e-4e95-8716-576363b5df82) successfully added.
# nmcli con mod team0 team.config '{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool"}}'
# nmcli con show
NAME      UUID              TYPE      DEVICE
team0-port1 35c4ce56-5c5e-4e95-8716-576363b5df82  802-3-ethernet  --
team0-port0 54c66033-da90-49d4-8bf0-42e62413ba5b  802-3-ethernet  --
team0    6991240c-c9b5-4591-93ff-2f7a5ecf0f40  team        team0
```

The teaming interface will come up when at least one of the ports assigned to it are brought up:

```
# ip link list team0
5: team0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT
    link/ether 7a:15:55:92:64:58 brd ff:ff:ff:ff:ff:ff
# nmcli connection up team0-port0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/6)
# nmcli connection up team0-port1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/7)
# ip link list team0
5: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
    link/ether 52:54:00:04:00:03 brd ff:ff:ff:ff:ff:ff
```

Configure other properties on the teaming interface as needed. For example:

```
# nmcli con mod team0 ipv4.method manual ipv4.addresses "10.100.0.3/24 10.100.0.254"
# nmcli device show team0
GENERAL.DEVICE:                  team0
GENERAL.TYPE:                    team
GENERAL.HWADDR:                 52:54:00:04:00:13
GENERAL.STATE:                   100 (connected)
GENERAL.CONNECTION:              team0
GENERAL.CON-PATH:                /org/freedesktop/NetworkManager/ActiveConnection/10
IP4.ADDRESS[1]:                  ip = 10.100.0.3/24, gw = 10.100.0.254
IP4.DNS[1]:                      10.100.0.254
IP4.DOMAIN[1]:                   example.com
```

Files will be created automatically in /etc/sysconfig/network-scripts/ to store the configuration.

The team interface can be brought up, and the state verified, by running:

```
# ifup team0
# teamdctl team0 state view -v
setup:
runner: activebackup
kernel team mode: activebackup
D-BUS enabled: yes
ZeroMQ enabled: no
debug level: 0
daemonized: no
PID: 4902
PID file: /var/run/teamd/team0.pid
ports:
eth0
ifindex: 2
addr: 52:54:00:04:00:04
ethtool link: 0mbit/halfduplex/up
link watches:
link summary: up
instance[link_watch_0]:
name: ethtool
link: up
link up delay: 0
link down delay: 0
eth1
ifindex: 3
addr: 52:54:00:04:00:04
ethtool link: 0mbit/halfduplex/up
link watches:
link summary: up
instance[link_watch_0]:
name: ethtool
link: up
link up delay: 0
link down delay: 0
runner:
active port: eth1
# teamdctl team0 state item set runner.active_port eth0
```

Interface Bridging

Why Bridge Interfaces?

- Join multiple networks
- Join virtual networks to physical
- Filter packets between networks
- Monitor packets between networks

Ethernet Bridging

Bridging means connecting two networks at OSI model layer 2. Because the packets are forwarded at the Ethernet layer, any protocol can be run over the bridge.

Most Ethernet bridging, by far, is done with dedicated switches. Using a switch will give much better performance than is possible with Linux bridging. The primary reasons to configure bridging on Linux are to connect a virtual network (e.g. tunnel, VPN, virtual machine) to a physical network and to filter or monitor traffic between two networks, using **ebtables** or **iptables**.

Configuring Bridges Manually

All bridge control and inspection is performed with the **brctl** command, which is in the **bridge-utils** package. First create a bridge using the **brctl addbr** command, and then add individual interfaces to it with the **brctl addif** command. List bridges and their associated interfaces with the **brctl show** command:

```
# brctl addbr br0
# brctl addif br0 eth0
# brctl addif br0 eth1
# brctl show
bridge name      bridge id      STP enabled    interfaces
br0              8000.545200635d5b  no            eth1
```

Example Bridging Configuration on RHEL7

On RHEL7, assigning interfaces to be part of a bridge is done by adding a line to the individual interface config files:

File: /etc/sysconfig/network-scripts/ifcfg-eth1

```
+ DEVICE=eth1
+ ONBOOT=yes
+ BOOTPROTO=none
- IPADDR=10.1.0.5
- NETMASK=255.255.0.0
+ BRIDGE=br0
```

For each configured bridge, optionally the IP configuration can be set for that interface:

File: /etc/sysconfig/network-scripts/ifcfg-br0

```
+ DEVICE=br0
+ ONBOOT=yes
+ BOOTPROTO=none
+ IPADDR=10.1.0.5
+ NETMASK=255.255.0.0
```

Example Bridging Configuration on SLES12

On SLES12, to configure bridging, create a new interface config file, /etc/sysconfig/network/ifcfg-br0 with this content:

File: /etc/sysconfig/network/ifcfg-br0

```
IPADDR=10.1.0.5/16
NETWORK=
BROADCAST=
STARTMODE=auto
USERCONTROL=no
BRIDGE='yes'
BRIDGE_PORTS='eth1 eth2'
BRIDGE_AGEINGTIME='300'
BRIDGE_FORWARDDELAY='0'
BRIDGE_HELLOTIME='2'
BRIDGE_MAXAGE='20'
BRIDGE_PATHCOSTS='19'
BRIDGE_PORTPRIORITIES=
BRIDGE_PRIORITY=
BRIDGE_STP='on'
```

802.1q VLANs

802.1q VLANs

- IEEE standard
- `ip link add link eth2 name eth2.101 type vlan id 101`
- Appears as a normal interface
Valid for `ifconfig` and packet capture

Virtual Local Area Network (VLAN) Theory

Splitting networks into multiple data-link layer segments is common and provides a number of benefits: broadcast isolation, security, ability to interconnect hosts using different data-link protocols (e.g. Ethernet, token-ring, FDDI), etc. Historically, new LANs were created by physical topology changes, (e.g. adding a new switch and rewiring devices by plugging a subset of devices into the new switch). A more convenient, and common way of defining data-link boundaries today is by modifying the configuration of a switch (changes to software not hardware).

Normally, when VLANs are used, individual switch ports are assigned to a single VLAN. However, it is sometimes useful to have a port assigned to multiple VLANs simultaneously (most commonly when the port is acting as a trunk and carrying the traffic of multiple VLANs to another switch). In this case, each data-link layer frame has an additional header prepended that identifies the VLAN that the frame belongs to. Configuring a Linux network interface to support VLANs allows the kernel to read this additional header and route the frame to the appropriate virtual interface based on the VLAN ID read from the header.

Controlling Traffic with VLANs

There are two main standards for creating VLANs with Ethernet switches, the IEEE standard 802.1q, and the Cisco proprietary ISL. To the system, a VLAN shows up as a regular network interface. Configuring VLANs can be accomplished through standard network configuration files. It also requires the use of the `ip` (or the older `vconfig`) command.

With the exception of their name, VLAN interfaces can be treated like a regular interface and can be examined, configured and sniffed using the standard commands and files. Note that various network cards and drivers can have issues with the larger VLAN Ethernet frames.

The VLANs must be defined and activated on the Ethernet switch.

RHEL7 VLAN Configuration Details

It is important to note that Red Hat Enterprise Linux uses the `DEV_PLUS_VID_NO_PAD` naming mode, which means that the VLAN network interface name will start with the physical interface name followed by a period followed by the VLAN number. For example the network interface `eth2.101` would be VLAN 101 on `eth2`.

To configure VLANs, ensure that the `8021q` kernel module will load on boot. This can be done by making the following edit:

File: /etc/sysconfig/network

`VLAN=yes`

Continuing the example from before, the interface configuration file for VLAN 101 on eth2 is:

```
File: /etc/sysconfig/network-scripts/ifcfg-eth2.101
DEVICE=eth2.101
BOOTPROTO=none
ONBOOT=yes
IPADDR=10.3.2.1
NETMASK=255.255.255.0
NETWORK=10.3.2.0
```

Load the 8021.q module and restart the network service, or reboot.

SLES12 VLAN Configuration Details

It is important to note that SLES12 uses the ifcfg-vlan*NNN* naming mode, which means that *NNN* should be replaced with the VLAN ID or VLAN number. For example, if the VLAN number is 101 then the interface configuration file will be:

/etc/sysconfig/network/ifcfg-vlan101

VLAN interface configuration is supported via the normal **ifup** and **ifdown** network scripts used by all the other network interfaces.

Continuing the example from before, the interface configuration file for VLAN 101 on eth2 would be:

```
File: /etc/sysconfig/network/ifcfg-vlan101
ETHERDEVICE='eth2'
IPADDR='10.3.2.1'
NETMASK='255.255.255.0'
NETWORK='10.3.2.0'
BROADCAST='10.3.2.255'
STARTMODE='onboot'
```

The ETHERDEVICE variable specifies to which Ethernet interface the VLAN interface should be attached. All other variables are standard for configuring an IP interface.

Tuning Kernel Network Settings

/proc/sys/net filesystem

- Interact with kernel parameters via pseudo-files
`# echo 1 > /proc/sys/net/ipv4/ip_forward`

sysctl

- Command-line interface to kernel parameters
`# sysctl -w net.ipv4.ip_forward=1`

Commonly tuned options include

- rp_filter
- ip_forward
- log_martians

be able to forward packets between interfaces. The option `ip_forward` can be used to enable support for packet forwarding when necessary.

Make the Changes Persistent

In the past, system administrators would put the `echo` commands in a startup script run on boot. Today, the official way to make these changes persistent is editing the config file for the `sysctl` command.

On boot, `systemd` loads its default kernel parameters, and overrides found in `/etc/sysctl.d/` and `/etc/sysctl.conf`.

Useful Network Tuning Options

Many networking-related parameters within the Linux kernel can be modified during runtime. Available parameters will vary from major kernel version to major kernel version. For information about what options can be manipulated on the kernel, and what various settings for those options mean, see the kernel documentation (found in the Documentation sub-directory), specifically the `networking/ip-sysctl.txt` text file.

One commonly tuned option is `rp_filter`, which does source validation of all incoming packets by conceptually reversing their path and verifying that their source route is the same as their reply route. This option is often enabled to enhance security, though this may break complicated asymmetric routing configurations.

Along with `rp_filter`, `log_martians` is often enabled to enhance system security. If enabled, attempts to connect to the system by spoofing impossible source addresses (such as 127.0.0.1) will be logged by the kernel.

Another option commonly modified to enhance system security is `accept_source_route`; by default, Linux will not accept source-routed packets as a security precaution, but this option can be used on routers which might need to enable support for source-routed packets.

By default, Linux systems will not forward packets between interfaces, again as a security precaution. Routers, by definition, must

Lab 12

Estimated Time:
S12: 25 minutes
R7: 25 minutes

Task 1: Multiple IP Addresses Per Network Interface

Page: 12-21 Time: 5 minutes

Requirements:  (1 station)  (classroom server)

Task 2: Configuring IPv6

Page: 12-26 Time: 10 minutes

Requirements:  (1 station)

Task 3: Troubleshooting Practice: Networking

Page: 12-30 Time: 10 minutes

Requirements:  (1 station)

Objectives

- » Add more than one IP address to a network interface
- » Add more than one IP address to a network interface in a persistent fashion

Requirements

- (1 station) ■ (classroom server)

Relevance

It is often useful (especially on systems acting as a router or firewall) to bind multiple IP addresses to a single physical interface. The modern Linux kernel provides several approaches to do this including virtual interfaces and IP aliases.

Notices

- » Run ip link. Verify the interface name of the local ethernet or wireless interface, (e.g. p3p1, em1). The lab task assumes the device eth0.

- 1) Examine the current IP address you have configured on your network card, and record the value of the last octet (shown as an X in the sample output below):

```
# ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:1b:21:24:fd:94 brd ff:ff:ff:ff:ff:ff
    inet 10.100.0.X/24 brd 10.100.0.255 scope global eth0
        . . . snip . . .
```

Result: _____

- 2) Add a second IP address to eth0 using the recorded value as the fourth octet:

```
# ip addr add 192.168.41.X/24 dev eth0
```

- 3) Ping the new alias:

```
# ping -c 1 192.168.41.X
PING 192.168.41.X (192.168.41.X) 56(84) bytes of data.
64 bytes from 192.168.41.X: icmp_seq=1 ttl=64 time=0.028 ms
```

Lab 12

Task 1

Multiple IP Addresses Per Network Interface

Estimated Time: 5 minutes

- 4) Attempt to view the new interface settings with both the ip and ifconfig commands:

```
# ip addr list dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:1b:21:24:fa:6e brd ff:ff:ff:ff:ff:ff
        inet 10.100.0.X/24 brd 10.100.0.255 scope global eth0
            inet 192.168.41.X/24 scope global eth0
# ifconfig -a
eth0      Link encap:Ethernet HWaddr 00:1B:21:24:FA:6E
          inet addr:10.100.0.X Bcast:10.100.0.255 Mask:255.255.255.0 • New IP address is *not* visible.
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:3322 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2801 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:301141 (294.0 KiB) TX bytes:406208 (396.6 KiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:268 (268.0 b) TX bytes:268 (268.0 b)
```

- 5) Re-add the additional address with a label so that the ifconfig command can associate a new virtual interface with the address:

```
# ip addr del 192.168.41.X/24 dev eth0
# ip addr add 192.168.41.X/24 label eth0:0 dev eth0
```

- 6) View the interface again with both commands:

```
# ip addr list dev eth0
. . . output omitted . . .
# ifconfig -a
eth0      Link encap:Ethernet HWaddr 00:1B:21:24:FA:6E
          inet addr:10.100.0.X Bcast:10.100.0.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:3259 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:2760 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:295489 (288.5 KiB) TX bytes:400715 (391.3 KiB)

eth0:0      Link encap:Ethernet HWaddr 00:1B:21:24:FA:6E
            inet addr:192.168.41.X Bcast:0.0.0.0 Mask:255.255.255.0 • New IP address is visible (due to use of a label).
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            . . . snip . . .
```

- 7) Remove the additional interface and address:

```
# ip addr del 192.168.41.X/24 dev eth0
```

- 8) [R7] This step should only be performed on RHEL7.

Create a new interface configuration file to make a persistent virtual interface:

File: /etc/sysconfig/network-scripts/ifcfg-eth0:0
+ DEVICE=eth0:0 + BOOTPROTO=none + IPADDR=192.168.41.X + NETMASK=255.255.255.0 + ONBOOT=yes

- 9) [S12] This step should only be performed on SLES12.

Add the following to the bottom of the existing interface configuration file to define an additional address for the interface:

File: /etc/sysconfig/network/ifcfg-eth0
+ IPADDR_1=192.168.41.X + NETMASK_1=255.255.255.0

- 10) [R7] This step should only be performed on RHEL7.

Stop the NetworkManager service:

```
# systemctl stop NetworkManager
```

11) [R7] This step should only be performed on RHEL7.

Use the ifup script to bring up your new virtual interface.

```
# ifup eth0:0
# ip addr show eth0:0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP qlen 1000
        link/ether 00:1b:21:24:fa:6e brd ff:ff:ff:ff:ff:ff
        inet 10.100.0.X/24 brd 10.100.0.255 scope global eth0
            inet 192.168.41.X/24 brd 192.168.41.255 scope global eth0:0
```

- Verify that the new virtual interface is seen in the output.

12) [S12] This step should only be performed on SLES12.

Use the ifup script to bring up the network interface with the new alias:

```
# ifdown eth0; ifup eth0
# ip addr show eth0
. . . output omitted . . .
```

- Verify that the new IP address is seen in the output.

13) If available in your lab environment, test the new address from another system:

```
# ping -c 1 192.168.41.Y
. . . output omitted . . .
```

Cleanup:

14) [R7] This step should only be performed on RHEL7.

Remove the additional interface and address to avoid conflicts with future lab tasks:

```
# rm -f /etc/sysconfig/network-scripts/ifcfg-eth0:0
```

15) [S12] This step should only be performed on SLES12.

Remove the interface configuration to prevent conflicts with future lab tasks:

File: /etc/sysconfig/network/ifcfg-eth0

- IPADDR_1=192.168.41.X
- NETMASK=255.255.255.0

- 16)** [R7] *This step should only be performed on RHEL7.*

Restart the network sub-system and verify connectivity to the classroom systems.

```
# systemctl start NetworkManager  
# systemctl restart network
```

- 17)** [S12] *This step should only be performed on SLES12.*

Restart the network sub-system and verify that you have connectivity to the classroom systems.

```
# systemctl restart network
```

Objectives

- » Verify Link-Local IPv6 Connectivity
- » Configure and Test IPv4 Mapped Address Connectivity

Requirements

- (1 station)

Relevance

An increasing number of systems are adopting IPv6.

Notices

- » Run ip link. Verify the interface name of the local ethernet or wireless interface, (e.g. p3p1, em1). The lab task assumes the device eth0.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) List the addresses currently assigned to your eth0 interface. Record your EUI, which is the last 64bits of the IPv6 assigned to your network interface. In this case it is everything after fe80:: (excluding the /64):

```
# ip addr show eth0  
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000  
    link/ether 00:01:03:de:57:af brd ff:ff:ff:ff:ff:ff  
    inet 10.100.0.X/24 brd 10.100.0.255 scope global eth0  
        inet6 fe80::XXX:XXX:XXXX:XXXX/64 scope link valid_lft forever preferred_lft forever
```

Result: _____

- 3) Ping your IPv6 loopback address to test basic connectivity:

```
# ping6 -c 3 ::1  
PING ::1(::1) from ::1 : 56 data bytes  
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.042 ms  
64 bytes from ::1: icmp_seq=2 ttl=64 time=0.043 ms  
64 bytes from ::1: icmp_seq=3 ttl=64 time=0.044 ms  
--- ::1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
```

Lab 12

Task 2

Configuring IPv6

Estimated Time: 10 minutes

rtt min/avg/max/mdev = 0.042/0.043/0.044/0.000 ms

- 4) Connect via the ssh command to localhost:

```
# ssh -6 guru@::1
The authenticity of host '::1 (::1)' can't be established.
RSA key fingerprint is 75:77:46:2c:ef:23:c3:a2:fb:3e:90:93:61:c7:4e:a7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '::1' (RSA) to the list of known hosts.
guru@::1's password: work 
. . . output omitted . . .
$ exit
```

- 5) Associate an IPv4 mapped IPv6 address on the *eth0* interface. Use the prefix of ffff with the same EUI address discovered earlier:

```
# ip addr add ::ffff:10.100.0.X/96 dev eth0
```

- 6) Verify that the new address has been correctly associated with the interface by listing, and then pinging the address:

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,NOTRAILERS,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:01:03:de:57:af brd ff:ff:ff:ff:ff:ff
    inet 10.100.0.X/24 brd 10.100.0.255 scope global eth0
        inet6 ::ffff:10.100.0.X/96 scope global
            valid_lft forever preferred_lft forever
        inet6 fe80::XXX:XXX:XXXX:XXXX/64 scope link
            valid_lft forever preferred_lft forever
# ping6 -c 3 ::ffff:10.100.0.X
PING ::ffff:10.100.0.X(::ffff:10.100.0.X) 56 data bytes
64 bytes from ::ffff:10.100.0.X icmp_seq=1 ttl=64 time=0.264 ms
64 bytes from ::ffff:10.100.0.X icmp_seq=2 ttl=64 time=0.185 ms
64 bytes from ::ffff:10.100.0.X icmp_seq=2 ttl=64 time=0.184 ms
--- ::ffff:10.100.0.X ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.184/0.211/0.264/0.037 ms, pipe 2
```

- 7) Verify that the ssh daemon is bound to your IPv6 addresses on port 22 using the netstat command:

```
# ss -tan | grep :::22
LISTEN      0      128          :::22          ::::*
```

- 8) Try connecting via the ssh command over IPv6 to your own address:

```
# ssh -6 guru@::ffff:10.100.0.X
The authenticity of host '::ffff:10.100.0.X (::ffff:10.100.0.X)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
RSA key fingerprint is ea:97:53:b6:f9:09:02:ed:17:87:67:49:f6:22:39:c9.
20298: Warning: Permanently added '::ffff:10.100.0.X' (RSA) to the list of known hosts.
guru@::1's password: work [Enter]
. . . output omitted . .
$ exit
```

Bonus

- 9) If supported by your lab environment, verify that you can connect to another system's IPv6 link local address with the ping6 command, then connect using SSH:

```
# ping6 -I eth0 -c 3 fe80::YYY:YYY:YYYY:YYYY
PING fe80::YYY:YYY:YYYY:YYYY(fe80::YYY:YYY:YYYY:YYYY) 56 data bytes
64 bytes from fe80::21b:21ff:fe25:e9: icmp_seq=1 ttl=64 time=4.79 ms
64 bytes from fe80::21b:21ff:fe25:e9: icmp_seq=2 ttl=64 time=0.225 ms
64 bytes from fe80::21b:21ff:fe25:e9: icmp_seq=3 ttl=64 time=0.140 ms

--- fe80::YYY:YYY:YYYY:YYYY%eth0 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.140/1.719/4.794/2.174 ms
# ssh -6 guru@fe80::YYY:YYY:YYYY:YYYY%eth0
The authenticity of host 'fe80::YYY:YYY:YYYY:YYYY (fe80::YYY:YYY:YYYY:YYYY)' can't be established.
RSA key fingerprint is ea:97:53:b6:f9:09:02:ed:17:87:67:49:f6:22:39:c9.
Are you sure you want to continue connecting (yes/no)? yes
20298: Warning: Permanently added 'fe80::YYY:YYY:YYYY:YYYY' (RSA) to the list of known hosts.
guru@fe80::YYY:YYY:YYYY:YYYY's password:work [Enter]
. . . output omitted . . .
```

- The -I option is required under IPv6, otherwise you will get the error connect: Invalid argument.

- 10) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

❖ Practice troubleshooting common system errors.

Requirements

☒ (1 station)

Relevance

Troubleshooting scenario scripts were installed on your system as part of the classroom setup process. You use these scripts to break your system in controlled ways, and then you troubleshoot the problem and fix the system.

- 1) Use tsmenu to complete the networking troubleshooting scenario:

Troubleshooting Group	Scenario Category	Scenario Name
Group 2	Networking	networking-01.sh

Lab 12

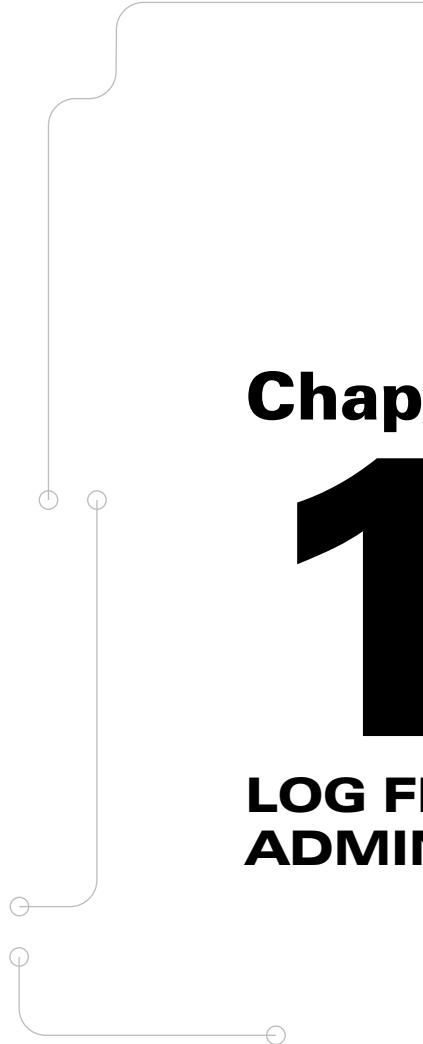
Task 3

Troubleshooting Practice: Networking

Estimated Time: 10 minutes

Content

System Logging	2
systemd Journal	4
systemd Journal's journalctl	6
Secure Logging with Journal's Log Sealing	8
gnome-system-log	10
Rsyslog	11
/etc/rsyslog.conf	12
Log Management	15
Log Anomaly Detector	16
Sending logs from the shell	17
Lab Tasks	18
1. Using the systemd Journal	19
2. Setting up a Full Debug Logfile	24
3. Remote Syslog Configuration	26
4. Remote Rsyslog TLS Configuration	32



Chapter

13

LOG FILE ADMINISTRATION

System Logging

Sysklogd

- Standard through RHEL5
- **syslogd** and **klogd**

Rsyslog

- Enhanced drop-in replacement for sysklogd
- **rsyslogd**

systemd Journal

- Wide adoption in modern Linux distributions
- Often used in combination with Rsyslog

Some daemons bypass syslog

- Typically log directly into a subdirectory under /var/log/
- Apache and Samba are examples

Logging Daemon Implementations

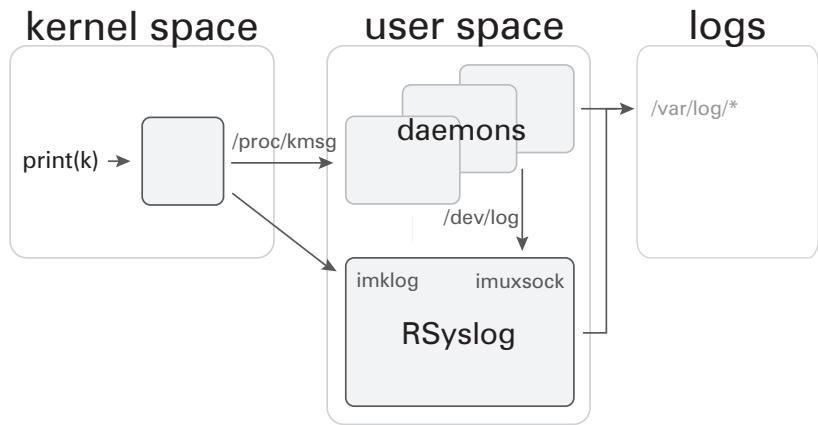
The original Unix system log daemon was written for Sendmail and was a standard part of Berkeley Unix (BSD). An enhancement called Sysklogd has been dominant on Linux systems until recent years. This provided **syslogd**, and a separate **klogd** for interpreting kernel messages, and was configured in `/etc/syslog.conf`. It has been replaced on some systems with Syslog-NG. However, enterprise distributions now ship Rsyslog, a drop in replacement for Sysklogd.

With the adoption of systemd, the primary logging services have shifted from being provided by a Syslog implementation to the systemd Journal. The new journal can operate independently, or can be configured to selectively pass messages to an Rsyslog process. Rsyslog can in turn be configured to selectively pass messages (for example those from a remote host) back into the journal.

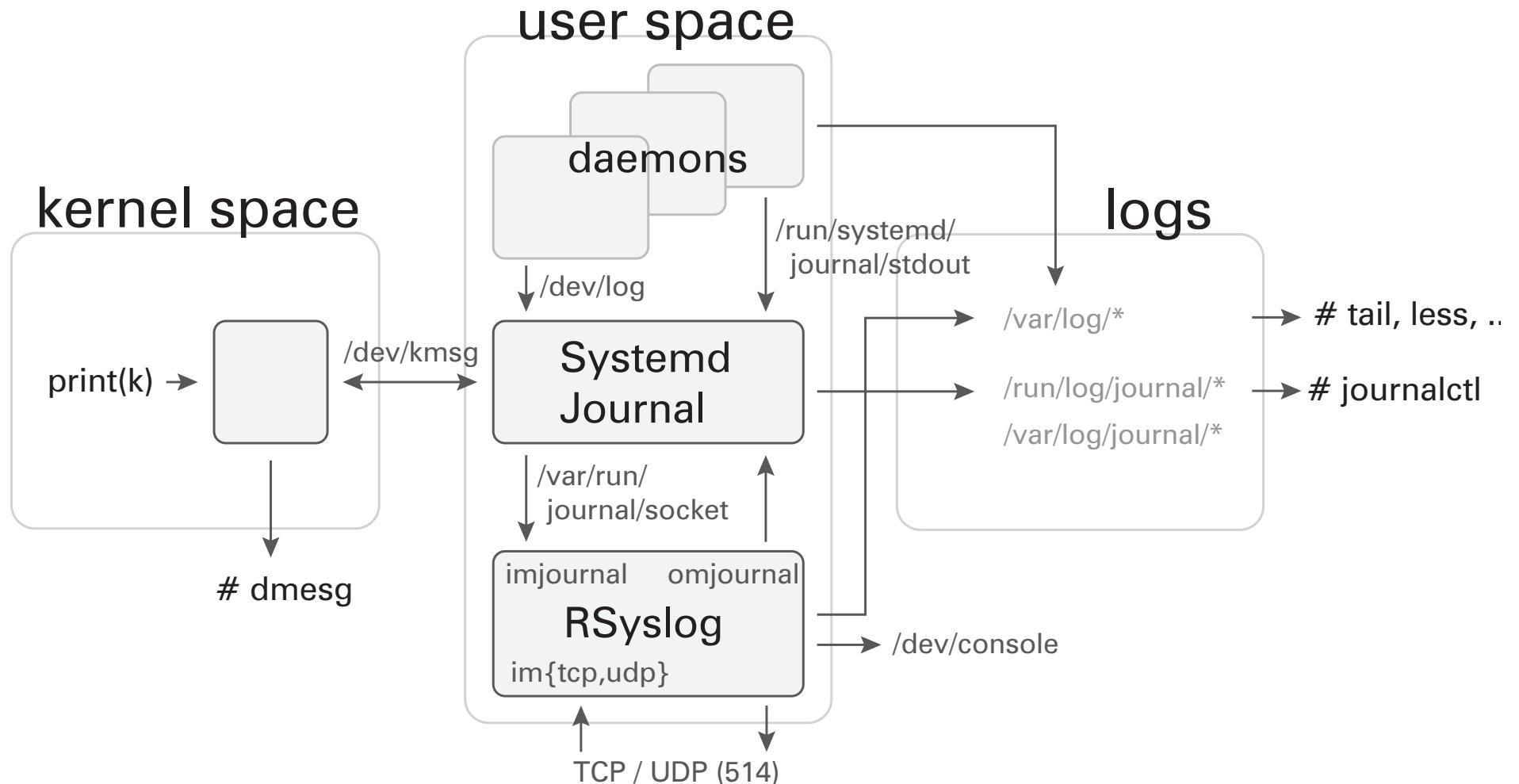
Logging Architecture

The following illustrations show the flow of log data between the various system components for a system using just Rsyslog and the newer architecture introduced by running both the systemd Journal and Rsyslog together as found on modern Linux distributions.

Traditional Logging Architecture



New Logging Architecture Incorporating the systemd Journal



systemd Journal

Journal adds many new features, and complements syslog

- Forwards to Syslog by default in systemd version less than 216

Several different approaches to integrating rsyslog and the journal are available

Volatile storage by default, persistent storage possible

Configurable in /etc/systemd/journald.conf

Journal Features

The traditional Syslog style logging daemons had many deficiencies. Messages generated before the logging daemon was started (early boot and initial RAM disk processes) were not captured. Messages written to STDOUT or STDERR were not captured. Logs were unstructured, unindexed text. The new systemd Journal addresses these issues and also adds several other useful features such as:

- ❖ Can collect messages from: early boot, kernel, and user processes (STDOUT, STDERR, or native Journal API)
- ❖ Logs are structured binary data with extensive meta-data
- ❖ Logs are indexed on all fields allowing for rapid filtering and retrieval.
- ❖ Built in log rotation.
- ❖ Supports cryptographic sealing and verification of logs for integrity.
- ❖ Includes powerful command for accessing log data (**journalctl**).
- ❖ Integrates well with Rsyslog allowing bi-directional passing of log messages between the services. Can also be used as a replacement to Rsyslog.

Enabling Persistent Storage for Journal

By default, the journal logs data to the `/run/log/journal/_MACHINE_ID` directory which is a tmpfs filesystem and subsequently volatile. The location and method of storage is controlled via the `Storage=` option in the

`/etc/systemd/journald.conf` file. The default setting of `Storage=auto` will automatically begin storing log data in the `/var/log/journal` directory if that directory exists.

```
# mkdir -p /var/log/journal/  
# systemctl restart systemd-journald
```

Once persistent log storage is enabled, settings in the `journald.conf` control the total amount of space the logs are allowed to consume, and the rotation intervals. See the man page `journald.conf(5)` for details. The total space in use by journal can be seen by running:

```
# journalctl --disk-usage  
Journals take up 712.3M on disk.
```

Free up disk space being used by the journal immediately by running:

```
# journalctl --vacuum-size=250M  
. . . output omitted . . .  
# journalctl --disk-usage  
Journals take up 246.6M on disk.
```

rsyslog and Journal Integration

Two different approaches can be used for Syslog integration. With early versions of systemd, the journal was configured to forward messages to **rsyslog** in a push architecture and the journal by default used the `ForwardToSyslog=yes` setting. Currently the more efficient pull architecture is used, and **rsyslog** uses `imjournal` to pull messages from the journal in a structured format, and not access the journal's unstructured output socket by using this configuration:

File: /etc/rsyslog.conf

```
$ModLoad imjournal # provides access to the systemd journal  
$OmitLocalLogging on # don't use the journal's output socket
```

In the recommended configuration, the systemd Journal is what receives the syslog messages from local applications by listening to the /dev/log file. This is configured in the [Socket] section of the /usr/lib/systemd/system/systemd-journald.socket. Then as previously noted, **rsyslog** pulls the messages from the journal using the imjournal input module.

File: /usr/lib/systemd/system/systemd-journald.socket

```
[Socket]  
ListenStream=/run/systemd/journal/stdout  
ListenDatagram=/run/systemd/journal/socket  
ListenDatagram=/dev/log  
SocketMode=0666  
PassCredentials=yes  
PassSecurity=yes  
ReceiveBuffer=8M
```

This also means that installing **rsyslog** is optional, and a pure journal configuration will work transparently for the syslog using applications on the box.

Receiving network syslog messages into the journal

One feature the journal doesn't have is the ability to receive standard syslog messages over the network. For this type of configuration, setup **rsyslog** to receive the messages and then forward the messages to the journal using the omjournal output module. For example:

File: /etc/rsyslog.conf

```
module(load="imudp") # input module for UDP syslog  
module(load="omjournal") # output module for journal  
input(type="imudp" port="514" ruleset="writeToJournal")  
ruleset(name="writeToJournal") {action(type="omjournal")}
```

systemd Journal's journalctl

systemd Journal is an indexed binary database

journalctl used to query database

- Without any parameters, shows full journal contents, oldest first
- Advanced filtering available
- Uses **less** to display information
 - No wrapping by default, use arrow keys to scroll horizontally
 - Set **SYSTEMD_LESS** environment variable to control **less** behavior

Accessing Logs with journalctl

The **journalctl** command is used to access log data stored by the journal. If called without any options it will display all log data automatically piped to the **less** pager wth the oldest log entries at the top. Remember that **less** allows for both vertical and horizontal paging, and that long log lines can be read by scrolling to the right or left as needed with the arrow keys.

In order to wrap long lines, set the **SYSTEMD_LESS** environment variable to **FRXMK** which omits the S that is normally used by default. For example:

```
# SYSTEMD_LESS=FRXMK journalctl
```

Normal users will only have access to their own user log data (if any). Members of the **adm** group or the root user have full access. The output of **journalctl** is similar to traditional Syslog format but is visually styled to highlight important data, and time-stamps apply time-zone offset to show the local time. Powerful filtering options are

Show logs, in reverse chronological order for a specific systemd unit:

```
# journalctl -r -u libvиртd
-- Logs begin at Fri 2013-07-26 15:31:10 MDT, end at Mon 2015-04-20 13:05:12 MDT. --
Dec 10 08:34:05 localhost dnsmasq[3589]: started, version 2.68 cachesize 150
Dec 10 08:33:59 localhost systemd[1]: Started Virtualization daemon.
. . . snip . . .
```

supported to specify exactly what log data is desired (see man **journalctl(1)** for details). Examples of useful options include:

- n *num* ⇒ Show the specified number of log events
- r ⇒ Reverse output showing most recent events at the top
- o *output_format* ⇒ Almost a dozen output formats ranging from the most minimal **cat**, to verbose which show all meta-data fields.
- f ⇒ Shows most recent log events and continues to display new entries as they are generated (similar to **tail -f logfile**)

Filtering Logs

One of the major advantages of the journal is its ability to quickly filter logs based on any of the indexed meta-data fields. This allows an administrator to quickly extract the relevant log data without resorting to extensive error-prone **grep** pipelines. The following examples show some of the filtering methods (see man **journalctl(1)** for more details).

Show details of how many boots are stored in the logs and what periods of time they cover. Then display logs for only the previous boot:

```
# journalctl --list-boots | head -n 2
-90 afc3137b82ad4650b95c231b62fbb636 Wed 2014-06-18 12:07:03 MDT Thu 2014-06-19 18:20:43 MDT
-89 afc3137b82ad4650b95c231b62fbb636 Wed 2014-06-18 12:07:03 MDT Thu 2014-06-19 18:20:43 MDT
# journalctl -b -1
. . . output omitted . . .
```

Display log entries that have collected since the most recent boot:

```
# journalctl -b
. . . output omitted . . .
```

Show logs only for events with priority levels of emergency, alert, critical, or error, and were generated by the **sshd** command:

```
# journalctl -p 0..3 _COMM=sshd
-- Logs begin at Wed 2014-06-18 12:07:03 MDT, end at Wed 2014-10-15 15:13:31 MDT. --
Oct 14 16:28:08 dhcp93.hq.gurulabs.com sshd[24936]: error: Couldn't create pid file "/var/run/sshd.pid": Permission denied
```

Show all possible values that can be used with the **_COMM** match based on what is in the journal:

```
# journalctl -F _COMM
sudo
pkexec
. . . snip . . .
```

Show all possible match fields that can be used:

```
# journalctl Tab Tab
_AUDIT_LOGINUID=           ERRNO=          _PID=          _SYSTEMD_SESSION=
. . . snip . . .
```

Show logs for the specified time period only for a specific systemd unit:

```
# journalctl --since "2014-7-20" --until "2014-7-22" -u crond
-- Logs begin at Wed 2014-06-18 12:07:03 MDT, end at Wed 2014-10-15 15:39:36 MDT. --
Jul 20 10:58:54 localhost systemd[1]: Started Command Scheduler.
Jul 20 10:58:54 localhost crond[636]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 60% if used.)
```

Show the last 20 log events for the graphical session of a specified user:

```
# journalctl -r -n 20 _UID=50002 _COMM=gnome-session
-- Logs begin at Wed 2014-06-18 12:07:03 MDT, end at Wed 2014-10-15 17:44:41 MDT. --
Oct 15 17:32:39 localhost gnome-session[1596]: ,["alt4.stun.l.google.com","19302"]
Oct 15 17:32:39 localhost gnome-session[1596]: [19883:218] 0x3d758dca5a0: C->F: ["jf",[[{"stun.l.google.com","19302"}]
. . . snip . . .
```

Secure Logging with Journal's Log Sealing

Attackers typically modify log files to cover their tracks

Forward Secure Sealing (FSS) enables detection of log modifications

- Uses Seekable Sequential Key Generators cryptographic primitive
- Most recent 15 minutes not protected by default Configurable interval
- Still consider remote logging FSS serves as a red flag only

Detecting Log Modification with Log Sealing

In the aftermath of system security compromise, logs become an essential resource in determining when and how the system security was breached. However, if an attacker obtained root level access, then any data on the system could have been modified and subsequently locally stored logs can not be trusted. The most common solution to this problem is for a copy of all log data to be forwarded to a secure remote system. However, the additional

```
# journalctl --setup-keys --interval=30m
Generating seed...
Generating key pair...
Generating sealing key...
```

The new key pair has been generated. The secret sealing key has been written to the following local file. This key file is automatically updated when the sealing key is advanced. It should not be used on multiple hosts.

/var/log/journal/3495c675d461450ba68decb774b69083/fss

Please write down the following secret verification key. It should be stored at a safe location and should not be saved locally on disk.

d8c8a7-149a6b-86c299-b5ae20/bfb78-6b49d200

The sealing key is automatically changed every 30min.

The keys have been generated for host nazgul.gurulabs.com/3495c675d461450ba68decb774b69083.

To verify the logs later run the following:

complexity and cost associated with remote logs is not always acceptable.

The systemd Journal allows the local logs to be periodically "sealed" in such a way that they can later be verified for integrity and an alert produced if modifications are detected. To initiate log sealing, first determine the sealing interval (default 15min). Then create the key pair (sealing and verification):

```
# journalctl --verify --verify-key=d8c8a7-149a6b-86c299-b5ae20/bfb78-6b49d200
PASS: /var/log/journal/3495c675d461450ba68decb774b69083/system@0005043471fa28e1-5f0c9b107b59e7d5.journal-
PASS: /var/log/journal/3495c675d461450ba68decb774b69083/user-42@0004ffb075e0b7dd-27ea58729668e9a1.journal-
PASS: /var/log/journal/3495c675d461450ba68decb774b69083/user-990@00050458dd685421-a4e6597e9fe35c31.journal-
1c47f18: invalid entry item (11/23 offset: 00000███████████ 37%
Invalid object contents at 1c47f18: Bad message
File corruption detected at /var/log/journal/3495c675d461450ba68decb774b69083/system@00050220578498b1-fdb8ec9328bb37ff.→
journal-:1c47f18 (of 33554432 bytes, 88%).
FAIL: /var/log/journal/3495c675d461450ba68decb774b69083/system@00050220578498b1-fdb8ec9328bb37ff.journal- (Bad message)
PASS: /var/log/journal/3495c675d461450ba68decb774b69083/user-50002@0004fdcc7c35bb97-0e133add9a9fcba9.journal-
```

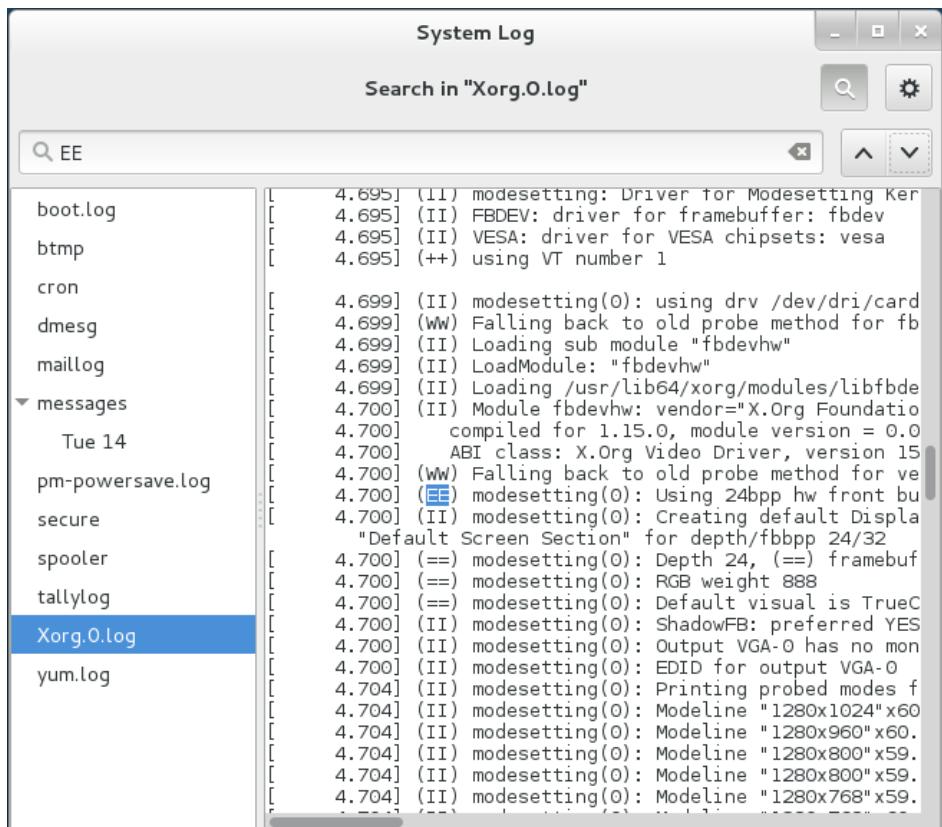
gnome-system-log

Graphical tool for examining current log files

- search tool
- can open rotated logs for viewing
- can open compressed logs

gnome-system-log

Newly introduced in RHEL7, the **gnome-system-log** graphical program can be used to view log contents. By default, it automatically opens the current file for the most common system logs. It requires root level access and will prompt for credentials if run by an unprivileged user. Additional logs can be opened, including compressed logs. A search tool is provided that uses regular expressions to search for the specified pattern. Search patterns can be stored for later use:



Rsyslog

Drop-in replacement for Sysklogd

IPv6 support

Reliable logging

- Logging over TCP
- Server failover

Precise logging

- Enhanced filtering
- Precise timestamps
- Definition of message output formats

Many Input and Output Modules

- Ability to log to SQL or NoSQL databases

TLS encryption

Precise Logging

Rsyslog also provides several enhancements over traditional logging implementations with regards to logging precision. For example, it's possible to filter messages on any part of a log message rather than the priority of the message and the originating facility. Among other things, this allows for per-host log file creation on the logging server. Additionally, it's possible (using templates) to choose the exact logging format of messages created by Rsyslog, including support for more precise timestamps than are used with standard Syslog log messages.

Additional Rsyslog Features

Rsyslog provides many additional features which are unavailable with Sysklogd, such as support for TLS encryption, advanced matching and filtering of messages (such as by content), and the ability to log to SQL databases. For a full list of available features, as well as a side-by-side feature comparison of Rsyslog and Syslog-ng, visit http://www.rsyslog.com/doc-rsyslog_ng_comparison.html.

Rsyslog

Rsyslog is a logging daemon intended to compete with Syslog-ng. When Rainer Gerhards, the primary author of Rsyslog, began development of his logging daemon, Syslog-ng was already well-established as a logging daemon boasting many features lacking from the ubiquitous Sysklogd. When describing his motivation for creating yet another alternative to Sysklogd, he explained his goal to use his creation to "prevent monocultures and provide a rich freedom of choice." Rsyslog has since become the default logging daemon for most Linux distributions. In order to ease the transition from Sysklogd to Rsyslog, the existing configurations will continue to work when using Rsyslog, after which any Rsyslog specific enhancements may be added.

Reliable Network Logging

Rather than being limited to UDP, like most logging daemons, Rsyslog supports TCP over IPv4 or IPv6. TCP is more reliable than UDP, because of its acknowledgment and retransmission capabilities. Another way Rsyslog can increase reliability is through the use of fallback logging destinations. When `rsyslogd` is unable to log to a particular destination, one or more additional destinations, either hosts or files, may be specified for message delivery.

/etc/rsyslog.conf

Syntax

- Rules consist of selector field(s) and action field separated by spaces or tabs
 - Multiple selectors per line, separated by semicolons
 - Only one action per line

Selector

- The selector consists of a facility and a priority, separated by a period, such as mail.info
 - An asterisk (*) stands for all facilities or all priorities

Action

- Can log to: file (including devices), FIFO, terminal, remote machine, specific users, all users

rsyslog.conf

All system and kernel messages get passed to **rsyslogd**. For every log message received, Rsyslog looks at its /etc/rsyslog.conf configuration file to determine how to handle that message. Rsyslog looks through the configuration file for all rule statements which match that message, and handles the message as each rule statement dictates. If no rule statement matches the message, Rsyslog discards it. Rule statements specify two things: what messages to match (selectors), and what to do with matched messages (actions).

Selectors

Messages to match are specified by a selector which matches facilities and priorities, while actions to apply to matched messages are specified by an action field. For example, the following configuration line tells Rsyslog to apply the action /var/log/kernlog to all messages with a facility of kern and a level of debug:

File: /etc/rsyslog.conf

+ kern.debug	/var/log/kernlog
--------------	------------------

Priority statements in selectors are hierarchical. Rsyslog will match all messages of the specified priority and higher. The selector kern.debug matches all messages produced by the kernel with priority debug or higher; since debug is the lowest possible priority, the selector kern.debug matches all messages with a kern facility. In addition, an asterisk can be used as a wild card to represent all

priorities, so kern.* would also match all messages produced by the kernel.

Unlike the priority field, the facility field is not hierarchical. It is still possible to match multiple messages from different facilities, however. Multiple selectors can be listed on a line, separated by semicolons. This can be useful when the same action needs to be applied to multiple messages. Similarly, the asterisk wild-card can be used to specify all facilities, providing another method for applying an action to a variety of messages.

Actions

Many actions are possible, though only one can be included in a rule:

- File names can be listed in the action field, specifying the location of files to which the selected message should be written. These files can be text files, as is usually the case, but they can also be device files such as a terminal or a printer.
- User names can also be specified. If the named user is logged into the system when Rsyslog processes the message, the message will be printed to all of that user's terminals.
- An asterisk for the action tells Rsyslog to write the message to all logged-in users (it goes to all active terminals).
- Messages can be sent to remote hosts. The action @host tells Rsyslog to forward the message to the machine *host*, where it will be processed again by that *host*'s Syslog daemon. Use @@(o)*host* for sending over TCP.

Syslog Facilities and Priorities

The facility is used to specify what type of program is generating the message. The Syslog daemon can then be configured to handle messages from different sources differently. For instance, Red Hat Enterprise Linux uses `/var/log/secure` for its authpriv facility and `/var/log/maillog` for email. Both Debian and SUSE separate its mail facility into several log files by priority, with a second all-priorities file (`/var/log/mail.log` on Debian, and `/var/log/mail` on SUSE). Debian uses `/var/log/daemon.log` for its daemon facility. Usually, those not separated go to `/var/log/messages`, or perhaps `/var/log/syslog`. This table lists the standard defined facilities with brief descriptions of what they are used for:

Facility	Description
auth/authpriv	security/authorization messages
cron	cron and atd daemons messages
daemon	other system daemons
kern	kernel messages
local0 – local7	reserved for local use
lpr	line printer subsystem
mail	mail subsystem
news	Usenet news subsystem
syslog	messages generated internally by the system log daemon
user	generic user-level messages
uucp	UUCP subsystem

The priority, or level, of a message is intended to determine the importance of a message. This table lists the standard priority levels with brief descriptions of their meanings:

Priority	Description
emerg	system is unusable
alert	action must be taken immediately
crit	critical conditions
err	error conditions
warning	warning conditions
notice	normal, but significant, condition
info	informational messages
debug	debugging messages

Enabling Remote Logging

The Rsyslog daemon is not configured to listen on the network by default. Make the following edit to enable remote logging:

[R7] File: /etc/rsyslog.conf

[S12] File: /etc/rsyslog.d/remote.conf

```
- #$ModLoad imudp.so
- #$UDPServerRun 514
+ $ModLoad imudp.so
+ $UDPServerRun 514
```

Enabling TCP/TLS Remote Logging

First, to support TLS logging with Rsyslog requires TCP instead of UDP. Second, the appropriate certificates must be generated using the **certtool** command, as Rsyslog uses GnuTLS not OpenSSL. Make the following edit to enable TCP remote logging:

File: /etc/rsyslog.conf

File: /etc/rsyslog.d/remote.conf

```
- #$ModLoad imtcp.so
- #$InputTCPServerRun 514
+ $ModLoad imtcp.so
+ $InputTCPServerRun 514
```

Though port 514 can be used, it may be more appropriate to use port 6514. The following is an example of a logging server configuration for TLS (directive order matters):

[R7] File: /etc/rsyslog.conf

[S12] File: /etc/rsyslog.d/remote.conf

```
- $ModLoad imtcp
+ $ModLoad imtcp
+ $DefaultNetstreamDriver gtls
+ $DefaultNetstreamDriverCAFfile /etc/ssl/CA.pem
+ $DefaultNetstreamDriverCertFile /etc/ssl/certs/rsyslog.crt
+ $DefaultNetstreamDriverKeyFile /etc/ssl/rsyslog.key
+
+ $InputTCPServerStreamDriverMode 1
+ $InputTCPServerStreamDriverAuthMode anon
- $InputTCPServerRun 514
+ $InputTCPServerRun 6514
```

The NetstreamDriver entries must be in order *preceding* the ServerMode configuration entries. The following is an example of a corresponding logging client configuration for TLS:

[R7] File: /etc/rsyslog.conf

[S12] File: /etc/rsyslog.d/remote.conf

```
+ $DefaultNetstreamDriverCAFfile /etc/ssl/CA.pem
+ $DefaultNetstreamDriver gtls
+ $ActionSendStreamDriverMode 1
+ $ActionSendStreamDriverAuthMode anon
```

[R7] *The following applies to RHEL7 only:*

The rsyslog-gnutls package is needed for supporting the gtls NetstreamDriver configuration.

[S12] *The following applies to SLES12 only:*

The rsyslog-module-gtls package is needed for supporting the gtls NetstreamDriver configuration.

Log Management

- /var/log/journal/
 - Persistent location for Systemd Journal
 - /var/log/*ApplicationName*/
 - Sub-directories for applications that directly log
 - /var/log/
 - Standard location for Syslog log files
- logrotate**
- Log rotation utility for Syslog and direct Application logs
 - Allows automatic rotation, compression, removal and processing of log files
 - Runs daily from /etc/cron.daily/logrotate
 - Uses /etc/logrotate.conf and /etc/logrotate.d/

The /etc/logrotate.d/ directory makes it easy for RPMs to install and remove configuration files on the system specifying how the log files produced by the software in that RPM should be managed.

Main logrotate Configuration File

The /etc/logrotate.conf file mainly acts to set defaults and then includes configuration files from the /etc/logrotate.d/ directory.

```
File: /etc/logrotate.conf
weekly
rotate 4
create
dateext
include /etc/logrotate.d
/var/log/wtmp {
    monthly
    create 0664 root utmp
        minsize 1M
    rotate 1
}
/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}
```

Log Management

Log files grow regularly over time and must be trimmed, or rotated, to prevent total disk space consumption. However, most daemons complain if their log files are rotated while a log file is open for writing. Normally, rotation of log files requires that daemons be stopped, log files be moved, new log files be created, and then daemons be re-started.

Log rotation can be tedious and repetitive. Linux provides utilities which can help with both of these chores. The **logrotate** program can be used to automate log file rotation.

The Systemd Journal has automatic disk space control. By default it will use a maximum of 10% of the filesystem and keep a minimum of 15% freespace. These are set in the /etc/systemd/journald.conf.

logrotate

The system ships with **logrotate** configured to manage system logs. Basic **logrotate** configuration is done in the /etc/logrotate.conf file. In this file, general options can be set like how frequently log files should be rotated, how many old log files should be kept, and whether or not old log files should be compressed. In addition, all the configuration files in /etc/logrotate.d/ are also interpreted by **logrotate**. Files in this directory configure **logrotate** to manage specific services, by telling it information like what log files generated by that service need to be rotated, how the service should be restarted if it needs to be restarted after log file rotation, and so forth.

Log Anomaly Detector

Day-to-day information log messages considered unimportant
logwatch

- Reports/acts upon anomalous messages

logwatch

System logs need to be analyzed periodically to ensure system security, or to prepare usage reports. Log analysis can be tedious, repetitive, and a time-consuming task.

Linux distributions usually ship Kirk Bauer's **logwatch** package.

logwatch is an easily customized suite of software for monitoring and analyzing system logs to identify interesting messages. **logwatch** can analyze log files from most popular services, and it can be configured to do so automatically on a regular basis, emailing the administrator any results. Commonly, it is configured to monitor log files on an hourly or nightly basis for suspicious activity on the system.

The default configuration invokes **logwatch** from a script placed in /etc/cron.daily/ and emails a report to the root user.

Sending logs from the shell

logger

Options include

- **-f**: provide a file for the log message
- **-n**: send the log to the specified remote server
- **-p**: set the log priority
- **-s**: print the message to STDOUT in addition to the system log

logger

The **logger** command takes a string of text as its arguments and sends the string to the system log. If a string is not provided as arguments, it reads its message from STDIN. (It can also read its message from a file using the **-f** flag.) For example:

```
$ logger hello, world
$ sudo tail -n 1 /var/log/messages
Apr  2 11:57:18 stationX guru: hello, world
```

The following is a usage summary for the **logger** command:

```
usage: logger [-dhisTV] [-f file] [-n server] [-P port]
              [-p priority] [-t tag] [-u socket] [message string]
```

Lab 13

Estimated Time:
S12: 65 minutes
R7: 65 minutes

Task 1: Using the systemd Journal

Page: 13-19 Time: 20 minutes

Requirements:  (1 station)

Task 2: Setting up a Full Debug Logfile

Page: 13-24 Time: 5 minutes

Requirements:  (1 station)

Task 3: Remote Syslog Configuration

Page: 13-26 Time: 15 minutes

Requirements:   (2 stations)

Task 4: Remote Rsyslog TLS Configuration

Page: 13-32 Time: 25 minutes

Requirements:   (2 stations)

Objectives

- ❖ Configure the Journal for persistent logging
- ❖ Search the Journal for log events

Requirements

- ▀ (1 station)

Relevance

The systemd Journal is the new primary facility for logs.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Write a message directly into the kernel message buffer and then show how it flows through to the Journal and Syslog:

```
# echo "kernel message test" > /dev/kmsg  
# dmesg | tail -n 1  
[ 174.828302] kernel message test  
# journalctl -k -r --no-pager  
-- Logs begin at Thu 2014-10-16 15:23:11 MDT, ↵  
end at Thu 2014-10-16 15:26:05 MDT. --  
Oct 16 15:26:05 stationX.example.com : kernel message test  
[R7] # grep 'kernel message' /var/log/messages  
[R7] Oct 16 15:26:05 stationX journal: kernel message test
```

Lab 13

Task 1

Using the systemd Journal

Estimated Time: 20 minutes

• -k = show kernel messages, -r = reverse sort (newest first), -n 1 = number of messages

- 3) Write a message to the traditional Syslog socket and then show it flows through the Journal and Syslog:

```
# logger "Rsyslog message test"  
# journalctl -r _COMM=logger --no-pager  
-- Logs begin at Thu 2014-10-16 15:23:11 MDT, end at Thu 2014-10-16 15:32:02 MDT. --  
Oct 16 15:32:02 stationX.example.com root[3429]: Rsyslog message test  
# grep Rsyslog /var/log/messages  
Oct 16 15:32:02 stationX root: Rsyslog message test
```

- 4) Discover how much space the current journal logs take and where they are stored:

```
# journalctl --disk-usage  
Journals take up 4.6M on disk.  
# ls -lh /run/log/journal/*  
total 4.7M  
-rw-r-----. 1 root root 4.7M Oct 16 15:40 system.journal
```

- 5) Enable persistent logging by creating the needed storage directory and verify that the logs are moved to that location:

```
# mkdir /var/log/journal  
# systemctl restart systemd-journald  
# ls -l /run/log/  
total 0  
# ls -lh /var/log/journal/*  
total 8.1M  
-rw-r-----. 1 root root 8.0M Oct 16 15:50 system.journal
```

- 6) List all the meta-data for the new persistent Journal:

```
# journalctl --header  
File Path: /var/log/journal/34957ea60a9a6468c4bd3e01bc0373f7/system.journal  
File ID: d36ce064960345599a470dd0c0cee346  
Machine ID: 34957ea60a9a6468c4bd3e01bc0373f7  
Boot ID: 73f5c088fec243acb588bec845c604c8  
Sequential Number ID: d36ce064960345599a470dd0c0cee346  
State: ONLINE  
Compatible Flags:  
Incompatible Flags: COMPRESSED  
Header size: 240  
Arena size: 8388368  
Data Hash Table Size: 46378  
Field Hash Table Size: 333  
Rotate Suggested: no  
Head Sequential Number: 1  
Tail Sequential Number: 1190  
Head Realtime Timestamp: Thu 2014-10-16 15:23:11 MDT  
Tail Realtime Timestamp: Thu 2014-10-16 15:55:02 MDT
```

Tail Monotonic Timestamp: 31min 50.844s
Objects: 4986
Entry Objects: 1190
Data Objects: 2815
Data Hash Table Fill: 6.1%
Field Objects: 51
Field Hash Table Fill: 15.3%
Tag Objects: 0
Entry Array Objects: 928
Disk usage: 8.0M

- 7) List the events logged by the restart of the Journal service and the move of the logs:

```
# journalctl _COMM=systemd-journal --no-pager
Oct 16 15:49:30 stationX.example.com systemd-journal[475]: Journal stopped
Oct 16 15:49:30 stationX.example.com systemd-journal[3802]: Permanent journal is using 8.0M (max 203.8M, -->
    leaving 305.7M of free 1.8G,
Oct 16 15:49:30 stationX.example.com systemd-journal[3802]: Permanent journal is using 8.0M (max 203.8M, -->
    leaving 305.7M of free 1.8G,
Oct 16 15:49:30 stationX.example.com systemd-journal[3802]: Time spent on flushing to /var is 14.301ms -->
    for 1178 entries.
Oct 16 15:49:30 stationX.example.com systemd-journal[3802]: Journal started
```

- 8) Use verbose output mode to view the details of one of the messages:

```
# journalctl MESSAGE="Journal started" -o verbose -rn1 --no-pager
-- Logs begin at Thu 2014-10-16 15:23:11 MDT, end at Thu 2014-10-16 16:10:01 MDT. --
Thu 2014-10-16 15:49:30.626723 MDT [s=d36ce064960345599a470dd0c0cee346;i=49d;b=73f5c088fec243acb588bec845c, -->
  604c8;m=5e24cd1f;t=505913924
  _PRIORITY=6
  _TRANSPORT=driver
  _UID=0
  _GID=0
  _COMM=systemd-journal
  _EXE=/usr/lib/systemd/systemd-journald
  _CMDLINE=/usr/lib/systemd/systemd-journald
  _CAP_EFFECTIVE=4402800cf
  _SYSTEMD_CGROUP=/system.slice/systemd-journald.service
  _SYSTEMD_UNIT=systemd-journald.service
```

```
_SYSTEMD_SLICE=system.slice
_BOOT_ID=73f5c088fec243acb588bec845c604c8
_MACHINE_ID=34957ea60a9a6468c4bd3e01bc0373f7
MESSAGE=Journal started
MESSAGE_ID=f77379a8490b408bbe5f6940505a777b
_SELINUX_CONTEXT=system_u:system_r:syslogd_t:s0
_HOSTNAME=stationX.example.com
_PID=3802
```

Bonus

- 9) Examine the status of the running SSH daemon. Note that the last few related Journal messages are also displayed:

```
# systemctl status sshd
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: active (running) since Tue 2014-10-28 14:08:02 MDT; 5min ago
     Process: 11485 ExecStartPre=/usr/sbin/sshd-keygen (code=exited, status=0/SUCCESS)
   Main PID: 11487 (sshd)
      CGroup: /system.slice/sshd.service
              `--11487 /usr/sbin/sshd -D

Oct 28 14:08:02 stationX.example.com systemd[1]: Started OpenSSH server daemon.
Oct 28 14:08:02 stationX.example.com sshd[11487]: Server listening on 0.0.0.0 port 22.
Oct 28 14:08:02 stationX.example.com sshd[11487]: Server listening on :: port 22.
```

- 10) Simulate the service crashing by sending an abort signal and quickly examine the service state:

```
# killall -ABRT sshd; systemctl status sshd
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: activating (auto-restart) (Result: core-dump) since Tue 2014-10-28 14:14:06 MDT; 24s ago
     Process: 11487 ExecStart=/usr/sbin/sshd -D $OPTIONS (code=dumped, signal=ABRT)
     Process: 11485 ExecStartPre=/usr/sbin/sshd-keygen (code=exited, status=0/SUCCESS)
   Main PID: 11487 (code=dumped, signal=ABRT)
 [R7] . . . snip . . .
```

11) Wait for systemd to automatically restart the service and check its status again:

```
# systemctl status sshd
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
     Active: active (running) since Tue 2014-10-28 14:14:49 MDT; 23s ago
       Process: 12803 ExecStartPre=/usr/sbin/sshd-keygen (code=exited, status=0/SUCCESS)
    Main PID: 12805 (sshd)
      CGroup: /system.slice/sshd.service
             `-12805 /usr/sbin/sshd -D

Oct 28 14:14:49 stationX.example.com systemd[1]: Started OpenSSH server daemon.
Oct 28 14:14:49 stationX.example.com sshd[12805]: Server listening on 0.0.0.0 port 22.
Oct 28 14:14:49 stationX.example.com sshd[12805]: Server listening on :: port 22.
```

- This may take a minute before sshd is started. The amount of time systemd waits before restarting this service is defined in `/usr/lib/systemd/system/sshd.service`, or if `undefined_stale_timeout` is found in `/etc/systemd/system.conf`.

12) Examine all the log messages for that unit to see the complete record of events:

```
# journalctl -u sshd --no-pager
. . . output omitted . . .
```

13) [R7] *This step should only be performed on RHEL7.*

Examine the information collected by the abrt service related to the service crash:

```
# abrt-cli list
id 5a0d3474fdafa24aecbd3d5e2d71a9e661b50831
Directory:      /var/spool/abrt/ccpp-2014-10-28-14:14:06-11507
count:          1
executable:     /usr/sbin/sshd
package:        openssh-server-6.4p1-8.el7
time:           Tue Oct 28 14:14:06 2014
uid:            0

# ls -l /var/spool/abrt/ccpp*
# cat /var/spool/abrt/ccpp*/var_log_messages
. . . output omitted . . .
```

Objectives

Configure syslog to log debug messages

Requirements

1 station

Relevance

Setting up a debug log file can help troubleshoot system issues

- 1) The following actions require administrative privileges. Switch to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Configure Rsyslog to log all debug messages to the file /var/log/debug. Add the following file to the /etc/rsyslog.d/ directory:

File: /etc/rsyslog.d/debug.conf
+ # Log anything (excluding mail/authpriv/cron) of debug level or higher.
+ *.debug;mail,authpriv,cron.none /var/log/debug

- 3) Reload the log daemon so the changes go into effect:

```
# systemctl restart rsyslog
```

- 4) Test the syslog configuration by sending a test log message of info priority:

```
# logger -p info -t info-lvl-msg "This is an info-priority message"
```

- 5) Check to see which file the test message was logged to:

```
# tail -n 2 /var/log/{debug,messages}  
==> /var/log/debug <==  
systemd: Started Session 19 of user root.  
info-lvl-msg: This is an info-priority message  
  
==> /var/log/messages <==  
systemd: Started Session 19 of user root.
```

Lab 13

Task 2

Setting up a Full Debug Logfile

Estimated Time: 5 minutes

• The info-priority messages go to both log files.

info-lvl-msg: This is an info-priority message

- 6) Test the syslog configuration by sending a test log message of debug priority:

```
# logger -p debug -t debug-lvl-msg "This is a debug-priority message"
```

- 7) Check to see which file the test message was logged to:

```
# tail -n 2 /var/log/{debug,messages}  
==> /var/log/debug <==  
info-lvl-msg: This is an info-priority message  
debug-lvl-msg: This is a debug-priority message
```

```
==> /var/log/messages <==  
systemd: Started Session 19 of user root.  
info-lvl-msg: This is an info-priority message
```

• On RHEL7, no debug priority message is logged.

SLES12 uses /var/log/messages as a catch-all for all messages, including debug and info level. As a bonus, create another entry logging only notice level messages, then send another debug level message (as per above). Note the message is not logged in the notice level file.

Cleanup

- 8) Configure Rsyslog to stop logging debug messages. Remove the Rsyslog configuration:

```
# rm -f /etc/rsyslog.d/debug.conf
```

- 9) Reload the daemon so the changes go into effect:

```
# systemctl restart rsyslog
```

- 10) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- ❖ Configure Rsyslog to listen on the network for remote log messages
- ❖ Configure Rsyslog to send messages to a remote logging server

Requirements

█ █ (2 stations)

Relevance

Centralized logging servers can help increase manageability and security of system logs.

Notices

- ❖ One system will need to act as the logging server (`systemS`), and one system will need to act as the client (`systemC`).

- 1) Find out what ports are used for remote Rsyslog logging:

```
$ grep syslog /etc/services
syslog      514/udp
. . . snip . . .
```

- 2) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso 
```

- 3) Determine whether or not Rsyslog is listening on the network:

```
# lsof -i :514
```

No output here means that Rsyslog is not currently listening on the network

Lab 13

Task 3

Remote Syslog Configuration

Estimated Time: 15 minutes

Configure Remote Logging Server

4) The following steps should only be done on the remote logging server.

5) [R7] This step should only be performed on RHEL7.

Configure Rsyslog to listen on the network for remote log messages (port 514 by default):

File: /etc/rsyslog.conf

```
# Provides UDP syslog reception
- #$ModLoad imudp
- #$UDPServerRun 514
+ $ModLoad imudp
+ $UDPServerRun 514
```

6) [S12] This step should only be performed on SLES12.

Configure Rsyslog to listen on the network for remote log messages:

File: /etc/rsyslog.d/remote.conf

```
# UDP Syslog Server:
- #$ModLoad imudp.so # provides UDP syslog reception
  ## UDPLogFile /var/log/remote.log # force to listen on this IP only,
  ##                                     # needs SYSLOG_REQUIRE_NETWORK=yes.
- #$UDPServerRun 514 # start a UDP syslog server at standard port 514
+ $ModLoad imudp.so # provides UDP syslog reception
  ## UDPLogFile /var/log/remote.log # force to listen on this IP only,
  ##                                     # needs SYSLOG_REQUIRE_NETWORK=yes.
+ $UDPServerRun 514 # start a UDP syslog server at standard port 514
```

7) Start (or restart) the syslog daemon so that changes go into effect:

```
# systemctl restart rsyslog
```

8) Determine whether or not the daemon is listening for remote network connections:

```
# lsof -i :514
```

```
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rsyslogd 5585 root 4u IPv4 9896403      0t0 UDP *:syslog
rsyslogd 5585 root 5u IPv6 9896404      0t0 UDP *:syslog
```

Configure Remote Logging Client

- 9) The following steps should only be done on the client that will send a log to the remote logging server.
- 10) [R7] This step should only be performed on RHEL7.

Configure Rsyslog to send messages to the remote logging server:

```
File: /etc/rsyslog.conf
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none          /var/log/messages
+ *.info;mail.none;authpriv.none;cron.none          @10.100.0.5
```

- 11) [S12] This step should only be performed on SLES12.

Using the /etc/rsyslog.conf file (or /etc/rsyslog.d/remote.conf, or a new file in the /etc/rsyslog.d/ directory), configure Rsyslog to send messages to the remote logging server:

```
File: /etc/rsyslog.conf
# the rest in one file
#
*.*;mail.none;news.none          -/var/log/messages
+ *.*;mail.none;news.none          @10.100.0.5
```

- 12) Restart the daemon so that the changes go into effect:

```
# systemctl restart rsyslog
```

- 13) Send a test log message:

```
# logger -p info -t info-lvl-msg "Message from stationC"
```

Confirm Results

- 14) Observe the contents of the system log on the remote logging server:

```
# grep info-lvl-msg /var/log/messages  
Apr 19 09:57:16 stationC info-lvl-msg: Message from stationC
```

- This is the message which was created on the client system.

Since messages are also logged locally, the client system's /var/log/messages log file should have a copy of this message.

Configure Remote Logging Server for TCP

- 15) The following steps should only be done on the remote logging server.

- 16) [R7] *This step should only be performed on RHEL7.*

Configure Rsyslog to listen on the network over TCP for remote log messages (port 514 by default):

File: /etc/rsyslog.conf

```
# Provides UDP syslog reception  
- $ModLoad imudp  
- $UDPServerRun 514  
+ #$ModLoad imudp  
+ #$UDPServerRun 514  
  
# Provides TCP syslog reception  
- #$ModLoad imtcp  
- #$InputTCPServerRun 514  
+ $ModLoad imtcp  
+ $InputTCPServerRun 514
```

- 17) [S12] *This step should only be performed on SLES12.*

Configure Rsyslog to listen on the network over TCP for remote log messages:

File: /etc/rsyslog.d/remote.conf

```
- #$ModLoad imtcp.so # load module
+ $ModLoad imtcp.so # load module
+ $InputTCPServerRun 514
    . . . snip . .
- $ModLoad imudp.so # provides UDP syslog reception
    ## UDPServerAddress 10.10.0.1 # force to listen on this IP only,
    ##
    ## needs SYSLOG_REQUIRE_NETWORK=yes.
- $UDPServerRun 514 # start a UDP syslog server at standard port 514
+ #$ModLoad imudp.so # provides UDP syslog reception
    ## UDPServerAddress 10.10.0.1 # force to listen on this IP only,
    ##
    ## needs SYSLOG_REQUIRE_NETWORK=yes.
+ #$UDPServerRun 514 # start a UDP syslog server at standard port 514
```

- 18) Start (or restart) the syslog daemon so that changes go into effect:

```
# systemctl restart rsyslog
```

- 19) Determine whether or not the daemon is listening for remote network connections:

```
# lsof -i :514
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rsyslogd 2225 root 4u IPv4 146256      0t0  TCP *:shell (LISTEN)
rsyslogd 2225 root 5u IPv6 146257      0t0  TCP *:shell (LISTEN)
```

Configure Remote Logging Client for TCP

- 20) The following steps should only be done on the client that will send a log to the remote logging server.

- 21) [R7] This step should only be performed on RHEL7.

Configure Rsyslog to send messages to the TCP remote logging server:

File: /etc/rsyslog.conf

```
→ *.info;mail.none;authpriv.none;cron.none          @@10.100.0.S
```

- 22) [S12] This step should only be performed on SLES12.

Configure Rsyslog to send messages to the remote logging server:

File: /etc/rsyslog.conf
→ *.*;mail.none;news.none @10.100.0.5

- 23) Restart the daemon so that the changes go into effect:

```
# systemctl restart rsyslog
```

- 24) Send a test log message:

```
# logger -p info -t info-lvl-msg "Message from stationC"
```

Confirm Results

- 25) Observe the contents of the system log on the remote logging server:

```
# grep info-lvl-msg /var/log/messages
stationC info-lvl-msg: Message from stationC
stationC info-lvl-msg: Message from stationC
```

- This is from the UDP remote log
- This is from the TCP remote log

Objectives

- Configure Rsyslog TLS remote logging configuration

Requirements

2 stations

Relevance

Logs secure from peeping eyes, and in-the-middle attacks, ensure logging data integrity and safety from those who might maliciously use the logging data.

Notices

- One system acts as the logging server (stationS), and one system acts as the client (stationC).

- On both the client and server, **install** the GnuTLS Rsyslog module.

```
[R7] [C & S]# yum install -y gnutls-utils rsyslog-gnutls  
[S12] [C & S]# zypper install -y gnutls rsyslog-module-gtls  
     . . . output omitted . . .
```

Generate Server CA signed TLS certificate

- Though normally certificates would be made with OpenSSL, the Rsyslog GnuTLS module doesn't work well with its certificates. Instead, GnuTLS provides the certtool command for generating certificates. First, create a self-signed certificate authority certificate and key. **Enter** Example CA for the "Common name", then press **Enter** for all questions other than the last. For example:

```
[S]# cd /etc/rsyslog.d/  
[S]# certtool --generate-privkey --outfile ca-key.pem  
Generating a 2432 bit RSA private key...  
[S]# certtool --generate-self-signed --load-privkey ca-key.pem --outfile ca.pem  
Generating a self signed certificate...  
Please enter the details of the certificate's distinguished name. Just press enter to ignore a field.  
Common name: Example CA  
UID:   
Organizational unit name:   
Organization name:   
Locality name:   
State or province name:   
Country name (2 chars): 
```

Lab 13

Task 4

Remote Rsyslog TLS Configuration

Estimated Time: 25 minutes

Enter the subject's domain component (DC):
This field should not be used in new certificates.
E-mail:
Enter the certificate's serial number in decimal (default: 1403294261):
Activation/Expiration time.
The certificate will expire in (days): 3650
Does the certificate belong to an authority? (y/N):
Is this a TLS web client certificate? (y/N):
Will the certificate be used for IPsec IKE operations? (y/N):
Is this a TLS web server certificate? (y/N):
Enter a dnsName of the subject of the certificate:
Enter a URI of the subject of the certificate:
Enter the IP address of the subject of the certificate:
Enter the e-mail of the subject of the certificate:
Will the certificate be used for signing (required for TLS)? (Y/n):
Will the certificate be used for encryption (not required for TLS)? (Y/n):
Enter the URI of the CRL distribution point:
... snip ...
Is the above information ok? (y/N): y

- 3) Create the private key for the Rsyslog remote server. Generate a certificate request from it, similar to the previous step:

```
[S]# certtool --generate-privkey --outfile key.pem
Generating a 2432 bit RSA private key...
[S]# certtool --generate-request --load-privkey key.pem --outfile request.pem
Generating a PKCS #10 certificate request...
Common name: Example server
Organizational unit name: 
Organization name: 
Locality name: 
State or province name: 
Country name (2 chars): 
Enter the subject's domain component (DC): 
UID: 
Enter a dnsName of the subject of the certificate: 
Enter a URI of the subject of the certificate: 
Enter the IP address of the subject of the certificate: 
Enter the e-mail of the subject of the certificate: 
Enter a challenge password: 
Does the certificate belong to an authority? (y/N): 
```

Will the certificate be used for signing (DHE and RSA-EXPORT ciphersuites)? (Y/n):
Will the certificate be used for encryption (RSA ciphersuites)? (Y/n):
Is this a TLS web client certificate? (y/N):
Is this a TLS web server certificate? (y/N):

- 4) Sign the certificate request with the CA certificate, pressing all questions other than the 3650 certificate expiration, and the final confirmation question. For example:

```
[S]# certtool --generate-certificate --load-request request.pem --load-ca-certificate ca.pem --load-ca-privkey ca-key.pem --outfile cert.pem
```

Generate a signed certificate...

Enter the certificate's serial number in decimal (default: 0123456789):

Activation/Expiration time.

The certificate will expire in (days): 3650

Extensions.

Do you want to honour the extensions from the request? (y/N): y

Does the certificate belong to an authority? (y/N):

Is this a TLS web client certificate? (y/N):

Will the certificate be used for IPsec IKE operations? (y/N):

Is this a TLS web server certificate? (y/N):

Enter a dnsName of the subject of the certificate:

Enter a URI of the subject of the certificate:

Enter the IP address of the subject of the certificate:

Enter the e-mail of the subject of the certificate:

Will the certificate be used for signing (required for TLS)? (Y/n):

Will the certificate be used for encryption (not required for TLS)? (Y/n):

... snip ...

Is the above information ok? (y/N): y

Signing certificate...

- 5) Rsyslog will never need to modify the keys. Secure the certificates and keys:

```
[S]# rm -f request.pem  
[S]# chmod 400 *key.pem
```

Configure Rsyslog Server For Remote TLS

- 6) Edit the Rsyslog server configuration file to enable logging using TLS encryption.
Specify the generated key and certificates.

[R7] File: /etc/rsyslog.conf

[S12] File: /etc/rsyslog.d/remote.conf

```
$ModLoad imtcp
+ $DefaultNetstreamDriver gtls
+ $DefaultNetstreamDriverCAFile /etc/rsyslog.d/ca.pem
+ $DefaultNetstreamDriverCertFile /etc/rsyslog.d/cert.pem
+ $DefaultNetstreamDriverKeyFile /etc/rsyslog.d/key.pem
+
+ $InputTCPServerStreamDriverMode 1 # run driver in TLS-only mode
+ $InputTCPServerStreamDriverAuthMode anon
→ $InputTCPServerRun 6514
```

Configure Rsyslog Client to Log to a Secure Remote Server

- 7) On the Rsyslog client, edit the configuration to use TLS encryption when sending remote logs.

For Red Hat Enterprise Linux these entries should go at the top of the GLOBAL DIRECTIVES section.

[R7] File: /etc/rsyslog.conf
[S12] File: /etc/rsyslog.d/remote.conf
+ \$DefaultNetstreamDriver gtls
+ \$DefaultNetstreamDriverCAFile /etc/rsyslog.d/ca.pem
+ \$ActionSendStreamDriverMode 1
+ \$ActionSendStreamDriverAuthMode anon

- 8) [R7] This step should only be performed on RHEL7.

On the client, ensure the delivery port is configured:

File: /etc/rsyslog.conf
→ *.info;mail.none;authpriv.none;cron.none @@10.100.0.5:6514

- 9) [S12] This step should only be performed on SLES12.

On the client, ensure the delivery port is configured:

File: /etc/rsyslog.conf
→ *.info;mail.none;news.none @@10.100.0.5:6514

- 10) On the client, copy the server's CA certificate to the local system:

[C]# scp stationS:/etc/rsyslog.d/ca.pem /etc/rsyslog.d/

• It may be necessary to run restorecon /etc/rsyslog.d/ca.pem after copying.

Verify Configuration and Successful Secure Logging

- 11) On both server and client, restart the Rsyslog daemon.

[C & S]# systemctl restart rsyslog

- 12) Verify that Rsyslog is listening on the server.

```
[S]# lsof -i :6514
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rsyslogd 6145 root 4u IPv4 45071 0t0 TCP *:syslog-tls (LISTEN)
rsyslogd 6145 root 5u IPv6 45072 0t0 TCP *:syslog-tls (LISTEN)
```

- 13) Send a test log message from the client:

```
[C]# logger -p info -t info-lvl-msg "Message from stationC"
```

Verify that the server received the log in the /var/log/messages file. If it did not, check the local log as well. Is Rsyslog running?

Server Cleanup:

- 14) Configure Rsyslog to stop listening on the network for remote log messages:

[R7] File: /etc/rsyslog.conf
[S12] File: /etc/rsyslog.d/remote.conf
<pre>- \$ModLoad imtcp - \$DefaultNetstreamDriver gtls - \$DefaultNetstreamDriverCAFile /etc/rsyslog.d/ca.pem - \$DefaultNetstreamDriverCertFile /etc/rsyslog.d/cert.pem - \$DefaultNetstreamDriverKeyFile /etc/rsyslog.d/key.pem - - \$InputTCPServerStreamDriverMode 1 - \$InputTCPServerStreamDriverAuthMode anon - \$InputTCPServerRun 6514</pre>

- 15) Restart the daemon so that the changes go into effect:

```
[S]# systemctl restart rsyslog
```

Client Cleanup

- 16) [R7] This step should only be performed on RHEL7.

On the client, configure Rsyslog to stop sending messages to your partner's system:

File: /etc/rsyslog.conf

# Log anything (except mail) of level info or higher.	
# Don't log private authentication messages!	
*.info;mail.none;authpriv.none;cron.none	/var/log/messages
- *.info;mail.none;authpriv.none;cron.none	@@10.100.0.0:S

- 17) [S12] This step should only be performed on SLES12.

On the client, configure Rsyslog to stop sending messages to your partner's system:

File: /etc/rsyslog.conf

.;mail.none;news.none	-/var/log/messages
- *.*;mail.none;news.none	@@10.100.0.0:S:6514

- 18) The client also needs to remove the gtls configuration:

[R7] File: /etc/rsyslog.conf

[S12] File: /etc/rsyslog.d/remote.conf

- \$DefaultNetstreamDriver gtls	
- \$DefaultNetstreamDriverCAFile /etc/rsyslog.d/ca.pem	
- \$ActionSendStreamDriverMode 1	
- \$ActionSendStreamDriverAuthMode anon	

- 19) Restart the daemon so that the changes go into effect:

```
# systemctl restart rsyslog
```

Bonus

- 20) Trade places, and have the client system act as server, and the server system act as client.

Exit root Shell

- 21) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Content

System Status – Memory	2
System Status – I/O	3
System Status – CPU	4
Performance Trending with sar	6
Determining Service to Process Mapping	7
Real-time Monitoring of Resources — Cgroups	8
Troubleshooting Basics: The Process	9
Troubleshooting Basics: The Tools	11
strace and ltrace	15
Common Problems	17
Troubleshooting Incorrect File Permissions	18
Inability to Boot	19
Typos in Configuration Files	20
Corrupt Filesystems	21
RHEL7 Rescue Environment	22
SUSE Rescue Environment	24
Lab Tasks	26
1. System Activity Reporter	27
2. Cgroup for Processes	32
3. Recovering Damaged MBR	38

Chapter

14

MONITORING & TROUBLESHOOTING

System Status – Memory

free

- Quick summary of memory usage

swapon -s

- Displays swap usage on a per swap file / partition basis

Process Files

- /proc/meminfo
- /proc/swaps

vmstat

- **vmstat 2** – displays running stats every 2 seconds

Memory Status

Linux provides several tools for examining the memory usage on running systems as shown in the following examples:

```
# free  
# cat /proc/meminfo
```

Swap usage can be summarized by either of the following:

```
# swapon -s  
# cat /proc/swaps
```

Field	Meaning
r	number of processes waiting to run
b	number of processes uninterruptedly sleeping
swpd	amount of swap used, in kilobytes
free	amount of memory free, in kilobytes

Field	Meaning
buff	amount of memory used for buffers, in kilobytes
si	memory being swapped from disk, in kilobytes/sec
so	memory being swapped to disk, in kilobytes/sec
bi	blocks written to disk, in blocks/sec
bo	blocks read from disk, in blocks/sec
in	number of interrupts per second
cs	number of context switches per second
us	percent of total CPU consumed by user processes
sy	percent of total CPU consumed by system processes
wa	percent of total CPU spent waiting on I/O
id	percent of total CPU idle time

Linux also provides the **vmstat** command, commonly used on Unix systems to examine memory usage. **vmstat** produces output like:

```
# vmstat  
procs      --memory--    --swap--   --io--   --system--   --cpu--  
 r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st  
 0 0     0  31108 413728 155980    0    0  147  142  458  838  16  4  78  3  0
```

System Status - I/O

vmstat

iostat

- Shows detailed I/O statistics on a per drive basis
- When using **-x** dereference any dm-Number files by running:
`# ls -al /dev/mapper/`
Or using **-N**

iotop

- Shows detailed I/O statistics on a per process basis

Process Files

- /proc/interrupts
- /proc/stat

More detailed information about specific devices can be found by:

```
# iostat -d -x /dev/sda  
. . . output omitted . . .
```

Field	Description
Device	name, in dev# format, where # is the major number
rrqm/s	merged read requests per second
wrqm/s	merged write requests per second
r/s	read requests per second
w/s	write requests per second
rsec/s	sectors read per second
wsec/s	sectors written per second
avgrq-sz	average size of requests, in sectors
avgqu-sz	average queue length of requests
await	average wait time until I / O request served
svctm	average time to service I / O requests, in seconds
%util	percentage of CPU time used servicing I/O requests

I/O Status

Several commands are also available to examine I/O statistics on Linux systems. Some basic I/O information is provided by **vmstat**. In addition, **iostat** can be run with a few different options to obtain more detailed statistics. This command presents summary information regarding I/O:

```
# iostat -d  
. . . snip . . .  
Device: tps Blk_read/s Blk_wrtn/s Blk_read Blk_wrtn  
sda 0.93 45.67 35.60 4433955 3456508  
fd0 0.00 0.00 0.00 32 0  
md0 0.12 0.36 0.51 35203 49684  
dm-0 0.64 0.31 1.51 30446 146162
```

Field	Description
Device	name, in dev# format, where # is the major number
tps	I / O requests per second
Blk_read/s	number of blocks read per second
Blk_wrtn/s	number of blocks written per second
Blk_read	total number of blocks read
Blk_wrtn	total number of blocks written

System Status – CPU

Process Files

- /proc/cpuinfo
Shows detailed info about physical CPU(s)

iostat -c

- provides a summary of CPU utilization

mpstat

- provides detailed CPU utilization
- including forced sleep by Xen hypervisor

CPU Details

Fairly detailed information about the processor(s) on the system can be obtained from the /proc/cpuinfo file as shown in the following example:

```
$ cat /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 13
model name    : Intel(R) Pentium(R) M processor 2.00GHz
... snip ...
```

CPU Status (uni-processor)

Several tools provide information about the system CPU configuration or utilization. This command displays an average, for all processors:

```
# iostat -c
... snip ...
avg-cpu: %user   %nice  %system %iowait  %steal   %idle
          0.20    0.07    0.37    0.01    0.01   99.35
```

Field	Description
%user	percentage of CPU used for user processes
%nice	percentage of CPU used for niced user processes
%sys	percentage of CPU used for system processes
%idle	percentage of CPU time spent idle

CPU Status (multi-processor)

Slightly more detailed information can be obtained with **mpstat**, which is usually run with the **-P ALL** option to display information about all processors on the system:

```
# mpstat -P ALL
04:02:14 PM   CPU   %user   %nice   %sys %iowait   %irq    %soft   %steal   %guest   %gnice   %idle   intr/s
04:02:14 PM   all    0.18    0.00    0.14    0.40    0.00    0.00    0.00    0.00    0.00    99.27   335.19
04:02:14 PM     0    0.20    0.00    0.17    0.63    0.00    0.01    0.00    0.00    0.00    98.99   171.50
04:02:14 PM     1    0.17    0.00    0.11    0.18    0.00    0.00    0.00    0.00    0.00    99.54   163.69
```

Field	Description
CPU	all is average of all CPUs; a number is a specific CPU
%user	percentage of CPU used for user processes
%nice	percentage of CPU used for niced user processes
%sys	percentage of CPU used for system processes
%iowait	percentage of CPU/CPUs time spent waiting for disk I/O
%irq	percentage of CPU/CPUs time spent servicing interrupts
%soft	percentage of CPU/CPUs time spent servicing software interrupts
%steal	percentage of time spent in involuntary wait by virtual CPU/CPUs while the hypervisor was servicing another virtual CPU
%idle	percentage of CPU time spent idle
intr/s	number of interrupts per second processed

Performance Trending with sar

Systemd oneshot service: sysstat.service

- records boot events
- SLES12: controls cronjob enablement

sadc

- system process located at `/usr/lib64/sa/sadc`
- run from `/etc/cron.d/sysstat`
cron job defaults to data from 10 minute intervals
- logs activity to `/var/log/sa/`

sar

- Shows system activity over time based on **sadc** logs

System Activity Reporter

The System Activity Reporter (**sar**) is a flexible tool which can be used to observe real time performance data, or take samples over time to show performance trends.

The basic syntax of the **sar** command is:

sar [options] [interval [count]]

The *options* indicates what output is desired, and the optional *count* says how many samples are desired. Useful options include:

Option	Result
-A	display (almost) all available statistics
-b	display disk I/O statistics
-B	display system paging statistics
-c	display process creation statistics
-d	display per-disk activity statistics
-I irq SUM PROC ALL XALL	display interrupt statistics; <i>irq</i> is the number of the IRQ to display; SUM displays the total number of interrupts; PROC displays the number of interrupts per processor; ALL displays the first 16 interrupts; XALL displays all interrupts;

Option	Result
-n DEV EDEV SOCK FULL	display network statistics; DEV displays statistics on network devices; EDEV displays network errors; SOCK displays statistics on network sockets; FULL displays all network statistics
-q	displays queue lengths and load averages
-r	displays memory and swap utilization statistics
-R	displays memory I/O statistics
-u	displays CPU utilization statistics
-v	displays kernel file table statistics
-w	displays switching statistics
-W	displays swapping statistics
-x pid SELF SUM ALL	displays process statistics; <i>pid</i> is the PID of the process to monitor; SELF indicates the sar process itself; SUM displays major and minor fault statistics; ALL displays statistics for all system processes
-X pid SELF ALL	displays stats about a process' child process; <i>pid</i> is the PID of the process whose children are monitored; SELF indicates children of the sar process itself; ALL displays statistics for child processes of all system processes
-y	display statistics about TTY activity

Determining Service to Process Mapping

Careful examination of PID and PPID numbers

- show basic hierarchy: `ps (f | -H)`
- processes can obscure or escape this

Based on systemd assigned unique control groups

- `systemd-cgls`

Historical Method of Mapping Processes to Services

When troubleshooting or doing other types of system analysis, it is often necessary to determine exactly which processes are associated with each service. For some services the mapping is an obvious one-to-one mapping where the service has started just a single process which may even be named after the service. For other services, there may be many processes whose names and even PID/PPID numbers make the relationship difficult or impossible to determine.

The `ps` command can display the relationship between processes, and the PID and PPID numbers can be examined to determine the relationship of a process with its parent, but this does not provide an authoritative link to the service that spawned the processes:

```
$ ps -ef f
. . . snip . .
guru 7654 1 0 Oct30 ? /usr/libexec/gnome-terminal-server
guru 7657 7654 0 Oct30 ? \_ gnome-pty-helper
guru 7658 7654 0 Oct30 pts/0 \_ /bin/bash
root 7737 7658 0 Oct30 pts/0 \_ su -
root 7743 7737 0 Oct30 pts/0 \_ -bash
. . . snip . .
```

Processes can change the name presented in the `ps` output, can have a parent die and inherit a new PPID of 1, or double fork making determining the original PPID impossible.

systemd Cgroup Method

`systemd` makes determining service to process mapping much easier by launching all processes for a service into unique named cgroups. The `ps` command can be used to display the cgroup information (e.g. `ps xawf -eo pid,user,cgroup,args`), but `systemd` also provides the `systemd-cgls` command specifically for this purpose:

```
# systemd-cgls
. . . snip . .
`-system.slice
  |-polkit.service
    `--31318 /usr/lib/polkit-1/polkitd --no-debug
  |-lvm2-lvmetad.service
    `--12415 /usr/sbin/lvmetad -f
  |-auditd.service
    |-5491 /sbin/auditd -n
    |-5504 /sbin/audispd
    `--5512 /usr/sbin/sedispatch
  . . . snip . .
```

Real-time Monitoring of Resources — Cgroups

Systemd starts slices and units in cgroups

Cgroups track resource usage

- totals for root slice enabled by default
- can enable manually for individual slices or units

Cgroups can also place limits on resources

`systemd-cgtop`

Viewing Cgroup Based Resource Tracking

Systemd provides the ability for real-time monitoring and limiting of system resource usage. In the default configuration, Systemd only monitors the global system resource usage through the root cgroup.

Resource usage tracking can be enabled for either everything or on a per-service basis. To enable resource usage tracking for everything edit the file `/etc/systemd/system.conf` and enable the options `DefaultCPUAccounting`, `DefaultBlockIOAccounting`, and `DefaultMemoryAccounting`:

```
File: /etc/systemd/system.conf
+ DefaultCPUAccounting=yes
+ DefaultBlockIOAccounting=yes
+ DefaultMemoryAccounting=yes
```

After editing, Systemd will have to have its configuration reloaded:

```
# systemctl daemon-reload
```

Resource usage tracking can also be enabled on a per-service basis with `systemctl`:

```
# systemctl set-property sshd MemoryAccounting=True
# systemctl set-property sshd MemoryLimit=100M
# systemctl restart sshd
# systemctl show sshd | grep Memory
MemoryCurrent=749568
MemoryAccounting=yes
```

`MemoryLimit=104857600`

`systemctl set-property` will create the drop-in unit files `/etc/systemd/system/ssh.service.d/50-MemoryAccounting.conf` and `/etc/systemd/system/ssh.service.d/50-MemoryLimit.conf`. To disable the memory limit in the future, the file `50-MemoryLimit.conf` can simply be deleted.

Once tracking has been enabled, the stats for any cgroup path can be accessed directly through the related files in the relevant, mounted cgroup filesystem:

```
# cat /sys/fs/cgroup/cpuacct/user.slice/user-1000-
.sslice/cpuacct.stat
user 2361
system 1486
# cat /sys/fs/cgroup/cpuacct/cpuacct.stat
user 25451
system 7976
```

Systemd also provides a `top`-like command for viewing the resource usage in real-time. For an individual target or slice to show details, monitoring must be enabled.

`# systemd-cgtop`

Path	Tasks	%CPU	Memory	Input/s	Output/s
/	71	0.4	579.1M	-	-
/user.slice	43	0.4	20.3M	15.9K	0B
/system.slice	43	0.0	7.3M	-	-
... snip ...					

Troubleshooting Basics: The Process

Investigate

Document Discoveries

Form Multiple Hypotheses

Test Each Hypothesis

Consider All Solutions (Then Pick The Best)

Document Changes

Repeat the Process (If Needed)

Investigate

In many ways, troubleshooting is like reading a good detective novel. One must not stop at the first possible explanation. Instead, evidence must be gathered and analyzed. The investigation can't end until there's enough proof to distinguish truth from fiction. Sometimes the truth is exactly as expected, and the challenge is proving it. Other times, the truth will be quite different from what was originally suspected.

Don't form an opinion too quickly when troubleshooting. A change that might correct one problem could make another problem worse. A superficial fix that only addresses a surface problem, without correcting the root cause, may hide the problem for awhile, but eventually the problem will return, perhaps making the problem worse!

Without a proper understanding of the problem, it is rare to arrive at a good solution. Be patient. Make sure the problem is understood before trying to solve it. Such obvious advice may seem ridiculous, but it is surprising the number of times it's ignored.

Document Discoveries

If knowing is half the battle, what's the other half? Remembering.

While it may be possible to keep everything in one's head during a short troubleshooting session, the longer the session, or the higher the stress of the situation, the more important it becomes to keep a record of everything learned during the process. Don't make things

too complicated. Paper and a pen is enough to get the job done, but if the troubleshooting session is long or the problem returns, a few basic notes taken during the troubleshooting process can be invaluable. Good notes can also be the basis of future documentation.

Form Multiple Hypotheses

Eventually, enough information will be gathered to support at least one hypothesis. At this point it is tempting to jump in and attempt to solve the problem. Resist this temptation! Take a moment to consider and rank other possible hypotheses. Many problems are easy to solve, but only after one stops focusing on the wrong hypothesis.

Test Each Hypothesis

Even after forming hypotheses, continue to collect information. If possible, perform one or more experiments to confirm the most likely explanation. Such confirmation is especially important when the fix is potentially dangerous or time consuming. A good experiment may also reveal a more serious underlying problem. Without careful confirmation, one may end up repeatedly addressing surface problems without ever arriving at the root cause.

Consider All Solutions (Then Pick The Best)

Many problems allow multiple solutions. Instead of immediately choosing the first solution that comes to mind, consider alternatives. Focusing too much on one solution may result in ignoring a faster,

safer solution. Taking time to consider the consequences of a solution is a critical part of not making the problem worse.

Document Changes

All changes made during the troubleshooting process should be documented. Ideally, any and all changes to a system should always be documented. If a change is later discovered to be inappropriate, something as simple as a chronological list of changes on a piece of paper can save significant time when reversing unintended damage.

Repeat the Process (If Needed)

Sometimes resolving one problem can reveal another. In these cases, it is tempting to assume the new problem is just an extension of the old. While that may often be true, care should still be taken to follow proper troubleshooting procedure, not just slap a quick fix on and hope for the best.

Troubleshooting Basics: The Tools

Every Tool Is Useful

Analyzing Log Files

- awk, dmesg, grep, head, tail

Reviewing System Status

- cat /etc/*-release, chkconfig, date, df, dig, free, getfacl, ls, lsattr, mount, netstat, ps, rpm, service, systemctl, top, w

Reviewing Network Settings

- cat /etc/resolv.conf, dig, ethtool, ip addr, ip route

Every Tool Is Useful

There is no replacement for a deep knowledge of a system's services and subsystems, and the tools that can be used to explore and control them. Under the right conditions, any command is a potential troubleshooting tool. However, some tools are more commonly used than everything else. These most common tools are presented in the next several tables.

Analyzing Log Files

Log files should always be examined early in the troubleshooting process. A lot can be learned quickly using basic tools to examine the system's log files.

Command	Description
awk	The ability to rapidly create simple reports from logs by filtering certain columns, summarizing, and applying formatting can reveal things that would otherwise be lost in the noise of the logs.
dmesg	While some kernel events are recorded in system log files, others may only be present in the kernel ring buffer.
grep grep -v	Log files are often filled with a mixture of useful and irrelevant information. While it is tempting to attempt mental filtering, it is better to use software to do it.
tail tail -n	While things sometimes break as a result of long past events, most often a recent change is at the root of new problems.
tail -f	Monitoring logs as new information is added can accelerate the troubleshooting process.

Reviewing System Status

Minor misconfigurations can have surprising side effects. Early in the troubleshooting process it is important to review the general status of the system.

Command	Description
cat /etc/*-release	When working in an environment with multiple Linux versions, knowing how to quickly determine what's running on a system can be helpful.
chkconfig --list service name status pgrep daemon ps -ef	Sometimes things break when a service is running that shouldn't be. More often, things break when a service should be running but isn't. If a service depends on multiple daemon's, make sure that all are running.
date	As services become increasingly time sensitive, systems can break in unexpected ways when their clocks are wrong.
df -h df -ih	Running out of disk space can cause many applications to exhibit strange behavior. Just as it is possible to run out of data blocks, it is also possible to run out of inodes. Some Linux filesystems can automatically allocate more inodes, but others can not.
free	When Linux runs low on available RAM, it is forced to page data out to swap, resulting in poor performance. Running low on memory can also force the kernel to activate the OOM Killer. This can even happen when memory is technically available but actually unusable.
ls -l ls -Z lsattr getfacl	Improper file permissions are a common cause of unexpected behavior. In addition to traditional Unix permissions, consider other options, like SELinux contexts, extended attributes, and FACLs.
mount cat /proc/mounts	Pay attention to not only what is mounted, but which options the filesystems are mounted with. Unfortunately, on older Linux distributions such as RHEL6, when the root filesystem is in readonly mode, the /etc/mtab file can not be updated and the output of the mount command may be incorrect. For this reason it may be necessary to query the kernel directly. In newer Linux versions, /etc/mtab is a symlink to /proc/self/mounts which avoids the problem of an out-of-sync mtab file.
netstat -tulpn lsof -i	A service may be running but not listening to the right port or address. Even if a service is configured to listen to the correct port and address, it will not be able to if another process has already claimed it.
rpm -Va	It is normal for configuration files to change. It is not normal for binaries and libraries to change. Changes may indicate failing storage, bad memory, or human action.
runlevel	When a system is brought up manually, it is easy accidentally leave it in an incorrect runlevel. Knowing the current runlevel is also important when confirming the correct services are running.
top	In addition to its default view, top includes several useful commands to tweak what information it shows and how. It is also a simple interface for certain types of process management.
w	When troubleshooting, it is important to know who is using the system.

Reviewing Network Status

Linux has a very network-oriented design. Even services only listening on localhost can break in unexpected ways when the network is misconfigured or misbehaving.

Command	Description
<code>dig host cat /etc/resolv.conf</code>	Many services depend on working name resolution.
<code>ethtool mii-tool</code>	When physical indicators are not visible, the <code>ethtool</code> command can be used to check link status. It can also be used to correct link level settings when autonegotiation fails.
<code>ip addr</code> <code>ip route</code>	Although it is possible to view most of a system's network configuration using <code>ifconfig</code> , the <code>ip</code> command will provide more correct and more complete information.
<code>tcpdump</code>	While it may be tedious to get specific information about network behavior using <code>tcpdump</code> , it is helpful for getting a general idea of what's happening on the network.

strace and ltrace

strace

- traces systems calls and signals
- **-p pid_num** – attach to a running process
- **-o file_name** – output to file
- **-e trace=expr** – specify a filter for what is displayed
- **-f** – follow (and trace) forked child processes
- **-c** – show counts

ltrace

- very similar to **strace** in functionality and options but shows dynamic library calls made by the process
- **-S** – show system calls
- **-n x** – indent output for nested calls

```
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
. . . snip . . .
```

Curly braces are used to indicate dereferenced C structures. Square braces are used to indicate simple pointers or an array of values.

Since **strace** often creates a large amount of output, it's often convenient to redirect it to a file. For example, the following could be used to launch the Z shell, trace any forked child processes, and record all file access to the files.trace file:

```
# strace -f -o files.trace -e trace=file zsh
. . . output omitted . . .
```

Run the **ls** command counting the number of times each system call was made and print totals showing the number and time spent in each call (useful for basic profiling or bottleneck isolation):

```
# strace -c ls
. . . output omitted . . .
```

The following example shows the three config files that OpenSSH's **sshd** reads as it starts. Note that **strace** sends its output to STDERR by default, so if you want to pipe it to other commands like **grep** for further altering you must redirect the output appropriately:

```
# strace -f -eopen /usr/sbin/sshd 2>&1 | grep ssh
. . . output omitted . . .
```

The strace Command

The **strace** command can be used to intercept and record the system calls made, and the signals received by a process. This allows examination of the boundary layer between the user and kernel space which can be very useful for identifying why a process is failing. Using **strace** to analyze how a program interacts with the system is especially useful when the source code is not readily available. In addition to its importance in troubleshooting, **strace** can provide deep insight into how the system operates.

Any user may trace their own running processes; additionally, the root user may trace any running processes. For example, the following could be used to attach to and trace the running **slapd** daemon:

```
# strace -p $(pgrep slapd)
. . . output omitted . . .
```

strace Output

Output from **strace** will correspond to either a system call or signal. Output from a system call is comprised of three components: The system call, any arguments surrounded by parenthesis, and the result of the call following an equal sign. An exit status of -1 usually indicates an error. For example:

```
$ strace ls enterprise
execve("/bin/ls", ["ls", "enterprise"], /* 38 vars */) = 0
brk(0) = 0x8aa2000
```

[R7] The following applies to RHEL7 only:

Trace only the network related system calls as Netcat attempts to connect to a local **in.telnetd** service:

```
# strace -e trace=network nc localhost 23  
. . . output omitted . . .
```

[S12] The following applies to SLES12 only:

Trace just the network related system calls as Netcat attempts to connect to a local **telnetd** service:

```
# strace -e trace=network netcat localhost 23  
. . . output omitted . . .
```

The ltrace Command

The **ltrace** command can be used to intercept and record the dynamic calls made to shared libraries. The amount of output generated by the **ltrace** command can be overwhelming for some commands (especially if the **-S** option is used to also show system calls). You can focus the output to just the interaction between the program and some list of libraries. For example, to execute the **id -Z** command and show the calls made to the **libsSelinux.so** module, execute:

```
$ ltrace -l /lib/libselinux.so.1 id -Z  
is_selinux_enabled(0xc1c7a0, 0x9f291e8, 0xclaffc, 0, -1) = 1  
getcon(0x804c2c8, 0xfee80fff4, 0x804b179, 0x804c020, 0) = 0  
user_u:system_r:unconfined_t
```

Remember that you can see what libraries a program is linked against using the **ldd** command.

Common Problems

- Incorrect file permissions**
- Inability to boot**
- Incorrect boot-loader installation**
- Corrupt file systems**
- Typos in configuration files**
- Disk full**
 - no free blocks
 - no free inodes
- Running out of (virtual) memory**
- Runaway processes**

Problems and Solutions in Linux

When problems occur in Linux, they manifest themselves via symptoms. Sometimes there is a strong correlation between the symptom and problem such that determining the root problem is very easy. Other times, the problem manifests itself in symptoms that are subtle or otherwise non-obvious. These latter types of problems can be especially difficult to solve the first time you encounter them.

Experience is a large, if not the largest, factor in speedy troubleshooting. The upcoming pages document common problems, their symptoms and solutions. Hopefully this will save you time in the future.

Troubleshooting Incorrect File Permissions

Overly strict permissions

/tmp/ not world writable and sticky (1777)

Filesystem is a hierarchy

What are the permissions on /

What UID & GID is daemon running as?

- What are the permissions on the files the daemon is trying to access?
- What are the permissions on parent directories?

Incorrect SELinux file context

Possible Solutions

Incorrect file permissions can be very difficult to debug. For example, if permissions on /tmp/ are incorrect, some services may start successfully, but when making an attempt to write a socket in /tmp/, will fail and silently exit.

strace and **ltrace** are invaluable for debugging opaque problems like these. Use **strace** on the daemon or program to see if it is getting any "permission denied" error messages. Then take the appropriate action to fix the permission problem.

One problem which can be difficult to diagnose is incorrect permissions on the root directory, /. To see the permissions on /, use the **ls -ld** / command. On Linux, you should see:

```
$ ls -ld /
drwxr-xr-x 22 root root 1024 Sep 18 07:25 /
```

Incorrect SELinux Security Context

On RHEL7/SLES12, problems with a wrong SELinux context can be fixed with the **chcon** or **restorecon** commands. For example, in order for Apache to be able to read a file it needs to have the **httpd_sys_content_t** SELinux context. Here is an illustration of a common problem:

```
# cd /var/www/html/; ls -alz
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 .
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 ..
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 index.html
-rw-r--r--. root root unconfined_u:object_r:var_t:s0    logo.png
```

The logo.png file has the incorrect context. It can be corrected with:

```
# chcon -t httpd_sys_content_t logo.png
```

Inability to Boot

Understanding exact steps performed during booting will greatly aid troubleshooting

BIOS, hardware, and boot loader problems

Kernel issues

- Does kernel have drivers for device that has root filesystem, and for the root filesystem type?

Filesystem Corruption or Label errors

- Filesystem is corrupted enough that the kernel cannot mount it read-only

Configuration file errors in critical files

- /etc/fstab, /etc/default/grub

x86 Linux Boot Process

BIOS

- ❖ Quick Hardware Check
- ❖ Initial Hardware configuration
- ❖ Selects and passes control to Master Boot Record on boot device

GRUB first stage in MBR

- ❖ Executes 2nd stage loader

Boot loader 2nd stage

- ❖ Provides boot prompt, displays graphic
- ❖ Starts execution of kernel

Linux kernel

- ❖ Initializes and configures hardware
- ❖ Finds and decompresses initramfs image(s)
- ❖ Loads drivers
- ❖ Creates root device, mounts root partition. Location of root filesystem is usually passed to kernel by boot loader
- ❖ Executes **init** (can be overridden with **init=/xxx/xxx** option)

init/systemd – (initial process)

- ❖ Executes core system services and environment configuration.

Changes accomplished by **systemd** include processing of command line parameters passed to kernel. Uses **libselinux** to mount the **/selinux/** filesystem. Loads SELinux policy. **systemd** starts system daemons, launches **login** for text terminal log in, and launches more specific services such as the display manager for the X Window System and **journald** for handling logging.

Typos in Configuration Files

Typos in certain configuration files can prevent successful boot

- /etc/default/grub
- /boot/grub2/grub.cfg
- /etc/fstab

Check /var/log/*

- Also see journalctl(1)

Check documentation for proper syntax

Essential Configuration Files

Typographical errors in essential configuration files can prevent Linux from booting correctly. This can occur when mistakes are made in the boot loader configuration files. In order to recover from an incorrectly installed boot loader, the system usually needs to be booted off of rescue media. To fix mistakes in other important files, such as fstab or inittab, you can usually do a minimal boot off of the local installation, either by booting with a rescue shell instead of init (**init=/bin/bash** at the boot loader command line) or by booting into run level 1 (appending **1** to the kernel parameters from the GRUB boot loader).

The GRUB configuration file that defines the menu items displayed on boot is the /boot/grub2/grub.cfg file. This file is generated by **grub2-mkconfig** or **grub-mkconfig**, using configurations found in /etc/grub.d/ and /etc/default/grub.

Corrupt Filesystems

**Filesystems need to be clean before they are mounted read-write
Unless major damage has occurred, unclean filesystems can still be
mounted read-only**

**Filesystem could be unclean because of hardware failure
fsck will return filesystem to consistent state**

- Filesystem structures repaired, not filesystem data
- Can take a LONG time on very large filesystems

Journaling filesystems partially prevent need to run fsck

- Btrfs, XFS, Ext3/4, JFS, ReiserFS

Solutions

At boot time, all filesystems are checked to see if they are in a clean state. Filesystems are marked clean as the final step of being unmounted. If a filesystem is not clean, **fsck** is run against it. **fsck** is a front-end program which calls an appropriate program, (e.g. **e2fsck(8)**, **reiserfsck(8)**, **xfs_repair(8)**) to check the filesystem.

If there is enough damage to the filesystem that data might potentially be lost, the automatic filesystem check halts, and you are prompted to manually run **fsck**. You will be prompted to press "y" at each repair step. This can become quite tedious, so you can use **fsck -y** to tell **fsck** to always assume "yes".

Files recovered by **fsck** will be stored in the `lost+found/` directory at the root of the file system from which they were recovered. On Ext2/3/4, `lost+found/` is not a normal directory file, but rather one which has a pool of pre-allocated data blocks available for its use. If it gets removed it can be recreated with the **mklost+found** command. The **xfs_repair** can safely create a `lost+found` directory during repair, so having one created in advance is not needed.

RHEL7 Rescue Environment

Some problems require use of a rescue mode

- Boot off install disk 1 and type `linux rescue` at the boot prompt or select Rescue installed system from the menu
- Filesystems will be mounted under `/mnt/sysimage/` if possible
- After fixing the problem, type `exit` at the prompt to reboot the system
If SELinux is enabled a filesystem re-label will be performed during the boot

The Rescue Environment

Using the rescue environment is a last-step effort, not a first step. Most problems do not require its use, and can be solved more easily in a more full-featured environment, such as single user mode. The rescue environment has a limited set of utilities. One useful utility is the `chroot` command:

```
# chroot /mnt/sysimage
```

`chroot` will change `/` (the root filesystem) to begin from `/mnt/sysimage/`. From there you can run `grub-install`, `vi`, `emacs`, and everything else as you would normally.

If Auto-mounting Filesystems Fail

When in the rescue environment, and auto-mounting fails, the first task is often to locate and mount your filesystems. To locate the available hard drives, examine the `/proc/partitions` file. For example:

```
# grep sd.* /proc/partitions
 8 0 42328800 sda
```

Next use the `fdisk` command to obtain a list of the available partitions on the available drives. For example:

```
# fdisk -l /dev/sda
...
/dev/sdal    2048   17101043  102322  501M 83 Linux
```

`/dev/sda2 1028096 727101043 7162322 34.2G 8e Linux LVM`

For each partition of type Linux, determine what part of your filesystem it contains using `blkid`:

```
# blkid
...
/dev/sda1: UUID="57019c5c-68f5660a612d" TYPE="xfs"
/dev/sda2: UUID="0XDg8q-K1Lo-PfFSff" TYPE="LVM2_member"
...

```

If there is a LVM PV partition, scan and activate LVM volume groups and view the results:

```
# vgs
# vgchange -a -y
# vgs
  VG #PV #LV #SN Attr   VSize  VFree
  vg0   1   4   0 wz--n- 34.18g 22.68g
# lvs
  LV   VG   Attr       LSize
  root vg0 -wi-a---- 8.00g
  swap vg0 -wi-a---- 512.00m
  tmp   vg0 -wi-a---- 1.00g
  var   vg0 -wi-a---- 2.00g
```

Find the partition of LVM logical volume that contains the root filesystem. Mount each filesystem in turn, and examine it to determine what it contains. The name of the LVM logical volume is usually a good indicator what it contains. For example:

```
# mount /dev/vg0/root /mnt/sysimage  
# cd /mnt  
# ls -al
```

When examining the mounted filesystem, remember that different filesystems will have different sub-directories:

```
/ ⇒ dev/, root/, lib/, etc/, media/, srv/  
/home ⇒ username1, username2, username3  
/boot ⇒ grub2/, vmlinuz.kernel_version, vmlinuz  
/var ⇒ log/, adm/, spool/, lock/, run/
```

Tip: Have a print out of the /etc/fstab file, and the output of the **fdisk -l** command for each drive.

The rescue environment provides several options for detecting and mounting Linux filesystems. One option is for the rescue environment to scan all drives on the system and automatically mount detected Linux filesystems in read-write mode. Another option will mount detected Linux filesystems in read-only mode. The last option is to skip the detection of Linux filesystems completely.

SUSE Rescue Environment

Some problems require use of a rescue mode

- Boot off SLES Install DVD and select Rescue System from the boot menu
- Login as root, no password required
- Hard drive filesystems need to be mounted manually
- After fixing the problem, type **reboot** at the prompt to reboot the system

Network rescue possible as well

- Requires a network install server available
- Boot from CD #1 and type:

```
install=nfs://10.100.0.254/netinstall usedhcp=1 rescue=1
```

The SLES12 Rescue Environment

The rescue environment has a limited set of utilities. A common trick is to use the "real system" during recovery. For example, to change your / to /mnt/ you can run:

```
# chroot /mnt
```

After chroot'ing you can run **grub2-install**, **vi**, **emacs**, and other command as you normally would.

Locating your Filesystems

When in the rescue environment, the first task is often to locate and mount your filesystems. To locate the available hard drives, examine the /proc/partitions file. For example:

```
# grep sd.$ /proc/partitions
 8 0 42328800 sda
```

Next use the **fdisk** command to obtain a list of the available partitions on the available drives. For example:

```
# fdisk -l /dev/sda
...
/dev/sda1 2048 17101043 102322 501M 83 Linux
/dev/sda2 1028096 727101043 7162322 34.2G 8e Linux LVM
```

For each partition of type Linux, determine what part of your filesystem it contains using **blkid**:

blkid

```
... snip ...
/dev/sda1: UUID="57019c5c-68f5660a612d" TYPE="xfs"
/dev/sda2: UUID="0XDg8q-K1Lo-PffSff" TYPE="LVM2_member"
... snip ...
```

If there is a LVM PV partition, scan and activate LVM volume groups and view the results:

```
# vgscan
# vgchange -a -y
# vgs
  VG #PV #LV #SN Attr   VSize  VFree
  vg0    1    4    0 wz--n- 34.18g 22.68g
# lvs
  LV   VG   Attr       LSize
  root vg0 -wi-a---- 8.00g
  swap vg0 -wi-a---- 512.00m
  tmp  vg0 -wi-a---- 1.00g
  var  vg0 -wi-a---- 2.00g
```

Find the partition of LVM logical volume that contains the root filesystem. Mount each filesystem in turn, and examine it to determine what it contains. The name of the LVM logical volume is usually a good indicator what it contains. For example:

```
# mount /dev/vg0/root /mnt
# cd /mnt
# ls -al
```

When examining the mounted filesystem, remember that different filesystems will have different sub-directories:

/ ⇒ dev/, root/, lib/, etc/, media/, srv/
/home ⇒ username1, username2, username3
/boot ⇒ grub/, vmlinuz.kernel_version, vmlinuz
/var ⇒ log/, adm/, spool/, lock/, run/

Tip: Have a print out of the /etc/fstab file, and the output of the **fdisk -l** command for each drive.

Lab 14

Estimated Time:
S12: 40 minutes
R7: 40 minutes

Task 1: System Activity Reporter

Page: 14-27 Time: 10 minutes

Requirements:  (1 station)

Task 2: Cgroup for Processes

Page: 14-32 Time: 15 minutes

Requirements:  (1 station)  (graphical environment)

Task 3: Recovering Damaged MBR

Page: 14-38 Time: 15 minutes

Requirements:  (1 station)  (classroom server)

Objectives

- Identify performance trends using the sar command.

Requirements

- (1 station)

Relevance

Performance tuning and capacity planning require the collection and analysis of resource usage statistics over a longer period of time to understand trends and spot patterns.

Notices

- This lab requires that the sysstat package has been installed, and running, for at least an hour so that sufficient data is available when running the commands. The lab output shown is based on approximately 2 days of data.

- Verify that the sysstat package is installed:

```
$ rpm -q sysstat  
sysstat-10.1.5-7.el7.x86_64
```

- [S12] This step should only be performed on SLES12.

Enable sysstat to run through systemd:

```
# sudo systemctl enable --now sysstat  
root's password: makeitso [Enter]
```

- [R7] This step should only be performed on RHEL7.

The /usr/lib64/sa/sadc command is called by the /usr/lib64/sa/sa1 script to update the system activity database. The /usr/lib64/sa/sa2 script calls the sar command to generate weekly reports. Use the more command to print the contents of the sysstat crontab:

```
$ more /etc/cron.d/sysstat  
# Run system activity accounting tool every 10 minutes  
*/10 * * * * root /usr/lib64/sa/sa1 1 1  
# 0 * * * * root /usr/lib64/sa/sa1 600 6 &  
# Generate a daily summary of process accounting at 23:53
```

Lab 14

Task 1

System Activity Reporter

Estimated Time: 10 minutes

- The sa1 script calls the sadc command to generate, or update, the system activity database.

```
53 23 * * * root /usr/lib64/sa/sa2 -A
```

- The sa2 script calls the sar command to generate weekly reports.

4) [S12] This step should only be performed on SLES12.

The /usr/lib64/sa/sadc command is called by the /usr/lib64/sa/sa1 script to update the system activity database. The /usr/lib64/sa/sa2 script calls the sar command to generate weekly reports. Use the more command to print the contents of the /etc/sysstat/sysstat.cron crontab file:

```
$ more /etc/cron.d/sysstat
# crontab for sysstat

*/10 * * * * root [ -x /usr/lib64/sa/sa1 ] && exec ↵
/usr/lib64/sa/sa1 -S ALL 1 1

# Update reports every 6 hours
55 5,11,17,23 * * *      root [ -x /usr/lib64/sa/sa2 ] && exec ↵
/usr/lib64/sa/sa2 -A
```

- The sa1 script calls the sadc command to generate, or update, the system activity database.
- The sa2 script calls the sar command to generate weekly reports.

5) Check to see if the database has been generated:

```
$ ls -l /var/log/sa/
total 476
-rw-r--r--. 1 root root 120720 Jun 20 23:50 sa20
-rw-r--r--. 1 root root 231804 Jun 21 16:50 sa21
-rw-r--r--. 1 root root 124378 Jun 20 23:53 sar20
[R7] $ file /var/log/sa/*
[S12] $ file /var/log/sa/206211/*
/var/log/sa/sa20:  data
/var/log/sa/sa21:  data
/var/log/sa/sar20: ASCII text
```

Results may vary depending on how long the system has been running, and how long the sysstat package has been installed. In the example above, two databases, and a report of the first database, have been generated.

6) Read the system activity report:

```
$ less /var/log/sa/sar20
. . . snip . . .
15:04:27          LINUX RESTART
```

	CPU	%usr	%nice	%sys	%iowait	%steal	%irq	%soft	%guest	%idle
15:10:01	all	0.30	0.00	0.16	0.05	0.00	0.00	0.00	0.00	99.50
15:20:01	0	0.30	0.00	0.16	0.05	0.00	0.00	0.00	0.00	99.50
15:30:01	all	0.23	0.00	0.15	0.04	0.00	0.00	0.00	0.00	99.58
15:30:01	0	0.23	0.00	0.15	0.04	0.00	0.00	0.00	0.00	99.58
. . . snip . . .										

A summary of the column meanings can be found under the **-u** option entry in the sar(1) manual.

- 7) Use the sar command to generate a report:

```
$ sar -A  
. . . snip . . .  
12:00:02 AM CPU %usr %nice %sys %iowait %steal %irq %soft %guest %idle  
12:10:01 AM all 0.06 0.00 0.14 0.04 0.00 0.00 0.00 0.00 0.00 99.76  
12:10:01 AM 0 0.06 0.00 0.14 0.04 0.00 0.00 0.00 0.00 0.00 99.76  
12:20:01 AM all 0.02 0.00 0.13 0.04 0.00 0.00 0.00 0.00 0.00 99.82  
12:20:01 AM 0 0.02 0.00 0.13 0.04 0.00 0.00 0.00 0.00 0.00 99.82  
. . . snip . . .
```

Notice that the report is current, showing the 10 minute update intervals from the database.

- 8) Use the sar command to print real time data as a report, pausing every two seconds, showing 4 lines and an average summary:

```
$ sar -u 2 4  
. . . snip . . .  
• -u = CPU utilization  
05:23:59 PM CPU %user %nice %system %iowait %steal %idle  
05:24:01 PM all 0.00 0.00 1.01 0.00 0.00 98.99  
05:24:03 PM all 0.50 0.00 0.00 0.00 0.00 99.50  
05:24:05 PM all 0.00 0.00 0.50 0.00 0.00 99.50  
05:24:07 PM all 0.00 0.00 0.00 0.00 0.00 100.00  
Average: all 0.12 0.00 0.38 0.00 0.00 99.50
```

- 9) Print a report showing statistics for each network interface, printing two entries, pausing two seconds in-between each entry, with a per-interface average summary.

```
$ sar -n DEV 2 2 • -n DEV = Network devices
. . . snip . .
05:31:38 PM IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s
05:31:40 PM lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
05:31:40 PM eth1 1.01 1.01 0.10 0.10 0.00 0.00 0.00 0.00
05:31:40 PM eth0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

05:31:40 PM IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s
05:31:42 PM lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
05:31:42 PM eth1 1.50 1.50 0.13 0.33 0.00 0.00 0.00 0.00
05:31:42 PM eth0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Average: IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s
Average: lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Average: eth1 1.25 1.25 0.11 0.21 0.00 0.00 0.00 0.00
Average: eth0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

See the entry for **-n** in the **sar(1)** manual for column descriptions.

- 10) Print a report showing statistics for Unix network sockets, printing two entries, pausing two seconds in-between entries, with a per-interface average summary.

```
$ sar -n SOCK 2 2 • -n SOCK = Network sockets
. . . snip . .
05:36:17 PM totsck tcpsck udpsck rawsck ip-frag tcp-tw
05:36:19 PM 398 6 11 0 0 0
05:36:21 PM 398 6 11 0 0 0
Average: 398 6 11 0 0 0
```

- 11) Print a report showing memory statistics, printing four entries, pausing two seconds in-between entries, with an average summary:

```
$ sar -r 2 4 • -r = Memory utilization
. . . snip . .
05:40:00 PM kbmemfree kbmemused %memused kbbuffers kbcached kbcommit %commit
05:40:02 PM 2901416 730208 20.11 78096 485072 267896 6.45
05:40:04 PM 2901416 730208 20.11 78096 485072 267896 6.45
05:40:06 PM 2901424 730200 20.11 78096 485072 267896 6.45
05:40:08 PM 2901424 730200 20.11 78104 485072 267896 6.45
Average: 2901420 730204 20.11 78098 485072 267896 6.45
```

See the entry for **-r** in the **sar(1)** manual for column descriptions.

- 12) Print an I/O statistics report, printing four entries, pausing two seconds in-between entries:

```
$ sar -b 2 4
```

	tps	rtps	wtps	bread/s	bwrtn/s
05:43:37 PM	0.00	0.00	0.00	0.00	0.00
05:43:39 PM	0.00	0.00	0.00	0.00	0.00
05:43:41 PM	0.00	0.00	0.00	0.00	0.00
05:43:43 PM	0.00	0.00	0.00	0.00	0.00
05:43:45 PM	0.00	0.00	0.00	0.00	0.00
Average:	0.00	0.00	0.00	0.00	0.00

• -b = I/O and transfer rate

See the entry for **-b** in the `sar(1)` manual for column descriptions.

Objectives

- ❖ Use cgroups to identify the processes associated with specific user sessions and services
- ❖ Use cgroup accounting to track real-time resource usage

Requirements

- ❑ (1 station) X (graphical environment)

Relevance

Historical methods of tracking which processes are associated with a given service and what resources they are consuming were prone to many problems. Newer cgroup based tools make it easy to get accurate answers to these questions.

Notices

- ❖ This lab assumes that only one guru user login session is active. If other user sessions are active then the output will differ from that shown.

1) [S12] This step should only be performed on SLES12.

Start the auditd service:

```
$ su -c "systemctl start auditd"  
Password: makeitso 
```

2) List all the currently active sessions:

```
$ logindctl  
SESSION      UID  USER          SEAT  
c1           42   gdm          seat0  
1            1000 guru         seat0
```

2 sessions listed.

Details for the guru session shown in the output would correspond to the /sys/fs/cgroup/systemd/user.slice/user-1000.slice/session-1.scope/cgroup files.

3) List all processes contained within that user's cgroup (user-1000.slice):

```
$ logindctl user-status guru
```

Lab 14

Task 2

Cgroup for Processes

Estimated Time: 15 minutes

```
guru (1000)
  Since: Tue 2014-11-11 09:57:04 MST; 2min 49s ago
  State: active
  Sessions: *1
    Unit: user-1000.slice
      `-session-1.scope
        |-2251 gdm-session-worker [pam/gdm-password]
        |-2262 /usr/bin/gnome-keyring-daemon --daemonize --login
        |-2264 gnome-session --session gnome-classic
        |-2272 dbus-launch --sh-syntax --exit-with-session
    . . . snip . . .
```

- 4) List all of the processes associated with that particular cgroup (the user-1000.slice):

```
$ systemd-cgls --no-pager /sys/fs/cgroup/systemd/user.slice/user-1000.slice/
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/:
`-session-1.scope
  |-2251 gdm-session-worker [pam/gdm-password]
  |-2262 /usr/bin/gnome-keyring-daemon --daemonize --login
  |-2264 gnome-session --session gnome-classic
  |-2272 dbus-launch --sh-syntax --exit-with-session
. . . snip . . .
```

- 5) List all of the current cgroups with their associated processes, and search for the audit.service to determine how many processes it has launched:

\$ **systemd-cgls** The cgroup listing program launches

Search by typing: **auditd.service**

The auditd section of the output is displayed:

```
|-auditd.service:
|  |-629 /sbin/auditd -n:
|  |-639 /sbin/audispd:
|  `-644 /usr/sbin/sedispatch
```

Exit by pressing:

Program exits.

Cgroup Resource Usage Tracking

- 6) List all current cgroups with their associated number of launched processes:

```
$ systemd-cgtop
Path          Tasks  %CPU   Memory  Input/s Output/s
/
  160    0.3  565.6M    -      -
/system.slice/NetworkManager.service  2     -      -      -      -
/system.slice/abrt-oops.service      1     -      -      -      -
... snip ...
Exit by pressing: [q]
```

Notice that currently only the root slice shows details for CPU, Memory, and I/O.

- 7) Examine the current config for the systemd user.slice looking specifically at the accounting related options:

```
$ systemctl show user.slice | grep Accounting
CPUAccounting=no
BlockIOAccounting=no
MemoryAccounting=no
```

• All cgroup based accounting is currently disabled for this slice.

- 8) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso [Enter]
```

- 9) Enable the cgroup based CPU accounting for the user.slice:

```
# systemctl set-property user.slice CPUAccounting=1
# systemctl show user.slice | grep Accounting
CPUAccounting=yes
BlockIOAccounting=no
MemoryAccounting=no
```

• Now active

This change will be permanent once the systemd daemon is reloaded.

- 10) Enable the cgroup based CPU accounting for the system.slice:

```
# systemctl --runtime set-property system.slice CPUAccounting=1  
# systemctl show -p CPUAccounting system.slice  
CPUAccounting=yes
```

This change will be temporary (only for this boot).

- 11) Reload the systemd daemon so that it commits the earlier change to disk:

```
# systemctl daemon-reload
```

- 12) Identify, and examine, the file that systemd stored the persistent change in:

```
# find /etc/systemd/ -name '*Accounting*'  
/etc/systemd/system/user.slice.d/50-CPUAccounting.conf  
# cat /etc/systemd/system/user.slice.d/50-CPUAccounting.conf  
[Slice]  
CPUAccounting=yes
```

- 13) Confirm that the config for the user.slice references the override config stored in the /etc directory:

```
# systemctl show -p DropInPaths user.slice  
DropInPaths=/etc/systemd/system/user.slice.d/50-CPUAccounting.conf
```

- 14) Verify that the user and system slices are now showing CPU stats:

```
# systemd-cgtop  
Path          Tasks   %CPU   Memory  Input/s Output/s  
/             158    0.4    526.2M    -       -  
/user.slice     44    0.3     -        -       -  
/system.slice    37    0.0     -        -       -  
. . . snip . . .  
Exit by pressing: [q]
```

It may take some time for the system.slice to show any CPU time.

- 15) Activate non-persistent memory accounting for the auditd.service:

```
# systemctl --runtime set-property auditd.service MemoryAccounting=1
```

- 16) Examine the cgroup based resource stats again:

```
# systemd-cgtop
```

Path	Tasks	%CPU	Memory	Input/s	Output/s
/	158	0.6	525.1M	0B	509B
/user.slice	44	0.4	-	-	-
/system.slice	40	0.0	1.9M	-	-
... snip ...					
/system.slice/auditd.service	3	-	8.0K	-	-
... snip ...					

Exit by pressing:

It may take some time for the auditd.service to show any memory usage.
Notice that the system.slice also now shows memory stats since it is higher in
the cgroup hierarchy.

- 17) Reboot and test the persistence of the CPU accounting set on the user.slice:

```
# systemctl reboot
```

The system reboots.

Log in as the guru user to the GUI.

The GUI desktop is displayed.

Open a terminal.

```
$ systemd-cgtop
```

The user.slice still shows CPU accounting stats.

Cleanup

- 18) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
```

Password: *makeitso*

- 19) Disable the CPU accounting for the user.slice:

```
# rm /etc/systemd/system/user.slice.d/50-CPUAccounting.conf  
# systemctl daemon-reload
```

Objectives

- Use the rescue environment to recover a damaged MBR

Requirements

- (1 station) (classroom server)

Relevance

Certain problems are severe enough that they render the system unbootable by normal means. In these cases, you must boot the system using an alternate boot media and possibly alternate root filesystem to repair the system. This task teaches how to enter and use the rescue environment.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Proper preparation for any disk or filesystem disaster includes having a copy of the partition table and what each partition contains. At a minimum you should know where your /boot/, /, and /var/ filesystems are located. Use df to view where they are currently located:

```
# df  
... output omitted ...
```

- 3) Record which partition contains /boot:

Result: _____

- 4) Record which partition contains /:

Result: _____

- 5) Record which partition contains /var:

Result: _____

Lab 14

Task 3

Recovering Damaged MBR

Estimated Time: 15 minutes

- 6) Overwrite your MBR by running the command listed below. Please double-check your syntax before running the command, as simple typos here can easily render the machine inoperable (beyond the scope of this course to recover):

```
# dd if=/dev/zero of=/dev/sda count=1 bs=446  
... output omitted ...  
# systemctl reboot
```

- Carefully type this command as any typos could completely destroy your Linux installation.

- 7) Your instructor will provide instructions on whether PXE is available, and if so how to PXE boot your lab system. If PXE is not available, skip to the next step.

PXE boot your workstation.

Most Intel systems do this with the **F12** key after POST.

From the PXE menu, select Rescue Mode by **typing XXX** (replace with correct value from menu).
This will boot to the Rescue environment.

Skip to step 11 for RHEL7 systems, or 12 for SLES12 systems.

- 8) [R7] This step should only be performed on RHEL7.

If you are using the installation DVD, or the Rescue CD, complete the following:

Boot from the optical media provided by your instructor.

At the menu, **select** Troubleshooting

At the menu, **select** Rescue a Red Hat Enterprise Linux system

- 9) [S12] This step should only be performed on SLES12.

If you are using the "disk 1" installation media, it contains all the required components to boot into the rescue environment without requiring a network server. Otherwise, skip to step 10 to use the custom boot ISO. Boot off the media then complete the following actions:

Boot from the CD

The system should boot to a list of bootable targets.

Select Rescue System

The system should boot directly to the rescue environment

Skip to step 12.

10) [S12] This step should only be performed on SLES12.

If you are using a CD created from your own suse-boot.iso image then you should complete the following actions:

Boot from the CD

The system should boot to a list of targets.

Use your **[I]** key to **highlight** the Install option

Then type the following:

```
install=nfs://10.100.0.254/export/netinstall/SLES12 usedhcp=1 rescue=1
```

11) [R7] This step should only be performed on RHEL7.

It will then ask if it should attempt to locate and automatically mount the Linux filesystems. In many situations this will work fine and you would choose Continue.

Continue. It would then mount the root filesystem at /mnt/sysimage/ and mount child filesystems as well.

In order to become versed with the steps required in the worst case scenario, do not have it attempt automatic mounting. **Select Skip** to go directly to the command prompt.

```
sh-4.2# fdisk -l
```

- Note the partition number for your boot partition.

12) [S12] This step should only be performed on SLES12.

The system will boot and then drop you to a prompt:

```
Rescue Login: root
```

- No password will be required.

13) If the system being rescued is using LVM based devices, it may be necessary to discover, import, activate, or create the necessary device files:

```
# lvmdiskscan
. . . snip . . .
/dev/sdb                  [      74.51 GiB]
6 disks
17 partitions
```

- In some cases, misconfigured external storage targets, such as iSCSI or FCoE, fail to automatically detect. Note in this case that 1 LVM physical volume is detected.

```
0 LVM physical volume whole disks
1 LVM physical volume
# lvm vgs
VG          #PV #LV #SN Attr   VSize  VFree
vg0        1   5   0 wz--n- 34.18g 25.68g
# lvm vgchange -a y vg0
5 logical volume(s) in volume group "vg0" now active
# ls /dev/vg0/
root  swap  tmp  var
```

- If there is an x in the attributes then the Volume Group has been exported and will need to be imported using `lvm vgimport vg0`.

14) Mount the normal filesystems under `/mnt/sysimage/`:

```
# mkdir /mnt/sysimage
# mount /dev/vg0/root /mnt/sysimage/
# mount -o bind /dev /mnt/sysimage/dev/
# mount -o bind /sys /mnt/sysimage/sys/
# mount -o bind /proc /mnt/sysimage/proc/
# mount /dev/vg0/var /mnt/sysimage/var/
# mount /dev/vg0/tmp /mnt/sysimage/tmp/
# mount /dev/sda1 /mnt/sysimage/boot/
# chroot /mnt/sysimage/
```

- Once the root filesystem is mounted it is possible to look at `/mnt/sysimage/etc/fstab` to see the normal mounts. If LABEL or UUID references are used instead of the device filename, use the `findfs` or `blkid` commands to identify the device filenames.
- The `chroot` command may not provide an indication that you are currently in a pseudo-root.

15) Repair the master boot record (MBR) on the primary drive:

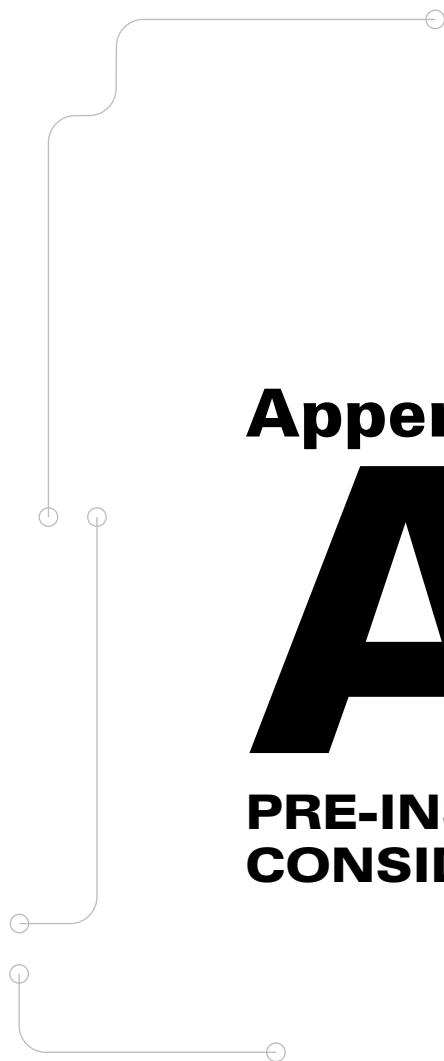
```
# grub2-install /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
```

16) Exit the `/mnt/sysimage/` isolated root. Halt the system:

```
# exit
# systemctl poweroff
```

- This terminates the chrooted shell.

17) Boot the system to the hard drive. If needed, remove the rescue media used. If the steps above were performed correctly the system can now be booted normally.



Appendix

A

PRE-INSTALLATION CONSIDERATIONS

Pre-Installation Considerations

Is the hardware compatible?

Will the system require dual booting?

- Resizing existing partitions?

Which partition scheme will be used?

- GPT or MBR

Will LVM scheme be used?

- Should LVM PVs be on bare LUN/disk or use a partition?

Hardware vs Software RAID vs LVM RAID vs Btrfs RAID

What filesystem(s) will be used?

What is the expected primary role of this system?

Life Cycle Considerations: 13 years

- 10 year life cycle, plus up to 3 years extended maintenance

Pre-installation Considerations

Certain aspects of the system can be difficult to change after the initial installation. Things that should be considered include the following:

Is the hardware compatible with Linux?

Linux has made huge strides in the area of hardware compatibility in recent years. Server class hardware is almost certainly supported, workstation class support is likely, but unusual devices may not be supported, especially some laptop hardware.

Will the System Dual-boot?

If the system will boot to more than one operating system, then attention should be given to what boot loader is used, and to the desired OS default.

What Partitioning Scheme Will Be Used?

Software RAID and LVM are easier to configure at installation time (especially for the root filesystem). When traditional partitions are used, it is best to select appropriate initial sizes since modifying partition sizes later can be difficult and dangerous.

What Filesystem Type(s) Will Be Used?

Each available filesystem type may have distinct advantages and disadvantages. System performance can be tuned by selecting a filesystem that best fits the expected usage of the system.

What is the Expected Role of the System?

The role of a system will determine the software packages that are installed. While it is easy to install and remove packages after installation, experience has shown that busy administrators often neglect to remove unneeded software packages after the initial install. Remember that every software package installed on the system should be considered a potential security vulnerability. One of the most important steps in securing a system is ensuring that it is not running any services that are not absolutely required.

Life Cycle Considerations

Each major release of Red Hat Enterprise Linux has a support life cycle of 10 years, plus up to 3 years of extended maintenance, from the date of release. The 10 year support time is further divided into 3 phases each with different levels of support. Red Hat publishes full details of their support cycle at the following URL:
<http://www.redhat.com/security/updates/errata/>

End of Phase 3 Support Dates for current RHEL Releases

- ⌚ Red Hat Enterprise Linux 5: March 31, 2017
- ⌚ Red Hat Enterprise Linux 6: November 30, 2020
- ⌚ Red Hat Enterprise Linux 7: June 30, 2024

Hardware Compatibility

Linux should be compatible with most hardware

Potentially problematic hardware

- Extremely new hardware
- Proprietary laptop components

Linux Hardware Compatibility

The two components of a Linux system that care about the hardware are the Linux kernel and the X Window System. In the early days of Linux, hardware support was spotty. The main issue was getting proper hardware documentation so that developers could write the drivers.

With the rise of Linux's popularity and market share, most hardware manufacturers have moved Linux support to being a check list item. Because of this, most hardware makers are openly making proper documentation available to Linux developers. Another big change is that it is becoming increasingly common for hardware makers to write and release their own internally developed drivers.

Binary vs. Source Drivers

To ensure maximum performance, reliability, and technical superiority, there is *no* guaranteed binary or source level compatibility between major kernel versions. Because of this, even when hardware manufacturers write their own drivers, open source drivers are strongly preferred. Source code availability enables the Linux community to maintain the drivers and keep the hardware supported in future versions of Linux, even if the hardware manufacturer loses interest (or goes out of business).

Running Linux on a Laptop

A laptop is much more likely to have proprietary and undocumented hardware than a desktop or server system. Before attempting to

install Linux on a laptop, system components, such as the video chipset, sound card, or network card should be carefully checked to make sure that the laptop's hardware is properly supported by the Linux kernel.

The "Linux on Laptops" website (<http://www.linux-laptop.net/>) is a great resource when installing Linux on a laptop. It lists thousands of laptop models, and for each model displays people's experiences running Linux on that particular laptop, including anything unusual they had to do to make Linux fully functional.

The use of Linux on mobile computers presents many situations and procedures that are not encountered in desktop use. There are many best practices and helpful tricks in the Linux Mobile Guide, an online resource that is updated on a regular basis.

Red Hat Hardware Catalog

The Red Hat Hardware Catalog can be found at <https://access.redhat.com/certifications>. This list documents what hardware is known to work, or not work, with Red Hat Enterprise Linux.

SUSE Linux Hardware Compatibility List

The SUSE Linux Hardware Compatibility List (HCL) can be found at <https://www.suse.com/yesssearch/>. This database documents what hardware is known to work with SUSE Linux Enterprise Server.

Multi-OS Booting

Normally only a desktop or laptop consideration

Consider OS partition and drive constraints

Consider possible sharing of partitions

- swap
- data

Consider making a backup of the master boot record (MBR)

OS Partition and Drive Constraints

Linux can happily co-exist with other operating systems on the same computer. Other operating systems may have constraints on which partition, or if more than one hard drive is available, to which drive they can be installed. Linux has no such constraints and can be installed on any partition(s) on any drive.

Since Linux plays nicely with other operating systems during an install, it is best to install it last. Other operating systems will often wipe the master boot record during install, causing grief if Linux is already installed.

Sharing Partitions Between Operating Systems

Depending on what operating systems are installed, you may be able to have several installed OSes make use of the same partitions. For example, if your laptop has several different versions of Linux installed then at a minimum you could create a single shared swap partition that is mounted by each of the three Linux installations.

Even if you were booting between completely different operating systems (such as both Linux and Microsoft Windows) you still may choose to create a data partition that will be mounted by both operating systems. In this case, it is recommended that you create the partition and filesystem while booted to Windows. Linux tends to be more flexible in the partitions that it will accept and should be able to mount whatever the Windows partitioning program creates.

Backing Up and Restoring the MBR Manually

If you are unsure what changes an installer might make to your master boot record, then you can create a backup of the boot record before starting the install. The procedure described here uses the `dd` command. If you already have Linux installed on the system, then issue the command from a terminal as root. If you do not have Linux already installed, then use a bootable Linux disk or CD:

```
# dd if=/dev/sda of=MBR bs=512 count=1
```

Then copy the resultant MBR file to a safe location. If you need to restore the master boot record later, you can issue this command after copying the saved MBR file into the current directory:

```
# dd if=MBR of=/dev/sda bs=512 count=1
```

SUSE Automatic MBR Backup

When performing an installation or upgrade of SLES12 the existing MBR is automatically saved into the `/var/lib/YaST2/backup_boot_sectors/` directory. The 512 byte file is named after the block device it was copied from.

```
# ls -l /var/lib/YaST2/backup_boot_sectors/
total 4
-rw----- 1 root root 512 May  4 09:25 _dev_sda
```

Partition Considerations

MBR Table Structure

- Primary Partitions (max of 4)
- Extended Partition (max of 1)
 - generally fills rest of disk
 - contains Logical Partitions
- Max number of partitions limited by kernel and partitioning tools
- 32bit LBA limits max disk size to 2TB

GPT Table Structure

- 128 partitions
- No extended or logical partitions
- Critical structures duplicated and CRC checked
- 64bit LBA limits max disk size to 9.4 billion TB

GUID Partition Table – GPT

The traditional BIOS booted MBR and corresponding partition table has several limitations that have become more significant over time. With 2TB disks now shipping, and SAN or RAID presented block devices often much larger, the 32bit LBA addressing (with its corresponding 2T max addressable size) used in the MBR is no longer sufficient. GPT uses 64bit LBA addressing and is capable of dealing with much larger disks. Additional benefits include:

- ❖ disks up to 9.4×10^{21} bytes (9.4 zettabytes) in size
- ❖ support for 128 partitions per disk
- ❖ important data structures duplicated at end of disk and CRCs stored to detect corruption
- ❖ uses 16byte GUIDs to identify partitions instead of 2byte type codes
- ❖ can store user readable Unicode label for each partition

GPT defines a compatibility MBR structure in the first sector to help protect the GPT data structures from modification by non-GPT aware utilities. These legacy programs will see a single partition of type code ee encompassing the whole disk (or 2TB which ever is smaller). The **parted** and **gdisk** programs can create GPT disk labels and partitions. Notably, **gdisk** can convert from MBR to GPT, and is modeled after (though not identical to) util-linux **fdisk**.

Partition Considerations

The various filesystems constituting a Linux installation will reside on one or more block devices. Determining the appropriate partitions for a new installation is closely tied to the filesystems that will be created unless LVM is in use. However, even when LVM Logical Volumes are used, the underlying Physical Volumes generally map to partitions on the disk.

MBR Partition Table Structure

On Intel-compatible hardware with a BIOS, Linux uses a partition scheme which allows up to four partitions to be created on a hard drive (now called primary partitions). Only one of these partitions may be flagged bootable at a time.

To extend this scheme beyond four partitions, a special type of primary partition, the extended partition, was created. The extended partition is not a functional partition, and no filesystem can be created on it; rather, it is a container for other functional partitions, called logical partitions. In the MBR partition scheme, up to four primary partitions can be created, one of which can be an extended partition. Within the extended partition, logical partitions can be created. On IDE, PATA, and SATA hardware, Linux supports a maximum of 63 usable partitions. Previous to kernel 2.6.28, all devices detected as SCSI (e.g. /dev/sda) had a maximum 15 partitions.

Filesystem Planning

Appropriate filesystem layout depends on machine function

- Only a root filesystem (/) is absolutely required
- Typical minimum partitions: /boot/, /, and swap.

Common additional filesystems

- /var/ – This directory contains logs, mail files and other various data
- /srv/ – A blank directory that can be used much like /var/
- /tmp/ – Space for temporary files
- /home/ – Users' home directories
- /opt/ – Additional program binaries (usually third party)
- /usr/local/ – Additional local programs and data

Typical Minimal Filesystems

Linux typically has a minimum of three filesystems on Intel-compatible hardware. The first of these is a small /boot/ filesystem (normally 250–500MB in size). Due to limitations with the GRUB boot loader, this filesystem should be stored on an Ext3/4 partition, not a Logical Volume, software RAID device, LUKS encrypted device, or GPT partition.

The second of these is a swap partition. This partition is typically 1-2 times the size of RAM installed on the system up to 2GB. For systems with more than 2GB of RAM, swap is normally equivalent in size to the amount of RAM installed (plus 2G). Choosing the appropriate size will vary with the machine's intended function, especially for systems with more than 32GB. Alternatively, swap can be a file on a filesystem instead of a partition.

The third filesystem is the root filesystem, (/). This needs to be large enough to contain all other files that will be installed (anywhere from 4–12GB is typical).

When to Create Additional Filesystems

The use of additional filesystems beyond the minimums discussed will depend on the intended use of the system, but can offer many potential benefits. Common reasons for creating additional filesystems include:

- Containment (prevent files from filling the root or another critical filesystem)

- Matching filesystem type for performance with expected I/O patterns (Ext4, XFS, tmpfs, block size, journal location, etc.)
- Different mount options for: security (ro, noexec, nosuid, nodev, etc.); performance (noatime, journal mode, etc.); features (quota, acl, user_xattr, etc.)
- Filesystem level backup and restores (e.g. **dump**, and **restore**)
- Separation of user data and system data

Care should be taken when separating a file system, as this may impact services, storage access, and the boot process. Also, it can be difficult to accurately predict the correct size for some filesystems. For this reason, generally the best practice is to use LVM and place all but the /boot/ filesystem on logical volumes. Logical volumes make it much easier to resize file systems later.

Top Level Directories that Must Be on /

The root filesystem is the first filesystem that is mounted during the boot process and contains files critical to completing boot up. Because of that, certain directories cannot be separated from the root filesystem. These directories include: /etc/, /bin/, /sbin/, /lib/, /dev/, /root/ (optional). Originally, the root user's home directory was / (hence the name root), but has since been put into /root/ on Linux, and some Unix, systems. The /etc/passwd database expects /root/ to be in place (such as when booting to single user mode), but the system will still function without it. In recent years, it has been demonstrated that some services fail silently when the /usr/ filesystem is separate. This has led to distributions requiring /usr/ to be part of the root filesystem.

Selecting a Filesystem

Linux supports several advanced journaling filesystems

Extended Filesystem: Ext2 (no journal), Ext3, and Ext4

- R7: Max supported filesystem size — 50TB
- R7: Max supported individual file size — 16TB

XFS – SGI's journaling filesystem

- Advanced filesystem, highly scalable, supports defragging
- Sparse inode chunk allocation to support peer-to-peer cluster filesystems: GlusterFS and Ceph
- R7: Max supported filesystem size — 500TB

Btrfs

- Built-in Volume Manager, RAID, compression, and many features
- Likely future default Linux filesystem

Ext Journaling Modes

The extended 3 filesystem added journaling support to Ext2. Several journaling modes are available. It can journal all file data and metadata (data=journal), or it can journal metadata but not file data (data=ordered or data=writeback).

When not journaling file data, Ext can be configured to write file system data before metadata (data=ordered; this causes all metadata to point to valid data) or to not do any file data journaling at all (data=writeback; the filesystem will be consistent, but old data may appear in files after an unclean system shutdown). These options give the administrator the power to make appropriate trade-offs between speed and file data consistency, and to tune performance for specific usage patterns on a filesystem-by-filesystem basis. Ext3 features include:

- ❖ in-place compatibility with ext2
- ❖ advanced options to customize journaling performance based on system usage
- ❖ excellent general use performance

Filesystem Features: XFS

- ❖ Provides high performance, and advanced features, for large filesystems.
- ❖ Default filesystem in RHEL7
- ❖ Default filesystem for non-OS filesystems in SLES12

Journaling Filesystem Implementation

All filesystems are essentially databases which store and organize data on a disk and use metadata (such as file names and timestamps) to allow later retrieval. When a file is modified in any way, two filesystem transactions are required. The first one updates the file data, while the other transaction updates the metadata (the filesystem's indexing information about that file). Since two operations are required, the potential exists for one operation, but not the other, to be completed, resulting in an inconsistent filesystem on which the data and metadata disagree. If the system loses power or is otherwise interrupted after writing the blocks out to disk, but before writing the metadata to disk, the filesystem is in an inconsistent state and must be repaired (using **fsck**) for that file to be accessible.

Journal-based filesystems remove this need for **fsck** to do a time-consuming recursive check to repair inconsistencies by keeping a journal of all intended transactions before they are carried out, and then updating the journal to reflect the success of transactions after they have been completed. On a journaled filesystem, creating a new file would first write an entry to the journal, then write the file and the file's metadata, then write again to the journal indicating that the file was successfully created. This approach provides more robustness and reduced recovery times for more overhead during normal operation (extra writes to the journal).

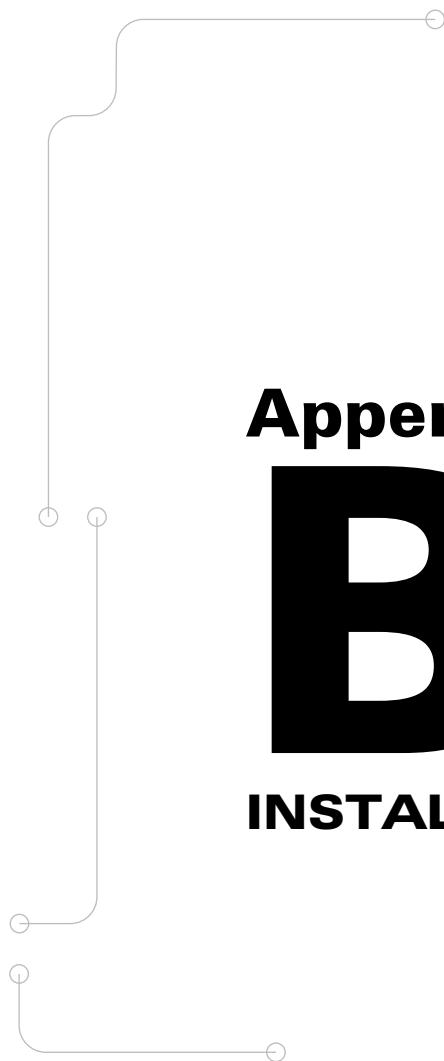
Filesystem Features: Btrfs

- ❖ Subvolumes
- ❖ Snapshots
- ❖ Compression
- ❖ Integrated device management
- ❖ Rollback
- ❖ Upgrade an extended filesystem to Btrfs
- ❖ Default filesystem for OS filesystems in SLES12

Filesystem Features: ReiserFS

Though SuSE Linux (6.2-10.1) used ReiserFS by default (4.2-6.1 used ext2), starting with SUSE Linux Professional 10.2, the Ext3 filesystem became the new default. Still supported for backwards compatibility with SLES12

- ❖ Performs well with small & large numbers of files
- ❖ Provides advanced options to customize journaling performance based on system usage



Appendix

B

INSTALLING RHEL7

Anaconda: An Overview

Installer for Red Hat Enterprise Linux

Multiple modes

- Install
- Rescue

Multiple components

- Boot the system
- Load anaconda
- Configure the system
- Download packages

Documentation

Anaconda: An Overview

Anaconda is the installer for Red Hat Enterprise Linux. Although referred to as a single application, its modular design allows it to fulfill multiple roles. The installation process can be interactive, fully or partially automated. By default, it will run the media test before attempting an installation. This test can be aborted so as to proceed with the installation. An installation without media test can be selected from the DVD installation menu.

Hit `Esc` to get a boot: prompt. The default label is `linux`. For instance, type `linux rescue` to boot to a rescue mode. (Rescue can also be selected from the Troubleshooting menu option.) When run in rescue mode, Anaconda attempts to detect and mount all filesystems for an existing install.

Choosing Components

Four questions must be answered when installing a system with Anaconda:

1. How should the system be booted?
2. How should Anaconda be loaded?
3. What package source will be used to install the system?

4. What interface should be used to configure the system?

Custom support for all major PC boot methods is provided: CD/DVD, USB, internal drives, and PXE. A similar mixture of options for loading both Anaconda and packages is supported: CD/DVD, USB, internal drives, and network servers running NFS, FTP, or HTTP.

Two types of interfaces are provided: text-based and graphical. The advantage of the text interface is that it doesn't require a video card or network interface. It is possible to force the use of the text-based interface by passing the `text` command to Anaconda. If the system has a network card, the graphical interface can be accessed with VNC. Whether using the text-based or graphical interface, it is possible to automate part or all of the installation process. Instead of prompting a user for information, Anaconda can read configuration details from a kickstart file.

Example Scenarios

For a simple install of a single system, it is easiest to follow the default installation procedure. Boot the official install DVD, and let the system automatically load Anaconda. Use Anaconda's graphical interface to configure the system, and install only the packages included on the DVD.

For multiple installations of the same type, a fully automated process reduces installation time. Boot each system using PXE and load Anaconda, and all packages, from an NFS server. Fully automate the system configuration using a kickstart file downloaded over HTTP.

In a testing lab, self-contained automated reinstalls can be helpful.
Boot from an internal hard drive. Load Anaconda, packages, and the kickstart file from a partition which will not be used as the installation target.

Even for manual installs, the modular design of Anaconda can be helpful. For example, it is possible to use a boot CD, but load Anaconda and packages from a USB drive. Because USB drives are generally much faster than optical media, the install will complete faster. In addition, a local FTP server could be configured as a second package source, making it possible to add custom software during install.

Additional Documentation

Red Hat provides detailed documentation of the installation process, including less common options, in the Installation Guide available on their web site.

Anaconda: Booting the System

Hardware issues

- AMD64 and Intel64
- UEFI

Boot methods

- Boot CD or Install DVD
- USB
- PXE
- Hard drive

Booting From CD/DVD

In addition to the standard install DVD, a boot CD without the installation tree is available for download, named as a boot.iso file, (except for IBM System z). This can be used to bootstrap the install process, accessing the installation tree from another source like a USB drive or a network server.

Booting From USB

Both the full sized install DVD and the boot CD (boot.iso) can be written directly to a USB thumb drive, then used to install systems that support booting from USB. The entire USB drive must be reformatted, including removing any existing partition table. On Linux, this can be easily accomplished using one of the following dd commands:

```
# dd if=/dev/sr0 of=/dev/sdb
# dd if=boot.iso of=/dev/sdb
```

For UEFI-based systems, a special disk image can be used. It is included on the install DVD as images/efiboot.img.

Booting From PXE

Building a PXE server is not trivial. At a minimum, a properly configured DHCP server, a Trivial FTP (TFTP) server, and either an NFS, FTP, or HTTP server is required. Basic instructions for setting up PXE are included in the Install Guide for Red Hat Enterprise Linux. More advanced configurations with menu and dynamic configuration are also possible. While the overhead may not be justified to install a single system, it quickly becomes worthwhile as the number of systems that need to be installed increases.

Booting From a Hard Drive

Two files are required to configure GRUB to boot from a hard drive into Anaconda: the kernel and an initial ramdisk. Both can be obtained from the isolinux/ directory of the install DVD. This is a great option for systems that will be reinstalled repeatedly, for example in a QA lab.

Anaconda: Common Boot Options

Kernel parameters

Anaconda options

- `inst.rescue`
- `inst.text`
- `inst.vnc`
- `inst.repo=...`
- `inst.ks=...`

Adding Boot Arguments

To add boot arguments when booting from the official install DVD, hit escape to leave the initial boot menu for a boot prompt. Just like an installed Linux system, boot arguments can be passed to the installer's kernel to resolve driver issues.

Some Available Boot Options

Anaconda supports many boot options that can be used to resolve hardware issues or customize its behavior. These often interact with kernel parameters and Dracut options. For example:

`inst.rescue` ⇒ Start Anaconda in rescue mode instead of install mode.

`inst.text` ⇒ Use the text-based interface to Anaconda instead of the graphical interface. For instance, this is useful with serial consoles defined by the `console=` kernel parameter.

`inst.vnc` ⇒ Instead of using the video card to display Anaconda's graphical interface, start a VNC server.

`inst.vncpasswd=...` ⇒ Require a password when connecting to Anaconda over VNC.

`inst.headless` ⇒ Don't detect the video card to display Anaconda's graphical interface.

`inst.resolution=...` ⇒ Start the GUI with an alternate resolution (e.g. 1024x768).

`inst.repo=...` ⇒ Load Anaconda from an alternate source, like USB or a network server, (replaces `method=`).

`inst.ks=...` ⇒ Load a kickstart file for automating an install.

`ip=...` ⇒ IP address, netmask, prefix, gateway, and hostname to set the system to. Also see `BOOTIF=` and Dracut's `rd.neednet` options.

`ifname=...` ⇒ Kernel parameter for the name and MAC address of the interface, e.g. `ifname=foo:01:01:01:01:01:42`.

If using the `inst.ks=` boot option, the following values can be used (also applies to `inst.repo=`):

Source	Option format
CD/DVD drive	<code>inst.ks=cdrom:optionalDevice:/directory/ks.cfg</code>
Hard drive	<code>inst.ks=hd:device:/directory/ks.cfg</code>
HTTP server	<code>inst.ks=https://host/directory/ks.cfg</code>
FTP server	<code>inst.ks=ftp://user:pass@host/directory/ks.cfg</code>
NFS server	<code>inst.ks=nfs:nfsOptions:host:/directory/ks.cfg</code>

If using the `inst.ip=` boot option, the following values can be used (all entries are optional):

Method	Option format
Automatic (any)	<code>ip=[dhcp dhcp6 auto6 ibft]</code>
Automatic (specific)	<code>ip=eth0:[dhcp dhcp6 auto6 ibft]</code>
Static	<code>ip=address::gateway:netmask:hostname->:eth0:[none dhcp:mtu]</code>

Anaconda: Loading Anaconda and Packages

DVD

Tree-based

- USB/HD
- NFS/FTP/HTTP

ISO-based

- USB/HD
- NFS

ISO-based Install Options

For ISO-based installs, instead of expanding the entire install DVD, the original ISO image is used. However, the file `images/install.img` must still be extracted. For example:

```
# mkdir -p /export/install/images  
# mv install.iso /export/install/  
# mount -o loop,ro /export/install/install.iso /mnt  
# cp /mnt/images/install.img /export/install/images/  
# umount /mnt
```

To install from USB or a hard drive, the format of the `inst.repo` option is `inst.repo=hd:dev:/path`. For example:

`inst.repo=hd:sdb1:/install`. If using an ISO image, either from USB or hard drive, or from NFS, the installation directory is the directory containing the ISO image.

Installing from DVD

When a system is booted from the official install DVD, it will automatically load Anaconda from the same DVD unless the `inst.repo=` option is passed at boot.

Installing from USB and Hard Drive

Four file systems are supported when installing from USB or a hard drive: Ext2, Ext3, Ext4, and VFAT. LVM and RAID are not supported.

Tree-based Install Options

For tree-based installs, the entire contents of the install DVD are copied into a directory. For example:

```
# mount /dev/sr0 /mnt/  
# cp -dR /mnt /export/install/
```

To install from USB or a hard drive, the format of the `inst.repo` option is `inst.repo=hd:dev:/path`. For example:

`repo=hd:sdb1:/install`

To install from NFS, the format of the `inst.repo` option is `inst.repo=nfs:[options:]host:/path`. For example:

`inst.repo=nfs:server1.example.com:/export/install`

To install from FTP or HTTP, the format of the `inst.repo` option is `inst.repo=URL`. For example:

`inst.repo=ftp://guru:work@server1.example.com/install` or
`inst.repo=https://server1.example.com/install`.

Anaconda: Storage Options

Devices

- Local disks
- Firmware RAID
- SAN and multipath devices
- iSCSI and FCoE

Formats

- Software RAID
- LVM
- LUKS

Storage Technologies

In addition to basic configuration of locally attached disks, Anaconda includes support for detecting and configuring a wide range of more advanced technologies. During installation, if the user chooses to configure "Specialized & Network Disks", Anaconda can detect and present additional options for many types of hardware-backed technologies such as firmware RAID, SAN and multipath devices. Support for pure software-based iSCSI and FCoE is also included.

Storage Formats

In addition to supporting standard filesystems like Ext3 and VFAT, Anaconda can be used to configure more advanced disk formats. Software RAID can be used to provide cheap redundancy, or improved performance. LVM can be used to make storage more flexible. LUKS can be used to protect sensitive data by encrypting entire filesystems.

Anaconda: Troubleshooting

Logging

Hardware compatibility

Logging

During different stages of the installation process, log messages are sent to various destinations. Some messages are sent only to virtual terminals (VTs); these can be accessed using the standard key combinations (e.g. **Ctrl** + **Alt** + **F2**). Other messages are saved to files on a RAM disk. To view these files, use the command line available on VT 2.

VT #	Log File	Description
VT 2	n/a	Shell Prompt
VT 3	/tmp/anaconda.log	Log messages from Anaconda
VT 4	/tmp/syslog	Log messages sent to syslog by various programs
VT 5	/tmp/X.log	Diagnostic output from the X server
VT 5	/tmp/program.log	Commands executed by Anaconda and their output
n/a	/tmp/storage.log	Storage related log messages from Anaconda
n/a	/tmp/packaging.log	Diagnostic output from Yum and RPM

Because all log files created by Anaconda are stored only in RAM, to preserve them they must be copied to another system. One convenient option is the **scp** command available from the shell

prompt on VT 2.

As an added convenience, when Anaconda crashes all of its log files are concatenated to create the `/tmp/anacdump.tb-string` file. As with the other log files, this file must be manually copied to another system to preserve it.

Hardware Compatibility

Occasionally, attempting to install on very new or very old hardware can reveal incompatibilities. Although not an exhaustive list, one helpful resource is Red Hat's hardware compatibility database: <http://hardware.redhat.com/hcl/>.

One boot option is especially worth mentioning. Most video cards support a BIOS-based interface called VESA. As a result, adding "inst.xdriver=vesa" at boot can often be used as a work around for video driver issues.

FirstBoot

Requirements

- Install graphical environment

Tasks

- Subscribe to RHN
- Create a non-root user
- Configure user authentication
- Configure the system clock
- Configure kdump

Alternatives to firstBoot

FirstBoot

After installation has completed, if the system meets certain requirements, a graphical configuration tool called FirstBoot will run. First, a graphical environment must have been chosen during installation. For example, the "Server with GUI" software set.

Running FirstBoot is not absolutely necessary. Alternate configuration tools can be used instead. Specifically:

Task	Tool
Subscribe to RHN	<code>subscription-manager-gui</code> <code>subscription-manager register</code>
Create a non-root user	<code>system-config-users</code>
Configure user authentication	<code>system-config-authentication</code> <code>authconfig</code>
Configure the system clock	<code>system-config-date</code>
Configure kdump	<code>system-config-kdump</code>

Kickstart

Enables partially or fully automated installs

- Create Kickstart file
Previous install: /root/anaconda-ks.cfg.
File Sections: install options, packages, pre-install script (optional), post-install script (optional).
Validate with **ksvalidator**.
- Invoke Anaconda specifying Kickstart file location, e.g.
`ks=nfs:s1.example.com:/export/kickstart/r70-dhcp.cfg`

Kickstart

A kickstart file is a text file containing the answers to some, or all, of the questions that Anaconda asks during the install process.

Normally, kickstart files are used to perform the fully automated installation of many systems. If certain configuration options are omitted from the kickstart file, Anaconda will pause to request manual configuration. For example, if the required "timezone" option is missing, Anaconda will pause to allow manual selection of a timezone before automating the rest of the installation process.

Documentation

Documentation of core kickstart features is generally high quality. Unfortunately, more advanced features are not consistently documented. In addition to Red Hat's Installation Guide, another useful source of information is the Fedora Project:

<http://fedoraproject.org/wiki/Anaconda/Kickstart>.

Unfortunately, because Fedora's documentation focuses on the latest Fedora release, it occasionally contains information that doesn't yet apply to Red Hat Enterprise Linux.

Tools

Because the kickstart file format is so simple, a basic text editor is often sufficient for intermediate and advanced administrators.

Alternatively, a graphical tool called **system-config-kickstart** is also available. This tool is not recommended: its kickstart syntax is that of RHEL6 and has not been updated for changes in RHEL7.

Before performing a kickstart-based install, checking the kickstart file for errors is recommended. The **ksvalidator** command, which is part of the `pykickstart` package, can be used to check for syntax errors. Note, however, that it is not able to detect logic errors like a filesystem that is too small for the requested packages.

File Sections

A kickstart file consists of four basic sections, two of which are optional:

1. A collection of configuration options. For example, install vs. upgrade, storage configuration, and package sources.
2. A list of selected and excluded packages.
3. An optional pre-installation script.
4. An optional post-installation script.

As a useful feature, the pre-installation script can be used to dynamically generate a text file which can be loaded by a "%include" statement in the installation options section of the kickstart file. For example, a common use of the feature is dynamically generated storage configuration.

The post-installation script can be used to configure the newly installed system before it is even rebooted. This can be used, for example, to ensure a specific security policy.

Example Kickstart File

```
# Any required options not listed will be interactively prompted for.
# Required: bootloader, keyboard, lang, part, rootpw, timezone
install
lang en_US.UTF-8
keyboard --vckeymap=us --xlayouts=us
network --device=eth0 --bootproto=dhcp --hostname=mx1.example.com
nfs --server=server1.example.com --dir=/export/netinstall/RHEL7
repo --name="errata" --baseurl="nfs:server1.example.com:/export/courserepos/errata/R7"
auth --enableshadow --passalgo=sha512
bootloader --location=mbr --boot-drive=sda
timezone America/Denver --utc
firewall --disabled
firstboot --disabled
rootpw --plaintext makeitso
user --name=guru --password=work
xconfig --startxonboot
reboot

zerombr
clearpart --all --initlabel
part /boot --fstype="xfs" --size=500
part pv.42 --size=2000 --grow
volgroup mx1 --pesize=4096 pv.42
logvol / --fstype=xfs --name=root --vgname=mx1 --size=8192
logvol /tmp --fstype=xfs --name=tmp --vgname=mx1 --size=2048
logvol /home --fstype=xfs --name=home --vgname=mx1 --size=4096
logvol swap --name=swap --vgname=mx1 --recommended

%packages
@base
@core
@gnome-desktop
@x11
curl
strace
-emacs
%end

%post
curl https://server1.example.com/cgi-bin/post-install.cgi | /bin/bash
%end
```

Network Booting with PXE

Preboot eXecution Environment (PXE)

- Uses DHCP to acquire an address
- Uses TFTP to obtain a boot image

TFTP Server Configuration

- Create and populate boot image directories
- Create the PXE boot configuration file

DHCP returns special options

- filename
- tftp-server-name

Starting the TFTP Server

The TFTP service does not require a configuration file to be started. However, a firewall must allow connections to TCP port 69. The TFTP service runs from Xinetd and can be started as shown here:

```
# chkconfig xinetd on  
# chkconfig tftp on  
# service xinetd restart
```

Preparing for Booting Clients

To support PXE clients, you must populate the TFTP server's directory with the necessary directory paths and files. First create a directory hierarchy to hold the boot files for each OS to be supported:

```
# mkdir -p /var/lib/tftpboot/linux-install/{RHEL,SLES}
```

Copy the network boot kernel and initial RAM-disk image for each OS from the installation media into its respective directory (the filenames sometimes differ between Linux distributions).

For example, if the RHEL7 installation DVD is mounted at /mnt/cdrom/, then run:

```
# cp /mnt/cdrom/images/pxeboot/{vmlinuz,initrd.img} /var/lib/tftpboot/linux-install/RHEL/
```

Copy the main PXE boot image to the TFTP server's directory tree:

```
# cp /usr/share/syslinux/pxelinux.0 /srv/tftpboot/linux-install/
```

Preboot eXecution Environment (PXE)

PXE is a standard that provides for the ability to boot the system from a file that is downloaded at boot time from a network server. Almost all modern business class PCs support PXE either via code on a ROM in the NIC, or directly via the system BIOS.

PXE is also a powerful companion for automated installation because it allows a new machine to be installed without requiring any local media. Plug the new system into the network, initiate a network boot and walk away. Depending on the package selections and network/system speed you can have a fully installed system in a matter of minutes.

Specialized rescue or recovery images can also be served by the PXE server to help when troubleshooting a system. A menu of network bootable images can be defined on the PXE server and sent to the PXE client screen for selection. A default can also be defined with a timeout so that clients configured to netboot can optionally revert to booting from the local disk after some defined interval.

To configure a Linux system to act as a PXE server, the `dhclient` and `syslinux` RPM packages need to be installed.

In addition, on Red Hat Enterprise Linux the `tftp-server` package should be installed.

Additional documentation about configuring PXE server support can be found at <http://syslinux.zytor.com/pxe.php>.

PXE Linux TFTP Configuration

Create the configuration files read by the **pxelinux.0** image. First, create the PXE boot image's configuration directory:

```
# mkdir /var/lib/tftpboot/linux-install/pxelinux.cfg/
```

Detailed documentation for these files can be found in the `/usr/share/doc/syslinux-version/syslinux.txt` file.

Second, create a `tftpboot/linux-install/pxelinux.cfg/default` file with content similar to:

File: `/var/lib/tftpboot/linux-install/pxelinux.cfg/default`

```
+ default 0
+ timeout 100
+ prompt 1
+ display msgs/boot.msg
+
+ label 0
+   localboot 0
+
+ label 1
+   kernel RHEL/vmlinuz
+   append initrd=RHEL/initrd.img ramdisk_size=10000,
+         ks=nfs:10.100.0.254:/export/kickstart/RHEL.ks
+
+ label 2
+   kernel SLES/vmlinuz
+   append initrd=SLES/initrd splash=silent,
+         install=nfs://10.100.0.254/export/netinstall/,
+         SLES usedhcp=1 textmode=1 autoyast=nfs://
+               10.100.0.254/export/netinstall/configs/SLES.ay
```

Finally, create the `tftpboot/linux-installmsgs/boot.msg` file with content like:

File: `/var/lib/tftpboot/linux-installmsgs/boot.msg`

```
+ PXE Network Installation
+ Enter number of the OS you wish to install:
+ 0. Boot Local Machine
+ 1. Install RHEL
+ 2. Install SLES
```

DHCP Server Configuration

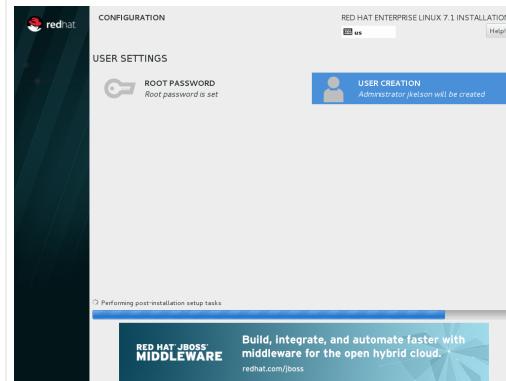
The first thing that a PXE client does is attempt to obtain an IP address. The DHCP server should be configured with an appropriate pool of addresses or static reservations for the clients. In addition to providing IP address information, the DHCP server must indicate the name and location of the boot file that will be retrieved by the PXE client. Since this boot filename should only be sent for requests from PXE clients, not normal DHCP requests, use the conditional functionality built into the ISC DHCP server. For example:

File: `/etc/dhcpd.conf`

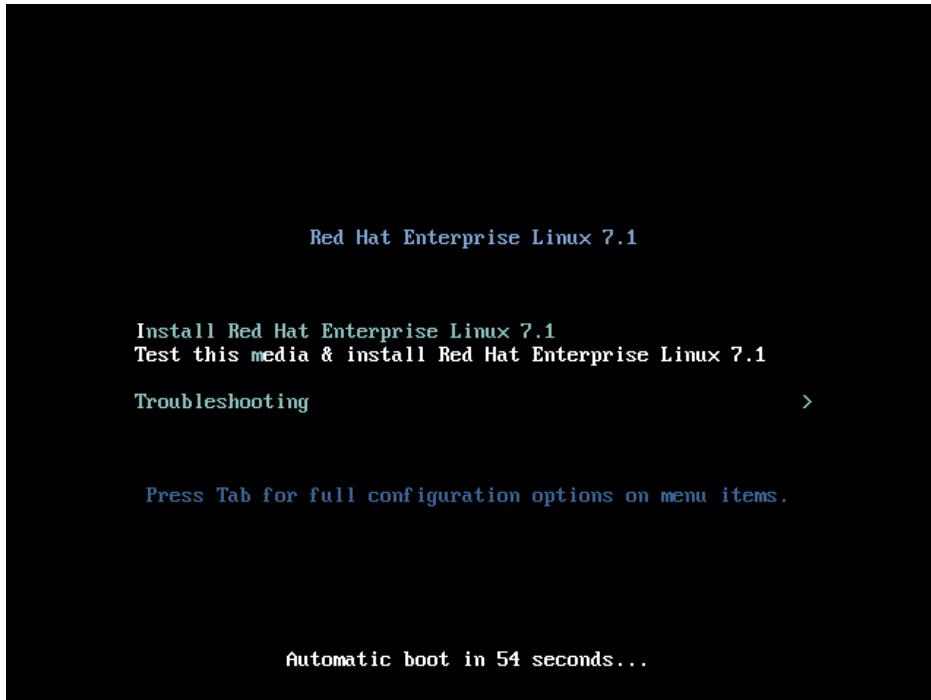
```
+ #PXE Specific Boot options
+ class "PXE" {
+   match if substring (option vendor-class-identifier, 0, 9) = "PXEClient" {
+     next-server "10.100.0.254";
+     filename "linux-install/pxelinux.0";
+   }
+ }
```

Additional example configuration options are described in the `/usr/share/doc/syslinux-version/pxelinux.txt` file.

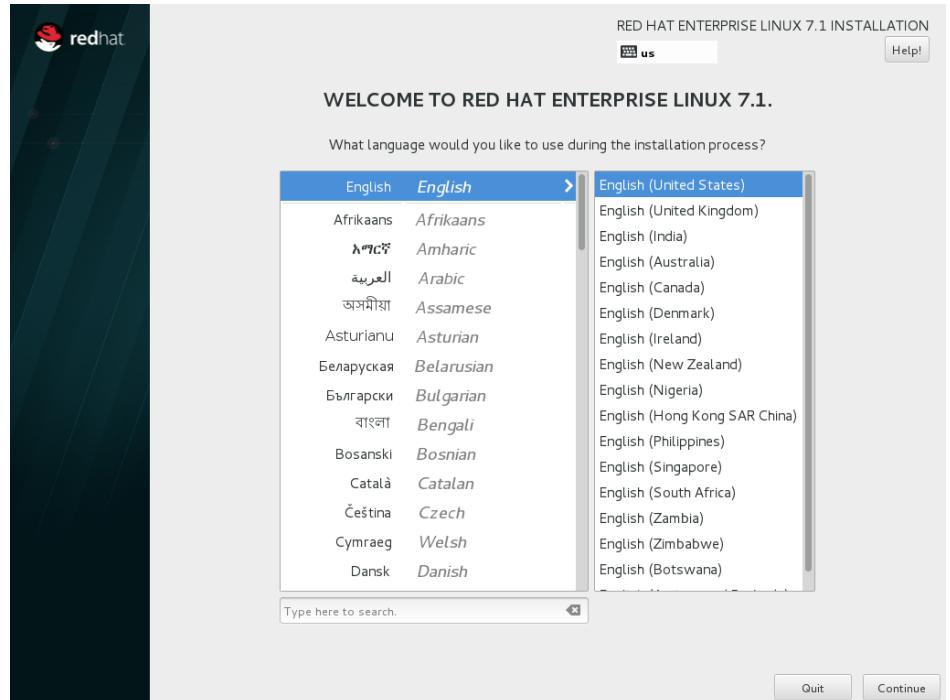
A Typical Install



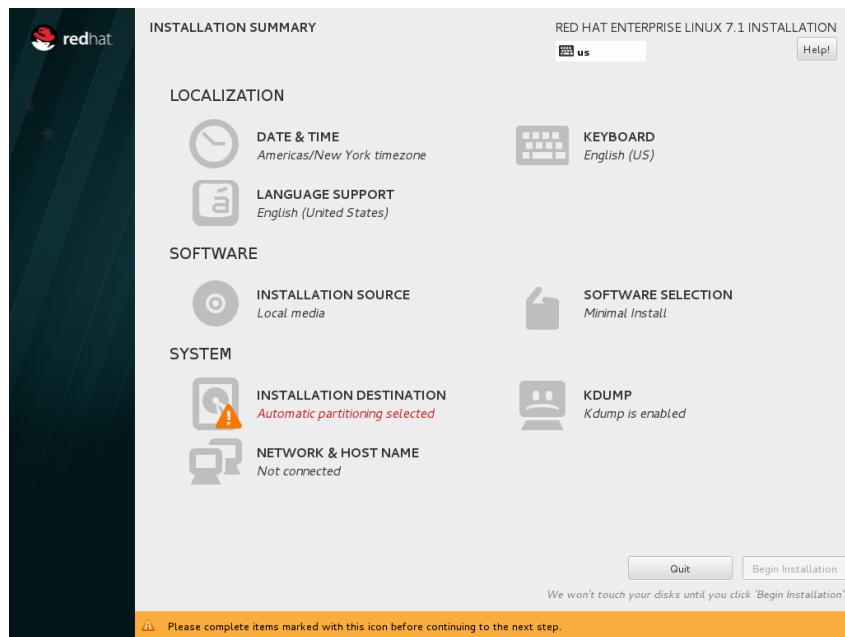
(1) Choosing boot options at the ISOLINUX menu:



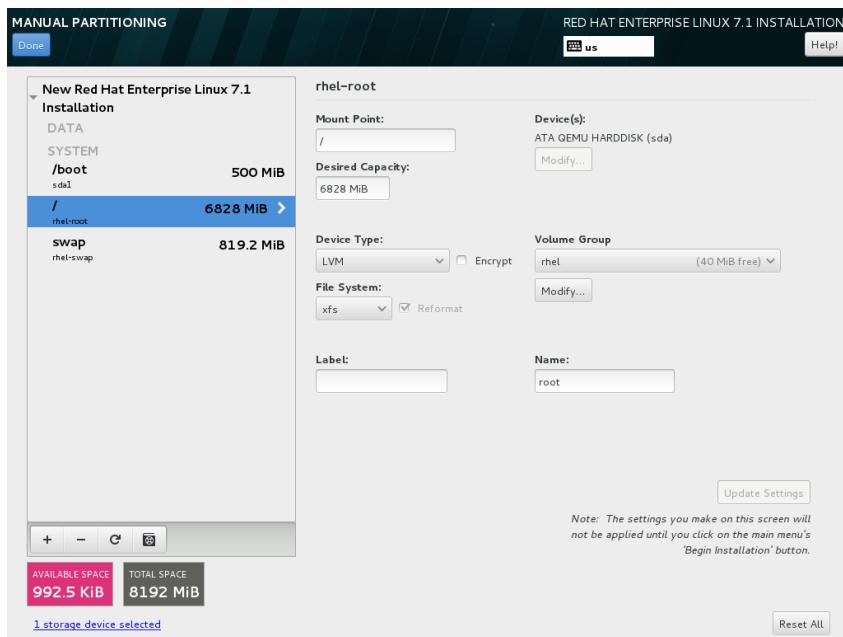
(2) Choosing the default language for the system:



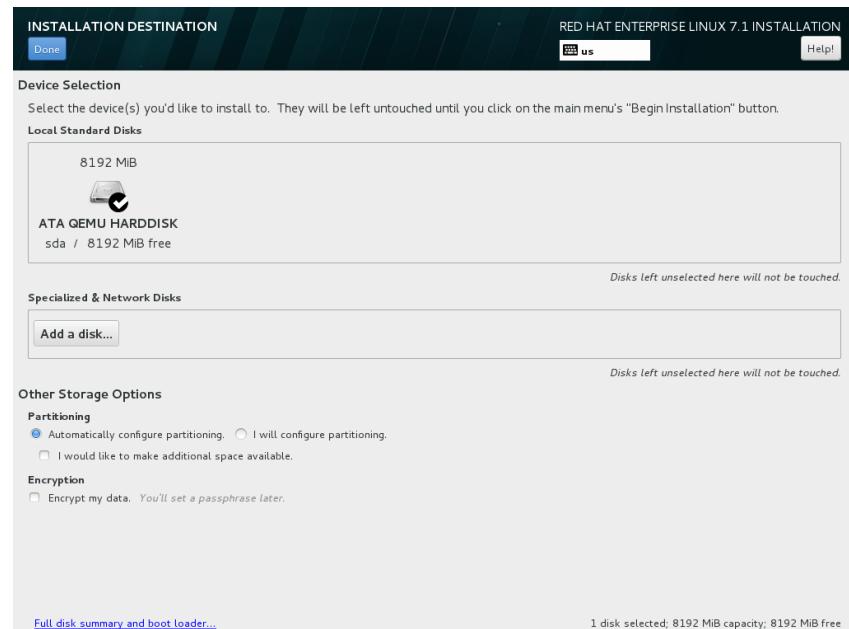
(3) Installation Summary:



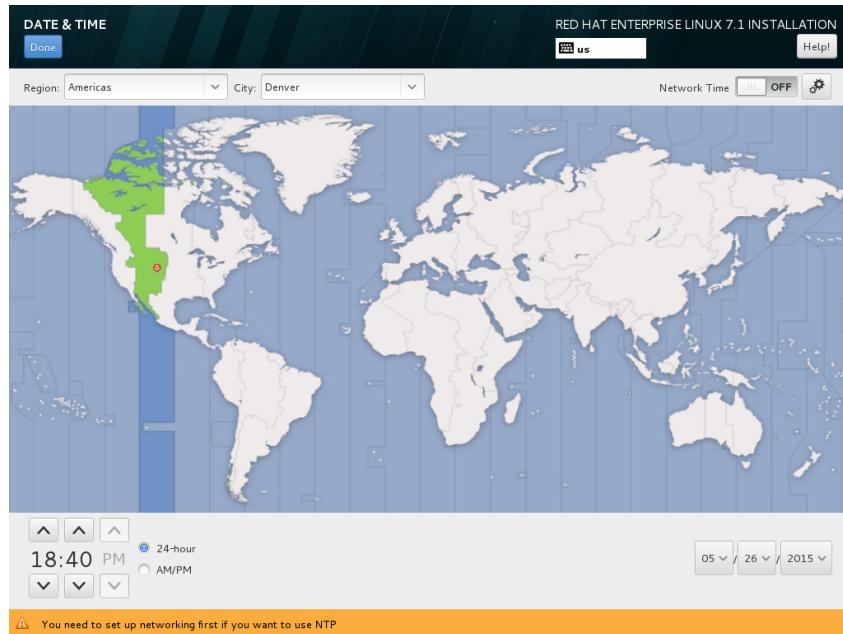
(5) Manual Partitioning:



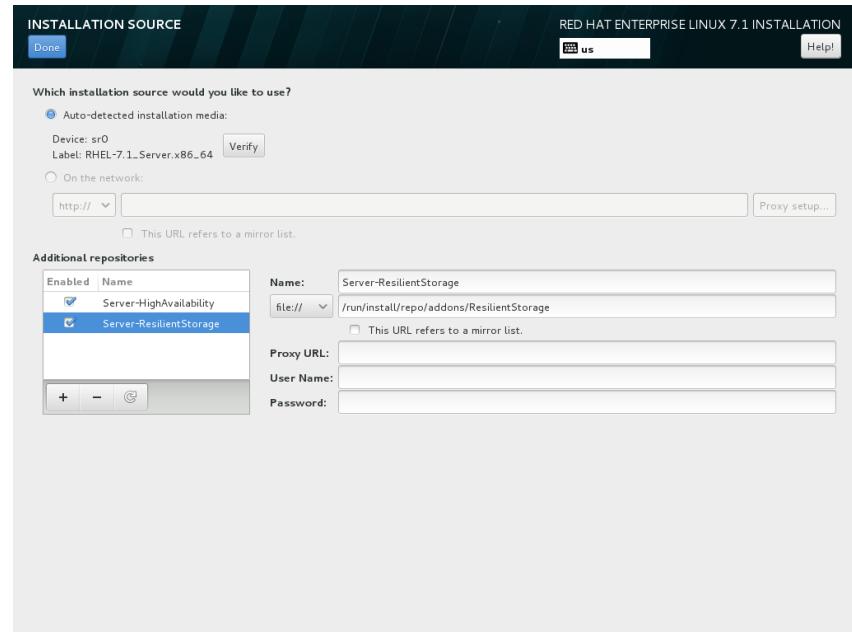
(4) Choosing what storage device to use for installation:



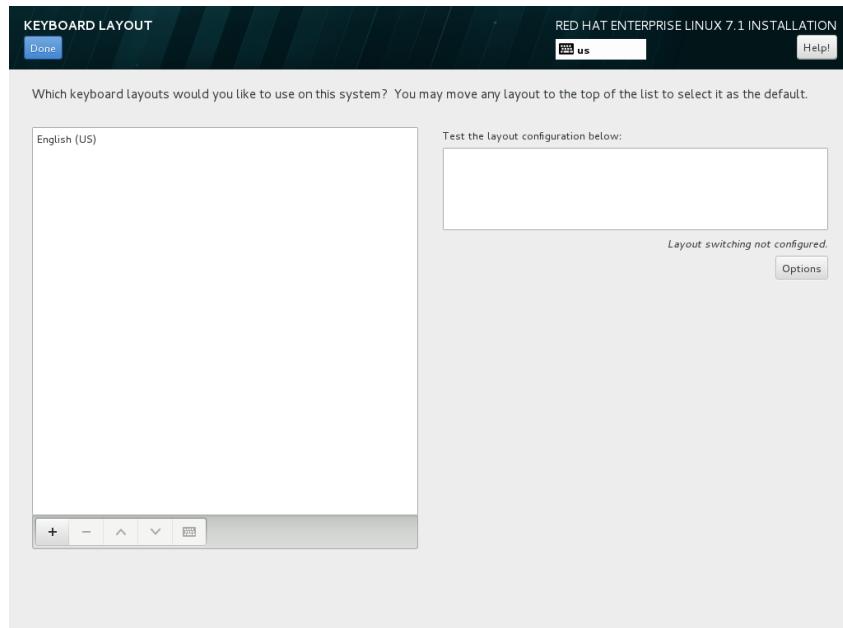
(6) Choosing a timezone and NTP:



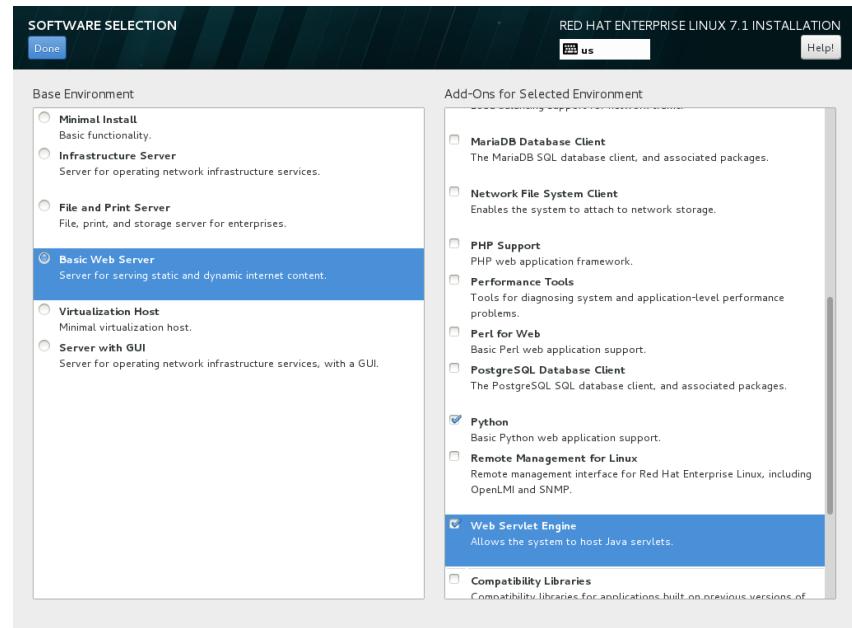
(8) Choosing an installation source:



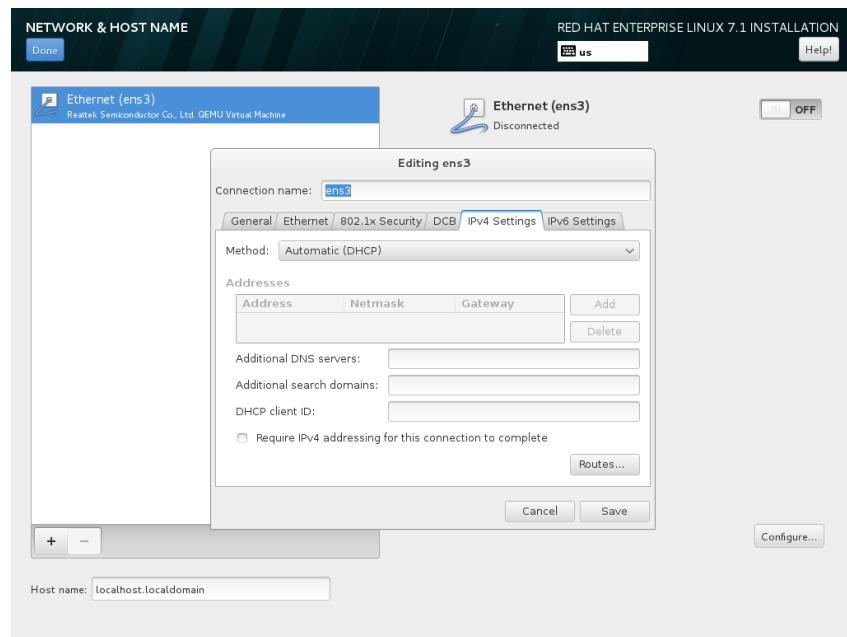
(7) Choosing the default keyboard layout for the system:



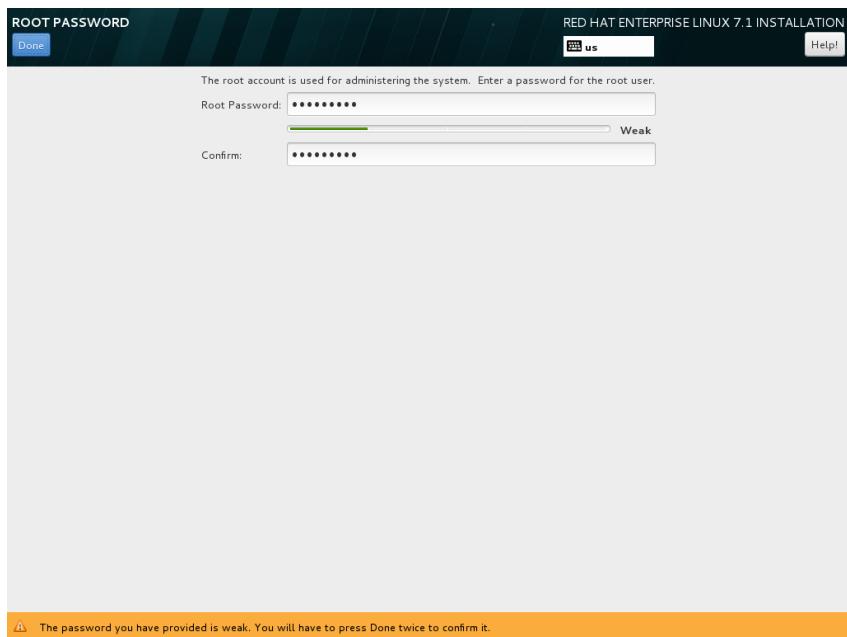
(9) Choosing package groups:



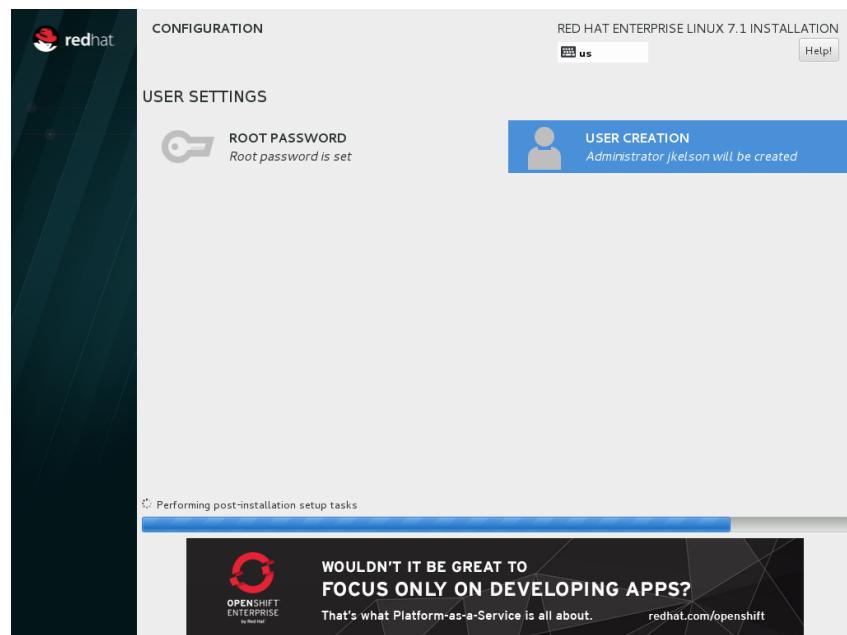
(10) Configure networking and choose a hostname:



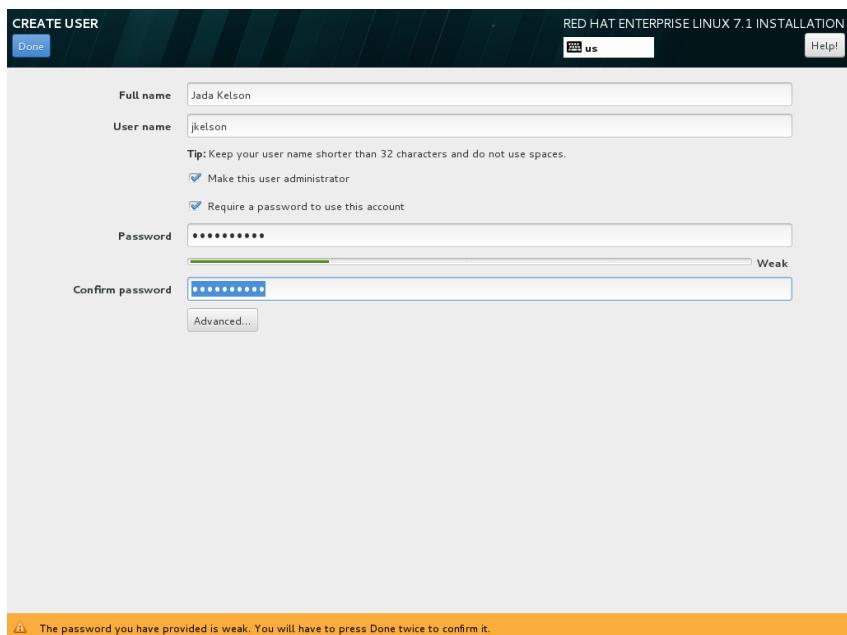
(12) Choosing a root password:



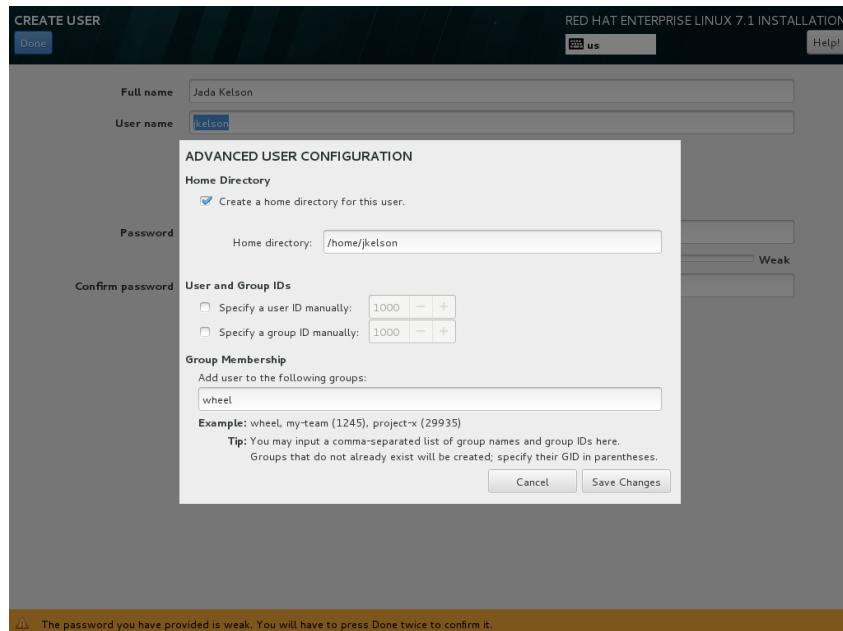
(11) Package installation and user settings overview:



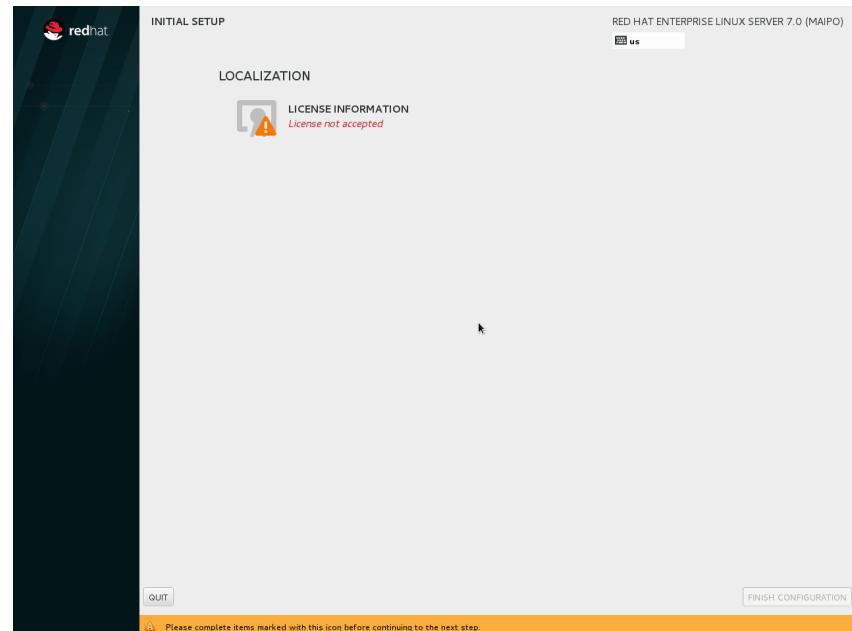
(13) Creating an initial user:



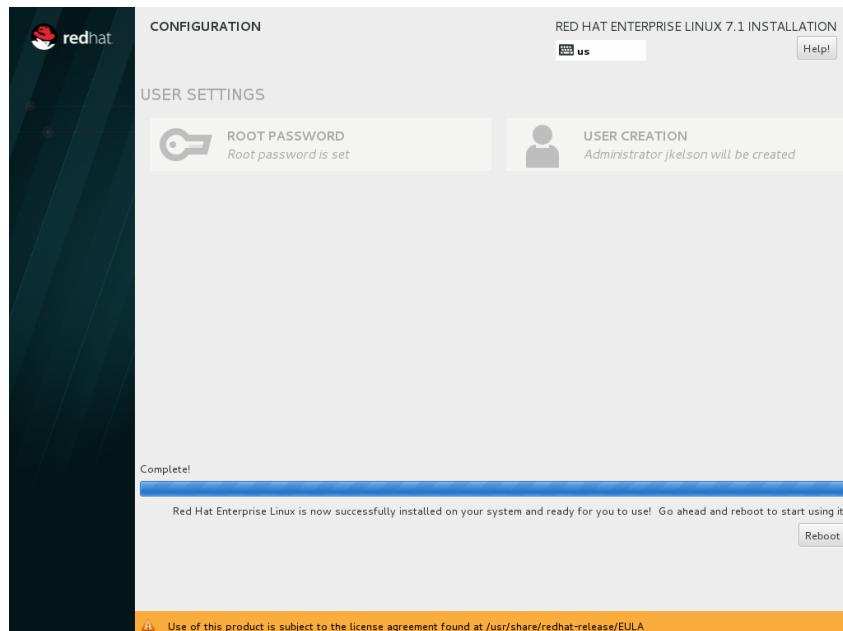
(14) Initial user advanced options:



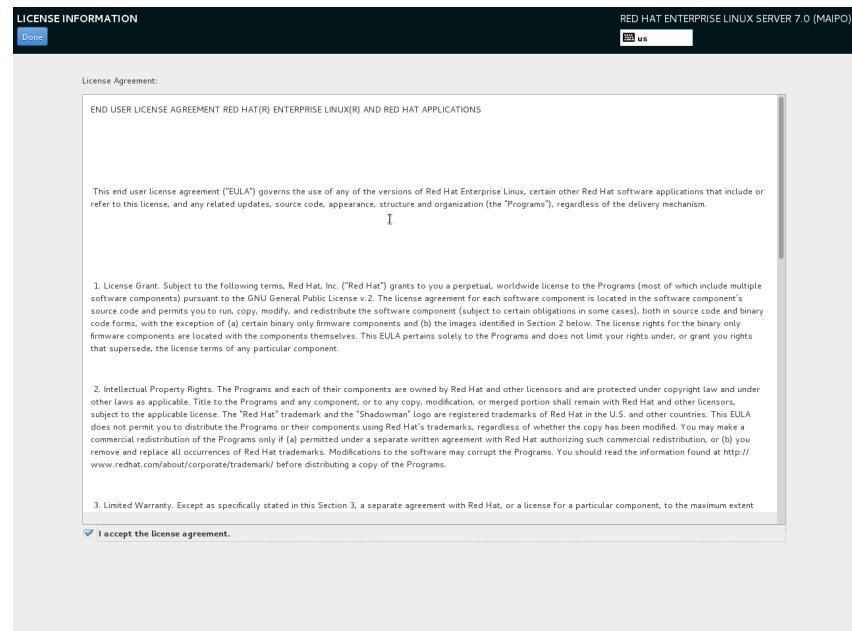
(1) Firstboot setup:



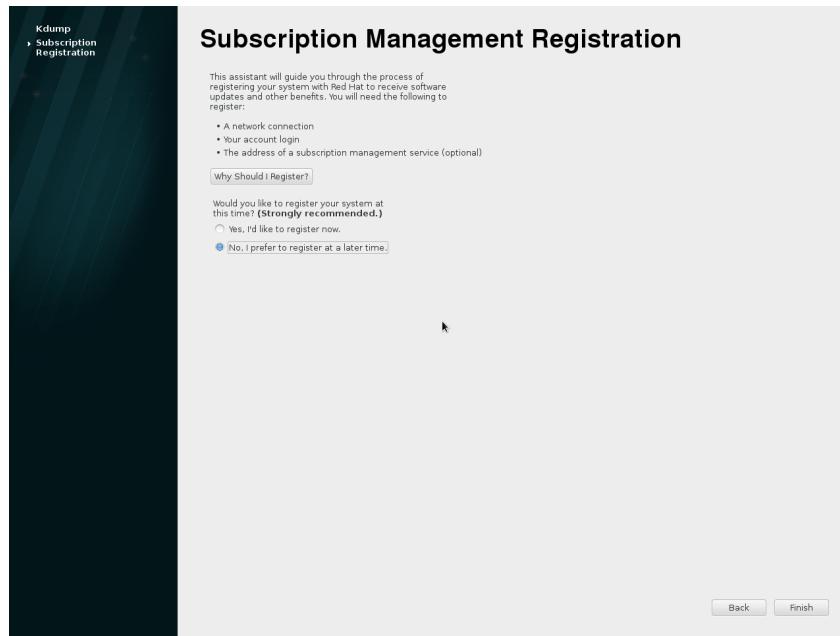
(15) Installation complete and reboot:



(2) Accept the Red Hat Enterprise Linux license:



(3) Registering with Red Hat Network:



Lab 2

Estimated Time:
R7: 55 minutes

Task 1: Linux Installation [R7]

Page: 2-13 Time: 30 minutes

Requirements: (1 station) (classroom server) (graphical environment)

Task 2: Automating Installation with Kickstart [R7]

Page: 2-19 Time: 25 minutes

Requirements: (1 station) (classroom server) (graphical environment)

Objectives

- ❖ Perform a workstation install via NFS with the graphical installer.
- ❖ Configure LVM and software RAID at installation time.

Requirements

- █ (1 station) █ (classroom server) X (graphical environment)

Relevance

Knowing the specifics of an install enables a system administrator to more efficiently prepare deployment options, and avoid extra work after installation.

Notices

- ❖ If this lab exercise is being run within a virtual environment, the use of special keystrokes may be needed to switch between virtual terminals.

- 1) Identify your hostname using the `uname -n` command:

```
$ uname -n  
stationX.example.com
```

Record for future use:

Result: _____

- 2) Select an option based on which boot media is available:

Available Install Media	Goto
Installation DVD	Step 3
A PXE capable system	Step 4

If using the install DVD-based boot method, ensure that the system is configured to boot from the DVD drive.

Goto the step indicated in the table.

Lab 2

Task 1

Linux Installation [R7]

Estimated Time: 30 minutes

3) Boot to the "Installation Method" menu:

Boot the install DVD.

Verify that "Install Red Hat Enterprise Linux 7.x" is selected in the menu.

If necessary, use the arrow key to select the correct menu item.

Skip to Step 5.

4) Boot to the "Installation Method" menu:

Ensure that the DVD drive is empty.

Boot the system using PXE.

The exact procedure is hardware specific. Ask your instructor for details.

5) Use the mouse to **select your preferred language. Click Continue to move to the next screen.**

6) Configure Anaconda Localization:

Under KEYBOARD, **verify** the keyboard layout is correct for your system.

If necessary, check with your instructor. To change your keyboard, **click** KEYBOARD, then click on +. Scroll through the list, **select** the keyboard layout, then **click** Add. **Select** your keyboard (after testing it), and click the up arrow to put it to the top of the list. **Click** Done after choosing a different keyboard.

Choose the correct time zone for your system.

Click DATE & TIME. **Click** your time zone region, or select the Region and City from the toggle menus. **Click** Done after choosing a different time zone.

7) Configure Anaconda for an NFS-based install:

Under SYSTEM, click NETWORK & HOSTNAME.

Click OFF to change it to ON. This should fill in the address, gateway, and DNS information, as well as the hostname.

Verify the network configuration using the table below. Some classrooms may not support DHCP, and will require that this information be added manually. Click Done.

Instead of using Local media, use the NFS shared installation repository.

Under SOFTWARE, click INSTALLATION SOURCE.

Select the On the network: radio button.

Change http:// in the toggle menu to nfs. Enter server1.example.com:/export/netinstall/RHEL7.

Parameter	Value
Hostname	stationX.example.com (see step 1)
IPv4 address	10.100.0.X / 255.255.255.0
Gateway	10.100.0.254
Name Server	10.100.0.254

8) Install a server with a graphical interface. Add the group that includes the autofs package.

Click SOFTWARE SELECTION.

Click the checkbox for Server with GUI. In the Add-Ons for Selected Environment, click the Network File System Client.

Click Done.

9) Remove all existing partitions and physical volumes:

Select "INSTALLATION DESTINATION".

The INSTALLATION DESTINATION is displayed.

Click the I will configure partitioning. radio button. Click Done.

Click the Red Hat Enterprise Linux Server Linux 7.x for x86_64.

This will expand the automatic partitioning configuration.

Select a mount point and click the minus button below to remove it.

Check the box to "Delete all other filesystems".

All partitions should now be removed.

10) Make sure that the New mount points will use the following partitioning scheme toggle menu has LVM selected. Use the add (+) button to build a new partition table as follows:

Type	Mount Point	FS Type	VG Name	LV Name	Size (MB)
Standard Partition	/boot	xfs	<N/A>	500	
LVM Logical Volume	/	xfs	vg0X	root	8672
LVM Logical Volume	swap	swap	vg0X	swap	1024

11) Verify that the device map shows two logical volumes in volume group vg0X, then click Done. Click Begin Installation.

12) As the installation proceeds, click ROOT PASSWORD to makeitso, entering it twice, then click Done.

13) Create a non-root user as follows:

Username	Password
guru	work

Click the Make this user administrator check box.

14) Depending on the system and the network, installation may take anywhere from just a few minutes to a half-hour or more. This could be a good time to take a short break.

15) When the install is done, **remove** the DVD if there is one.

Click Reboot. Some virtual environments may require a system halt so that the hard drive can be selected as an alternative boot device.

16) Work through the first half of FirstBoot Initial Setup:

Click LOCALIZATION.

Click the I accept the license agreement check box, **click** Done, then **click** Finish Configuration.
Agree to the license.

At the Kdump screen, accept the defaults by **clicking** Forward.

Select "No, I prefer to register at a later time." **Click** Finish.
Do not register with RHN.

17) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

18) Configure Yum so that package can be installed:

```
# systemctl start autofs  
# cp /net/server1/export/netinstall/postinstalldata/configs/classroom.RHEL7.repo /etc/yum.repos.d/
```

Objectives

- ❖ Modify the anaconda-ks.cfg kickstart file
- ❖ Install a system using the modified kickstart file.

Requirements

- █ (1 station) █ (classroom server) X (graphical environment)

Relevance

Installing systems manually can be time consuming and tedious. When deploying larger numbers of new systems, the ability to automate the installation saves time and decreases the probability of mistakes. It is useful to modify the existing anaconda-ks.cfg file to be more generic, so that it can be used on a variety of different hardware of the same architecture.

- 1) Copy the Kickstart file created by Anaconda into the guru user's home directory:

```
# cp /root/anaconda-ks.cfg /home/guru/ks.cfg  
# chown guru: /home/guru/ks.cfg
```

- 2) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

- 3) In the ks.cfg file, add text to the top. This will allow for easier troubleshooting if mistakes are detected during the installation process with the modified kickstart file.

File: ~/ks.cfg

System authorization information
+ text

- 4) In the ks.cfg file, disable firstboot.

File: ~/ks.cfg

- firstboot --enable
+ firstboot --disable

Lab 2

Task 2

Automating Installation with Kickstart [R7]

Estimated Time: 25 minutes

- 5) In the ks.cfg file, sanitize the network entry (**--device** should always be used for *each* network option):

File: ~/ks.cfg

```
→ network --bootproto=dhcp --device=eth0 --ipv6=auto --activate→  
      --hostname=stationX.example.com  
- network --hostname=stationX.example.com
```

- 6) In the ks.cfg file, enable autofs:

File: ~/ks.cfg

```
→ services --disabled="chronyd" --enabled="autofs"
```

- 7) In the ks.cfg, modify the partitioning configuration to allow more space for the volume group, use **--recommended** for the swap file size, and create a separate /tmp/ directory:

File: ~/ks.cfg

```
→ part pv.42 --fstype="lvmpv" --ondisk=sda --size=970410000  
→ volgroup vg0X --pesize=4096 pv.42  
→ logvol swap --fstype="swap" --size=1024 --name=swap --vgname=vg0X --recommended  
→ logvol / --fstype="xfs" --size=86725000 --name=root --vgname=vg0X  
+ logvol /tmp --fstype="xfs" --size=1024 --name=tmp --vgname=vg0X
```

- 8) Add reboot to ensure a truly hands-off install:

File: ks.cfg

```
logvol /tmp/ --fstype="xfs" --size=1024 --name=tmp --vgname=vg01  
+ reboot
```

- 9) In the ks.cfg file, reduce and simplify the package groups that are to be used.

File: ~/ks.cfg

```
@core
-
@desktop-debugging
-
@dial-up
@fonts
@gnome-desktop
-
@guest-agents
-
@guest-desktop-agents
-
@input-methods
@internet-browser
-
@multimedia
-
@network-file-system-client
-
@print-client
@x11
+
autofs
    chrony
+
emacs
```

- 10) Enter the following %post script:

File: ks.cfg

```
+ %post
+
+
+ mount -o ro,nolock 10.100.0.254:/export/netinstall/postinstalldata /mnt/
+ if test -f /mnt/configs/classroom.RHEL7.repo
+ then cp /mnt/configs/classroom.RHEL7.repo /etc/yum.repos.d/
+ fi
+ umount /mnt/
+ ) > /root/post.log 2>&1
+
%end
```

- 11) Copy the Kickstart file to server1:

```
$ chmod a+r ks.cfg
$ cp ks.cfg /net/server1.example.com/export/tmp/stationX.ks
```

- Ensures the file is readable by the installer.
- Replace the X with your station number.

If this gives an error, verify that autofs is running: **systemctl restart autofs**.

12) Select an option based on which boot media is available:

Available Install Media	Goto
Installation DVD	Step 13
A PXE capable system	Step 14

If using the install DVD-based boot method, ensure that the system is configured to boot from the DVD drive.

Goto the step you selected from the previous table.

13) Start a Kickstart-based install:

Boot the install DVD.

Type `Esc`.

At the boot: prompt type `linux inst.ks=nfs:server1.example.com:/export/tmp/stationX.ks` `Enter`

Add "ks=nfs:server1.example.com:/export/tmp/stationX.ks" to the Anaconda boot arguments to use the kickstart file for installation.

Skip to Step 15.

14) Start a Kickstart-based install:

Ensure that the DVD drive is empty.

Boot the system using PXE. Add "ks=nfs:server1.example.com:/export/tmp/stationX.ks" to the boot arguments.

The exact procedure is hardware specific. Ask your instructor for details.

15) Depending on the system and the network, installation may take anywhere from just a few minutes, to an hour or more. This could be a good time to take a short break. If the installer returns to the installation menu, then either the `inst.ks=` entry was invalid, or there is a problem in the kickstart file. If needed, check with your instructor so that the kickstart file can be corrected.

After a successful install, if the system reboots back to the graphical installer, halt the system, then boot to the hard drive.

- 16) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 17) Examine the new Kickstart file created by Anaconda:

```
# less /root/anaconda-ks.cfg  
. . . output omitted . . .
```

- Type q to quit.

- 18) Examine the log created by the post-install script:

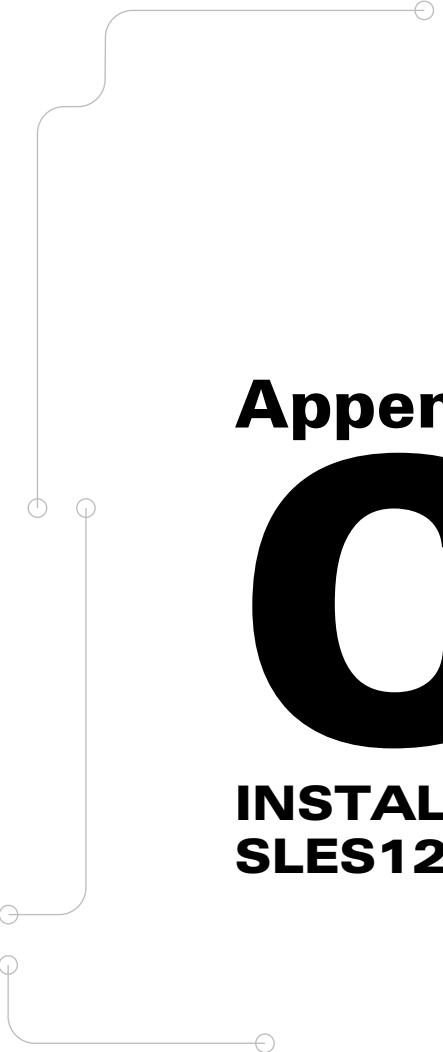
```
# less /root/post.log  
. . . output omitted . . .
```

- 19) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Cleanup

- 20) Ask the instructor if your system should be reinstalled with the standard class image.



Appendix

C

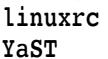
INSTALLING

SLES12

YaST Install Program Interface

YaST is the install program

- Hub and spoke architecture
- Offers both GUI and text-based installs
- Additionally supports serial console and SSH installs
- Two components



The YaST Installer

The SUSE Linux YaST installer is designed in a hub and spoke configuration. From the top level (the hub), it is possible to drill down into each sub section and configure details, then return to the top level hub.

Sub-sections include:

- ❖ System (hardware)
- ❖ Keyboard Layout
- ❖ Partitioning
- ❖ Booting
- ❖ Add-On Products
- ❖ Software
- ❖ Time Zone
- ❖ Language
- ❖ Default Run Level
- ❖ Kdump

Stage One of the SUSE Linux Enterprise Server Install Process

The SUSE Linux Enterprise Server installation process is split into two parts. The first part runs the text/curses based **linuxrc** program. The two main installation tasks of **linuxrc** are:

- ❖ Kernel module loading
- ❖ Installation method selection

Via the kernel modules menu, manual selection of kernel modules is

possible. When performing a network install, it will be necessary to select and load the appropriate kernel module for your network adapter. When performing a hard disk installation, loading an appropriate filesystem driver and possibly an IDE, SCSI, or RAID module will be required. Modules included with SUSE Linux and third-party kernel modules are both supported.

The installation method allows for CD-ROM/DVD, network, or hard disk. For network installations you will be prompted to enter additional information (e.g. DHCP). You can perform network installations via NFS, FTP, HTTP, or SMB/CIFS.

As well as initiating the installation, the **linuxrc** program has additional functionality:

- ❖ System Information
- ❖ Booting off an existing SUSE Linux installation
- ❖ Starting the rescue environment

Through the System Information screen, the **linuxrc** program can be used to inspect hardware details such as available hard disks, PCI devices, processor, memory, interrupts, I/O ports, DMA channels, network cards, and kernel messages.

When an existing SUSE Linux installation's boot loader has become damaged, **linuxrc** can be used to boot the existing installation. If more serious damage or configuration corruption has occurred the rescue environment can be accessed for system repair and recovery.

Network Installation

Often faster than DVD-ROM installation

Requires the creation of a SUSE Linux Network Installation Server

- Copy contents of all DVD-ROMs to server

Clients boot off of floppy images, CD-ROM made with the suse-boot.iso image, SUSE Linux installation DVD-ROM 1, or PXE server

Installation client supports 5 different protocols

- NFS
- FTP/TFTP
- HTTP
- SMB

Creating a SUSE Linux Network Installation Server

To setup a network installation server for SUSE Linux, copy the contents of the DVD-ROM to a directory on the installation server:

```
# mkdir /media/cdrom  
# mount /dev/cdrom /media/cdrom  
# cp -a /media/cdrom/* /location/of/disk/space/ && eject -  
/dev/cdrom
```

Sharing the Directory

The */location/of/disk/space/* directory must then be made accessible via the chosen network installation method. For example, if users need to be able to install over NFS, that directory must be exported via NFS.

NFS is the most commonly used method. This is due, in part, to the fact that it offers the fastest installation times, but also because of the post-install flexibility NFS can provide. If the location containing the installation directory was */netinstall/*, then adding this line to the */etc/exports* file would share it properly:

```
File: /etc/exports  
+ /netinstall *(sync,ro)
```

The YaST instserver module

For a wizard like interface that automates the above procedure, the YaST instserver module creates a network share, then prompts for the media and spools it correctly.

The SLP protocol can be used to announce the availability of SUSE Linux network installation sources. The instserver YaST module can create the SLP service announcement configuration while populating the installation directory.

Media-less Network Installs via PXE

Historically, big Unix systems have offered the ability to perform a network boot before any operating system code is executed. This has traditionally been used to initiate network installations without requiring any floppies or CD-ROMs. The PC world lagged behind the Unix world in this area for a long time. Fortunately, most non-home-use PCs sold since 2001 have followed the PC99 standard by Intel and Microsoft. The PC99 standard, among many details, defines that computers with network interfaces must implement wake-on-LAN and network booting via Intel's PXE (Pre-Execution Environment) standard.

The standard Linux DHCP server and TFTP server can support PXE clients.

Creating a boot.iso Boot CD

SUSE provides the required files that can be used to create a CD-ROM image. This file can be burned to a CD-R.

To create the CD-ROM image change to the /boot/arch/loader/ (where *arch* is commonly i386 or x86_64) directory (on the first installation DVD) and run the command:

```
# mkisofs -o /tmp/suse-boot.iso -b isolinux.bin -c boot.cat  
-no-emul-boot -boot-load-size 4 -boot-info-table .
```

The created image can then be burned using the command:

```
# cdrecord dev=/dev/cdrom -v -eject /tmp/suse-boot.iso
```

SLP for SUSE Linux Installation

Service Location Protocol implementations

- OpenSLP
<http://www.openslp.org/>
- Apple Rendezvous/Bonjour
<http://www.apple.com/macosx/features/bonjour/>

Static SLP service registration files in /etc/slp.reg.d/

Add `install=slp` on the boot: prompt to have `linuxrc` find installation servers via SLP

File: /etc/slp.reg.d/service.reg

```
service:install.suse:nfs://$HOSTNAME/export/→  
    netinstall/SLES12,en,65535  
baseproduct=SUSE CORE  
baseversion=12  
defaultbase=x86_64  
description=SUSE SLES Version 12 x86_64  
label=SUSE SLES Version 12  
machine=x86_64  
vendor=SUSE Linux AG  
version=12
```

Launching a SUSE Linux Installation Using SLP

Because an AutoYaST2 file can not specify an installation source, administrators are forced to type in a long URL on the boot: prompt in order for YaST to be able to find it. The use of PXE allows you to avoid all that typing and perform a truly, fully automatic installation.

However, it is sometimes necessary or desirable to have more than one installation server available. In these cases, PXE can not as easily be used to allow one to select which installation server to use.

A SUSE Linux installation can be instructed to use SLP to find available installation sources. However, this will require user intervention to select which installation source to use, even if there is only one provided on the subnet.

Service Location Protocol

The Service Location Protocol (SLP) allows a system to automatically find services available on the same subnet as the system. The OpenSLP implementation ships as of SLES9 and SL9.2+. Apple's Rendezvous/Bonjour technology is compatible with OpenSLP.

SLP is described in several RFCs:

RFC2608 ⇒ Service Location Protocol, Version 2

RFC2609 ⇒ Service Templates and Service: Schemes

RFC2610 ⇒ DHCP Options for Service Location Protocol

RFC2614 ⇒ An API for Service Location

OpenSLP implements SLP version 2. It is important to remember that SLP only works within a single subnet, unless a proxy server is used to connect separate subnets. OpenSLP has support for such proxies.

SUSE Linux Installation Server SLP Configuration

The Installation Server YaST module creates a static service registration file in the /etc/slp.reg.d/ directory. Files in this directory must end in .reg in order for OpenSLP to pick them up. Note that OpenSLP can not detect if the server (such as NFS) providing the files goes down. This sample can be used to create your own SLP installation server service announcements on any SLP server:

Installation Choices

DVD-ROM

Hard Drive

- Requires locally attached partition
- Supports FAT16/32, NTFS, ext2/3/4, XFS, or Btrfs
- Installation tree can be copied from DVD installation media or from loopback mounted ISO image of the DVD-ROM
- Boot from install media or suse-boot.iso CD-ROM
- Add `install=hd:/<path/to/source>?device=sdaX` at the boot: prompt

Network

Installing SUSE Linux Enterprise Server

A variety of sources can be used for the installation media, and a variety of methods for running the installation program. The YaST installer uses the same configuration code when doing post-install configuration.

SUSE Linux Enterprise Server also offers a variety of advanced installation methods. For example, AutoYaST2 can be used to script and automate installs. Similarly, SUSE Linux Enterprise Server supports installation over VNC or serial consoles, popular when installing on headless servers, or via SSH.

Boot Options

The YaST2 installer supports the passing of options when initiating the installation. These options allow you to provide key details such as the networking information that you want to use during the installation (thus avoiding the later interactive prompts for that information). All options are passed in the form `variable=value`. This list contains examples of useful options with descriptions of their effects:

install ⇒ URL that points to the installation source media. For example, `nfs://server1/export/netinstall/SLES12/`. Note that if usernames and passwords are required, they can be passed in the URL as well.

hostip, netmask, gateway, nameserver ⇒ the IP address information for the host being installed

netdevice ⇒ specify which network interface is used to complete the install (for example `eth1`)

textmode ⇒ set to 1 to start YaST in text mode

usedhcp ⇒ set to 0 or 1 to (dis|en)able DHCP for install (disabled by default)

autoyast ⇒ specifies the URL used to obtain the autoinstall file and causes an autoinstall to be attempted.

rescue ⇒ set to 1 to enter rescue mode

vnc ⇒ set to 1 to cause YaST to start a VNC server

vncpassword ⇒ sets the password for the VNC server

usessh ⇒ set to 1 to have the installer launch an SSH server

sshpassword ⇒ password for the SSH server

insmod ⇒ cause YaST to load the specified kernel module (using any options that are specified)

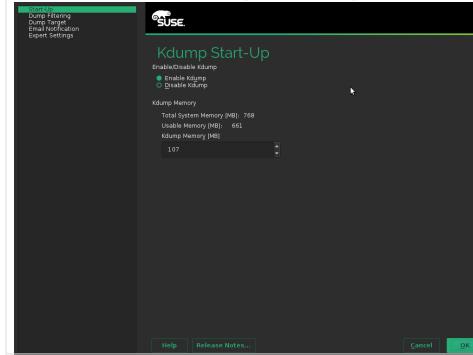
loghost ⇒ IP address of a remote syslog server to log to

Installing From the Local Hard Drive

If Windows or another version of Linux is already installed on the computer, and the computer has free un-partitioned space, SUSE Linux Enterprise Server can be installed by placing the installation tree on the computer's hard drive. The computer should then be booted off of the installation disk 1, or a `suse-boot.iso` CD-ROM, and told to install from the hard drive. The installer will then ask for the partition and directory on the hard drive which contains the installation tree or downloaded ISO images.

Kernel Crash Dump Configuration

Kernel Crash Dump (Kdump)



Saving Memory Images from Kernel Crashes

When a Kernel panic or crash occurs, saving a copy of the memory may greatly assist in determining the cause of the crash.

The Kdump facility is implemented using the kexec feature of the kernel. At boot time, a portion of memory is reserved, and a second, dormant, kernel is loaded into that reserved memory. In the event of a fatal kernel crash, as its last act the crashing kernel executes the dormant kernel. The Kdump kernel has a special initramfs which then saves a copy of kernel memory to /var/crash/. Kdump is actually quite flexible and it can be configured to save the memory image to a remote NFS server, a partition (with the `dd` command), use `scp` to transfer to a remote server, or save to a directory on a filesystem. The actions can be configured in the `/etc/sysconfig/kdump` configuration file. See `kdump(5)`.

If enabled, Kdump modifies GRUB 2, which results in additional `/boot/grub2/grub.cfg` kernel parameters.

Configuring Kdump After Installation

The YaST2 `kdump` module provides a GUI tool to modify and configure Kdump settings after installation.

Network Booting with PXE

Preboot eXecution Environment (PXE)

- Uses DHCP to acquire an address
- Uses TFTP to obtain a boot image

TFTP Server Configuration

- Create and populate boot image directories
- Create the PXE boot configuration file

DHCP returns special options

- filename
- tftp-server-name

Preboot eXecution Environment (PXE)

PXE is a standard that provides for the ability to boot the system from a file that is downloaded at boot time from a network server. Almost all modern business class PCs support PXE either via code on a ROM in the NIC, or directly via the system BIOS.

PXE is also a powerful companion for automated installation because it allows a new machine to be installed without requiring any local media. Plug the new system into the network, initiate a network boot and walk away. Depending on the package selections and network/system speed you can have a fully installed system in a matter of minutes.

Specialized rescue or recovery images can also be served by the PXE server to help when troubleshooting a system. A menu of network bootable images can be defined on the PXE server and sent to the PXE client screen for selection. A default can also be defined with a timeout so that clients configured to netboot can optionally revert to booting from the local disk after some defined interval.

To configure a Linux system to act as a PXE server, the `dhclient` and `syslinux` RPM packages need to be installed.

In addition, on SUSE Linux Enterprise Server the `tftp` package should be installed.

Additional documentation about configuring PXE server support can be found at <http://syslinux.zytor.com/pxe.php>.

Starting the TFTP Server

The TFTP service does not require a configuration file to be started. However, a firewall must allow connections to TCP port 69. The TFTP service runs from Xinetd and can be started as shown here:

```
# chkconfig xinetd on  
# chkconfig tftp on  
# service xinetd restart
```

Preparing for Booting Clients

To support PXE clients, you must populate the TFTP server's directory with the necessary directory paths and files. First create a directory hierarchy to hold the boot files for each OS to be supported:

```
# mkdir -p /srv/tftpboot/linux-install/{RHEL,SLES}
```

Copy the network boot kernel and initial RAM-disk image for each OS from the installation media into its respective directory (the filenames sometimes differ between Linux distributions).

For example, if the SLES12 installation DVD is mounted at `/mnt/cdrom/`, then run:

```
# cp /mnt/cdrom/boot/x86_64/loader/{linux,initrd} /  
    /srv/tftpboot/linux-install/SLES/
```

Copy the main PXE boot image to the TFTP server's directory tree:

```
# cp /usr/share/syslinux/pxelinux.0 /  
    /srv/tftpboot/linux-install/
```

PXE Linux TFTP Configuration

Create the configuration files read by the `pxelinux.0` image. First, create the PXE boot image's configuration directory:

```
# mkdir /srv/tftpboot/linux-install/pxelinux.cfg/
```

Detailed documentation for these files can be found in the `/usr/share/doc/packages/syslinux/syslinux.txt` file.

Second, create a `tftpboot/linux-install/pxelinux.cfg/default` file with content similar to:

File: `/srv/tftpboot/linux-install/pxelinux.cfg/default`

```
+ default 0
+ timeout 100
+ prompt 1
+ display msgs/boot.msg
+
+ label 0
+   localboot 0
+
+ label 1
+   kernel RHEL/vmlinuz
+   append initrd=RHEL/initrd.img ramdisk_size=10000
+     ks=nfs:10.100.0.254:/export/kickstart/RHEL.ks
+
+ label 2
+   kernel SLES/vmlinuz
+   append initrd=SLES/initrd splash=silent
+     install=nfs://10.100.0.254/export/netinstall/
+       SLES usedhcp=1 textmode=1 autoyast=nfs://
+         10.100.0.254/export/netinstall/configs/SLES.ay
```

DHCP Server Configuration

The first thing that a PXE client does is attempt to obtain an IP address. The DHCP server should be configured with an appropriate pool of addresses or static reservations for the clients. In addition to providing IP address information, the DHCP server must indicate the name and location of the boot file that will be retrieved by the PXE client. Since this boot filename should only be sent for requests from PXE clients, not normal DHCP requests, use the conditional functionality built into the ISC DHCP server. For example:

File: `/etc/dhcpd.conf`

```
+ #PXE Specific Boot options
+ class "PXE" {
+   match if substring (option vendor-class-identifier, 0, 9) = "PXEClient" {
+     next-server "10.100.0.254";
+     filename "linux-install/pxelinux.0";
+   }
+ }
```

Additional example configuration options are described in the `/usr/share/doc/packages/syslinux/pxelinux.txt` file.

Finally, create the `tftpboot/linux-installmsgs/boot.msg` file with content like:

File: `/srv/tftpboot/linux-installmsgs/boot.msg`

```
+ PXE Network Installation
+ Enter number of the OS you wish to install:
+ 0. Boot Local Machine
+ 1. Install RHEL
+ 2. Install SLES
```

Creating AutoYaST2 Files

Created completely from scratch "by hand"

- Excellent documentation exists

YaST2 module – autoyast

- GUI tool for creating autoinst.xml

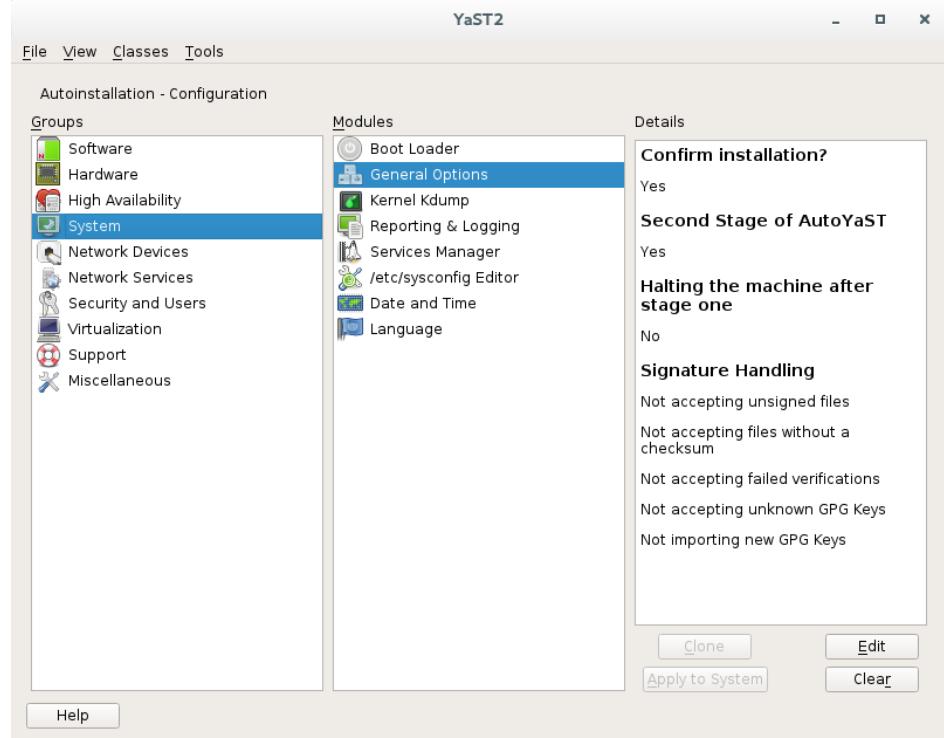
Making AutoYaST2 Files

AutoYaST2 files are XML files. Though creating an AutoYaST file entirely by hand, in your favorite text editor, can be done, the complexity of the format makes this unwieldy. Instead, create the file using the autoyast YaST module in either text or graphical mode. Once a file exists (See Clone System Configuration during installation), it can be edited by hand, or updated using the YaST2 autoyast module.

The autoyast module provides a nice point-and-click interface to select all options which can be configured in an AutoYaST2 file. Additionally, you can load existing AutoYaST2 files into the autoyast module for editing. These files are normally stored in the /var/lib/autoinstall/repository/ directory, though this path can be changed.

YaST2 autoyast module

The YaST2 autoyast module can also import the older ALICE autoinstallation info files and Red Hat's Kickstart files. You can then edit the parameters and save your new AutoYaST2 file.



Using AutoYaST2 files

AutoYaST2 file location specified manually

- From floppy disk
- Embedded into custom CD-ROM
- Over network (using boot.iso boot media)

AutoYaST2 file location obtained via DHCP

- DHCP option "next-server"
- DHCP option "filename"

Using Local AutoYaST2 Files

Once an AutoYaST2 file has been created, the YaST installer still has to be configured to use it. The installer can retrieve the file over the network, or it can load it from a local disk or thumb drive.

To start an AutoYaST2 install using a local device, name the file autoinst.xml, and copy it to the respective device (e.g. a thumb drive: /dev/sdb1). At the boot: prompt type:

```
boot: autoyast=sdb1://path_to_autoyast_file
```

The file can be taken directly from the filesystem using the autoyast=file://path/autoinst.xml parameter with the path to the file.

Using Network AutoYaST2 Files

Retrieving the AutoYaST2 file over the network is even simpler. To initiate a network AutoYaST install at the boot prompt, give it the autoyast option:

```
boot: autoyast=nfs://server/path/to/autoyast/file.ay
```

The installer will retrieve the script via NFS. For HTTP, NFS or TFTP retrieval, the install clients must be using DHCP to configure their network interface. Standard FTP is not supported for retrieval of an AutoYaST file, only for accessing the installation media.

Specifying AutoYaST2 File Location with DHCP

Some DHCP servers can be configured to push the name and location of the AutoYaST2 file to install clients automatically as the clients obtain their DHCP leases. If you are using the ISC DHCP server under SUSE Linux Enterprise Server (the default) then you can add the following configuration options to the /etc/dhcpd.conf file (using the IP address and AutoYaST2 file path correct for the network in use):

```
File: /etc/dhcpd.conf
next-server 10.100.0.254;
filename "/export/autoyast/stations.ay"
```

The IP address specified is the address of the NFS server that has the AutoYaST2 file. This does not need to be the same as the server that is sharing the installation packages.

The filename specified is the name of the AutoYaST2 file that should be retrieved. If there is no filename specified, only a directory, then YaST will attempt to use an AutoYaST2 file in that directory whose name is based on the IP address of the computer being installed. For example, if the IP address was 10.100.0.1 then it will attempt to retrieve the /path/to/autoyast/640001 file.

Even if the DHCP server is correctly configured to send these options, you must still make **linuxrc** aware that an AutoYaST2 installation is desired by passing the **autoyast** option at the boot prompt.

linuxrc Automation

linuxrc usually requires input to configure the network
Options linuxrc needs to be fully automatic

- usedhcp
- install

Pass options to linuxrc using

- the boot prompt
- info file
 - info=cd:/pathTo/info
 - info=http://example.com/pathTo/info
 - info=disk:/pathTo/info
- embedded within the AutoYaST2 file
- custom initrd

Fully Automating Automated Installations

Once the AutoYaST2 file is complete and ready to use, automated installations will take place with little to no user or administrator interaction at all. However, the **linuxrc** program needs to know how and where to find the installation media along with the AutoYaST2 file. **linuxrc** is the initial program that loads, configures hardware, and identifies the YaST repository. A more detailed explanation can be found at <http://en.opensuse.org/Linuxrc>.

There are two options which **linuxrc** needs in order to initiate installations without user interaction. These are the `usedhcp` and `install` options.

The `usedhcp` option tells **linuxrc** whether or not to automatically configure the first NIC in the system and attempt to configure it using DHCP. The default value for `usedhcp` is 0.

The `install` option provides **linuxrc** with the location of the installation media. This is the location that **linuxrc** will use to load the YaST installation program from. It will also pass this value to YaST for use, though it may be overridden by the user during installation configuration for manual installations, or by the AutoYaST2 file for automated installs.

Passing Options To linuxrc

There are several ways to get options to **linuxrc**. The first method is to specify them on the kernel boot line. When a Linux kernel sees options that it does not understand it will pass those options on to whatever program it starts. This is usually `/sbin/init` for a running system, but for the installation kernels this would be **linuxrc**.

Here is an example that would allow **linuxrc** to find and launch YaST without user interaction:

```
install=nfs://10.100.0.254/path/to/media usedhcp=1
```

When **linuxrc** starts, a file named `info` can be used, which contains options that would be specified in the following form:

File: `info`

```
install: nfs://10.100.0.254/path/to/media
usedhcp: 1
```

Typically, the `info` file is specified using the boot parameter `info=`, which can point to a CD, web server, or local disk. An `info` file can also be embedded within an AutoYaST2 file. In this case the AutoYaST2 file must be named `info` and must be in the `initrd`.

Installation Diagnostics

Virtual Terminal	Contents
<code>Ctrl</code> + <code>Alt</code> + <code>F2</code>	Installation Dialog
<code>Ctrl</code> + <code>Alt</code> + <code>F3</code> , <code>Ctrl</code> + <code>Alt</code> + <code>F5</code> , <code>Ctrl</code> + <code>Alt</code> + <code>F6</code> , <code>Ctrl</code> + <code>Alt</code> + <code>F9</code>	Shell Prompt
<code>Ctrl</code> + <code>Alt</code> + <code>F3</code>	Install log (from installation program)
<code>Ctrl</code> + <code>Alt</code> + <code>F4</code>	System-related Messages
<code>Ctrl</code> + <code>Alt</code> + <code>F7</code>	X Graphical Display

Installation Virtual Terminals

The shell prompt can be very useful for getting out and running commands during the install. One useful technique is to `chroot` to the root filesystem that is being installed. When using the shell prompt during install, care must be taken to avoid interfering with the installation process.

The shell prompts are only available after the source installation media has been selected and successfully found.

Post-Installation Virtual Terminals

Virtual terminals are also available after installation. By default, the system runs gettys on the first six virtual terminals (providing text logins). If an X server is started then it will bind to the next available `tty` (virtual terminal number seven by default).

In addition to the standard key bindings (such as `Ctrl` + `Alt` + `F2`) to switch between virtual terminals, the root user can also use the `chvt` command to switch between virtual terminals.

YaST Installer Log files

During installation a detailed log file is recorded. To view the last 10 lines of the log file, change to a virtual terminal with a shell and run the command:

```
# tail /var/log/YaST2/y2log
```

After The First Reboot

Tasks

- Subscribe to the SUSE Customer Center
`yast2 online_update`
- Other tasks not done during installation

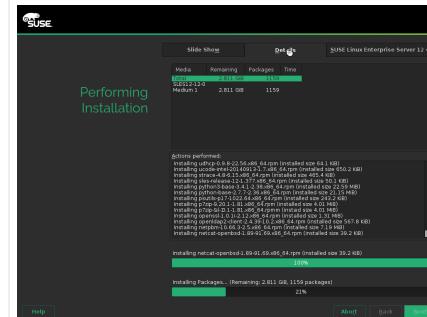
After The First Reboot

After the initial installation has completed, the system will reboot. Most tasks skipped during installation, such as configuring the network (including hostname), firewall, and additional hardware (e.g. printer) can be configured after the first reboot. The release notes can be found on the installed system in the `/usr/share/doc/release-notes/` directory.

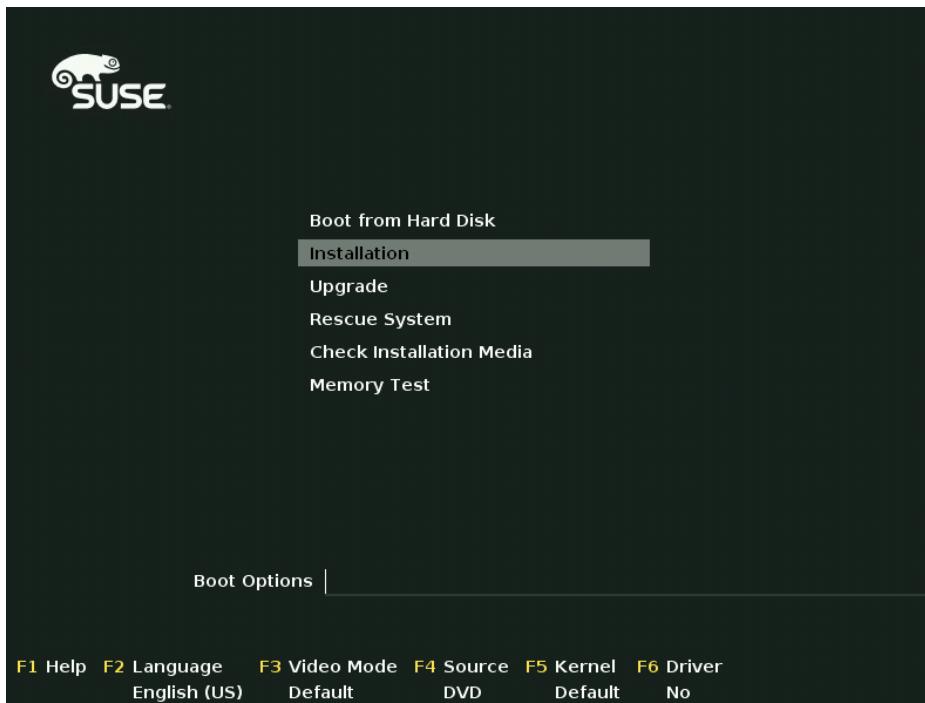
Yet another Setup Tool (YaST) can be used for configuration of any of these tasks, as well as many tasks performed during installation, as follows:

Task	Tool
Subscribe to the SUSE Customer Center update service	SUSEConnect
Change Password Encryption Type	yast2 security
Create a non-root user, including authentication methods	yast2 users
Printer Configuration	yast2 printer
Network Configuration	yast2 lan
Sound Card Configuration	yast2 sound
Date, Time, NTP	yast2 timezone (see <code>/etc/sysconfig/clock</code>)
Keyboard Configuration	yast2 keyboard
Language Configuration	yast2 language
Software Configuration	yast2 sw_single

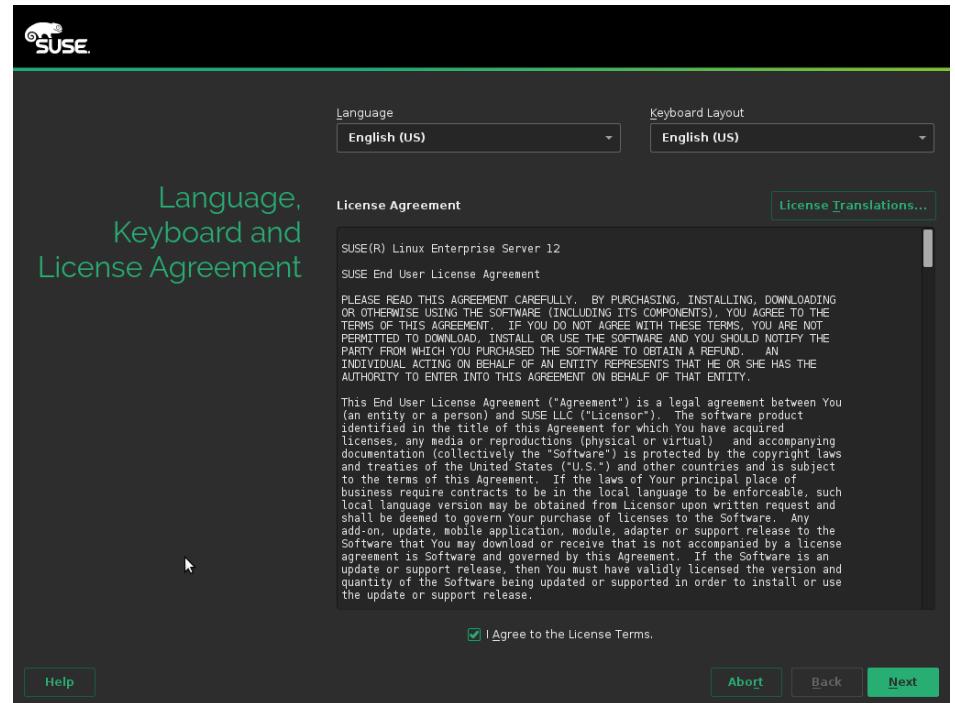
A Typical Install



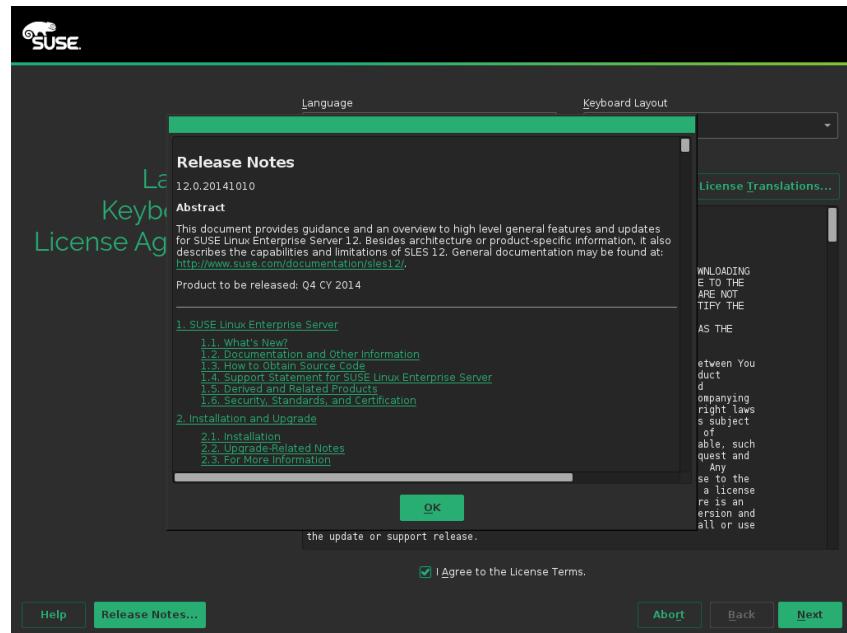
(1) Choosing boot options at the ISOLINUX menu:



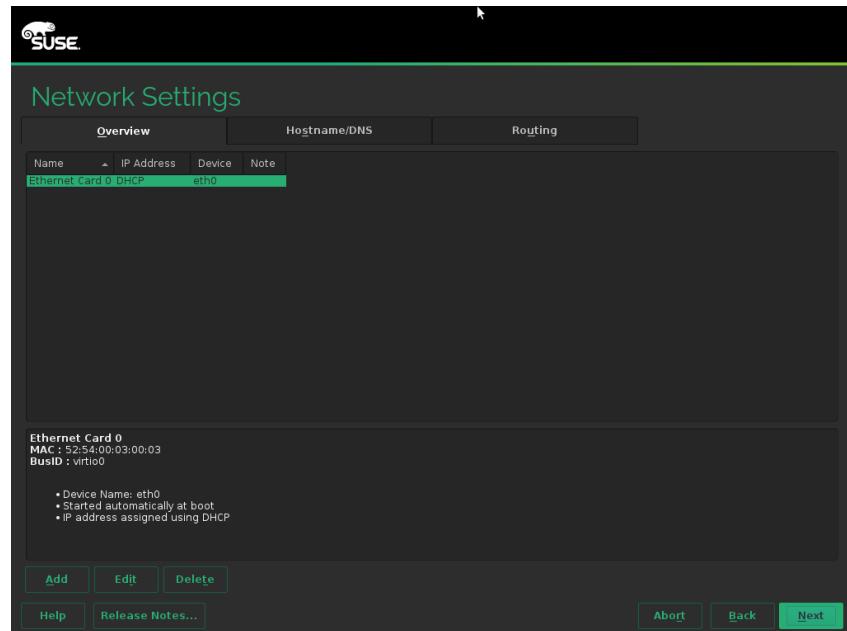
(2) Welcome:



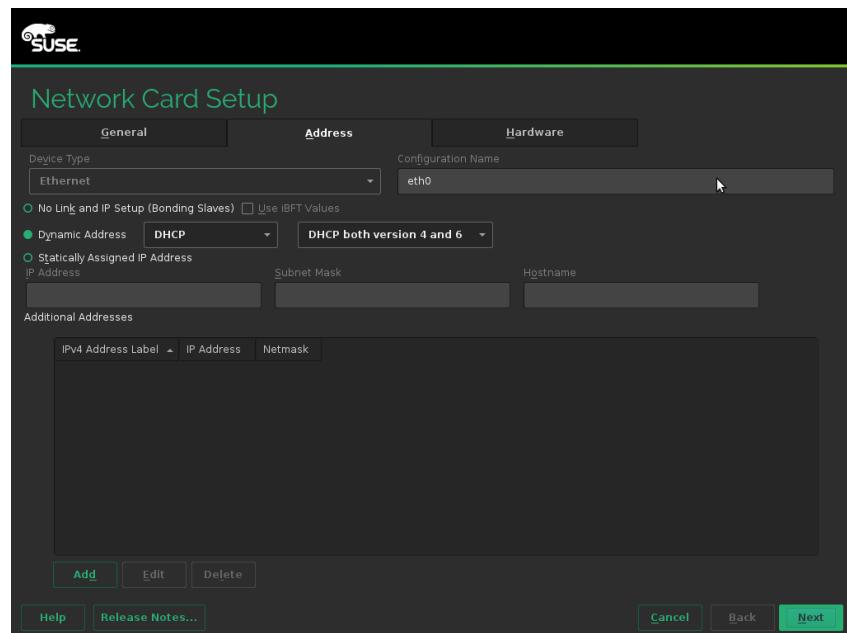
(3) Release Notes:



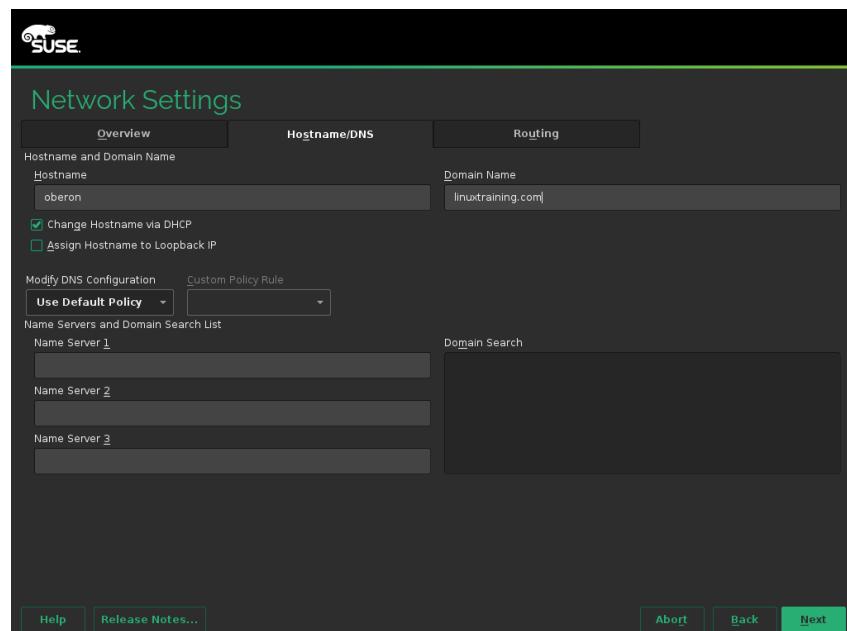
(4) Network Configuration:



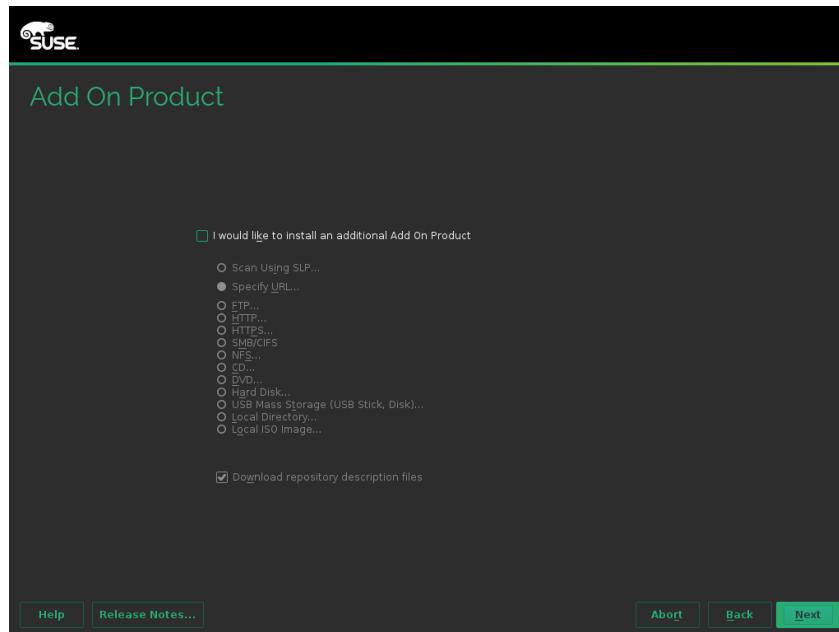
(5) Network Card Setup:



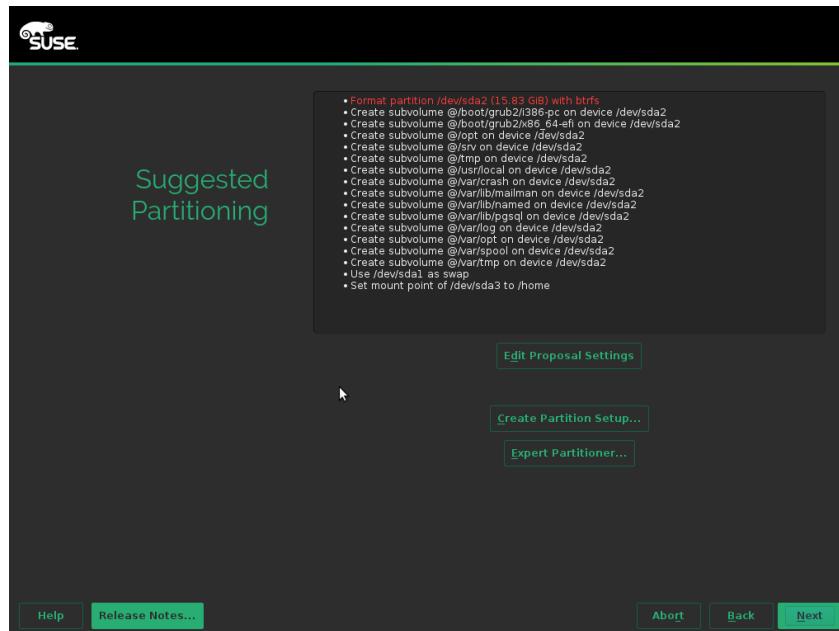
(6) Network Settings (Hostname/DNS):



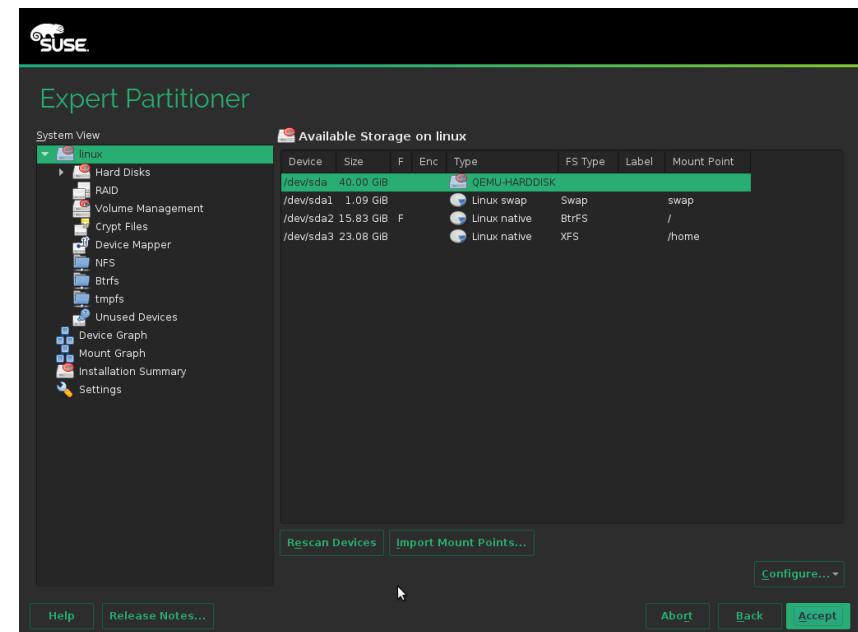
(7) Add On Product:



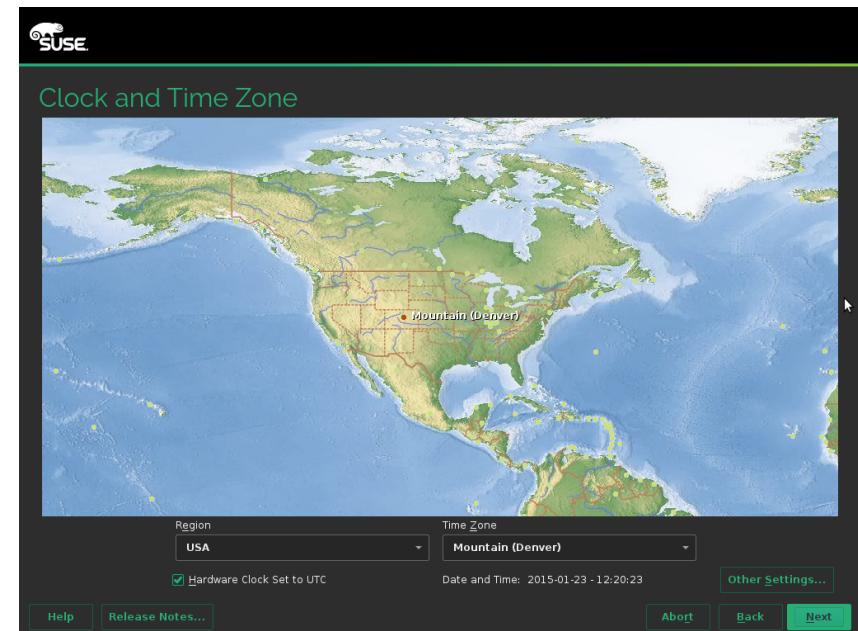
(8) Suggested Partitioning:



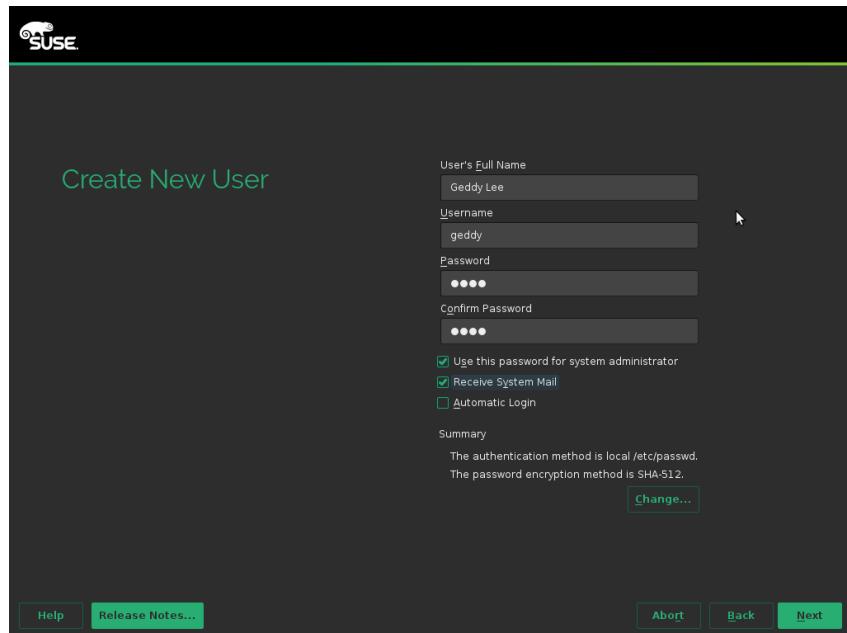
(9) Expert Partitioner:



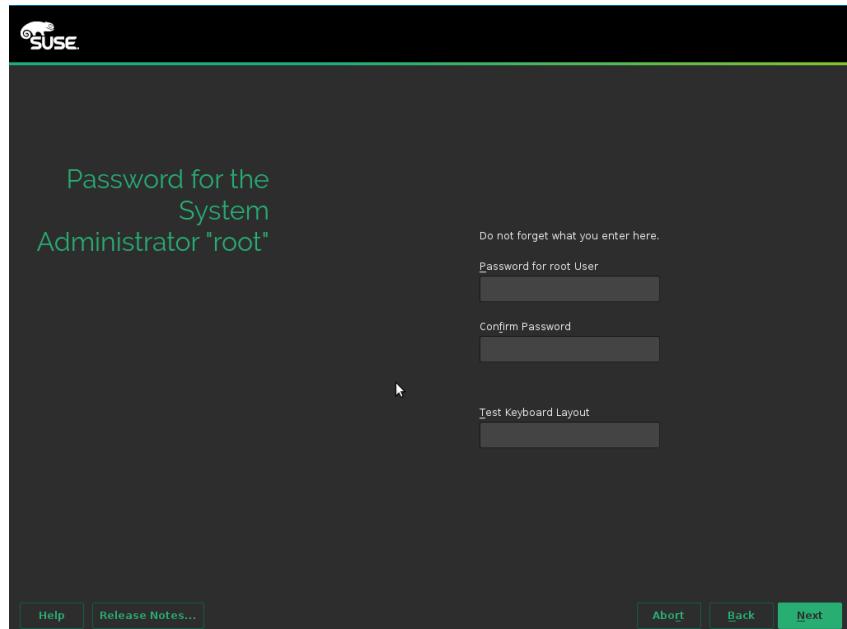
(10) Clock and Time Zone:



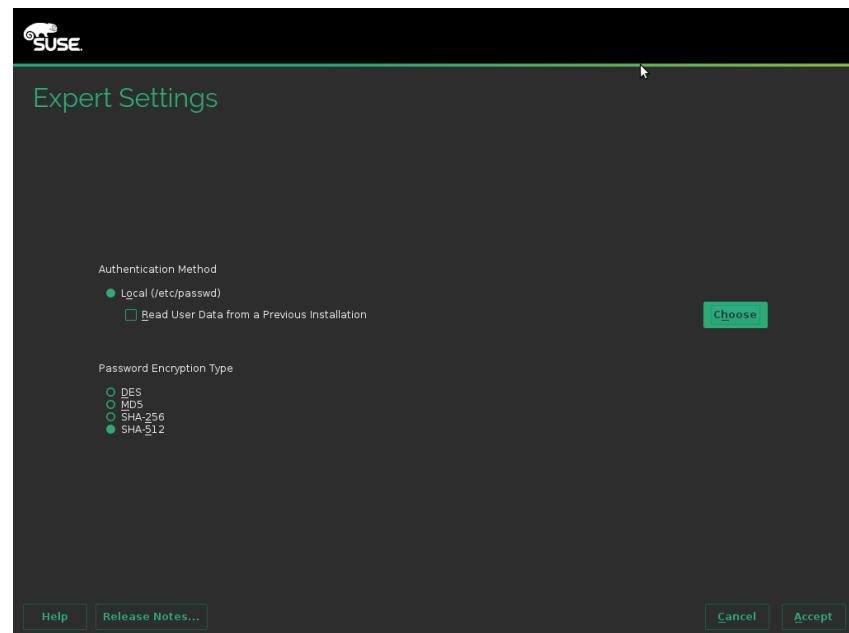
(11) Create New User:



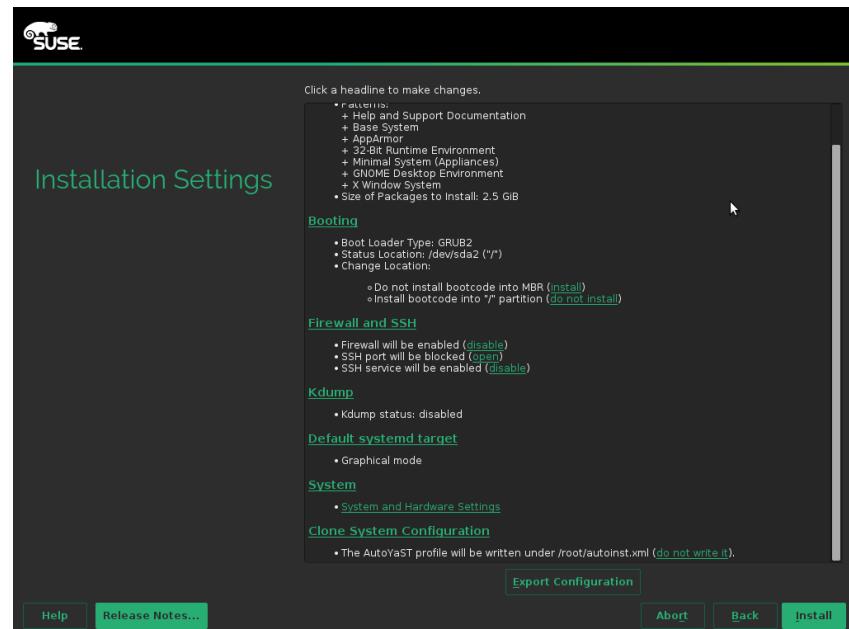
(12) Password for the System Administrator (root):



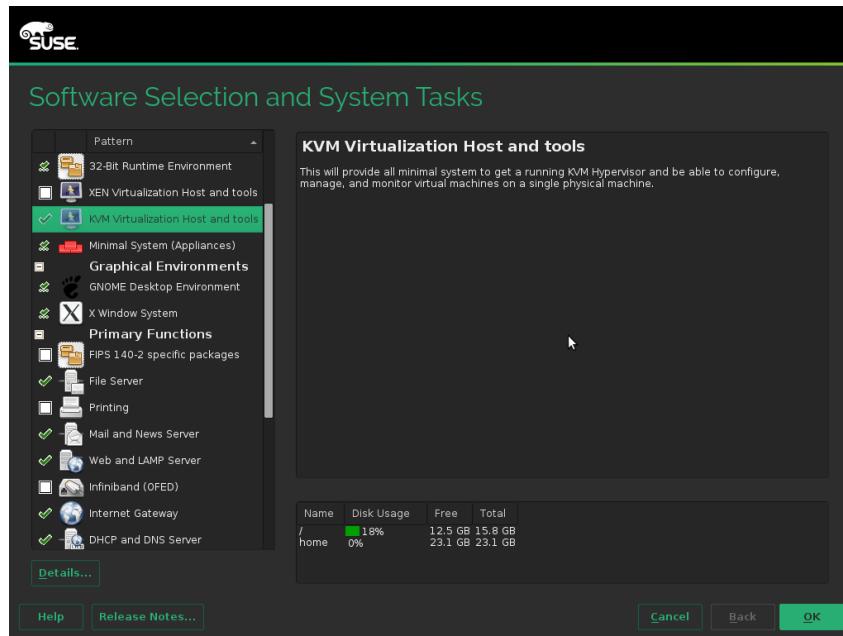
(13) Expert Settings (Authentication Method/Password Encryption):



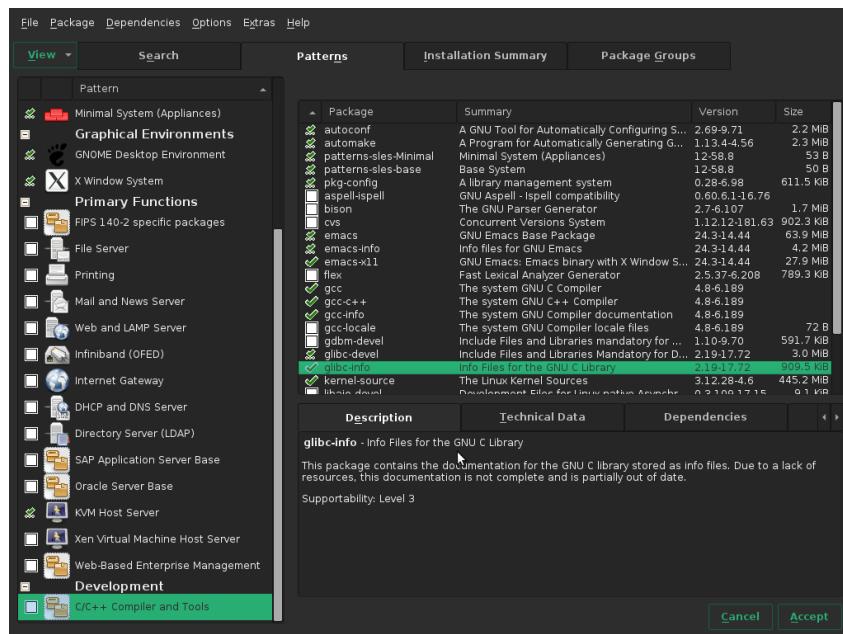
(14) Installation Settings:



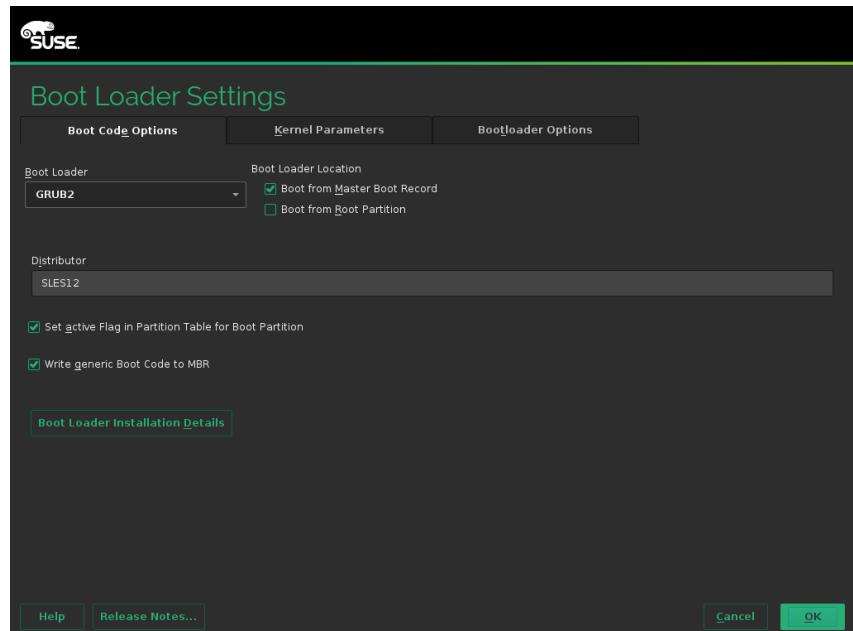
(15) Software Selection and System Tasks:



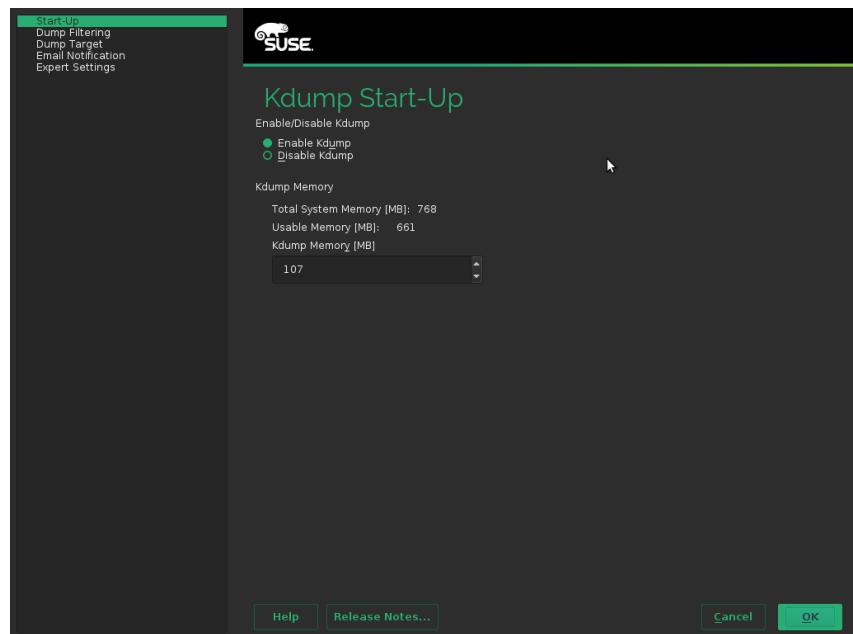
(16) Software Selection Details:



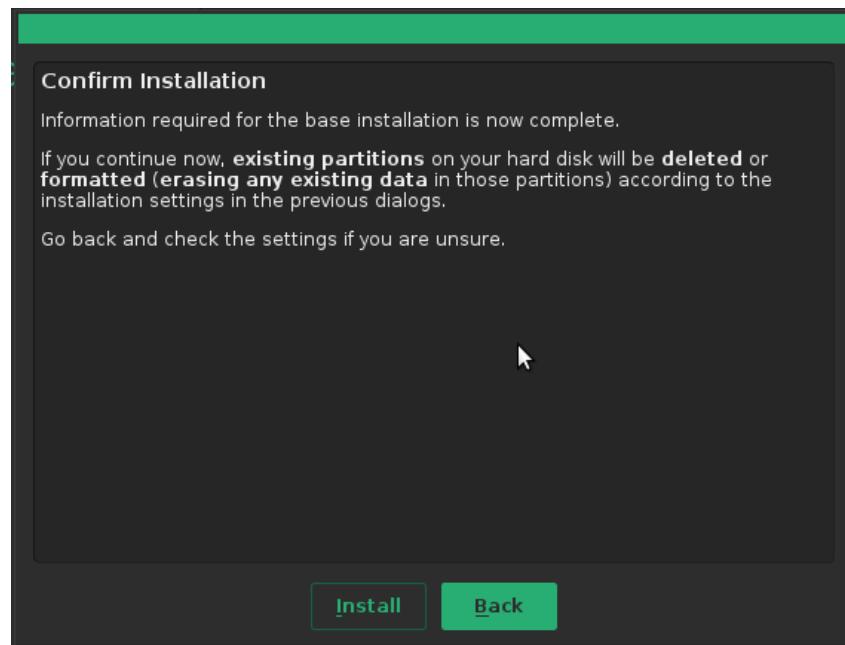
(17) Boot Loader Settings:



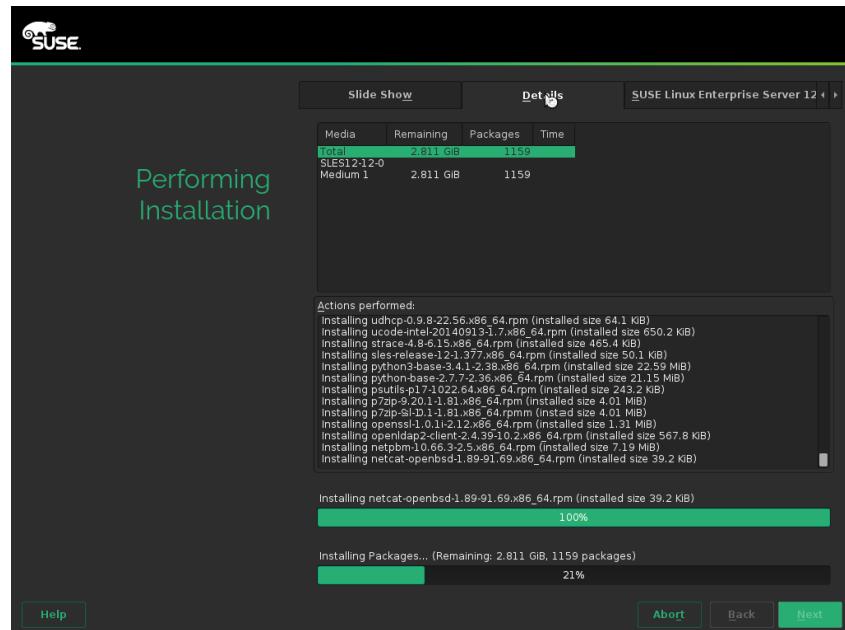
(18) Kdump Start-Up:



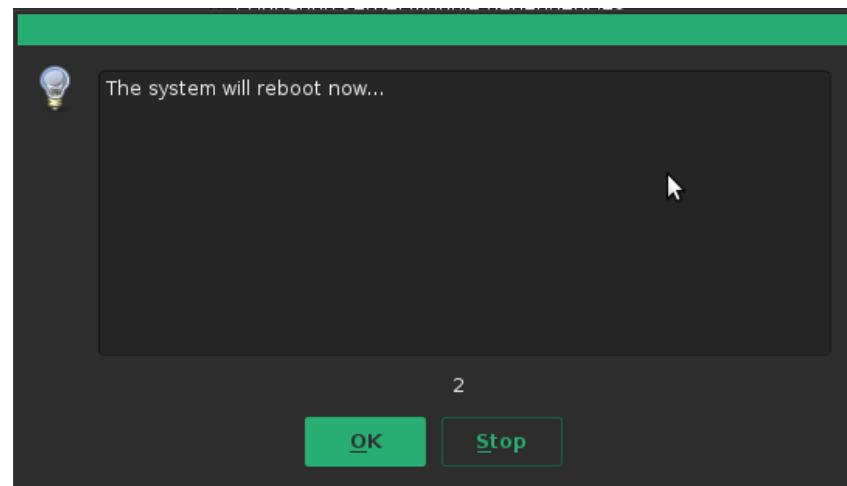
(18) Confirm Installation (the point of no return):



(19) Performing Installation:



(20) Installation Completed:



Lab 3

**Estimated Time:
S12: 90 minutes**

Task 1: SUSE Linux Enterprise Server Installation [S12]

Page: 3-26 Time: 45 minutes

Requirements: (1 station) (classroom server) (graphical environment)

Task 2: Automating Installation with AutoYaST [S12]

Page: 3-30 Time: 45 minutes

Requirements: (1 station) (classroom server) (graphical environment)

Objectives

- ❖ Perform a workstation install via NFS with the graphical installer.

Requirements

- █ (1 station) █ (classroom server) X (graphical environment)

Relevance

Installing the operating system is a basic system administrator duty. The machine should be configured at install time with the packages appropriate for the system role. Some things, partitioning for example, are much easier to configure at install time.

Notices

- ❖ This lab installs SLES12 over a network. This requires the classroom installation server, and either PXE (requires DHCP on the server) or the SLES12 installation DVD for the lab systems.

1) Boot to the SUSE Linux Enterprise Server installation Welcome screen. This is done by booting using PXE or booting to the SLES12 installation DVD. If booting PXE, skip to 3.

2) To use the installation server from the DVD, at the Boot Options prompt of the Installation menu item, type the following then Press **Enter**.

Boot Options **install=nfs://10.100.0.254/export/netinstall/SLES12 hostip=10.100.0.X/24**

Skip to step 4.

3) To boot PXE, use the following basic instructions (ask your instructor for help if needed, as system requirements may vary):

Press **F2** to bring up the PXE boot menu.

Different motherboards vary. Ask the instructor for assistance if needed.

Select the network **boot** option to boot from the NIC.

As the PXE-boot begins, there should be messages on the screen regarding DHCP, after which a menu of available distributions and installation options should appear. If there is more than one NIC available, ask the Instructor which one to use. If no NICs are available, ask the Instructor to reboot and enable the NIC PXE option in the BIOS configuration.

From the class installation menu, type the number under the **Manual Install** column for the distribution which is to be installed on this system. Press **Enter**.

Lab 3

Task 1

SUSE Linux Enterprise Server Installation [S12]

Estimated Time: 45 minutes

- 4) Verify the Language and Keyboard Layout toggle menus are suitable for your installation environment, typically English (US). Click I agree to the License Terms, then click Next
- 5) Click the Release Notes... button at the bottom of the screen to review. Click OK when done.
- 6) At the Registration screen, there is an option to configure the network. Click Network Configuration... at the top right of the screen.
- 7) By default the system will use DHCP to get its hostname, IP address, and default routing information. In the Overview tab you can see the default settings for the ethernet cards. To use these default settings click Next and skip to step 9, otherwise continue with the next step.
- 8) To de-select DHCP click on your ethernet card in the Overview tab then click on Edit. Click on the Statically Assigned IP Address radio button then enter the following information: **10.100.0.X** for the IP address, **255.255.255.0** for the Netmask, and **stationX.example.com** for the Hostname. Click next. Click on the Hostname/DNS tab. Enter **stationX** for the hostname, **example.com** for the Domain Name, and **10.100.0.254** for Name Server 1. Click on the Routing tab. Set the Default Gateway as **10.100.0.254**. A summary of the network settings is available in the table below. Click Next.

Parameter	Value
IP address	10.100.0.X
Netmask	255.255.255.0
Default gateway (IP)	10.100.0.254
Search domains	example.com
Primary nameserver	10.100.0.254
IP of NFS server	10.100.0.254

- 9) Click Skip Registration. Select Yes when prompted to verify skipping registration.
- 10) The next prompt will ask if you would like to install additional add-on products.
Click Next.
- 11) SUSE Linux Enterprise Server will offer a default partition and filesystem scheme. To make any changes you can click on Edit Proposal Settings, Create Partition Setup..., or Expert Partitioner. It is best to configure the partition layout before altering the selection of software packages to install, so click on Expert Partitioner.
- 12) Remove all of the existing partitions first, so as to start with a clean slate:

Under System View, expand Hard Disks.
Select sda. Any existing partitions are displayed.
Click on the Expert button and select Create New Partition Table.
A pop-up dialog opens which allows you to choose the partition table format.
Select MSDOS and click OK.
A display will warn that continuing will remove all data on /dev/sda/ and all RAIDs and volume groups using partitions on /dev/sda.
Click Yes.
- 13) Use the Add button to specify the new partition table setup listed here:

Device	Type	Size	Role	Filesystem	Mount Point
/dev/sda1	primary	512MB	Data and ISV Applications	xfs	/boot
/dev/sda2	primary	Maximum Size	Raw Volume (unformatted)	LVM	Blank

14) Once these partitions have been created, setup logical volumes:

Click Volume Management to setup a volume group and logical volumes.

Click Add then **click** Volume Group.

Enter **vg0** into the Volume Group Name dialog.

Select /dev/sda2 then **click** Add.

Click Finish.

Click Add and select Logical Volume.

15) Setup the following logical volumes:

Name	Size	Role	File System	Mount Point
root	10GB	Operating System	xfs	/
swap	512MB	Swap	Swap	swap

Click Accept when done, then Next.

16) The time zone settings affect both how the system and hardware clocks are set, and the display of time by many parts of the system and applications. On the Clock and Time Zone screen:

Select the time zone as appropriate.

Usually, the location of the server on the face of the Earth should indicate which time zone to use.

Ensure that the Hardware Clock Set To UTC checkbox is selected.

Click Next.

17) The next dialog will ask for an unprivileged user account to be created.

Type guru in the Username field.

Type work in the Password and Confirm Password fields.

It must be entered twice and both copies must match.

Click Next.

Click Yes on the The password is too simple dialog.

18) Set the root password:

Type **makeitso**  in the top two boxes.

Note that the root password must be entered twice and that both copies must match exactly.

Click Next.

Click Yes on the The password is too simple dialog.

19) It can take a moment for the system to populate the screen with the contents of the first installation hub Installation Settings. It is very common to tweak a few things on this hub. One of the most common items to customize is the list of packages to install. Click on Software to alter the list of packages that will be installed.

20) Select a package group, and then find and select an individual package to add to the package set, as follows:

Click the checkbox for C/C++ Compiler and Tools.
It's at the bottom of the list.

Click the Details... button.
It's located beneath the list of package groups.

Select the Search tab.
It's in the upper left of the screen.

Type **bind** in the Search: box and press .
It will take just a moment for the results to fill into the window on the right.

Click the checkbox next to bind in the window.
This will check bind and any related package dependencies.

Click the Accept button.

Click the Continue button (or press ) to accept the Automatic Changes YaST wants and return to the Installation Settings hub.
These changes are not only due to the selection of the bind package, but a normal part of the dependency selection of package groupings.

21) Unless the Instructor has indicated that any other options need changing, click Install to start the installation.

- 22) One more dialog box asking for confirmation of the decision to perform the installation steps that have been specified will pop up. **Click Install** to start the installation.
- 23) After YaST finishes formatting the partitions, the installation progress screen appears. The install will take a few minutes. This is a good time for a quick break.
- 24) After the files have been copied to the hard drive, the bootloader installed, and a couple of other tasks, the system will be automatically rebooted.
- 25) The system is now ready for users to log in and use. The accounts that have been created on the machine are:

Username	Password
guru	work
root	makeitso

Objectives

- ❖ Modify an AutoYaST2 file using a text editor.
- ❖ Create an AutoYaST2 file using YaST's autoyast module.
- ❖ Start an install using a pre made AutoYaST2 file.

Requirements

- (1 station)
- (classroom server)
- X (graphical environment)

Relevance

Installing systems manually can be time consuming and tedious. When deploying larger numbers of new systems, the ability to automate the installation process saves time and decreases the chance of mistakes being made.

Notices

- ❖ For this task, an AutoYaST file will be created similar to the one that will be used to set up the system.
- ❖ The instructor will provide boot media, along with the correct boot instructions from the options shown below.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Make a copy of the /root/autoinst.xml AutoYaST file:

```
# cp /root/autoinst.xml /root/default.ay
```

- 3) Ensure that AutoFS is configured for network filesystem mounting in the /net/ directory, and that automount is running:

```
# sed -i 's:^#/net:/net:' /etc/auto.master  
# systemctl restart autofs
```

Lab 3

Task 2

Automating Installation with AutoYaST [S12]

Estimated Time: 45 minutes

- 4) Locate the partitioning stanzas and change the size of the swap partition:

File: ~/default.ay

```
<partition>
<create config:type="boolean">true</create>
<crypt_fs config:type="boolean">false</crypt_fs>
<filesystem config:type="symbol">swap</filesystem>
<filesystem_id config:type="integer">130</partition_id>
<format config:type="boolean">true</format>
<loop_fs config:type="boolean">false</loop_fs>
<mount>swap</mount>
. . . snip . . .
- <size>526417408</size>
+ <size>1G</size>
</partition>
```

- 5) Launch the YaST2 autoyast module to create a new AutoYaST2 file:

yast2 autoyast

The AutoYaST2 tool appears. Click on file -> open and open default.ay. Explore the menu system to see what can be configured in an autoyast file.

Select the System section and choose a module to Edit.

Consider configuring the system language or timezone.

Click Next after each change.

Make other changes as desired.

Click File -> Save As from the Menu.

From the Save File Dialog, choose root's home directory and enter default2.ay as the file.

Click Save A dialog will appear indicating the save was successful.

Exit AutoYaST

- 6) Consider the data inside the newly created default2.ay file. How does it compare with the autoinst.xml file from the previous install? Some tweaking may be required here. Verify the information changed matches what was desired.

- 7) When ready, copy the new AutoYaST file to server1, so it can be used for installation. This step assumes that AutoFS is configured for using the /net/ directory and is started.

```
# cp default2.ay /net/server1.example.com/export/tmp/stationX.ay
```

If you are unable to complete this step please ask the instructor for assistance.

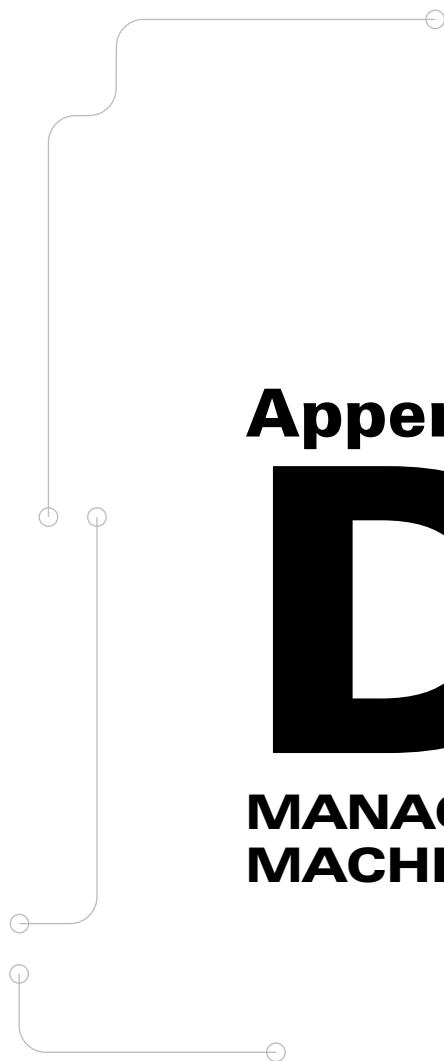
- 8) Install your system with the new AutoYaST file, entering the following at the Boot Options prompt from the Installation selection of the Installation DVD menu, or if using PXE, from the boot: prompt:

```
install=nfs://10.100.0.254/export/netinstall/SLES12→  
autoyast=nfs://server1/export/tmp/stationX.ay
```

If using PXE, prepend the Manual Install selection number 712 to the above:

```
boot: 712 install=nfs:...
```

- 9) Check with your instructor if the classroom system image needs to be used to restore your system for future chapters.



Appendix

D

MANAGE VIRTUAL MACHINES

Virtualization: What and Why?

What is virtualization?

- Virtualization makes it possible to run multiple operating systems on a single computer at the same time

Why virtualize?

- Hardware consolidation can reduce operating costs
- Virtualization can make it cheaper to build secure environments

Why not virtualize?

- Virtual system proliferation can increase operating costs
- Virtualization does not inherently improve security
- Centralized systems provide a single point of failure

What is virtualization?

Virtualization introduces an abstraction layer between operating system and hardware, making it possible to run multiple operating systems on a single computer at the same time.

Why virtualize?

Modern hardware is often underutilized. By running multiple systems on a single piece of hardware, utilization can be improved. Improved utilization reduces overhead, such as power, cooling, and rack space. Deployment of new and legacy systems on existing hardware is often faster and cheaper.

Virtualization makes it possible to move virtual systems across a network to new hardware. Modern solutions are able to migrate virtual systems live without shutting them down. This makes it easier to respond to changes in hardware. Virtual systems can be rebalanced by migrating from overloaded, to under-utilized, hardware. Migration also makes it possible to achieve near 100% uptime. When hardware maintenance is required, virtual systems can be migrated before powering down the hardware.

Production environments can be reproduced as virtual systems for testing and development purposes. While creating an exact copy of the production environment is desirable, budget constraints may not make it possible. Creating a virtual environment to validate changes, before modifying production systems, may reveal potential problems. Likewise, virtualization provides software developers with greater

flexibility to experiment without damaging critical development environments.

Virtualization does not inherently improve security, but virtualization can make it cheaper to build secure environments. When virtualization results in reduced complexity, and increased separation, it improves security. When it results in increased complexity, it is harmful, and can increase operating costs.

Since several virtual machines can be run on one physical system, if a critical piece of hardware fails, all of the virtual machines could become unavailable at once. However, if the storage for those virtual machines is available to other physical boxes (such as with a SAN solution), then the downed virtual machines could be brought up quickly on another server.

AMD-V (Pacifica)

During development of AMD's processor hardware virtualization technology, it was codenamed Pacifica. With the Q3 2006 release of the first Opteron, Athlon64, Turion64 and Sempron64 processors with the new hardware virtualization instructions, AMD renamed it AMD-V.

Visit http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796_14287,00.html for more information.

Intel VT (Vanderpool)

Intel's technology was codenamed Vanderpool during development and christened VT, or IVT for Intel Virtualization Technology upon release in Q2 of 2006, just a little ahead of AMD's AMD-V.

Visit <http://www.intel.com/technology/virtualization/> for more information.

Introducing libvirt

What does libvirt provide?

- Support for KVM/QEMU, Xen, VMware, LXC, and more
- Remote access; monitoring; domain, network, and storage management

What are advantages of using libvirt?

- Vendor independence
- Heterogeneous integration
- Rich ecosystem of technologies
sVirt, libguestfs, etc.

What are disadvantages of using libvirt?

- Xen-based terminology
- Delayed access to advanced features

What does libvirt provide?

libvirt is an abstraction layer designed to make it possible to manage many different virtualization technologies using the same tools. While achieving this goal is an ongoing process, libvirt continues to improve at an impressive rate.

Originally designed to create, access, and monitor Xen domains, today libvirt supports both "full" virtualization solutions, like KVM or VMware, and "container" solutions, like LXC. Support has been added for network and storage management. Fine grained security control is also an area of growing interest.

What are advantages of using libvirt?

Some virtualization technologies, like KVM, only provide low level native tools, relying on libvirt for higher level tools. Other technologies include their own management tools, for example VMware's excellent suite. Unfortunately, building infrastructure on a single technology leads to lock-in. In part, libvirt was created to make it easier to transition from one technology to another; for example moving from Xen to KVM.

Heterogeneous infrastructure is an unfortunate reality in many organizations. Whether because of legacy decisions, or technology-specific features, it is not uncommon to find multiple virtualization solutions coexisting. Using libvirt as a common interface can simplify integration.

As libvirt has gained acceptance, a rich ecosystem of technologies

has grown to take advantage of it. Multiple graphical and Web-based interfaces have been created and unique features have been added to libvirt itself, for example sVirt.

What is sVirt?

In a virtualized environment, vulnerabilities in a guest could enable attacks on the host or other guests. sVirt provides libvirt with more robust security constraints by using either SELinux or AppArmor to confine guests. Using SELinux for confinement, both the host and guests are protected. Using AppArmor, only the host is protected.

What are disadvantages of using libvirt?

Because libvirt was first created to manage Xen, much of its core terminology can feel unfamiliar when working with other technologies. For example, instead of using the terms "VM", "guest", or "container", libvirt commands and documentation use the term "domain". This makes it harder to translate technology specific documentation for use in a libvirt context.

While the libvirt authors have stated a desire to support all features for every virtualization technology, there is always an inherent delay between a feature introduction in a native tool and its availability in libvirt.

libvirt: Basic Concepts

Terminology

- Driver
- Hypervisor
- Node
- Domain
- URI

Architecture

- `libvirt.so`
- `libvirtd`
- `virsh`, `virt-manager`, etc.

libvirtd should not be thought of as a parent but instead a peer. Restarting it should not affect any active domains or volumes.

Terminology

Understanding several basic terms makes it much easier to read libvirt documentation and discover tools or commands.

Driver ⇒ a backend implementation used by libvirt to manage a specific virtualization technology

Hypervisor ⇒ a specific virtualization technology; for example KVM, Xen, or LXC

Node ⇒ a single physical machine

Domain ⇒ a collection of virtualized resources combined to provide an execution environment; also known as a "VM", "guest", "instance", or "container" in other settings

URI ⇒ used to identify a specific hypervisor, node, and (optionally) network protocol

Architecture

Technically, the name "libvirt" refers to a software library, but it is often used to refer to a helper daemon named **libvirtd** and a standard collection of tools built on the library (such as **virsh**, or **virt-manager**). The library contains drivers that translate generic API calls to backend specific actions. Drivers requiring only local, unprivileged access can be fully realized without depending on **libvirtd**. For privileged or remote access **libvirtd** is often required. However, this is not always true. For example, when managing a VMware ESX server the libvirt library uses VMware's SOAP API and when managing a Microsoft Hyper-V instance it uses WS-Management over HTTPS.

libvirt: Storage Architecture

libvirt Manages Storage Devices for use by VM guests

Local and Networked Storage Options

- Networked Storage required for live migrations

libvirt Supports different backends (pools)

- A directory of files, a whole disk, an NFS share, an iSCSI target, SAN, LVM, etc.

libvirt pools contain volumes

- A volume is assigned to a guest to use as a virtual storage device

libvirt Storage Architecture

Virtual machines can use storage resources provided by libvirt. libvirt has storage pools which contain volumes. A volume is assigned to a guest to use as a block device. libvirt pools can use different backends. For example, by default a pool called default is created using the directory backend. The default directory backend is configured so that each volume corresponds to a separate file in the /var/lib/libvirt/images directory.

Pool Backend Types

There are two main categories of pool backends supported: local storage pools, and networked (shared) storage pools. Local storage pools are normally used for small deployments, development, and when there is a single host. Networked storage is required for easy migration of guest virtual machines between hosts.

dir (local storage) ⇒ An existing directory with files in one of the supported volume types. The SELinux context type of `virt_image_t`

fs (local storage) ⇒ Variant of dir but requires a source block device and a destination mount point to be defined. libvirt will mount the filesystem itself, and it should not be auto mounted by the /etc/fstab file.

logical (local storage) ⇒ Use an LVM Volume Group as a pool, libvirt volumes will map to an LVM logical volume. Special volume formats such as qcows2 are not supported; effectively all volumes are raw.

disk (local storage) ⇒ Use a physical disk as a pool, libvirt volumes will map to partitions on the disk with the typical constraints on size and placement that partitions have. Using the GUID Partition Table (GPT) type as opposed to the DOS partition table is recommended as GPT allows up to 128 partitions on each disk.

netfs (networked storage) ⇒ Variant of dir but requires source NFS share and destination mount point to be defined. libvirt will mount the filesystem itself, and it should not be auto mounted by the /etc/fstab file.

iscsi (networked storage) ⇒ Use an iSCSI target as a pool. Volumes must be pre-defined on the iSCSI target as LUNs.

scsi (networked storage) ⇒ Use SCSI/SAN HBA as a pool. Volumes map to LUNs which must pre-exist.

Volume Types

The pools which use individual files for volumes allow the file type to be specified. Each volume file can either be a plain raw file, or a special format as supported by the `qemu-img` program. Supported special formats are as follows:

raw ⇒ A plain file

cow ⇒ A User Mode Linux disk image format

qcow ⇒ A QEMU v1 disk image format

qcow2 ⇒ A QEMU v2 disk image format

vmdk ⇒ A VMWare disk image format

vpc ⇒ A VirtualPC disk image format

When creating new virtual machines, typically either raw is used or

qcow2 if features such as AES encryption, compression, snapshots, and the ability to use a single read-only base image for multiple virtual machines, each with their own qcow2 image for writes, are desired.

Volume Creation

For most of the pool backend types, libvirt is able to create the volume itself. However, for iscsi and scsi, the LUNs which will be used for volumes must be manually created in advance.

SELinux considerations

The SELinux context type of `virt_image_t` must be set on the parent directory of the image files. The individual image files will automatically get assigned a unique SELinux context type each time they are booted if sVirt is enabled. When using NFS, either the `context` mount option or the SELinux boolean `virt_use_nfs` must be used.

libvirt: Network Architecture

**libvirt manages virtual networks used by VM guests
Implemented via Linux Ethernet bridges and Netfilter (iptables)
Virtual Networks**

- Private network either isolated, routed, or NATed to the outside
NAT network `virbr0` created by default
- Bridged network
Requires pre-configuration
The Linux bridge and `macvtap` bridges supported

Supports Single-Root I/O Virtualization (SR-IOV) as bridge alternative

- Single PCIe device appears as multiple devices
- Provides near native I/O performance

libvirt manages Netfilter rules automatically

server to service the virtual machines. libvirt accomplishes this by launching a per-network instance of `dnsmasq`.

Bridged Interfaces

libvirt supports two different type of bridges. Bridges created with the classical Linux bridge, and `macvtap` bridges. The Linux bridge switches Ethernet frames internally to the host so that VM to VM communication, when connected to the same bridge or virtual network, doesn't leave the host. To connect virtual interfaces to a Linux bridge, a TAP device is used.

Some data centers want to leverage the performance and management infrastructure of their existing switches. There are two standards, Virtual Ethernet Port Aggregator (VEPA aka 802.1Qb), and VNLink (aka Bridge Port Extension 802.1Qbh). In both cases switching of Ethernet frames takes place outside of the host in the physical switch, even for VM to VM or VM to host communication. This way the physical switch can enforce policy such as traffic filtering, bandwidth shaping, and monitoring of VMs instead of having the host do it. This can reduce processor utilization on the host, and allow network administrators to manage the layer 2 network all the way to the VMs. libvirt supports both standards via modes of `macvtap`.

Additionally, `macvtap` can implement a standard bridge with internal switching. In this later case it is slightly simpler and lower overhead since TAP virtual network kernel devices are not used. To see `macvtap` bridges, the `ip link` command can be used.

libvirt Network Architecture

There are two general types of networks that virtual machine guests can use: internal and bridged networks. Internal virtual networks can be either isolated or connected to the outside world via NAT or routing and are built on top of Linux's in-kernel layer 2 switch, the Linux bridge. Bridged networks place the virtual machine directly on an external network and use either the Linux bridge or a `macvtap` bridge.

Virtual machines can have multiple virtual interfaces connected to arbitrary networks. KVM based virtual machines have an Ethernet OUI of `52:54:00` while Xen based virtual machines use an Ethernet OUI of `00:16:3e`.

Optionally, libvirt network filters can be used to filter traffic to and from virtual interfaces. The network filters are implemented using `ebtables` for layer 2 filtering and `iptables` for the upper layers.

Virtual Networks

All virtual networks have at their core, a Linux bridge. These bridges are created automatically by libvirt, and are visible on the host using the `brctl` command from the `bridge-utils` package. Connectivity to the bridge determines if the bridge is isolated, NATed, or routed. NAT and routing are accomplished using the standard Linux methods which are automatically configured by libvirt. Virtual machines connect to the bridge via TAP devices.

Each virtual network can optionally have a DHCP server and DNS

To use Linux bridges to connect to an external interface, the external interface connection must be defined manually first, then libvirt takes care of the rest automatically. With macvtap bridges, libvirt is able to configure everything itself.

[R7] *The following applies to RHEL7 only:*

On the host, to create a bridge ext01 and connect to the eth0 interface, the following configuration can be used:

File: /etc/sysconfig/network-scripts/ifcfg-ext01

```
DEVICE=ext01
TYPE=Bridge
BOOTPROTO=none
ONBOOT=yes
DELAY=0
NM_CONTROLLED=no
IPADDR="10.70.166.67"
NETMASK="255.255.255.240"
GATEWAY="10.70.166.65"
```

File: /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
NM_CONTROLLED=no
HWADDR="00:1B:21:1D:93:F9"
BRIDGE="ext01"
```

Single-Root I/O Virtualization (SR-IOV)

As virtualization has greatly increased in popularity, hardware vendors have been looking for ways to optimize virtualized environments. The PCI standards body created Single-Root I/O Virtualization (SR-IOV) as a method to allow multiple VMs to natively share a PCI Express device. The goal being to provide the highest performance method of sharing a physical device with the lowest CPU utilization on the host. While this can be done with any class of PCI device, network interface cards were the first class of devices to receive broad SR-IOV support and enables the highest performing bridging method.

In SR-IOV terminology a physical device is known as a physical function (PF). A PF is able to be seen by multiple virtual functions (VFs). libvirt is able to assign a VF to a VM using the nodedev feature.

The VM loads a native driver for the VF and uses the VF directly, bypassing the hypervisor.

With network cards, each physical port becomes a PF. Although the hardware vendors have the flexibility to do what they like, typically all the VFs for a PF are attached to a layer 2 switch inside the NIC which is bridged to the physical port. Alternatively, some cards support VEPA and VNLink.

Previous to the SR-IOV technique, the PCI pass-through technique could be used to get the same performance benefit, but PCI pass-through requires a 1:1 assignment of physical device to VM. This cannot scale due to the limited number of expansion slots. With SR-IOV each PF can provide up to 256 VFs, although 64 VFs is more typical since each VF requires actual hardware resources.

Network Filters

With libvirt, network filters are defined as objects which are applied to guest NICs. The filters are implemented using Netfilter (**iptables**, **ip6tables**) and **ebtables**. The filters are defined with libvirt's XML syntax which are then translated to the underlying technology's syntax and created dynamically by **libvирtd**.

If for some reason the filters created by **libvирtd** are manually deleted or inadvertently broken, send a SIGHUP to **libvирtd** to have it recreate its rules.

Out of the box, a set of default rules are shipped that are typically used as building blocks to create other rules. An example of this is the clean-traffic rule which uses the following building block rules: no-mac-spoofing, no-ip-spoofing, allow-incoming-ipv4, no-arp-spoofing, no-other-l2-traffic, and qemu-announce-self.

libvirt: Graphical Tools

GUI Installations and Management

- **virt-manager**

Full Install Wizard

Allows monitoring, modification, and console access to existing VMs

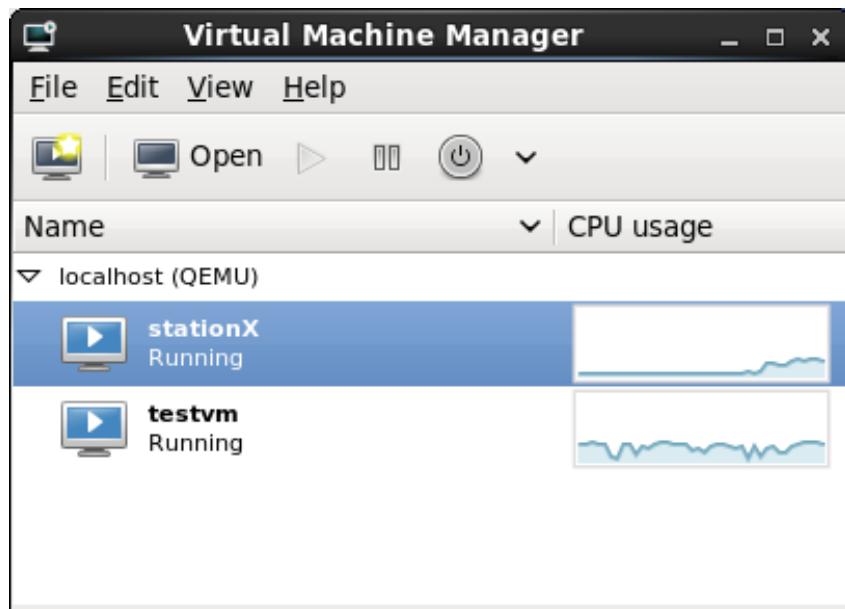
virt-viewer

- Lightweight graphical console access to VMs

The Virtual Machine Manager

The **virt-manager** command is a full fledged GUI VM management tool. It is able to launch installations, monitor VMs, and modify the virtual hardware. It is normally run directly on the host machine running **libvирtd**, but can also connect over the network using a secure encrypted connection.

When first launched, it connects to **libvирtd** at the specified URI, with a default of localhost, and displays a list of VMs:

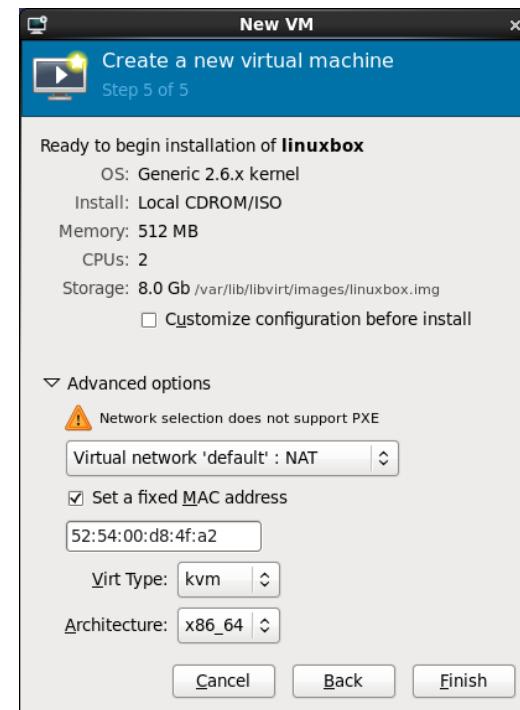
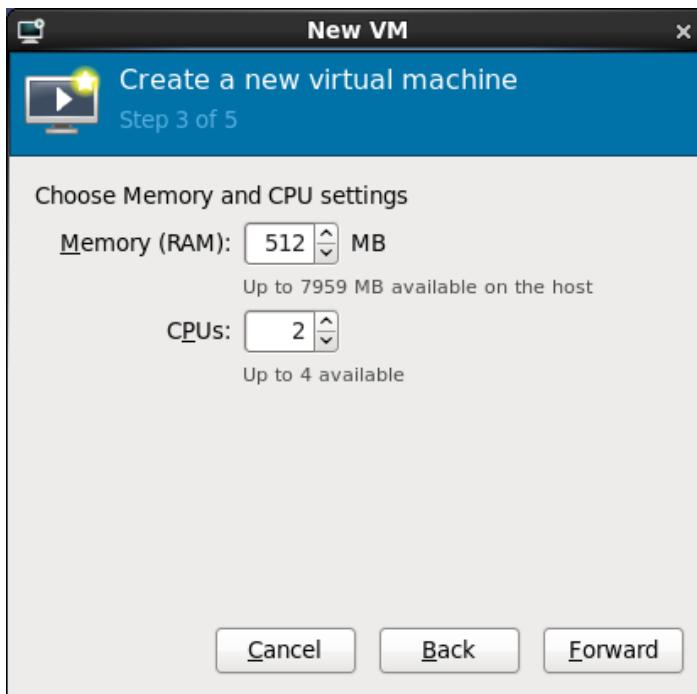
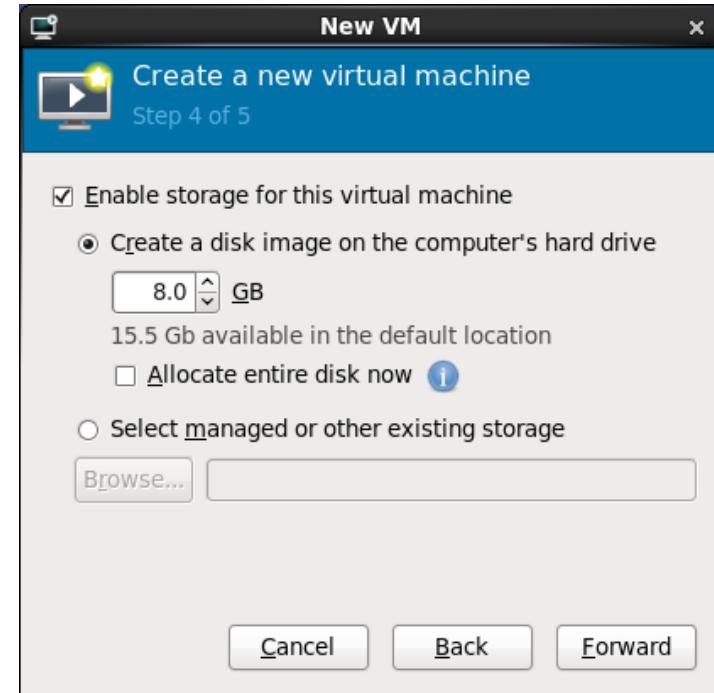
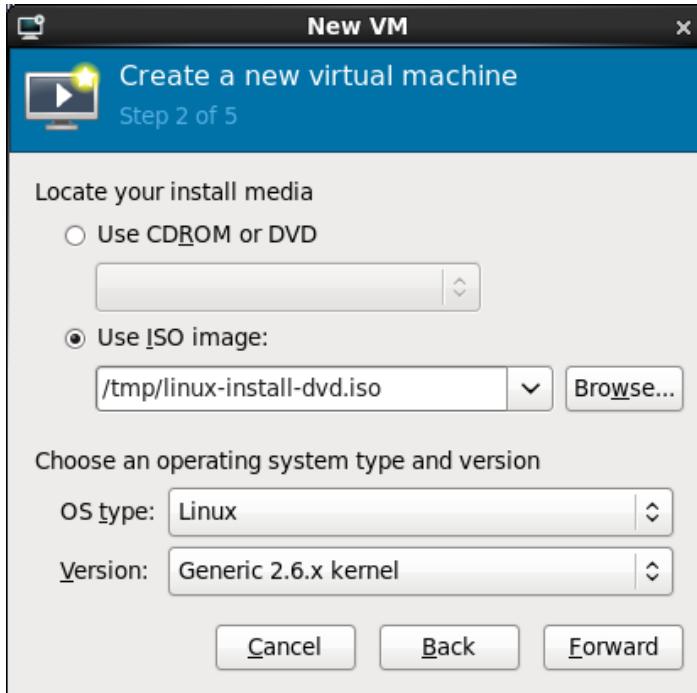


Installing a Virtual Machine

Within **virt-manager** the installation wizard provides a step-by-step process to install a virtual machine:

[R7] *The following applies to RHEL7 only:*





libvirt: Command Line Tools

General Purpose

- `virsh`
- RHEL7: `virt-top`

Special Purpose

- `virt-install`
- `virt-image`
- `virt-clone`
- `virt-convert`
- `virt-xml-validate`

General Purpose Commands

The following commands can be used to manage domains:

virsh ⇒ Primary command line libvirt interface. Can be run interactively (`virsh shell`) or non-interactively.

[R7] *The following applies to RHEL7 only:*

virt-top ⇒ Show domain statistics like CPU and memory usage.

Special Purpose Commands

The following are more focused commands used for virtual machine administration:

virt-install ⇒ Create new domains. Supports both text and graphical installation. Can run interactive or unattended.

virt-image ⇒ Define a new domain using an existing disk image.

virt-clone ⇒ Clone an existing domain. Creates a new disk image and generates new unique data.

virt-convert ⇒ Convert a domain from one type to another. For example, convert from VMware to KVM.

virt-xml-validate ⇒ Check a libvirt XML definition file for errors.

virsh: Basics

Advantages of virsh

Basic concepts

- `create` vs `define`
- `edit` vs `define`
- `destroy` vs `undefine`

Working with XML definitions

Finding documentation

Advantages of virsh

virsh shares the same advantages as other command line applications. It is often the fastest tool for simple manual control. It is easy to use in shell scripts. Remote administration is very light weight.

Another advantage is that new features are generally added to **virsh** before they are added to GUI or Web-based tools. Also, because there is a close relationship between the libvirt API and **virsh** commands, it can be a useful tool for exploring libvirt while creating custom tools in a language like C or Python.

Basic Concepts

There are repeating patterns that make **virsh** easier to understand. For example, "create" vs. "define". A creating command is a defining command immediately followed by a starting command. For example:

```
# virsh net-create dmz.xml  
Network dmz created from dmz.xml
```

Is the same as:

```
# virsh net-define dmz.xml  
Network dmz defined from dmz.xml  
# virsh net-start dmz  
Network dmz started
```

Defining commands are not limited to adding new definitions but can

also be used to replace existing definitions. For example:

```
# virsh pool-edit default  
Pool default XML configuration edited.
```

Can be replaced with:

```
# virsh pool-dumpxml default > default.xml  
# vim default.xml  
# virsh pool-define default.xml  
Pool default defined from default.xml
```

Destroying commands disable but do not delete. Undefining commands delete permanently, but only if the target is disabled. For example:

```
# virsh destroy linuxbox  
Domain linuxbox destroyed  
# virsh start linuxbox  
Domain linuxbox started  
# virsh undefine linuxbox  
error: Failed to undefine domain linuxbox  
# virsh destroy linuxbox  
Domain linuxbox destroyed  
# virsh undefine linuxbox  
Domain linuxbox has been undefined  
# virsh start linuxbox  
error: failed to get domain 'linuxbox'
```

Working with XML Definitions

Like the libvirt API, many **virsh** commands require reading or editing small XML documents. For example:

```
# virsh dumpxml linuxbox
<domain type='kvm' id='2'>
  <name>linuxbox</name>
  <uuid>7d6b9849-b1f9-7453-1b59-3bede448e8b1</uuid>
  . . . snip . . .
</domain>
```

Unfortunately, the syntax and semantics of these documents can be confusing. Many elements and attributes must be defined manually. Some are generated automatically but can be edited. Others are purely informational and can not be edited.

In the previous example, the `name` element and the `type` attribute must be specified when a new domain is defined. The `uuid` element may be specified, but when it is not a `uuid` is added automatically. The `id` attribute provides information about the currently running domain and attempts to edit it are ignored.

These details are generally included in the libvirt documentation, but some experimentation may be required to fully understand specific elements or attributes.

Finding Documentation

The **virsh** man page is succinct but comprehensive. In addition, the **virsh** command has integrated help like **ip** or **git**. This can be used to get a list of all commands, or details for a specific command. For example:

```
# virsh help
. . . output omitted (list of commands) . . .
# virsh help create
. . . output omitted (description of create command) . . .
```

Unfortunately, the documentation included with libvirt can be limited and cryptic. The libvirt Web site is an important additional resource.
<http://libvirt.org/>

The libvirt-client package installs Relax-NG schema files in
`/usr/share/libvirt/schemas/`.

virsh: Common Tasks

Starting And Stopping Domains

- `start` and `autostart`
- `shutdown` and `destroy`
- `suspend` and `resume`
- `save` and `restore`

Getting Serial Console Access

Gotchas

- incomplete lists
- editing definitions without restarting
- editing readonly attributes
- fighting for the serial console

Starting And Stopping Domains

The command `virsh start domain` will activate a domain that has not already been started. This is a temporary change. To control whether a domain is activated every time the host is rebooted, use the `autostart` command. For example:

```
# virsh autostart linuxbox
Domain linuxbox marked as autostarted
# virsh autostart --disable linuxbox
Domain linuxbox unmarked as autostarted
```

There are two ways to stop a running domain: shutdown and destroy. The command `virsh shutdown domain` will attempt to stop the domain cleanly. For example by triggering an ACPI power button event. The domain may choose to ignore shutdown requests. When shutdown doesn't work, the command `virsh destroy domain` will always kill the running domain. It is analogous to pulling the power cord on a physical machine; as such it should be avoided when possible.

If the driver supports it, libvirt can pause and resume domains. Both KVM and Xen support this feature. Instead of using the term "pause", libvirt uses the term "suspend". This does not free any memory, it merely prevents the domain from using CPU time. For example:

```
# virsh suspend linuxbox
Domain linuxbox suspended
# virsh list | grep linuxbox
 42 linuxbox           paused
```

```
# virsh resume linuxbox
```

Domain linuxbox resumed

If the driver supports it, libvirt can save and restore running domains. Both KVM and Xen support this feature. When a domain is saved to a file, it is paused, its memory is dumped to the file, then the memory it was using is freed. Saving and restoring a domain is much faster than shutting it down and booting it again later. This is useful when the host system has to be rebooted. For example:

```
# virsh save linuxbox ~/linuxbox.save
Domain linuxbox saved to ~/linuxbox.save
# virsh list | grep linuxbox
# virsh restore ~/linuxbox.save
Domain restored from ~/linuxbox.save
```

Getting Serial Console Access

Although **virsh** does not include support for accessing graphical consoles, it does provide support for accessing serial consoles. Depending on how the system is installed and configured, accessing the serial console may be as easy as running **virsh console domain**

For RHEL 7, SLES12, and Ubuntu guests, enable the serial console for both GRUB 2 and Linux by making the following edit:

File: /etc/default/grub

```
+ GRUB_CMDLINE_LINUX_DEFAULT="console=tty0,console=ttyS0,115200n8"
+ GRUB_TERMINAL="console serial"
+ GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1"
```

For RHEL 6 guests, a serial console is automatically started by the /etc/init/serial.conf if a serial console is the primary console (last console defined) on the kernel command line.

File: /boot/grub/grub.conf

```
→ ...console=tty0 console=ttyS0,115200n8
```

For SLES11 guests, to enable the serial console, edit /etc/inittab:

File: /etc/inittab

```
- #S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102
+ S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102
```

After editing /etc/inittab, either reboot or use **telinit** to reload the file:

```
# telinit q
```

If root should be allowed to login on the serial console, add **ttyS0** to /etc/securetty:

```
# echo ttys0 >> /etc/securetty
```

Gotchas

By default, **virsh** only lists active items. To list both active and inactive items, include the **--all** option. For example:

```
# virsh list
```

```
42 linuxbox           running
# virsh list --all
42 linuxbox           running
- windowsbox          shut off
```

Editing commands like **virsh edit domain** or **virsh net-edit network** use the currently active XML definition. Changes do not become permanent until after the target is restarted. When multiple edits are performed without restarting, only the most recent edit is saved. For this reason, stopping the target before editing is safest.

Some elements and attributes in a libvirt XML definition are readonly. Attempts to edit them will have no effect. No warning will be provided. Changes will merely be ignored.

If multiple processes access a domain's serial console at the same time, the serial console will behave non-deterministically. Each process will fight for data from the serial console, resulting in each process seeing only part of the domain's output. No warning will be provided. The domain will be fine, but appear to behave strangely.

virt-install

Command Line Installations

- **virt-install**

Supports text and GUI installs
VM hardware specs defined by options
Scriptable
Very useful to perform unattended installs

virt-install

The **virt-install** command is useful for initiating an installation of a guest VM under libvirt. The libvirt instance could be the local system, or on a remote system. All guest VM hardware configuration is specified by options to the command as well as paths to the installation files.

While it is possible to use **virt-install** to initiate a manual install, the real power of **virt-install** comes when combined with the operating system's unattended installation feature. This scenario can be further expanded by using **virt-install** as part of a script using variables. A single script could install one or more customized guest VMs based on the parameters passed to the script which modify the options used by **virt-install**.

When installing a para-virtualized Linux guest VM, the **--location** option is used to specify the path to the kernel+initrd pair. With full hard virtualized guests (Linux or otherwise) use either the same **--location** option, or use the **--cdrom** option to specify an ISO/CDROM image. Either option can reference an NFS, HTTP, or FTP path. Alternatively, the **--pxe** option can be used to initiate a network boot.

Specifying Hardware Specs

The following table lists commonly used hardware related options:

Option	Purpose
-r or --ram	Amount of memory in megabytes
--vcpus	Number of virtual CPUs for the guest
--disk <i>opt1=val1,...</i>	Storage configuration. Create or use libvirt volume in a pool. Can be used multiple times.
--network <i>NETWORK,opt1=val1,...</i>	Create a virtual NIC and connect it to the specified libvirt network or bridge. Can be used multiple times to create multiple virtual NICs.
--noautoconsole	Don't automatically connect to the console
-x or --extra-args	When using --location with a kernel+initrd pair, kernel command line arguments to use. Useful to start OS's unattended install.

Examples of virt-install

Installing Red Hat Enterprise Linux and SUSE Linux Enterprise Server using Kickstart and creating a new sparse 9GB virtual disk in libvirt's default storage pool:

```
# virt-install --name testvm --ram 2048 --vcpus 4 --noautoconsole --network network=default,  
--disk pool=default,size=9,sparse=true --location nfs:server1:/export/netinstall/RHEL7/SLES12,  
--extra-args "ks=nfs:server1:/export/ks/RHEL.ks noipv6"
```

Starting install...

Retrieving file .treeinfo...	2.6 kB	00:00	...
Retrieving file vmlinuz...	7.0 MB	00:00	...
Retrieving file initrd.img...	55 MB	00:00	...
Allocating 'testvm.img'	9.0 GB	00:00	
Creating domain...	0 B	00:00	

Domain installation still in progress. You can reconnect to
the console to complete the installation process.

To monitor the installation process, connect to the guest using **virt-manager** or **virt-viewer**:

```
# virt-viewer testvm
```

Launching a LiveCD distro:

```
# virt-install --name demo --ram 4096 --vcpus 4 --network network=default --nodisks --vnc --livecd --cdrom /tmp/linuxlive.iso
```

Virtual Machine Guest Tools & Drivers

Host can control/manage guests via QEMU Guest Agent

- Reboot, shutdown, suspend, information gathering
- Freezing/thawing guest filesystem before/after host initiated VM snapshots
- Install `qemu-guest-agent` package on Linux guests
- Install `qemu-ga-x64.msi` package on Windows guests
- Requires virtual serial port defined for communication

KVM Para-virtualized (Virtio) Drivers

- Optimized drivers for the guest
- Built into modern Linux kernels
- Use `virtio-win.iso` for Windows guests

VMware `open-vm-tools`

Using the QEMU Guest Agent

To interact with the guest agent from the host, use the `qemu-agent-command` command with `virsh`. Additionally, the `reboot` and `shutdown` `virsh` commands can then use the `--mode=agent` option to initiate a graceful reboot and shutdown via the agent.

When issuing a `reboot` or `shutdown` command, by default the `acpi` mode is used which simulates a physical power button press and is handled by the `acpid` daemon in the guest.

```
# virsh reboot --mode=agent linuxbox
Domain linuxbox is being rebooted
```

Blacklisting Commands the Host is Allowed to Run

[R7] *The following applies to RHEL7 only:*

On RHEL7, by default, the `BLACKLIST_RPC` variable in the `/etc/sysconfig/qemu-ga` disables all the `guest-file-*` commands. Which prevents the host from reading, creating or modifying files within the guest via the agent.

[S12] *The following applies to SLES12 only:*

On SLES12, the `systemd` unit file, the `qemu-ga.service` must be edited and the `-b` option added with a comma separated list of commands to black list.

VMware Guest Agent

When running a Linux VM under VMware, the `open-vm-tools` package can be installed, which is an open source VMware guest agent that is authored by VMware and included with all current major Linux distributions and fully supported by VMware. See <http://kb.vmware.com/kb/2073803>

QEMU Guest Agent Installation and Configuration

A sysadmin on the host system can issue commands to the virtual machine's guest agent via a virtio serial port. This allows the host to initiate a graceful reboot, shutdown, suspend within the guest, flush the disk cache within the guest, hot plug/remove vCPUs, obtain the guest's IP address, and even create, delete, and modify files. The guest agent is installed on Linux guests with the `qemu-guest-agent` package, and `qemu-ga-x64.msi` on Windows guests. The guest agent supports a blacklist of commands that the host is not allowed to run.

To enable the Guest Agent, a virtio serial port with a specific name must be added to the guest's hardware definition. This can be done with the following XML element which should be added inside the `<devices>` element:

```
<channel type='unix'>
  <source mode='bind' />
  <target type='virtio' name='org.qemu.guest_agent.0' />
</channel>
```

Using virsh qemu-agent-command

```
# virsh qemu-agent-command linuxbox '{"execute":"guest-info"}' | sed 's/},/,\\n/g'  
{"return": {"version": "2.0.0", "supported_commands": [  
    {"enabled": true, "name": "guest-set-vcpus", "success-response": true},  
    {"enabled": true, "name": "guest-get-vcpus", "success-response": true},  
    {"enabled": true, "name": "guest-network-get-interfaces", "success-response": true},  
    {"enabled": true, "name": "guest-file-flush", "success-response": true},  
    {"enabled": true, "name": "guest-file-seek", "success-response": true},  
    {"enabled": true, "name": "guest-file-write", "success-response": true},  
    {"enabled": true, "name": "guest-file-read", "success-response": true},  
    {"enabled": true, "name": "guest-file-close", "success-response": true},  
    {"enabled": true, "name": "guest-file-open", "success-response": true},  
    . . . snip . . .  
  
## Get a read-only file handle for /etc/shadow  
# virsh qemu-agent-command linuxbox '{"execute":"guest-file-open", "arguments": {"path": "/etc/shadow", "mode": "r"}}'  
{"return": 1001}  
  
## Read from the file handle, a JSON document is returned, file contents encoded in BASE64  
# virsh qemu-agent-command linuxbox '{"execute": "guest-file-read", "arguments": {"handle": 1001}}'  
{"return": {"count": 802, "buf-b64": "cm9vdDoh0jE2MjY50jA60Tk50Tk6Nzo60gpkYWVtb246KjoxNjE3Njow, eW5j0io6MTYxNz. . . snip . . . Y6MDo50Tk50To30jo6Cg==", "eof": true}}  
  
## Rewind file pointer back to top of file  
# virsh qemu-agent-command linuxbox '{"execute": "guest-file-seek", "arguments": {"handle": 1001, "offset": 0, "whence": 0}}'  
  
## Pipe output to jq to extract the specific field and decode the BASE64  
# virsh qemu-agent-command linuxbox '{"execute": "guest-file-read", "arguments": {"handle": 1001}}' | jq -r '.return["buf-b64"]' | base64 -d  
root:$6$lnRH39$1R04x05dXxl3J9mjLVpofwOugnoJwD1FER:16269:0:99999:7:::  
daemon:*:16176:0:99999:7:::  
bin:*:16176:0:99999:7:::  
sync:*:16176:0:99999:7:::  
man:*:16176:0:99999:7:::  
lp:*:16176:0:99999:7:::  
. . . snip . . .  
  
## Close the file handle  
# virsh qemu-agent-command linuxbox '{"execute": "guest-file-close", "arguments": {"handle": 1001}}'  
{"return": {}}
```

libguestfs and guestfish

What is libguestfs?

- A library and collection of tools to read and modify disk images.

How does libguestfs work?

- Runs a lightweight Linux appliance built on libvirt.

What tools does libguestfs provide?

- `guestfish` and `virt-rescue`
- `virt-inspector` and `virt-df`
- `virt-edit`, `virt-resize`, and `virt-win-reg`
- more added regularly

Using guestfish

`virt-cat` ⇒ display files in a disk image or domain

`virt-df` ⇒ check free space in a disk image or domain

`virt-edit` ⇒ edit files in a disk image or domain

`virt-filesystems` ⇒ list partitions, logical volumes, and filesystems

`virt-inspector` ⇒ detect OS version and drivers

`virt-inspector2` ⇒ updated version of `virt-inspector`

`virt-ls` ⇒ list files in a disk image or domain

`virt-make-fs` ⇒ create a filesystem from a directory or tar file

`virt-rescue` ⇒ an interactive repair shell

`virt-resize` ⇒ create a new disk image with a different size and partitions than the original

`virt-tar` ⇒ extract or add files to a disk image or domain

`virt-win-reg` ⇒ view and modify Windows Registry entries

Using guestfish

The **-i** option to **guestfish** can be used to automatically detect and mount filesystems. For example:

```
# guestfish --ro -d linuxbox -i  
Welcome to guestfish, the libguestfs filesystem interactive  
shell for editing virtual machine filesystems.  
... snip ...  
><fs> cat /etc/passwd  
... output omitted ...  
><fs> cat /etc/shadow  
... output omitted ...
```

One or more **guestfish** commands can be included on the command line. To run multiple commands, separate each with a colon (""). For example:

```
# guestfish --ro -d linuxbox -i cat /etc/passwd : cat /etc/shadow  
... output omitted ...
```

The same can also be done manually:

```
# guestfish  
Welcome to guestfish, the libguestfs filesystem interactive  
shell for editing virtual machine filesystems.  
... snip ...  
><fs> add-domain linuxbox readonly:true  
2  
><fs> run  
><fs> inspect-os  
/dev/vg_linuxbox/lv_root  
><fs> inspect-get-filesystems /dev/vg_linuxbox/lv_root  
/dev/vg_linuxbox/lv_root  
/dev/vda1  
/dev/vg_linuxbox/lv_swap  
><fs> inspect-get-mountpoints /dev/vg_linuxbox/lv_root  
/: /dev/vg_linuxbox/lv_root  
/boot: /dev/vda1  
><fs> mount-ro /dev/vg_linuxbox/lv_root /  
><fs> cat /etc/passwd  
... output omitted ...  
><fs> cat /etc/shadow  
... output omitted ...  
><fs> quit
```

Resizing Virtual Machine Disk Images

To resize a virtual machine file image, use the **virt-resize** command. The following example is from resizing a Windows XP disk image that uses the NTFS filesystem internally:

```
# qemu-img create -f raw wXP.img.final 7G  
Formatting 'wXP.img.final', fmt=raw size=7516192768  
# virt-resize --expand /dev/sda1 wXP.img wXP.img.final  
... snip ...  
Summary of changes:  
  
/dev/sda1: This partition will be resized from 3.9G to 7.0G. The  
filesystem ntfs on /dev/sda1 will be expanded using the  
'ntfsresize' method.  
... snip ...  
Resize operation completed with no errors. Before deleting the old  
disk, carefully check that the resized disk boots and works correctly
```

Lab 4

Estimated Time:
S12: 45 minutes
R7: 45 minutes

Task 1: Installing a Virtual Machine

Page: 4-20 Time: 45 minutes

Requirements:  (1 station)  (classroom server)  (graphical environment)

Objectives

❖ Install a usable domain.

Requirements

█ (1 station) █ (classroom server) X (graphical environment)

Relevance

Notices

❖ The lab assumes a bare metal installation. A virtual environment may not be able to host another virtual environment.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Install the virtualization packages needed to install a guest domain:

```
[R7] # yum install -y virt-{manag,view}er libvirt qemu-kvm  
[S12] # zypper install -t pattern kvm_server  
[S12] # zypper install -y virt-manager virt-viewer yast2-vm libvirt-daemon-qemu  
. . . output omitted . . .
```

- 3) Start the libvirt virtualization daemon:

```
# systemctl start libvirtd
```

- 4) [S12] This step should only be performed on SLES12.

Run the YaST virtualization module to configure KVM:

```
# yast2 virtualization
```

Select KVM server and KVM tools when asked about which Hypervisor(s) to install. Click Accept when complete. Click Yes when asked to Configure a default network bridge.

Lab 4

Task 1

Installing a Virtual Machine

Estimated Time: 45 minutes

- 5) Create a logical volume sufficient to fit the VM's disk image:

```
# lvcreate -L 9G -n libvirt-images vg0
Logical volume "libvirt-images" created
# mkfs -t xfs /dev/vg0/libvirt-images
. . . output omitted . . .
```

- 6) Mount and set persistent the new file system:

```
# cat >> /etc/fstab <<EOF
> /dev/vg0/libvirt-images /var/lib/libvirt/images xfs defaults 1 2
> EOF
# mount -a
```

- 7) To create a new virtual machine, launch **virt-manager**:

```
# virt-manager
```

Note that SLES also has a SUSE specific tool, **vm-install**.

- 8)

Click the Create a new virtual machine button. This will launch the Create a new virtual machine wizard.

Select Network Install (HTTP, FTP, or NFS).

Click Forward. This will advance to Step 2 of 5.

- 9) [R7] This step should only be performed on RHEL7.

Enter **nfs://10.100.0.254/export/netinstall/RHEL7** in the URL field.

Click Forward. This will advance to Step 3 of 5.

- 10) [S12] This step should only be performed on SLES12.

Enter **nfs://10.100.0.254/export/netinstall/SLES12** in the URL field.

Click Forward. This will advance to Step 3 of 5.

11)

Set the Memory to 768MB and use the default single CPU. **clicking** Forward.

This will advance to Step 4 of 5.

Enter 8 GiB for the disk image and then **click** Forward.

This will advance to Step 5 of 5.

Enter **guest01** for the Name.

Ensure networking is enabled with the default NAT option. As an alternate option, from the Source device toggle menu, **select** the Network

Valid ethernet devices include eth0, p3p1, em1, etc.

Make sure the Source mode is set to **Bridge**.

Check Customize configuration before installation. **Click** Finish.

This will open a customization window.

Click Begin Installation.

This will boot the system, and begin the Red Hat Enterprise Linux and SUSE Linux Enterprise Server installer.

12) [R7] *This step should only be performed on RHEL7.*

In general, install using defaults. Use a root password of makeitso. When asked about partitioning, **select** Automatic Partitioning. **Select** the Server with GUI software selection. **Reboot** the machine after the installation is complete. Skip the registration, and create the user guru with password work; **select** Make this user administrator. The log in screen will then be presented.

13) [S12] *This step should only be performed on SLES12.*

Click the box for I Agree to the License Terms and click **Next**. In general, install using defaults. Create the user guru with password work, use a root password of makeitso, and skip the Customer Center registration. After the installation is complete, the system will reboot.

14) [R7] *This step should only be performed on RHEL7.*

Log in to the guest, switch to root, and **configure** the guest console to be accessible to **virsh**. The following edit is *not* done on the hypervisor:

File: /etc/sysconfig/grub
→ GRUB_CMDLINE_LINUX="...snip...rhgb quiet console=tty0 console=ttyS0,115200

- 15) [R7] This step should only be performed on RHEL7.

Once configured, generate a new grub.cfg and power off:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
. . . output omitted . .
# systemctl poweroff
```

- 16) [R7] This step should only be performed on RHEL7.

To launch your virtual machine use the virsh command:

```
# virsh start guest01
```

The **virt-manager** command can also be used to start the guest before accessing it.

- 17) If needed, reaccess your virtual machine console, using the virt-viewer or **virsh** commands:

```
# virt-viewer guest01
# virsh console guest01
```

Type **[ctrl]+[I]** to break from the text terminal login. If needed, use **xm**, **virsh**, or **virt-manager** to start the guest before accessing it.

Clean up

- 18) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -
Password: makeitso 
```

- 19) Remove the guest:

```
# virsh destroy guest01
```

```
Domain guest01 destroyed
# virsh undefine guest01
Domain guest01 has been undefined
```

- 20) Remove the libvirt-images logical volume:

```
# umount /dev/vg0/libvirt-images
# lvremove /dev/vg0/libvirt-images
# sed -i '/libvirt/d' /etc/fstab
```

Appendix

E

BACKUPS

Backup Software

Low Level Backup Software

- **dump/restore, tar, cpio, star, pax, rsync**

High Level Backup Software

- Amanda, Bacula, BackupPC, Mondo Rescue, rdiff-backup, rsnapshot
- Relax-and-Recover
<http://relax-and-recover.org/>
- Commercial Software

Backup Planning

- Interval
- Security
- Backup vs archiving

Low Level Backup Software

Linux provides many low level backup and archival tools. Popular commands include **dump/restore**, **tar**, **cpio**, **star**, **pax**, and **rsync**.

The **dump** command works at the filesystem level, making it easier to backup extended file metadata like ACLs or perform incremental backups. The **restore** command can be used to restore an entire filesystem or individual files. The **restore** command also includes an interactive method for selecting files to restore. Unfortunately, **dump** and **restore** can only backup ext2/3/4 filesystems. Some filesystems have equivalent commands. For example **xfsdump** and **xfsrestore** can backup and restore XFS filesystems.

The **tar** command is a popular backup solution. However, older versions of **tar** on Unix had portability and reliability issues. In response, a more powerful and robust file format was created for the **cpio** command. However, **cpio** can not backup extended file metadata like ACLs or SELinux labels, whereas the latest versions of **tar**, **star** and **rsync** can.

High Level Backup Software

Many full featured backup applications exist for Linux, both free and commercial. Popular free solutions include Amanda, Bacula, BackupPC, Mondo Rescue, rdiff-backup and rsnapshot. Some free backup solutions include commercial support options. For example, both backula and Amanda Enterprise have commercial options.

BareOS is a fork of backula with many additional features that has

gained popularity after backula directed more of their efforts to their own commercial enterprise solution.

Relax and Recover (ReaR) is a compliment to traditional backup software by creating a rescue system image that is capable of a baremetal recovery. Enhances a traditional backup solution into a Disaster Recovery solution.

- ◎ Amanda: <http://www.amanda.org/>
- ◎ Bacula: <http://www.bacula.org/>
- ◎ BareOS: <http://www.bareos.org>
- ◎ BackupPC: <http://backuppc.sourceforge.net/>
- ◎ Mondo Rescue: <http://www.mondorescue.org/>
- ◎ Rear: <http://relax-and-recover.org/>
- ◎ rdiff-backup: <http://www.nongnu.org/rdiff-backup/>
- ◎ rsnapshot: <http://www.rsnapshot.org/>

Backup Strategy

Although often overlooked, to ensure data integrity and recoverability, a sound backup strategy is at least as important as the hardware and software being used to back up data. A routine of continuous backups at intervals appropriate for the site's data levels and data modification patterns is essential for increasing the probability of being able to restore needed files. One popular approach is full backups Friday night and incrementals the rest of the week. Files that are not commonly accessed, or that are rarely changed, should be archived using cheaper hardware. This may mean that only a single backup is needed for an archive. Ensure regular validation of backups

and archives with periodic restores and testing.

Secure Backup Media

Archival media should be physically secured to prevent theft, modification or destruction of the data. Depending on the sensitivity and value of the data it may be appropriate to have copies stored at multiple remote physical locations.

Managing Optical Media

CD, DVD, and Blu-ray (BD)

- /dev/cdrom and /dev/dvd (symlinks)
- eject
- mkisofs
- readcd
- cdrecord

Managing CDs, DVDs, and BDs

Managing optical media, such as a Digital Video Disc, can be useful for backups, transporting and sharing data, and manipulating data (such as with a custom installation disk). Linux uses the /dev/sr# device files, along with symbolic links (e.g. /dev/cdrom). For instance, to eject a CD-ROM, type the following:

```
$ eject /dev/cdrom
```

Some systems have the capability to close the CD tray with the **-t** option. The target can also be the mount point for the device, e.g. /mnt/.

Most modern optical drives support asynchronous notifications of events, such as media insertion, via either MMC or SATA ATAPI extensions.

Under GNOME 3, Nautilus can be configured to not automount, or autorun, when a disk, or removable media, is detected. See org.gnome.desktop.media-handling in the **dconf-editor**.

Creating a New Disk Image

The **mkisofs** command can read the file in a specified directory hierarchy and create a new image. For example, to create an ISO standard 9660 CD-ROM image of a directory, type the following:

```
$ mkisofs -o cd.iso -V "LABEL-NAME" -iso-level 2 DIR-TARGET
```

Three ISO levels are provided, with a directory nesting depth of eight

(255 total characters in the path and file name), and names limited to upper-case letters, numbers, or an underscore. Level 1 limits file names to 8 characters, with a dot separated 3 character extension. Level 2 allows for longer file names (30 characters not including the dot if using an extension). Level 3 provides for increased file size (beyond 4GB-1) with the use of file fragmentation, but is limited in use on some systems (e.g. OS X). Some tools follow a proposed extension to ISO 9660 (sometimes called ISO 9660:1999), which **mkisofs** calls Level 4.

Image Formats and mkisofs Options

To accommodate the needs of modern disk filesystems, several extensions to the base standard exist. The Joliet extensions are common on Microsoft Windows platforms, have a limitation of 64 character filenames, and use UCS-2 character coding (mostly compatible with UTF-16LE encoding used with modern Windows, see **-jcharset**). **mkisofs -J** provides the Joliet extension, (or **-joliet-long** for filenames up to 103 characters). For Unix systems, it is preferable to use the Rockridge protocol instead of, or in addition to, Joliet. To use Rockridge use the **-R** option. The similar **-r** option has preset values for UID and GID 0 (i.e. root ownership of files), gives global read access but not write, and where execute access is given to a file, the x-bit is set for user, group, and other for that file. 255 character file names are allowed, with case-sensitive naming, device files and symbolic link preservation, and with the 8 nested directory limit removed.

Compatibility with other systems, such as MS-DOS, is becoming less

of an issue. However, a TRANS.TBL translation file can be used in each directory to preserve the full file name for translation when using ISO Level 1, (which can include symbolic link paths, and block and character device major/minor numbers). Enable TRANS.TBL with the **-T** option.

Creating an Image from an Existing Disk

The **readcd** command is useful for taking existing media, and creating an image for later reproduction. It is preferred over other methods such as the **dd** command because in the event of CRC read errors, it will automatically retry. **readcd** uses a simple interactive mode that prompts for options not passed on the command line. For example:

```
# readcd -v dev=/dev/cdrom
```

Modern optical drives will automatically slow down if they encounter read errors. When trying to read from particularly dirty or damaged disks, use the **speed=#** option to set a slow read speed, which may help.

Creating Discs from Images

To burn an existing image to CD, DVD, or BD use the **cdrecord** command. In general, the **cdrecord** command will auto-detect the CD-ROM drive. A typical invocation which would show progress and eject when complete would be:

```
# cdrecord -eject -v cd.iso
```

To specify the device to be used, use the **dev=** option. This option can use the syntax: [scsibus], target, lun (e.g. **dev=15,1,0**) or can take the device path as an argument (e.g. **dev=/dev/sr0**). These can be discovered as follows:

```
# wodim --devices
wodim: Overview of accessible drives (1 found) :
-----
0 dev='/dev/scd0'    rww-- : 'Optiarc' 'DVD RW AD-7710H'
-----
# wodim -scanbus
scsibus1:
1,0,0 100) 'Optiarc' 'DVD RW AD-7710H' 'A833' Removable CD-ROM
... snip ...
```

For convenience, commonly used options (such as the device to use)

can be placed into the **/etc/wodim.conf** file. Note that a default config file is already present which sets the device to **/dev/cdrom** and specifies **burnfree**, an option that enables a feature of most modern optical drives that prevents buffer underrun errors.

Reading and Writing Audio CDs

Audio CDs use a different on disc format than data CDs. The **cddparanoia** command has extra features focused to the audio disc format that increase the chances of creating a bit perfect image from an audio disc. The **cdrdao** command can then be used to burn audio back out to a disc. This command operates the drive in Disk-at-Once burning mode which allows for things such as zero length track pre-gaps desirable on many audio disks.

Additional Examples

Originally, a PC BIOS was designed to boot floppies or IDE drives. To boot a CD, the BIOS must treat the CD as one of the two. The use of a floppy image is called the El Torito method. To make a bootable CD with a raw floppy image (**boot.img**) from the current working directory, do the following:

```
# mkisofs -o ~/cd.iso -b boot.img -c boot.cat -Jr -iso-level 2 .
```

If the image is from a hard disk, instead of a floppy (or floppy image created with **dd**), use the **-hard-disk-boot** option in addition to **-b**. To avoid using an emulated floppy, and compatibility issues with a hard disk image, programs like ISOLINUX use a custom boot image with the **-no-emul-boot** option.

To create an exact duplicate of an optical data disk that includes not just the same data, but also sub-channel data and identical TOC use the **-clone** option as shown in this example:

```
# readcd dev=/dev/dvd -clone f=disk.img
# cdrecord -clone -raw96r disk.img
```

To burn the contents of the current local directory to an optical disk loaded in the drive on a remote system:

```
# mkisofs -r . | ssh user@remote_host "cdrecord -data -"
```

Mount an optical image file using the loopback option:

```
# mount -o loop -t iso9660 disk.iso /mnt/cdrom
```

See and set the DVD region code for a drive, (warning: only a fixed number of changes may be possible):

```
# regionset
regionset version 0.1 -- reads/sets region code on DVD drives
Current Region Code settings:
RPC Phase: II
type: SET
vendor resets available: 4
user controlled changes resets available: 4
drive plays discs from region(s): 1, mask=0xFE
Would you like to change the region setting of drive? [y/n]:y
Enter the new region number for your drive [1..8]:3
New mask: 0xFC, correct? [y/n]: y
Region code set successfully!
```

Tape Libraries

SCSI Tape Device Files

- /dev/st*
- /dev/nst*

Tape Libraries accessed via SCSI generic devices

- /dev/sg0, /dev/sg1, etc

Controlled via the mtx command

- Supports barcode readers

/usr/share/doc/packages/mt_st/stinit.def.examples example file.

Manipulating Magnetic Media

The **mt** command can be used to rewind, erase, and position a tape. Each backup produces a single file on the tape, to position the tape at the start of the 3rd file, the following command can be used:

```
# mt -f /dev/nst0 asf 3
```

This does an absolute position by first rewinding the tape. Relative forwards and backwards position changes can be made by replacing **ASF** with either **FSF** or **BSF**.

Controlling Tape Libraries

Tape Libraries typically consist of one or more tape drives, a robotic arm with possibly a bar code scanner, and 2 or more tapes in slots. The **mtx** command can be used to control the library and unload and unload tapes. The **mtx** command is often scripted in conjunction with **mt**, as well as **tar**, **cpio**, or the **dump** backup commands.

The **mtx** command is used to control the robot within tape libraries. Once a tape is loaded into a tape driver, then the **mt** command can be used to manipulate the tape. The **mtx** command uses the **-f** option to specify which SCSI generic device to use. If you aren't sure which SCSI generic device maps to your tape library, then use the **inquiry** option and cycle through the SCSI generic devices until you find the library with a Product Type: Medium Changer. For example:

```
# mtx -f /dev/sg1 inquiry
```

Tape Devices and Density Modes

Linux supports a wide variety of backup hardware, ranging from floppy tape drives to enterprise-level DLT and similar large-capacity devices. Almost all SCSI devices will work without any further configuration. IDE or other non-SCSI-interface devices will require additional driver support. For devices which may or may not be supported, most distributions have a hardware compatibility list.

The first SCSI tape device is accessible via /dev/st0 and via /dev/nst0. The second file is the no-rewind device file for the same tape drive. Likewise, IDE tape drive device files are /dev/ht0 and /dev/nht0.

Most tape drives can operate in multiple modes, which include hardware compression, or different bit densities. Linux supports accessing each tape drive in four different configurations of modes: /dev/st0 == mode1, /dev/st01 == mode2, /dev/st0m == mode3, /dev/st0a == mode4

[R7] The following applies to RHEL7 only:

What configuration is mapped to each mode is defined in the /etc/stinit.def file. For DLT drives, use the /usr/share/doc/mt-st-*/stinit.def.examples example file.

[S12] The following applies to SLES12 only:

What configuration is mapped to each mode is defined in the /etc/stinit.def file. For DLT drives, use the

Product Type: Medium Changer

Vendor ID: 'ATL

Product ID: '1500

Revision: '2.08'

Attached Changer: No

Other commonly used options include:

status ⇒ Displays slot and tape drive tape media status including
any barcodes if available

inventory ⇒ Forces robot to rescan all slots and drives

load slotnum [drivenum] ⇒ Loads media for specified slot into
tape drive zero

unload [slotnum] [drivenum] ⇒ Unloads media. By default
unloads tape from drive zero to original slot.

transfer slotnum slotnum ⇒ Transfer media

Examples from the mtx Command

Display status of tape library, load a tape from slot 3 into drive zero,
and then display the status again.

```
# mtx -f /dev/sg3 status
  Storage Changer /dev/sg3:2 Drives, 20 Slots ( 1,
    Import/Export )
Data Transfer Element 0:Empty
Data Transfer Element 1:Empty
  Storage Element 1:Full :VolumeTag=BLP451S
  Storage Element 2:Full :VolumeTag=BLP452S
  Storage Element 3:Full :VolumeTag=BLP453S
. . . snip . .
# mtx -f /dev/sg3 load 3
# mtx -f /dev/sg3 status
  Storage Changer /dev/sg3:2 Drives, 20 Slots ( 1,
    Import/Export )
Data Transfer Element 0:Full (Storage Element 3 Lo-
    aded):VolumeTag = BLP453S
Data Transfer Element 1:Empty
  Storage Element 1:Full :VolumeTag=BLP451S
  Storage Element 2:Full :VolumeTag=BLP452S
  Storage Element 3:Empty
. . . snip . .
```

Backup Examples

Local Devices and Files

- `tar --xattrs -cpf /dev/nst0 /some/dir`
- `rsync -a --link-dest=../backup.1 /some/dir /data/backup.0`
- `mkisofs -r /home | cdrecord dev=/dev/hdc -v -eject -data -`
- `dump -0uf /root/bkp.dump /dev/vg01/foo
restore -xaf /root/bkp.dump`

Remote Devices and Files

- `tar -cpf user@backup.example.com:/backupdir/backup.tar --rsh-command=/usr/bin/ssh /some/dir`
- `rsync -a /some/dir backup.example.com:/data/backup.0`

Remote Devices and Files

The `tar` command can backup to remote files and devices with the help of `rmt`, the Remote MagTape protocol module. With the addition of the `--rsh-command` option, it is possible to securely connect using `ssh` instead of the insecure `rsh`. In the following example, the `tar` command is used with the `ssh` command to create a tape archive file on a remote system:

```
# tar -cvpf user@backup.example.com:/backupdir/backup.tar --rsh-command=/usr/bin/ssh /some/dir
```

Please note that it is not possible to pass arguments, such as compression and compression level, to the `ssh` command directly. Instead the `~/.ssh/config` or `/etc/ssh/ssh_config` files should be modified to give the desired options.

`rsync` is able to compare two versions of a file and transfer only the parts of the file that changed. This is much more efficient than transferring the entire file every time. However, `tar` is more efficient when no files exist on one of the systems.

Recent versions of `rsync` use `ssh` to securely transfer data. Older versions instead used insecure `rsh` by default, but the `-e` option could be used to switch to `ssh`. For example:

```
# rsync -a /some/dir backup.example.com:/data/backup.0  
# rsync -ae ssh /some/dir backup.example.com:/data/backup.0
```

Local Devices and Files

`tar`, the Tape ARchiver, was originally designed for use with tape drives, but it can also backup to normal files. Unlike some backup programs, recent versions of GNU `tar` can backup extended attributes like FACLs and SELinux contexts. The following commands could be used to backup a directory, including extended attributes, to a local tape device or file:

```
# tar --xattrs -cpf /dev/nst0 /some/dir  
# tar --xattrs -cpf /data/backup/full.tar /some/dir
```

Among its many features, `rsync` can use hard links to conserve disk space when a file has not changed since the previous backup. This means that `rsync` can be used to perform incremental backups that have the convenience of full backups. The following creates a backup in the directory `backup.0/` with hard links to identical files in the directory `backup.1/`:

```
# rsync -a --link-dest=../backup.1 /some/dir /data/backup.0
```

If the data to be backed up is not that large, CD-R/RW or DVD+/-R/RW backups may be worth considering. Assuming the system can stream data fast enough to prevent a buffer under-run, the following command would backup `/home` without needing to create a temporary disc image on the hard drive:

```
# mkisofs -r /home | cdrecord dev=/dev/hdc -v -eject -data -
```

Lab 5

Estimated Time:
S12: 35 minutes
R7: 35 minutes

Task 1: Using rsync and ssh for Backups

Page: 5-28 Time: 10 minutes

Requirements: (2 stations)

Task 2: Using tar for Backups

Page: 5-35 Time: 5 minutes

Requirements: (2 stations)

Task 3: Using cpio for Backups

Page: 5-37 Time: 5 minutes

Requirements: (2 stations)

Task 4: Creating ISO Images for Backups

Page: 5-19 Time: 5 minutes

Requirements: (1 station)

Task 5: Using dump and restore for Backups

Page: 5-21 Time: 10 minutes

Requirements: (1 station)

Objectives

- Backup files using rsync over ssh

Requirements

2 stations

Relevance

Using rsync to synchronize the files on two different systems is very efficient. By itself, rsync does not provide encryption. Combining rsync and ssh produces a very secure, efficient and easy to use solution.

Notices

- Several commands in this lab task have you connect to stationZ. You have the option to work alone and use your own station number for Z. (Working alone is less realistic, but does not require you to coordinate your efforts.)

- The following actions require administrative privileges. Switch to a root login shell:

```
$ su -  
Password: makeitso 
```

- For exploration of the rsync command, create a new data directory with several files of varying sizes:

```
# mkdir /tmp/data  
# cd /tmp/data  
# for i in $(seq 30)  
> do dd if=/dev/urandom of=file$i bs=5 count=$RANDOM  
> done  
9165+0 records in  
9165+0 records out  
45825 bytes (46 kB) copied, 0.0701888 seconds, 653 kB/s  
. . . output omitted . . .
```

Lab 5

Task 1

Using rsync and ssh for Backups

Estimated Time: 10 minutes

• Results will vary

3) Examine the newly created data files:

```
# ls -lh  
total 2.4M  
-rw-r--r-- 1 root root 125K Feb 17 11:49 file1  
-rw-r--r-- 1 root root 79K Feb 17 11:49 file10  
-rw-r--r-- 1 root root 93K Feb 17 11:49 file11  
. . . output omitted . . .
```

4) Create a duplicate of the data files in stationZ's /tmp/backup/ directory:

```
# rsync -avz -e ssh /tmp/data/ root@stationZ:/tmp/backup/  
root@stationZ's password: makeitso   
[R7] sending incremental file list  
[R7] ./  
[R7] file1  
[R7] file10  
[R7] . . . snip . . .  
[R7] file9  
[S12] building file list ... done  
[S12] data/  
[S12] data/file1  
[S12] data/file10  
[S12] . . . snip . . .  
[S12] data/file9  
sent 2282624 bytes received 620 bytes 913297.60 bytes/sec  
total size is 2279890 speedup is 1.00
```

- due to the random sizes of the created files these numbers will almost certainly be different on your system.

5) Make modifications to some of the data files. In this case, change the file's contents without changing its size (proof of this is seen—the file sizes do not change, but the MD5 sums do):

```
# for i in file1*  
> do echo -n "Before: "  
> sha256sum $i  
> ls -l $i  
> sed -i 's/a/b/' $i  
> echo -n "After: "
```

- loop through the files with names that start with file1

- Do a search and replace on the file contents.

```
> sha256sum $i  
> ls -l $i  
> echo  
> done  
Before: 0067ae6cf2ca153abc5587e595cd63ab7f1fefc6d44c31c0d20b0771ca07cdcf file1  
-rw-r--r-- 1 root root 88065 Feb 22 12:28 file1  
After: 648597eec85421c99bd0c16095cb25d582301e467f02b1763c9cd2c04c263b20 file1  
-rw-r--r-- 1 root root 88065 Feb 22 12:40 file1  
. . . snip . . .
```

- 6) Delete some of the remaining data files, and add content to others:

```
# rm -f file2[1-4]  
# for i in file2[5-8]; do cat file30 >> $i; done
```

- 7) Now that the files in the data directory have changed, use the rsync command to efficiently synchronize the backup directory to the new state by transmitting only the changes:

```
# rsync -avz -e ssh --delete /tmp/data/ root@stationZ:/tmp/backup/  
root@stationZ's password: makeitso   
sending incremental file list  
[R7] deleting file24  
[R7] deleting file23  
[R7] deleting file22  
[R7] deleting file21  
[S12] deleting data/file24  
[S12] deleting data/file23  
[S12] deleting data/file22  
[S12] deleting data/file21  
. . . snip . . .  
. /  
[R7] file1  
[R7] file10  
[S12] data/file1  
[S12] data/file10  
. . . snip . . .  
sent 1012389 bytes received 9872 bytes 292074.57 bytes/sec  
total size is 2458980 speedup is 2.41
```

- remove files no longer found in the source directory—this only occurs if the --delete option has been used
- Transfer the deltas file1 for changed files.
- Transfer the deltas file1 for changed files.

speedup is an indication of the increased efficiency gained by transferring only deltas—in this example (your exact numbers will vary), over two and a half times faster than transferring the entire file set.

- 8) As further proof that only changes are transmitted, try executing the same rsync command again and note the results:

```
# !!
rsync -avz --rsh=ssh --delete /tmp/data root@stationZ:/tmp/backup/
root@stationZ's password: makeitso 
sending incremental file list

sent 362 bytes received 20 bytes 69.45 bytes/sec
total size is 2458980 speedup is 6437.12
```

- This assumes Bash, or perhaps CSH; KSH uses the r command.

- 9) Clean up the temporary working directories to make space for future lab tasks:

```
# cd
# rm -Rf /tmp/{backup,data}
```

- 10) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Backup files using tar over ssh

Requirements

- (2 stations)

Relevance

tar is useful to perform quick backups, even across a network connection.

Notices

- Several commands in this lab task have you connect to stationZ. You have the option of selecting a lab partner and using their station number as Z, or you can opt to work alone and use your own station number for Z. (Working alone is less realistic, but does not require you to coordinate your efforts.)

- Start by making a temporary directory to work in:

```
$ cd /tmp  
$ mkdir backup
```

Lab 5

Task 2

Using tar for Backups

Estimated Time: 5 minutes

- Use tar to backup some of the files in /etc/ to the backup directory on stationZ:

(Note: the following command is one long line, but is shown with line continuation for ease in readability.)

```
$ tar --ignore-failed-read -c /etc/[a-d]* |   
> ssh guru@stationZ "cat > /tmp/backup/etc.tar"  
tar: Removing leading `/' from member names  
The authenticity of host 'stationZ (10.100.0.2)' can't be established.  
DSA key fingerprint is 04:c3:88:02:57:68:2f:29:35:80:b4:20:c7:b4:c4:bd.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'stationZ' (DSA) to the list of known hosts.  
guru@stationZ's password: work 
```

Some files, or directories, are not read accessible to the guru user. The Permission denied errors can be safely ignored on this, and future, steps.

- This step is required for the next step to work from the remote station

- c = create archive, adding files starting with a-d.
- On the other end of the ssh tunnel, use cat to "catch" the file and then redirect to a file.

- If you don't see this prompt, typing yes blindly should still work.

- 3) After the archive has been created—check with your lab partner if you are working together—use the tar command to verify its integrity and contents:

```
$ tar -tf /tmp/backup/etc.tar  
[R7] etc/abrt/  
[R7] etc/abrt/abrt-action-save-package-data.conf  
[R7] etc/abrt/arbt.conf  
[S12] etc/aclocal_dirlist  
[S12] etc/adjtime  
[S12] etc/aliases  
... snip ...
```

- -t = test (lists files in archive)
- f = operate on a file instead of STDIN

- 4) Create a duplicate of some of the directory and file structure from /etc/ (trade CPU time for network bandwidth by using compression on both ends of the pipe):

```
$ tar -cz --ignore-failed-read /etc/[a-d]* |   
> ssh stationZ "cd /tmp/backup; tar -xz"  
tar: Removing leading `/' from member names  
guru@stationZ's password: work 
```

- tar receives the piped output, decompresses and extracts the files

- 5) Verify that the files were successfully mirrored, then delete the files:

```
$ ls /tmp/backup/etc/  
[R7] a2ps.cfg      anacrontab avahi  cron.daily dbus-1  
[R7] a2ps-site.cfg at.deny    bashrc cron.deny default  
[S12] aclocal_dirlist  auto.master.d      brltty.conf      cupshelpers  
[S12] adjtime       auto.misc        ca-certificates  dbus-1  
... snip ...  
$ rm -Rf /tmp/backup/
```

- Results may vary, depending on the distribution and configuration of the remote station.

Objectives

- Backup files using cpio over ssh

Requirements

2 stations

Relevance

cpio is useful to perform quick backups, even across a network connection. While tar is easier to use, historically cpio is more portable.

Notices

- Several commands in this lab task have you connect to stationZ. You have the option of selecting a lab partner and using their station number as Z, or you can opt to work alone and use your own station number for Z. (Working alone is less realistic, but does not require you to coordinate your efforts.)

- Start by making a temporary directory to work in:

```
$ cd /tmp/  
$ mkdir backup
```

Lab 5

Task 3

Using cpio for Backups

Estimated Time: 5 minutes

- Create a partial copy of the files in /etc/ using cpio:

```
$ ls -d /etc/[a-d]* | [Enter]  
> cpio -ocB | [Enter]  
> ssh guru@stationZ \[Enter]  
  
> "cd /tmp/backup/; cpio -icBdm --no-absolute-filenames"  
guru@stationZ's password: work [Enter]  
. . . snip . . .  
11 blocks  
cpio: Removing leading `/' from member names  
11 blocks
```

- This step is required for the next step to work from the remote station

- generate the list of file names for cpio
- o = copy-out mode (create archive)
- c = use SYSV4 portable archive format
- B = use 5120 byte blocks.
- i = copy-in mode (extract from archive)
- d = make directories when needed
- m = retain previous file modification times
- no-absolute-filenames = create files relative to current directory ignoring leading slash if needed.
- Block size may vary, depending on system.

3) Verify that the files were successfully mirrored:

```
$ ls /tmp/backup/etc/  
[R7] abrt      auto.misc    cgsnapshot_blacklist.conf  cups  
[R7] acpi      auto.net     chkconfig.d            cupshelpers  
[S12] aclocal_dirlist  auto.master.d        brltty.conf       cupshelpers  
[S12] adjtime    auto.misc      ca-certificates   dbus-1  
... snip ...
```

Results may vary, depending on the distribution and configuration of the remote station.

4) Create a cpio archive of all files in /etc that have changed in the last day:

```
$ find /etc/ -mtime -1 |   
> cpio -ocB |   
> ssh guru@stationZ "cat > /tmp/backup/$(date +%j)-etc.cpio"  
guru@stationZ's password: work   
2 blocks
```

- Print the names of files that have changed in the last day and pipe them to cpio for archiving.
- Results will vary depending on changes to /etc/.

5) Verify the integrity of the newly created archive, then delete the result to clean up.
Be sure to replace XXX with the correct day of the year generated by the date command earlier:

```
$ cpio -tv < /tmp/backup/XXX-etc.cpio 2>/dev/null  
... output omitted ...  
$ rm -Rf /tmp/backup
```

Results may vary, depending on the distribution and configuration of the remote station.

Objectives

- >Create and test an ISO9660 image

Requirements

- (1 station)

Relevance

Because the media is so cheap, using CD/DVD/Blu-ray for backups is a popular solution. Using `mkisofs` it is very easy to create ISO images on Linux.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Create an ISO9660 image of the `/boot/` filesystem that would be suitable for backup to a CD-R:

```
# mkisofs -J -jcharset utf-8 -r -iso-level 2 -o /tmp/bootfs.iso /boot/  
... snip ...  
88.05% done, estimate finish Fri Oct 17 11:41:24 2014  
Total translation table size: 0  
Total rockridge attributes bytes: 27955  
Total directory bytes: 51200  
Path table size(bytes): 96  
Max brk space used 44000  
39764 extents written (77 MB)
```

Lab 5

Task 4

Creating ISO Images for Backups

Estimated Time: 5 minutes

• `-J` = Joliet directory extensions
• `-r` = Rock Ridge file name extensions
• `-o` = output to a file instead of STDOUT

- 3) Examine the `bootfs.iso` file:

```
# ls -l /tmp/bootfs.iso  
-rw-r--r-- 1 root root 81436672 Oct 17 11:41 /tmp/bootfs.iso  
# file /tmp/bootfs.iso  
/tmp/bootfs.iso: # ISO 9660 CD-ROM filesystem data 'CDROM'
```

- 4) Mount the image using a loopback device and extract a file from it:

```
# cd; mount -o loop /tmp/bootfs.iso /mnt/  
mount: /dev/loop0 is write-protected, mounting read-only  
# ls -l /mnt/  
... output omitted ...  
# cp /mnt/vmlin* /tmp/  
# ls -l /tmp/vmlin*  
... output omitted ...
```

At this point, it would make sense to use a program like cdrecord to burn this to a CD, DVD, or Blu-Ray Disc.

- 5) Unmount and delete the image:

```
# umount /mnt/  
# rm -f /tmp/bootfs.iso
```

- 6) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```

Objectives

- Backup and restore files with dump and restore

Requirements

- (1 station)

Relevance

Because dump accesses the block device directly, using dump and restore for backups provides several advantages. Even unmounted filesystems can be backed-up by dump. As the backup is happening, dump will never make any changes to the filesystem. Finally, dump has proven robust even when dealing with partially corrupted filesystems.

- 1) The following actions require administrative privileges. **Switch** to a root login shell:

```
$ su -  
Password: makeitso 
```

- 2) Install the package that provides the dump and restore commands:

```
[R7] # yum install -y dump  
[S12] # zypper install -y dump  
. . . output omitted . . .
```

- 3) Create a logical volume with an extended filesystem. Mount it and copy the contents of the /boot/ directory to it:

```
# lvcreate -L 250M -n foo vg0  
  Rounding up size to full physical extent 252.00 MiB  
  Logical volume "foo" created  
# mkfs -t ext3 /dev/vg0/foo  
. . . output omitted . . .  
# mount /dev/vg0/foo /mnt/  
# cp -R /boot/* /mnt/  
# umount /mnt/
```

Lab 5

Task 5

Using dump and restore for Backups

Estimated Time: 10 minutes

- 4) Identify the amount of space needed to do a full backup of the /boot filesystem:

```
# dump -0S /dev/vg0/foo  
[S12] DUMP: WARNING: no file `/etc/dumpdates'  
81418240
```

- Replace *sdaX* with the actual disk and partition number obtained in the previous step.
- Total space (in bytes) needed for *dump*. This will vary depending on the system.

- 5) Perform a full (level zero) backup of the /boot filesystem:

```
# dump -0uf /tmp/boot.dump /dev/vg0/foo  
[S12] DUMP: WARNING: no file `/etc/dumpdates', making an empty one  
DUMP: Date of this level 0 dump: Fri Oct 17 14:27:38 2014  
DUMP: Dumping /dev/vg0/foo (an unlisted file system) to /tmp/boot.dump  
. . . snip . . .  
DUMP: DUMP IS DONE
```

- Again replace *sdaX* with the correct info.
- *-f* = backup to this file (this would commonly be the block device file for the tape device)

- 6) List the files contained in the *dump* archive:

```
# restore -tf /tmp/boot.dump  
Dump date: Fri Oct 17 14:27:38 2014  
Dumped from: the epoch  
Level 0 dump of /boot on stationX.example.com:/dev/vg0/foo  
Label: none  
      2      .  
      11     ./lost+found  
. . . snip . . .
```

- 7) Create a directory to restore files into, and non-interactively restore a single file, /grub/stage2, from the archive:

```
# mkdir /tmp/restore/  
# cd /tmp/restore/  
# restore -xaf /tmp/boot.dump /grub2/grub.cfg  
set owner/mode for '.'? [yn] n 
```

- *-x* = extract (non-interactively)
- *-a* = do not prompt for Volume number
- *-f* = operate on a file (instead of STDIN)

- 8) Verify that the file was restored successfully:

```
# ls -l grub2/  
total 4  
-rw-r--r-- 1 root root 4089 Oct 17 14:22 grub.cfg
```

- 9) Restore a bunch of files (with names matching a specific pattern) non-interactively:

```
# restore -tf /tmp/boot.dump |  
> awk '/.mod/ {print $2}' |  
> xargs restore -xaf /tmp/boot.dump  
restore: ./grub: File exists  
set owner/mode for '.'? [yn] n
```

- Generate a list of all the files in the archive
- Match lines with regex '.mod' and output file names
- Collect the file names with xargs and pass them to the restore command for extraction
- The prompt for yn, but may return to a shell without allowing for a response. If this happens, feel free to ignore it, and proceed as if a response had been given.

- 10) Verify that the files were restored successfully:

```
# ls -l grub2/i386-pc/  
... output omitted ...
```

- 11) Delete the restored files in preparation for another restore:

```
# (cd; rm -Rf /tmp/restore/*)
```

- 12) Interactively restore a collection of files from the dump archive. Attach to the dump archive and list the contents:

```
# restore -if /tmp/boot.dump  
restore > ls  
... output omitted ...
```

Output will vary, based on installation.

- 13) Mark a file to be restored:

```
restore > add config*  
restore > ls  
.:  
System.map-XXX  
*config-XXX  
... snip ...
```

- The asterisk indicates that this file has been tagged for restore.

- 14) Mark additional files located inside the grub directory to be restored:

```
restore > cd grub2/
```

```
restore > ls
./grub2:
device.map  grub.cfg    i386-pc/      themes/
fonts/       grubenv    locale/
restore > add device.map
```

- File list may differ depending on system and distribution.

15) Extract all the tagged files into the current directory:

```
restore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume # (none if no more volumes): 1 
set owner/mode for '.'? [yn] n 
restore > quit
```

- This prompt could have safely been avoided (in this case) by using the -a parameter when you first launched restore

16) Examine the files to verify that the restore completed successfully:

```
# ls -R
.:
config-XXX
./grub2:
device.map
```

Cleanup

17) Remove the backup file:

```
# cd; rm -Rf /tmp/restore
# rm -f /tmp/boot.dump
# lvremove -f /dev/vg0/foo
```

18) Administrative privileges are no longer required; **exit** the root shell to return to an unprivileged account:

```
# exit
```