A guide to RHCSA & RHCE Certifications

Luis M. Arranz

0. Introduction

This guide has been made up from different documents, some of them self-written, some other transcribed or adapted from different technical documentation, man pages and web sites (especially Red Hat and CentOS web sites), as a reference guide covering each of the objectives set by Red Hat in order to demonstrate the abilities required of a system administrator taking the Red Hat Certified System Administrator (RHCSA) and Red Hat Certified Engineer (RHCE) exams on Red Hat Enterprise Linux 6.

Subjects in this document widely cover the exam objectives required to obtain RHCSA and RHCE certifications.

http://www.redhat.com/training/courses/ex200/examobjective
http://www.redhat.com/training/courses/ex300/examobjective

I hope this study guide will help you as it helped me.

L. Arranz

1. RHCSA/RHCE Objectives

1.1. RHCSA

Understand and use essential tools

- Access a shell prompt and issue commands with correct syntax
- Use input-output redirection (>, >>, |, 2>, etc.)
- Use grep and regular expressions to analyze text
- Access remote systems using ssh and VNC
- Log in and switch users in multiuser runlevels
- Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2
- Create and edit text files
- Create, delete, copy, and move files and directories
- Create hard and soft links
- List, set, and change standard ugo/rwx permission
- Locate, read, and use system documentation including man, info, and files in /usr/share/doc

*** **Note**: Red Hat may use applications during the exam that are not included in Red Hat Enterprise Linux for the purpose of evaluating candidate's abilities to meet this objective.

Operate running systems

- Boot, reboot, and shut down a system normally
- Boot systems into different runlevels manually
- Use single-user mode to gain access to a system
- Identify CPU/memory intensive processes, adjust priority with renice, and kill processes
- Locate and interpret system log files
- Access a virtual machine's console
- Start and stop virtual machines
- Start, stop, and check the status of network services

Configure local storage

- List, create, delete, and set partition type for primary, extended, and logical partitions
- Create and remove PVs, assign physical volumes to volume groups, and create and delete logical volumes
- Create and configure LUKS-encrypted partitions and LVs to prompt for password and mount a decrypted file system at boot
- Configure systems to mount file systems at boot by Universally Unique ID (UUID) or label
- Add new partitions and logical volumes, and swap to a system non-destructively

Create and configure file systems

- Create, mount, unmount, and use ext2, ext3, and ext4 file systems
- Mount, unmount, and use LUKS-encrypted file systems
- Mount and unmount CIFS and NFS network file systems
- Configure systems to mount ext4, LUKS-encrypted, and network file systems automatically
- Extend existing unencrypted ext4-formatted logical volumes
- Create and configure set-GID directories for collaboration
- Create and manage Access Control Lists (ACLs)
- Diagnose and correct file permission problems

Deploy, configure, and maintain systems

- Configure networking and hostname resolution statically or dynamically

1.2. RHCE

System configuration and management

- Route IP traffic and create static routes / Adding and deleting routes from IP routing table
- Use iptables for packet filtering & configure Network Address Translation (NAT)
- Use /proc/sys and sysctl to modify and set kernel runtime parameters
- Configure a system to authenticate using Kerberos
- Build a simple RPM that packages a single file
- Configure a system as an iSCSI initiator that persistently mounts an iSCSI target
- Produce and deliver reports on system utilization (processor, memory, disk and network)
- Use shell scripting to automate system maintenance tasks
- Configure a system to log to a remote system
- Configure a system to accept logging from a remote system

Network services

*** **Note**: Network services are an important subset of the exam objectives. RHCE candidates should be capable of meeting the following objectives **for each of the network services listed below**:

- Install the packages needed to provide the service
- Configure SELinux to support the service
- Configure the service to start when the system is booted
- Configure the service for basic operation
- Configure host-based and user-based security for the service

HTTP/HTTPS

- Configure a virtual host
- Configure private directories
- Deploy a basic CGI application
- Configure group-managed content
- Quick configuration procedure

DNS

- Configure a caching-only name server
- Configure a caching-only name server to forward DNS queries
- Quick configuration procedure

*** **Note**: Candidates are not expected to configure master or slave name servers

FTP

- Configure anonymous-only download
- Quick configuration procedure

NFS

- Provide network shares to specific clients
- Provide network shares suitable for group collaboration
- Quick configuration procedure

SMB

- Provide network shares to specific clients
- Provide network shares suitable for group collaboration
- Quick configuration procedure

2. SysV Init Runlevels

The SysV init runlevel system provides a standard process for controlling which programs **init** launches or halts when initializing a runlevel. SysV init was chosen because it is easier to use and more flexible than the traditional BSD-style init process.
The configuration files for SysV init are located in the **/etc/rc.d/** directory. Within this directory, are the **rc**, **rc.local**, **rc.sysinit**, and, optionally, the **rc.serial** scripts as well as the following directories:

init.d/ rc0.d/ rc1.d/ rc2.d/ rc3.d/ rc4.d/ rc5.d/ rc6.d/

The **init.d/** directory contains the scripts used by the **/sbin/init** command when controlling services. Each of the numbered directories represents the six runlevels configured by default under Red Hat Enterprise Linux.

2.1. Runlevels

The idea behind SysV init runlevels revolves around the idea that different systems can be used in different ways. For example, a server runs more efficiently without the drag on system resources created by the X Window System. Or there may be times when a system administrator may need to operate the system at a lower runlevel to perform diagnostic tasks, like fixing disk corruption in runlevel 1.
The characteristics of a given runlevel determine which services are halted and started by **init**. For instance, runlevel 1 (single user mode) halts any network services, while runlevel 3 starts these services. By assigning specific services to be halted or started on a given runlevel, **init** can quickly change the mode of the machine without the user manually stopping and starting services.
The following runlevels are defined by default under Red Hat Enterprise Linux:

- **0** - Halt
- **1** - Single-user text mode
- **2** - Not used (user-definable)
- **3** - Full multi-user text mode
- **4** - Not used (user-definable)
- **5** - Full multi-user graphical mode (with an X-based login screen)
- **6** - Reboot

In general, users operate Red Hat Enterprise Linux at runlevel 3 or runlevel 5 - both full multi-user modes. Users sometimes customize runlevels 2 and 4 to meet specific needs, since they are not used.
The default runlevel for the system is listed in **/etc/inittab**. To find out the default runlevel for a system, look for the line similar to the following near the bottom of **/etc/inittab**:

```
id:5:initdefault:
```

The default runlevel listed in this example is five, as the number after the first colon indicates. To change it, edit **/etc/inittab** as root.

Be very careful when editing /etc/inittab. Simple typos can cause the system to become unbootable. If this happens, either use a boot diskette, enter single-user mode, or enter rescue mode to boot the computer and repair the file.

To change the runlevel immediately, use the command **telinit** followed by the runlevel number. You must be root to use this command. The **telinit** command does not change the **/etc/inittab** file; it only changes the runlevel currently running. When the system is rebooted, it continues to boot the runlevel as specified in **/etc/inittab**.

2.2. Changing Runlevels (the process of)

Two sets of init scripts are involved in switching runlevels: K-scripts from the runlevel on which we are currently running and S-scripts from a new (target) runlevel. It is important to understand that there are two runlevels involved in switching:

Runlevel from which we switch. Let's call it **L_FROM**

Runlevel to which we switch. Let's call it **L_TO**

Simplifying, the rc script which is responsible for changing runlevels performs the following two operations:

All **K**-scripts that exist on the current runlevel **L_FROM** are processed. For each **K**-script for which **S**-script does not exist at the target runlevel **L_TO**, the **K**-script from **L_FROM** is executed.

Each **S**-script that exist at target runlevel **L_TO** is executed

In other words when you change the runlevel, first stop scripts of the current runlevel are launched, closing down some daemons running on the current runlevel that are not necessary on a new runlevel (do not have **S**-script defined for them). Then all start scripts of the new runlevel are run in the order defined by their numeric priorities.

For example, the following occurs when changing from runlevel 3 to 5:

- The administrator (root) enters the command **init 5** which tells init to change the current runlevel to 5.

- The init consults its configuration file (**/etc/inittab**) and determines it should start /etc/init.d/rc with the new runlevel as a parameter.

- Now rc calls all the stop scripts of the current runlevel, but only those for which there is no start script in the new runlevel. In this example, these are all the scripts that reside in **/etc/init.d/rc3.d** (old runlevel was 3) and start with a **K**. The number following **K** specifies the order to start, because there are some dependencies to consider.

- The last things to start are the start scripts of the new runlevel. These are, in this example, in **/etc/init.d/rc5.d** and begin with an **S**. The same procedure regarding the order in which they are started is applied here.

When changing into the same runlevel as the current runlevel, init only checks **/etc/inittab** for changes and starts the appropriate scripts.

2.3. Managing services scripts with 'chkconfig' command

If one of **on**, **off**, **reset**, or **resetpriorities** is specified after the service name, **chkconfig** changes the startup information for the specified service. The **on** and **off** flags cause the service to be started or stopped, respectively, in the runlevels being changed.

The **reset** flag resets the on/off state for all runlevels for the service to whatever is specified in the init script in question, while the **resetpriorities** flag resets the start/stop priorities for the service to whatever is specifed in the init script.

By default, the **on** and **off** options affect only runlevels 2, 3, 4, and 5, while **reset** and **resetpriorities** affects all of the runlevels. The **--level** option may be used to specify which runlevels are affected.

Note that for every service, each runlevel has either a start script or a stop script. When switching runlevels, init will not restart an already-started service, and will not re-stop a service that is not running.

chkconfig also can manage **xinetd** scripts via the means of **xinetd.d** configuration files. Note that only the **on**, **off**, and **--list** commands are supported for **xinetd.d** services.

For example, random.init has these three lines:

```
# chkconfig: 2345 20 80
# description: Saves and restores system entropy pool for \
#              higher quality random number generation.
```

This says that the random script should be started in levels 2, 3, 4, and 5, that its start priority should be 20, and that its stop priority should be 80. One should be able to figure out what the description says; the \ causes the line to be continued. The extra space in front of the line is ignored.

2.4. Changing Runlevels at Boot Time

It is possible to change the default runlevel at boot time by modifying the arguments passed by the boot loader to the kernel.
To change the runlevel of a single boot session, use the following instructions:

- When the GRUB menu bypass screen appears at boot time, press any key to enter the GRUB menu (within the first three seconds).

- Press the '**a**' key to append to the **kernel** command.

- Add <**space**> *<runlevel>* at the end of the boot options line to boot to the desired runlevel.

For example, the following entry would initiate a boot process into runlevel 3:

grub append> ro root=/dev/VolGroup00/LogVol00 rhgb quiet 3

3. Shutting down and rebooting a system

To shut down Red Hat Enterprise Linux, the root user may issue the **/sbin/shutdown** command.

The shutdown man page has a complete list of options, but the two most common uses are:

```
# /sbin/shutdown -h now
```

and

```
# /sbin/shutdown -r now
```

After shutting everything down, the **-h** option halts the machine, and the **-r** option reboots.

Changing to init 6 will reboot as well, which is what init 6 does:

```
# init 6
```

On the same note, init 0 calls all of the shutdown scripts and gracefully shuts down your machine:

```
# init 0
```

If the computer does not power itself down, be careful not to turn off the computer until a message appears indicating that the system is halted.

Failure to wait for this message can mean that not all the hard drive partitions are unmounted, which can lead to file system corruption.

PAM console users can use the **reboot** and **halt** commands to shut down the system.

Normal reboot:

```
# reboot
```

3.1. Boot systems into different runlevels manually

Red Hat Enterprise Linux is similar to most other linux distributions in its core functionality. The ability to run the operating system in multiple run levels is an important skill to have.

If you type into your terminal:

```
# runlevel
```

you should see a couple of numbers/characters as the output (first number indicates previous status; second one indicates current status):

```
N 3
```

There are 6 runlevels:

Runlevel **0** - **Halt**
Runlevel **1** - **Single User mode**. Most services turned off, including networking. Used to perform maintenance on the server usually. Boots logged into roots account, no password.
Runlevel **2** - This is basic functions, multi-user mode, **without any networking**.
Runlevel **3** - This is **what servers usually run** in, as it provides all of the services of the normal server, without the graphical user interface.
Runlevel **4** - Doesn't really get used.
Runlevel **5** - This provides the same functions of runlevel 3, along with services to allow for **desktop functionality** (graphical user interface).
Runlevel **6** - **Reboot**

The command to jump runlevels is actually really easy. Just type init followed by the runlevel you want to switch into.

```
# init 1
```

The above command would turn off most services and drop you into single user mode.

4. Forgotten 'root' password / using single-user to gain access

Booting into single user mode is the easiest way to gain access to a Red Hat Enterprise Linux server (only feasible if you have access to the physical console).

To enter single-user mode, reboot your computer. If you use the default boot loader, GRUB, you can enter single user mode by performing the following:

Method A:

1.- At the boot loader menu, use the arrow keys to highlight the installation you want to edit and type '**a**' to enter into append mode.

2.- You are presented with a prompt that looks similar to the following:

```
grub append> ro root=LABEL=/
```

3.- Press the Spacebar once to add a blank space, then add the word '**single**' to tell GRUB to boot into single-user Linux mode. The result should look like the following:

```
ro root=LABEL=/ single
```

4.- Press [Enter] and GRUB will boot single-user Linux mode. After it finishes loading, you will be presented with a shell prompt.

5.- You are now in single user mode, and be auto logged in as root. You can now change the root password by typing:

```
# passwd root
```

*** **Note**: For Red Hat Enterprise Linux 6.0 there is a bug that will prevent you from changing your root password in single user mode. This is a result of SELinux. For this situation you would want to temporarily disable SELinux.

```
# setenforce 0
```

Now you should be allowed to change your root password.

Method B:

1.- At the beginning of the boot process you should see the grub menu pop up with a countdown and some kernel options (or perhaps just one option). It should be counting down at this point and says: "**Press any key to enter the menu**". In this case you would hit any key.

2.- At the bottom of the screen there is an explanation of the few options that are available to use on this page. One of these options is '**e**' for edit. Hit '**e**' to edit the boot kernel options.

3.- You would now edit the main kernel options, adding '**single**' at the end.

4.- Once you have completed that hit enter, then '**b**' for boot.

5.- You are now in single user mode, and be auto logged in as root. You can now change the root password by typing:

```
# passwd root
```

***** Note**: For Red Hat Enterprise Linux 6.0 there is a bug that will prevent you from changing your root password in single user mode. This is a result of SELinux. For this situation you would want to temporarily disable SELinux.

```
# setenforce 0
```

Now you should be allowed to change your root password.

5. Basic system recovery

5.1. Booting into Rescue Mode

Rescue mode provides the ability to boot a small Red Hat Enterprise Linux environment entirely from CD-ROM, or some other boot method, instead of the system's hard drive.

As the name implies, rescue mode is provided to rescue you from something. During normal operation, your Red Hat Enterprise Linux system uses files located on your system's hard drive to do everything — run programs, store your files, and more.
However, there may be times when you are unable to get Red Hat Enterprise Linux running completely enough to access files on your system's hard drive. Using rescue mode, you can access the files stored on your system's hard drive, even if you cannot actually run Red Hat Enterprise Linux from that hard drive.
To boot into rescue mode, you must be able to boot the system using one of the following methods:

- By booting the system from a boot CD-ROM or DVD.
- By booting the system from other installation boot media, such as USB flash devices.
- By booting the system from the Red Hat Enterprise Linux installation DVD.

Once you have booted using one of the described methods, add the keyword **rescue** as a kernel parameter. For example, for an x86 system, type the following command at the installation boot prompt:

```
linux rescue
```

If a driver that is part of the Red Hat Enterprise Linux distribution prevents the system from booting, blacklist that driver with the **rdblacklist** option. For example, to boot into rescue mode without the foobar driver, run:

```
linux rescue rdblacklist=foobar
```

You are prompted to answer a few basic questions, including which language to use. It also prompts you to select where a valid rescue image is located. Select from Local CD-ROM, Hard Drive, NFS image, FTP, or HTTP. The location selected must contain a valid installation tree, and the installation tree must be for the same version of Red Hat Enterprise Linux as the Red Hat Enterprise Linux disk from which you booted. If you used a boot CD-ROM or other media to start rescue mode, the installation tree must be from the same tree from which the media was created.

If you select a rescue image that does not require a network connection, you are asked whether or not you want to establish a network connection. A network connection is useful if you need to backup files to a different computer or install some RPM packages from a shared network location, for example.

The following message is displayed:

The rescue environment will now attempt to find your Linux installation and mount it under the directory /mnt/sysimage. You can then make any changes required to your system. If you want to proceed with this step choose 'Continue'. You can also choose to mount your file systems read-only instead of read-write by choosing 'Read-only'. If for some reason this process fails you can choose 'Skip' and this step will be skipped and you will go directly to a command shell.

If you select **Continue**, it attempts to mount your file system under the directory /mnt/sysimage. If it fails to mount a partition, it notifies you. If you select Read-Only, it attempts to mount your file system under the directory /mnt/sysimage, but in read-only mode. If you select **Skip**, your file system is not mounted. Choose **Skip** if you think your file system is corrupted.

Once you have your system in rescue mode, a prompt appears on VC (virtual console) 1 and VC 2 (use the **Ctrl-Alt-F1** key combination to access VC 1 and Ctrl-Alt-F2 to access VC 2):

```
sh-3.00b#
```

If you selected **Continue** to mount your partitions automatically and they were mounted successfully, you are in single-user mode.

Even if your file system is mounted, the default root partition while in rescue mode is a temporary root partition, not the root partition of the file system used during normal user mode (runlevel 3 or 5). If you selected to mount your file system and it mounted successfully, you can change the root partition of the rescue mode environment to the root partition of your file system by executing the following command:

```
chroot /mnt/sysimage
```

This is useful if you need to run commands such as **rpm** that require your root partition to be mounted as **/**. To exit the chroot environment, type **exit** to return to the prompt.

If you selected **Skip**, you can still try to mount a partition or LVM2 logical volume manually inside rescue mode by creating a directory such as **/foo**, and typing the following command:

```
mount -t ext4 /dev/mapper/<VolGroup00-LogVol02> /foo
```

In the above command, **/foo** is a directory that you have created and **/dev/mapper/VolGroup00-LogVol02** is the LVM2 logical volume you want to mount. If the partition is of type **ext2** or **ext3** replace **ext4** with **ext2** or **ext3** respectively.

If you do not know the names of all physical partitions, use the following command to list them:

```
fdisk -l
```

From the prompt, you can run many useful commands, such as:

ssh, **scp**, and **ping** if the network is started
dump and **restore** for users with tape drives
parted and **fdisk** for managing partitions
rpm for installing or upgrading software
vi for editing text files

5.2. Booting into Single-User Mode

One of the advantages of single-user mode is that you do not need a boot CD-ROM; however, it does not give you the option to mount the file systems as read-only or not mount them at all.

If your system boots, but does not allow you to log in when it has completed booting, try single-user mode.

In single-user mode, your computer boots to runlevel 1. Your local file systems are mounted, but your network is not activated. You have a usable system maintenance shell. Unlike rescue mode, single-user mode automatically tries to mount your file system. Do not use single-user mode if your file system cannot be mounted successfully. You cannot use single-user mode if the runlevel 1 configuration on your system is corrupted.

On an x86 system using GRUB, use the following steps to boot into single-user mode:

- At the GRUB splash screen at boot time, press **any key** to enter the GRUB interactive menu.
- Select Red Hat Enterprise Linux with the version of the kernel that you wish to boot and type "**a**" to append the line.
- Go to the end of the line and type single as a separate word (press the Spacebar and then type **single**). Press Enter to exit edit mode.

5.3. Booting into Emergency Mode

In emergency mode, you are booted into the most minimal environment possible. The root file system is mounted read-only and almost nothing is set up. The main advantage of emergency mode over single-user mode is that the init files are not loaded. If init is corrupted or not working, you can still mount file systems to recover data that could be lost during a re-installation.
To boot into emergency mode, use the same method as described for single-user mode in Section with one exception, replace the keyword **single** with the keyword **emergency**.

6. kdump: Config. kernel dumps & analyzing the crash dump

Kernel dumps may provide invaluable insights when debugging serious issues.

1.- Install following RPMs:

```
kexec-tools
crash
```

To be able to analyze crash dumps, following packages should be also installed:

```
kernel-debuginfo-common
kernel-debuginfo
```

2.- Append "**crashkernel=128M@16M**" to the kernel parameters in /boot/grub/grub.conf (see next points to adjust the size of crashkernel):

```
        kernel /vmlinuz-2.6.18-238.el5 ro root=/dev/rvg/rootlv nodmraid rhgb quiet
crashkernel=128M@16M
```

3.- Reboot the system

4.- Check that the crash kernel has been loaded:

```
# cat /proc/iomem | grep Crash\ kernel
01000000-08ffffff : Crash kernel
```

5.- Configure kdump to dump to:

- either locally:

```
# echo > /etc/kdump.conf << EOF
path /var/crash
core_collector makedumpfile -d 31 -c
EOF
```

***** Note**: This config can be done also by running:

```
# system-config-kdump
```

Please check the option box "Enable kdump" at the top of the Dialog.

Next, you have to define the memory to reserve for Kdump In the dialog you see the memory information for your system and the usable memory for Kdump. On most systems a value of "128MB" Kdump memory should be enough.

Finally, you need to define a location where to store the dump file. You have the choice between 'file', 'nfs', 'ssh', 'raw', 'ext2', and 'ext3'.
This setup is straight forward, please configure the kdump as it fit's best into your environment. The simplest configuration for the location is "**file:///var/crash**".

- or to a remote server:

```
# echo > /etc/kdump.conf << EOF
net root@<kdump_remote_server>
core_collector makedumpfile -d 31 -c
EOF
```

6.- Propagate SSH keys so that the vmcore could be sent via scp without the need to enter any password (Take this point into account only if dumping to a remote server):

```
# service kdump propagate
```

7.- Configure kdump to start automatically

```
# chkconfig kdump on
# service kdump start
```

8.- Sync all filesystems:

```
# sync
```

9.- Provoke a kernel panic with:

```
# echo "1" > /proc/sys/kernel/sysrq
# echo "c" > /proc/sysrq-trigger
```

10.- Now the crash kernel should get booted and on the remote system a vmcore should get created under /var/crash:

```
# tree /var/crash
/var/crash
|-- 192.168.12.227-2010-01-21-20:16:16
`-- vmcore.flat
```

11.- The vmcore.flat needs to be processed in order to analyze the core dump via the crash utility:

```
# cat "vmcore.flat" | makedumpfile -R "/tmp/vmcore"
The dumpfile is saved to /tmp/vmcore.
makedumpfile Completed.
```

12.- Now you may analyze the vmcore with the crash utility:

```
# crash /usr/lib/debug/lib/modules/2.6.18-128.1.10.el5/vmlinux /tmp/vmcore

crash 4.0-8.9.1.el5
Copyright (C) 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009  Red Hat, Inc.
Copyright (C) 2004, 2005, 2006  IBM Corporation
Copyright (C) 1999-2006  Hewlett-Packard Co
Copyright (C) 2005, 2006  Fujitsu Limited
Copyright (C) 2006, 2007  VA Linux Systems Japan K.K.
Copyright (C) 2005  NEC Corporation
Copyright (C) 1999, 2002, 2007  Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002  Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public License,
and you are welcome to change it and/or distribute copies of it under
certain conditions.  Enter "help copying" to see the conditions.
This program has absolutely no warranty.  Enter "help warranty" for details.

GNU gdb 6.1
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-unknown-linux-gnu"...

 KERNEL: /usr/lib/debug/lib/modules/2.6.18-128.1.10.el5/vmlinux
 DUMPFILE: /tmp/vmcore  [PARTIAL DUMP]
 CPUS: 1
 DATE: Thu Jan 21 20:21:20 2010
 UPTIME: 00:03:10
 LOAD AVERAGE: 1.09, 0.46, 0.17
 TASKS: 445
 NODENAME: vrhel03
 RELEASE: 2.6.18-128.1.10.el5
 VERSION: #1 SMP Wed Apr 29 13:53:08 EDT 2009
 MACHINE: x86_64  (2666 Mhz)
 MEMORY: 1 GB
 PANIC: "SysRq : Trigger a crashdump"
 PID: 7835
 COMMAND: "bash"
 TASK: ffff81040699d0c0  [THREAD_INFO: ffff8103fed24000]
 CPU: 1
 STATE: TASK_RUNNING (SYSRQ)

crash>
```

6.1. The kdump procedure

1.- The normal kernel is booted with **crashkernel=...** as a kernel option, reserving some memory for the kdump kernel. The memory reserved by the crashkernel parameter is not available to the normal kernel during regular operation. It is reserved for later use by the kdump kernel.
2.- The system panics.
3.- The kdump kernel is booted using kexec, it uses the memory area that was reserved via the crashkernel parameter.
4.- The normal kernel's memory is captured into a vmcore.

***** Note**: Not reserving enough memory for the kdump kernel can lead to the kdump operation failing.

***** Warning**: Unless the system has enough memory, the Kernel Dump Configuration utility will not start and you will be presented with an error message.

6.2. Configuring crashkernel on RHEL6.0 and RHEL6.1 kernels

Some mappings of ram and appropriate crashkernel values:

ram size	crashkernel parameter	ram / crashkernel factor
>0GB	128MB	15
>2GB	256MB	23
>6GB	512MB	15
>8GB	768MB	31

The last column contains a ram/crashkernel factor.

The table is covered by the following crashkernel configuration:

```
crashkernel=0M-2G:128M,2G-6G:256M,6G-8G:512M,8G-:768M
```

For servers with more RAM it is recommended to compute the crashkernel parameter using the factors that have been observed so far: 15 to stay on a safe side (maybe wasting memory), using a factor of 20 should also work. Please also note that the maximum size of RAM that should be reserved here is 896M.

6.3. Configuring crashkernel on RHEL6.2 (and later) kernels

Starting with RHEL6.2 kernels **crashkernel=auto** should be used. The kernel will automatically reserve an appropriate amount of memory for the kdump kernel.

Additionally some improvements have been made in the RHEL6.2 kernel which have reduced the overall memory requirements of kdump.

The amount of memory reserved for the kdump kernel can be estimated with the following scheme:

base memory to be reserved = 128MB
an additional 64MB added for each TB of physical RAM present in the system. So
for example if a system has 1TB of memory 192MB (128MB + 64MB) will be reserved.

*** **Note**: It is recommended to verify that kdump is working on all systems after installation of all applications. The memory reserved by **crashkernel=auto** takes only typical RHEL configurations into account. If 3rd party modules are used more memory might have to be reserved. Thus, if a testdump fails it is a good strategy to verify if it works with **crashkernel=768M@0M** and if it does do further debugging of the memory requirements using the **debug_mem_level** option in **/etc/kdump.conf**.

*** **Note**: **crashkernel=auto** will only reserve memory on systems with 4GB or more physical memory. If the system has less than 4GB of memory the memory must be reserved in explicitly configuring **crashkernel=128M**. Since RHEL6.3GA (*kernel-2.6.32-279.el6*) this limit has been lowered to 2GB.

*** **Warning**: You need to take care that you have enough disk space on the configured location.

7. Setting local hostname and domain name

To change the hostname of the system, modify it in **/etc/hosts** and **/etc/sysconfig/network** and run following command

```
# hostname <newname>
```

8. Users and Groups

Managing users and groups can be a tedious task; this is why Red Hat Enterprise Linux provides tools and conventions to make them easier to manage.

The easiest way to manage users and groups is through the graphical application, User Manager (**system-config-users**).

8.1. Users and Groups management

8.1.1. Adding a user

To add a user to the system issue the **useradd** command that will create a locked user account:

```
# useradd <username>
```

Unlock the account by issuing the **passwd** command to assign a password and set password aging guidelines:

```
# passwd <username>
```

Command line options for **useradd**:

Option	Description
-c '*<comment>*'	*<comment>* can be replaced with any string. This option is generally used to specify the full name of a user.
-d *<home-dir>*	Home directory to be used instead of default **/home/*<username>*/**
-e *<date>*	Date for the account to be disabled in the format YYYY-MM-DD
-f *<days>*	Number of days after the password expires until the account is disabled. If **0** is specified, the account is disabled immediately after the password expires. If **-1** is specified, the account is not be disabled after the password expires.
-g *<group-name>*	Group name or group number for the user's default group. The group must exist prior to being specified here.
-G *<group-list>*	List of additional (other than default) group names or group numbers, separated by commas, of which the user is a member. The groups must exist prior to being specified here.
-m	Create the home directory if it does not exist.
-M	Do not create the home directory.
-n	Do not create a user private group for the user.
-r	Create a system account with a UID less than 500 and without a home directory
-p *<password>*	The password encrypted with **crypt**
-s	User's login shell, which defaults to **/bin/bash**
-u *<uid>*	User ID for the user, which must be unique and greater than 499

8.1.2. Adding a group

To add a group to the system, use the command **groupadd**:

```
# groupadd <group-name>
```

Command line options for **groupadd**:

Option	Description
-g *<gid>*	Group ID for the group, which must be unique and greater than 499
-r	Create a system group with a GID less than 500
-f	When used with **-g** *<gid>* and *<gid>* already exists, **groupadd** will choose another unique *<gid>* for the group.

8.1.3. Password aging

For security reasons, it is advisable to require users to change their passwords periodically.

To configure password expiration for a user use the **chage** command with of the following options followed by the username:

Option	Description
-m *<days>*	Specifies the minimum number of days between which the user must change passwords. If the value is 0, the password does not expire.
-M *<days>*	Specifies the maximum number of days for which the password is valid. When the number of days specified by this option plus the number of days specified with the **-d** option is less than the current day, the user must change passwords before using the account.
-d *<days>*	Specifies the number of days since January 1, 1970 the password was changed
-I *<days>*	Specifies the number of inactive days after the password expiration before locking the account. If the value is 0, the account is not locked after the password expires.
-E *<date>*	Specifies the date on which the account is locked, in the format YYYY-MM-DD. Instead of the date, the number of days since January 1, 1970 can also be used.
-W *<days>*	Specifies the number of days before the password expiration date to warn the user.
-l	Lists current account aging settings.

***** Tip**: If the **chage** command is followed directly by a username (with no options), it displays the current password aging values and allows them to be changed interactively.

You can configure a password to expire the first time a user logs in. This forces users to change passwords immediately:

```
# chage -d 0 <username>
```

This command sets the value for the date the password was last changed to the epoch (January 1, 1970). This value forces immediate password expiration no matter what password aging policy, if any, is in place.

8.2. Standard users

The groupid (GID) in this table is the primary group for the user

User	UID	GID	Home Directory	Shell
root	0	0	/root	/bin/bash
bin	1	1	/bin	/sbin/nologin
daemon	2	2	/sbin	/sbin/nologin
adm	3	4	/var/adm	/sbin/nologin
lp	4	7	/var/spool/lpd	/sbin/nologin
sync	5	0	/sbin	/bin/sync
shutdown	6	0	/sbin	/sbin/shutdown
halt	7	0	/sbin	/sbin/halt
mail	8	12	/var/spool/mail	/sbin/nologin
news	9	13	/etc/news	
uucp	10	14	/var/spool/uucp	/sbin/nologin
operator	11	0	/root	/sbin/nologin
games	12	100	/usr/games	/sbin/nologin
gopher	13	30	/var/gopher	/sbin/nologin
ftp	14	50	/var/ftp	/sbin/nologin
nobody	99	99	/	/sbin/nologin
rpm	37	37	/var/lib/rpm	/sbin/nologin
vcsa	69	69	/dev	/sbin/nologin
dbus	81	81	/	/sbin/nologin
ntp	38	38	/etc/ntp	/sbin/nologin
canna	39	39	/var/lib/canna	/sbin/nologin
nscd	28	28	/	/sbin/nologin
rpc	32	32	/	/sbin/nologin
postfix	89	89	/var/spool/postfix	/sbin/nologin
mailman	41	41	/var/mailman	/sbin/nologin
named	25	25	/var/named	/bin/false
amanda	33	6	var/lib/amanda/	/bin/bash
postgres	26	26	/var/lib/pgsql	/bin/bash
exim	93	93	/var/spool/exim	/sbin/nologin
sshd	74	74	/var/empty/sshd	/sbin/nologin
rpcuser	29	29	/var/lib/nfs	/sbin/nologin
nsfnobody	65534	65534	/var/lib/nfs	/sbin/nologin
pvm	24	24	/usr/share/pvm3	/bin/bash
apache	48	48	/var/www	/sbin/nologin
xfs	43	43	/etc/X11/fs	/sbin/nologin
gdm	42	42	/var/gdm	/sbin/nologin
htt	100	101	/usr/lib/im	/sbin/nologin
mysql	27	27	/var/lib/mysql	/bin/bash
webalizer	67	67	/var/www/usage	/sbin/nologin
mailnull	47	47	/var/spool/mqueue	/sbin/nologin
smmsp	51	51	/var/spool/mqueue	/sbin/nologin
squid	23	23	/var/spool/squid	/sbin/nologin
ldap	55	55	/var/lib/ldap	/bin/false
netdump	34	34	/var/crash	/bin/bash
pcap	77	77	/var/arpwatch	/sbin/nologin
radiusd	95	95	/	/bin/false
radvd	75	75	/	/sbin/nologin
quagga	92	92	/var/run/quagga	/sbin/login
wnn	49	49	/var/lib/wnn	/sbin/nologin
dovecot	97	97	/usr/libexec/dovecot	/sbin/nologin

8.3. Standard groups

Group	GID	Members
root	0	root
bin	1	root, bin, daemon
daemon	2	root, bin, daemon
sys	3	root, bin, adm
adm	4	root, adm, daemon
tty	5	
disk	6	root
lp	7	daemon, lp
mem	8	
kmem	9	

wheel	10	root
mail	12	mail, postfix, exim
news	13	news
uucp	14	uucp
man	15	
games	20	
gopher	30	
dip	40	
ftp	50	
lock	54	
nobody	99	
users	100	
rpm	37	
utmp	22	
floppy	19	
vcsa	69	
dbus	81	
ntp	38	
canna	39	
nscd	28	
rpc	32	
postdrop	90	
postfix	89	
mailman	41	
exim	93	
named	25	
postgres	26	
sshd	74	
rpcuser	29	
nfsnobody	65534	
pvm	24	
apache	48	
xfs	43	
gdm	42	
htt	101	
mysql	27	
webalizer	67	
mailnull	47	
smmsp	51	
squid	23	
ldap	55	
netdump	34	
pcap	77	
quaggavt	102	
quagga	92	
radvd	75	
slocate	21	
wnn	49	
dovecot	97	
radiusd	95	

8.4. User private groups

Red Hat Enterprise Linux uses a *user private group (UPG)* scheme, which makes UNIX groups easier to manage.

A UPG is created whenever a new user is added to the system. A UPG has the same name as the user for which it was created and that user is the only member of the UPG.

UPGs make it safe to set default permissions for a newly created file or directory, allowing both the user and the group of that user to make modifications to the file or directory.

The setting which determines what permissions are applied to a newly created file or directory is called a umask and is configured in the **/etc/bashrc** file. Traditionally on UNIX systems, the **umask** is set to **022**, which allows only the user who created the file or directory to make modifications. Under this scheme, all other users, including members of the creator's group, are not allowed to make any modifications. However, under the UPG scheme, this "group protection" is not necessary since every user has their own private group.

8.4.1. Group directories

Many IT organizations like to create a group for each major project and then assign people to the group if they need to access that project's files. Using this traditional scheme, managing files has been difficult; when someone creates a file, it is associated with the primary group to which they belong. When a single person works on multiple projects, it is difficult to associate the right files with the right group. Using the UPG scheme, however, groups are automatically assigned to files created within a directory with the setgid bit set. The setgid bit makes managing group projects that share a common directory very simple because any files a user creates within the directory are owned by the group which owns the directory.

Let us say, for example, that a group of people need to work on files in the **/usr/share/emacs/site-lisp/** directory. Some people are trusted to modify the directory, but certainly not everyone is trusted. First create an **emacs** group, as in the following command:

```
# groupadd emacs
```

To associate the contents of the directory with the **emacs** group, type:

```
# chown -R root.emacs /usr/share/emacs/site-lisp
```

Now, it is possible to add the proper users to the group with the **gpasswd** command:

```
# gpasswd -a <username> emacs
```

To allow users to create files within the directory, use the following command:

```
# chmod 775 /usr/share/emacs/site-lisp
```

When a user creates a new file, it is assigned the group of the user's default private group. Next, set the setgid bit, which assigns everything created in the directory the same group permission as the directory itself (**emacs**). Use the following command:

```
# chmod 2775 /usr/share/emacs/site-lisp
```

At this point, because the default umask of each user is 002, all members of the **emacs** group can create and edit files in the **/usr/share/emacs/site-lisp/** directory without the administrator having to change file permissions every time users write new files.

8.5. Shadow passwords

In multiuser environments it is very important to use shadow passwords (provided by the **shadow-utils** package). Doing so enhances the security of system authentication files. For this reason, the installation program enables shadow passwords by default.

The following lists the advantages of shadow passwords over the traditional way of storing passwords on UNIX-based systems:

- Improves system security by moving encrypted password hashes from the world-readable **/etc/passwd** file to **/etc/shadow**, which is readable only by the root user.
- Stores information about password aging.
- Allows the use of the **/etc/login.defs** file to enforce security policies.

Most utilities provided by the **shadow-utils** package work properly whether or not shadow passwords are enabled. However, since password aging information is stored exclusively in the **/etc/shadow** file, any commands which create or modify password aging information do not work.

The following is a list of commands which do not work without first enabling shadow passwords:

```
# chage
# gpasswd
# /usr/sbin/usermod -e or -f options
# /usr/sbin/useradd -e or -f options
```

8.6. Most relevant commands & files

User and Group Administrative Applications

chage — A command to modify password aging policies and account expiration.
gpasswd — A command to administer the /etc/group file.
groupadd — A command to add groups.
grpck — A command to verify the /etc/group file.
groupdel — A command to remove groups.
groupmod — A command to modify group membership.
pwck — A command to verify the /etc/passwd and /etc/shadow files.
pwconv — A tool to convert standard passwords to shadow passwords.
pwunconv — A tool to convert shadow passwords to standard passwords.
useradd — A command to add users.
userdel — A command to remove users.
usermod — A command to modify users.

User Management Command Line Tools

Application	Function
/usr/sbin/useradd	Adds user accounts. This tool is also used to specify primary and secondary group membership.
/usr/sbin/userdel	Deletes user accounts.
/usr/sbin/usermod	Edits account attributes including some functions related to password aging. For more fine-grained control, use the **passwd** command. **usermod** is also used to specify primary and secondary group membership.
passwd	Sets passwords. Although primarily used to change a user's password, it also controls all aspects of password aging.
/usr/sbin/chpasswd	Reads in a file consisting of username and password pairs, and updates each users' password accordingly.
chage	Changes the user's password aging policies. The **passwd** command can also be used for this purpose.
chfn	Changes the user's GECOS information.
chsh	Changes the user's default shell.

Group Management Command Line Tools

Application	Function
/usr/sbin/groupadd	Adds groups, but does not assign users to those groups. The **useradd** and **usermod** programs should then be used to assign users to a given group.
/usr/sbin/groupdel	Deletes groups.
/usr/sbin/groupmod	Modifies group names or GIDs, but does not change group membership. The **useradd** and **usermod** programs should be used to assign users to a given group.
gpasswd	Changes group membership and sets passwords to allow non-group members who know the group password to join the group. It is also used to specify group administrators.
/usr/sbin/grpck	Checks the integrity of the **/etc/group** and **/etc/gshadow** files.

Permission Management Command Line Tools

Application	Function
chgrp	Changes which group owns a given file.
chmod	Changes access permissions for a given file. It is also capable of assigning special permissions.
chown	Changes a file's ownership (and can also change group).

Configuration Files

/etc/passwd: This file is world-readable and contains a list of users, each on a separate line. On each line is a colon delimited list containing the following information:

Username — The name the user types when logging into the system.

Password — Contains the encrypted password (or an **x** if shadow passwords are being used).

User ID (***UID***) — The numerical equivalent of the username which is referenced by the system and applications when determining access privileges.

Group ID (***GID***) — The numerical equivalent of the primary group name which is referenced by the system and applications when determining access privileges.

GECOS — Named for historical reasons, the GECOS field is optional and is used to store extra information (such as the user's full name). Multiple entries can be stored here in a comma delimited list. Utilities such as **finger** access this field to provide additional user information.

Home directory — The absolute path to the user's home directory, such as **/home/juan/**.

Shell — The program automatically launched whenever a user logs in. This is usually a command interpreter (often called a shell). Under Red Hat Enterprise Linux, the default value is **/bin/bash**. If this field is left blank, **/bin/sh** is used. If it is set to a non-existent file, then the user will be unable to log into the system.

/etc/shadow: The file is readable only by the root user and contains password (and optional password aging information) for each user. As in the /etc/passwd file, each user's information is on a separate line. Each of these lines is a colon delimited list including the following information:

> ***Username*** — The name the user types when logging into the system. This allows the login application to retrieve the user's password (and related information).
>
> ***Encrypted password*** — The 13 to 24 character password. The password is encrypted using either the crypt(3) library function or the md5 hash algorithm. In this field, values other than a validly-formatted encrypted or hashed password are used to control user logins and to show the password status. For example, if the value is ! or *, the account is locked and the user is not allowed to log in. If the value is !! a password has never been set before (and the user, not having set a password, will not be able to log in).
>
> ***Date password last changed*** — The number of days since January 1, 1970 (also called the epoch) that the password was last changed. This information is used in conjunction with the password aging fields that follow.
>
> ***Number of days before password can be changed*** — The minimum number of days that must pass before the password can be changed.
>
> ***Number of days before a password change is required*** — The number of days that must pass before the password must be changed.
>
> ***Number of days warning before password change*** — The number of days before password expiration during which the user is warned of the impending expiration.
>
> ***Number of days before the account is disabled*** — The number of days after a password expires before the account will be disabled.
>
> ***Date since the account has been disabled*** — The date (stored as the number of days since the epoch) since the user account has been disabled.
>
> ***A reserved field*** — A field that is ignored in Red Hat Enterprise Linux.

/etc/group: It is world-readable and contains a list of groups, each on a separate line. Each line is a four field, colon delimited list including the following information:

> ***Group name*** — The name of the group. Used by various utility programs as a human-readable identifier for the group.
>
> ***Group password*** — If set, this allows users that are not part of the group to join the group by using the newgrp command and typing the password stored here. If a lower case x is in this field, then shadow group passwords are being used.
>
> ***Group ID*** (***GID***) — The numerical equivalent of the group name. It is used by the operating system and applications when determining access privileges.
>
> ***Member list*** — A comma delimited list of the users belonging to the group.

/etc/gshadow: Is readable only by the root user and contains an encrypted password for each group, as well as group membership and administrator information. Just as in the **/etc/group** file, each group's information is on a separate line. Each of these lines is a colon delimited list including the following information:

> ***Group name*** — The name of the group. Used by various utility programs as a human-readable identifier for the group.
>
> ***Encrypted password*** — The encrypted password for the group. If set, non-members of the group can join the group by typing the password for that group using the **newgrp** command. If the value of this field is **!**, then no user is allowed to access the group using the **newgrp** command. A value of **!!** is treated the same as a value of **!** — however, it also indicates that a

password has never been set before. If the value is null, only group members can log into the group.

Group administrators — Group members listed here (in a comma delimited list) can add or remove group members using the **gpasswd** command.

Group members — Group members listed here (in a comma delimited list) are regular, non-administrative members of the group.

8.7. User Accounts, Groups and permissions

Under Red Hat Enterprise Linux, a user can log into the system and use any applications or files they are permitted to access after a normal user account is created. Red Hat Enterprise Linux determines whether or not a user or group can access these resources based on the permissions assigned to them.

There are three different permissions for files, directories, and applications. These permissions are used to control the kinds of access allowed. Different one-character symbols are used to describe each permission in a directory listing. The following symbols are used:

r — Indicates that a given category of user can read a file.

w — Indicates that a given category of user can write to a file.

x — Indicates that a given category of user can execute the contents of a file.

A fourth symbol (**-**) indicates that no access is permitted.

Each of the three permissions are assigned to three different categories of users. The categories are:

owner — The owner of the file or application.

group — The group that owns the file or application.

everyone — All users with access to the system.

As stated earlier, it is possible to view the permissions for a file by invoking a long format listing with the command **ls -l**. For example, if the user **juan** creates an executable file named **foo**, the output of the command **ls -l foo** would appear like this:

```
-rwxrwxr-x   1   juan   juan   0 Sep 26 12:25   foo
```

The permissions for this file are listed at the start of the line, beginning with **rwx**. This first set of symbols define owner access — in this example, the owner **juan** has full access, and may read, write, and execute the file. The next set of **rwx** symbols define group access (again, with full access), while the last set of symbols define the types of access permitted for all other users. Here, all other users may read and execute the file, but may not modify it in any way.

One important point to keep in mind regarding permissions and user accounts is that every application run on Red Hat Enterprise Linux runs in the context of a specific user. Typically, this means that if user **juan** launches an application, the application runs using user **juan**'s context. However, in some cases the application may need a more privileged level of access in order to accomplish a task. Such applications include those that edit system settings or log in users. For this reason, special permissions have been created.

There are three such special permissions within Red Hat Enterprise Linux. They are:

setuid — used only for binary files (applications), this permission indicates that the file is to be executed with the permissions of the owner of the file, and not with the permissions of the user executing the file (which is the case without setuid). This is indicated by the character **s** in the place of the **x** in the owner category. If the owner of the file does not have execution permissions, a capital **S** reflects this fact.

```
# chmod 4755 share
# ls -l | grep share
drwsr-xr-x.   2 root it    4096 Apr  6 10:28 share
```

setgid — used primarily for binary files (applications), this permission indicates that the file is executed with the permissions of the group owning the file and not with the permissions of the group of the user executing the file (which is the case without setgid).

If applied to a directory, all files created within the directory are owned by the group owning the directory, and not by the group of the user creating the file. The setgid permission is indicated by the character **s** in place of the **x** in the group category. If the group owning the file or directory does not have execute permissions, a capital **S** reflects this fact.

```
# chmod 2755 share
# ls -l | grep share
drwxr-sr-x.   2 root it    4096 Apr  6 10:28 share
```

sticky bit — used primarily on directories, this bit dictates that a file created in the directory can be removed only by the user that created the file. It is indicated by the character **t** in place of the **x** in the everyone category. If the everyone category does not have execution permissions, the **T** is capitalized to reflect this fact.

```
# chmod 1755 share
# ls -l | grep share
drwxr-xr-t.   2 root it    4096 Apr  6 10:28 share
```

Under Red Hat Enterprise Linux, the sticky bit is set by default on the **/tmp/** directory for exactly this reason.

8.8. Create and configure set-GID directories for collaboration

Set-GID directories are used for group collaboration. Everything that is created in a directory with that special permission bit is automatically owned by the group.

To set that permission bit you have to add one more digit than you usually see to permission sets, or another character depending on how to usually set permissions.

The digit-based way to setup a **set-GID** directory:

We start off with a directory under **root**, named **share**. Right now it's owned by root, along with everything under it:

```
# ls -l | grep share
drwxr-xr-x.   2 root root  4096 Apr  6 10:28 share
# ls -l share/
total 0
-rw-r--r--. 1 root root 0 Apr  6 10:28 file213
```

First step in this process is changing group ownership of the directory to the group. '**it**' will be the group in this example.

```
# chgrp -R it  share/
# ls -l share/
total 0
-rw-r--r--. 1 root it 0 Apr  6 10:28 file213
```

As you can see, we recursivley change the directory and contents to the '**it**' group, but it's still owned by root. Next we would apply the set-gid permission on the directory by adding a **2** before the standard permissions set.

```
# chmod 2755 share/
# ls -l | grep share
drwxr-sr-x.   2 root it    4096 Apr  6 10:28 share
```

Now you can see the directory has a permission set that contains an '**s**' now in place of the x for group. Let's test this out. I am going to touch a file as root in this directory.

```
# touch newfile
# chmod 760 newfile
# ls -l
total 0
-rw-r--r--. 1 root it 0 Apr  6 10:28 file213
-rwxrw----. 1 root it 0 Apr 13 02:03 newfile
```

So you can see that automatically a group ownership is added to this file, which means anyone in the group '**it**' can now read and write to this file, while anyone else cannot.

The character-based way to setup a set-GID directory:

The only thing done differently with this method is the way the permissions are set. Instead of numerical/digit permissions, we use letters. First we list the permissions.

```
# ls -l | grep share
drwxr-xr-x.   2 root it    4096 Apr 13 02:03 share
```

Then apply the permissions with g+s, for group plus set-guid bit.

```
# chmod g+s share/
# ls -l | grep share
```

```
drwxr-sr-x.   2 root it    4096 Apr 13 02:03 share
```

As you can see it has the same effect, just a different approach.

9. Authentication Configuration

The **authconfig** command-line tool updates all of the configuration files and services required for system authentication, according to the settings passed to the script. Along with allowing all of the identity and authentication configuration options that can be set through the UI, the **authconfig** tool can also be used to create backup and kickstart files.

9.1. Tips for using authconfig

There are some things to remember when running **authconfig**:

- With every command, use either the **--update** or **--test** option. One of those options is required for the command to run successfully. Using **--update** writes the configuration changes**.** **--test** prints the changes to stdout but does not apply the changes to the configuration.

- Each enable option has a corresponding disable option.

9.2. Configuring LDAP User Stores

To use an LDAP identity store, use the **--enableldap**. To use LDAP as the authentication source, use **--enableldapauth** and then the requisite connection information, like the LDAP server name, base DN for the user suffix, and (optionally) whether to use TLS. The **authconfig** command also has options to enable or disable RFC 2307bis schema for user entries, which is not possible through the Authentication Configuration UI.

Be sure to use the full LDAP URL, including the protocol (**ldap** or **ldaps**) and the port number. Do not use a secure LDAP URL (**ldaps**) with the **--enableldaptls** option.

```
authconfig --enableldap --enableldapauth -ldapserver=ldap:
//ldap.example.com:389,ldap://ldap2.example.com:
389 --ldapbasedn="ou=people,dc=example,dc=com" --enableldaptls
--ldaploadcacert=https://ca.server.example.com/caCert.crt --update
```

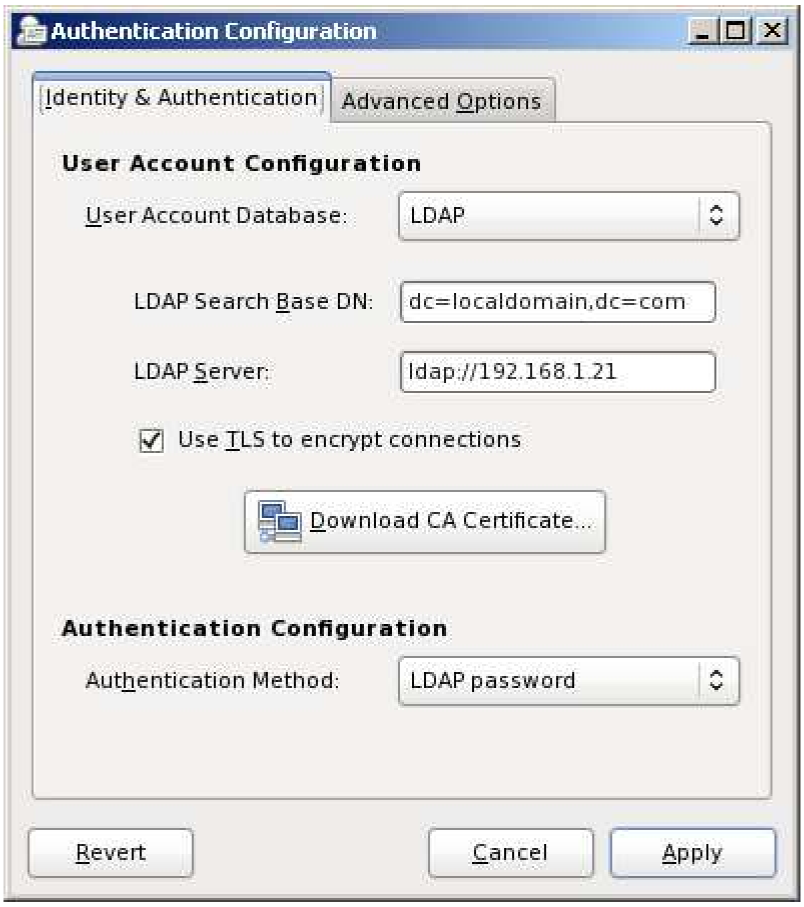
Instead of using **--ldapauth** for LDAP password authentication, it is possible to use Kerberos with the LDAP user store.

9.3. Configure a system to use an existing LDAP directory service

Install required packages:

```
# yum  -y  install openldap-clients  openldap  nss-pam-ldap
# yum  -y  system-config-authentication
# system-config-authentication
```

Check **enable ldap support**. Then configure should open up a windows that allows you to just add the ldap info, check the "Use TLS" and be done. Not much to this objective, seeming as you only have to be able to connect and authenticate.


Authentication Configuration

Identity & Authentication | Advanced Options

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: dc=localdomain,dc=com

LDAP Server: ldap://192.168.1.21

[✓] Use TLS to encrypt connections

Authentication Configuration

Authentication Method: LDAP password

9.4. Configuring NIS User Stores

To use a NIS identity store, use the **--enablenis**. This automatically uses NIS authentication, unless the Kerberos parameters are explicitly set, so it uses Kerberos authentication. The only parameters are to identify the NIS server and NIS domain; if these are not used, then the **authconfig** service scans the network for NIS servers.

authconfig --enablenis --nisdomain EXAMPLE --nisserver nis.example.com --update

9.5. Configuring Winbind User Stores

Windows domains have several different security models, and the security model used in the domain determines the authentication configuration for the local system.
For user and server security models, the Winbind configuration requires only the domain (or workgroup) name and the domain controller hostnames.

authconfig --enablewinbind --enablewinbindauth --smbsecurity user|server

--enablewinbindoffline --smbservers=ad.example.com --smbworkgroup=EXAMPLE --update

For ads and domain security models, the Winbind configuration allows additional configuration for the template shell and realm (ads only). For example:

```
authconfig --enablewinbind --enablewinbindauth --smbsecurity ads
--enablewinbindoffline --smbservers=ad.example.com --smbworkgroup=EXAMPLE
--smbrealm EXAMPLE.COM --winbindtemplateshell=/bin/sh --update
```

There are a lot of other options for configuring Windows-based authentication and the information for Windows user accounts, such as name formats, whether to require the domain name with the username, and UID ranges. These options are listed in the **authconfig** help.

9.6. Configuring Kerberos Authentication

Kerberos is an authentication method that allows users to authenticate without directly providing their password.

***** Note**: Just be aware, a Kerberos client does not work unless Network Time Protocol (NTP) clients on both systems are configured to synchronize with the same NTP server.

- Backup files under **/etc/sssd** directory, along with the **/etc/nsswitch.conf** configuration file.

What you should know is that Kerberos servers as configured on RHEL 6 don't have their own authentication databases. So for a valid client connection to a Kerberos server, you'll also need a connection to a network authentication database such as LDAP.

- Ensure all packages are installed.

To begin with we'll need to install **openldap-clients**, **authconfig** (usually installed by default).

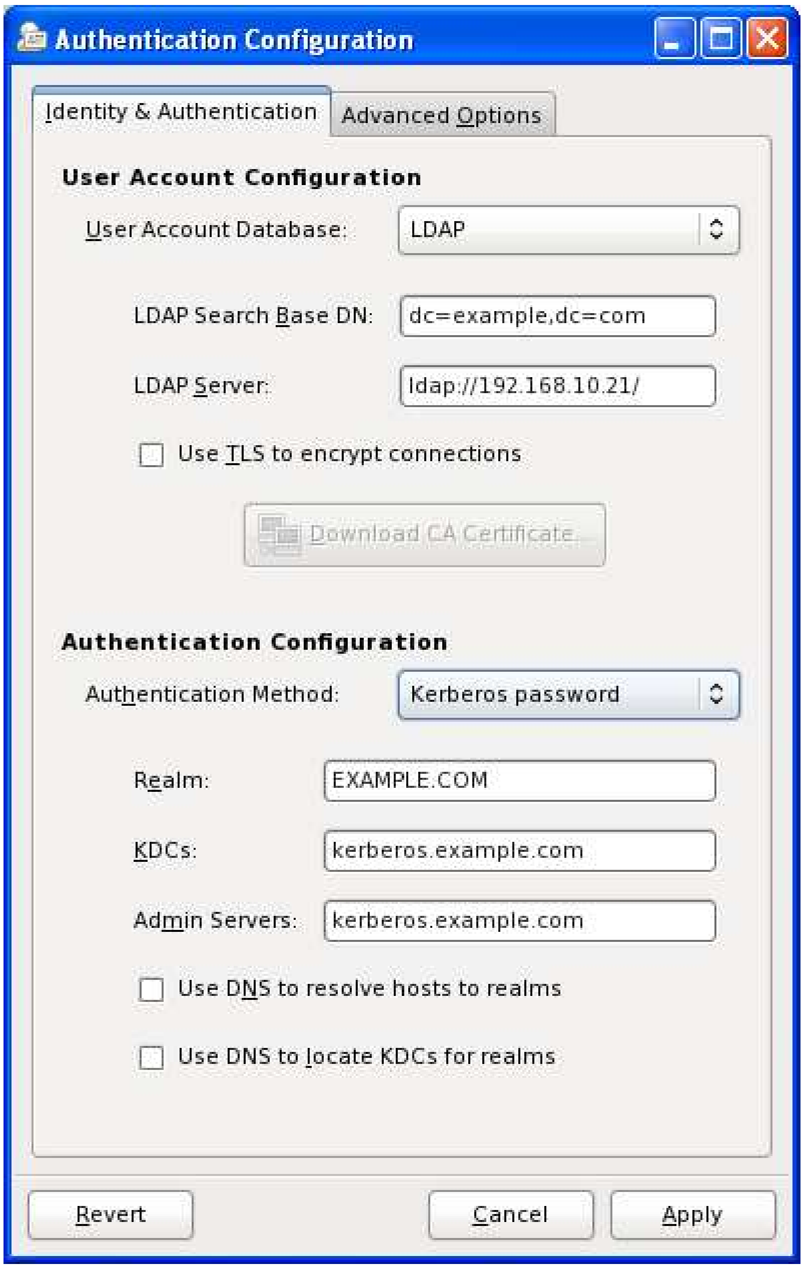
In addition, we need the kerberos and samba packages:

```
# yum install krb5-server pam_krb5 samba samba-common samba-winbind \
        samba-client samba-winbind-clients
```

- Run the Authentication Configuration tool

```
# system-config-authentication
        or
# authconfig-gtk
```

While other authentication databases are supported, the focus is on **LDAP**.


Authentication Configuration

Identity & Authentication | Advanced Options

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: dc=example,dc=com

LDAP Server: ldap://192.168.10.21/

☐ Use TLS to encrypt connections

Authentication Configuration

Authentication Method: Kerberos password

Realm: EXAMPLE.COM

KDCs: kerberos.example.com

Admin Servers: kerberos.example.com

☐ Use DNS to resolve hosts to realms

☐ Use DNS to locate KDCs for realms

The focus of this section is on the second half of the tab. For a Kerberos-based client, you'd retain Kerberos Password as the Authentication Method. The other options are
- **Realm**: By convention, the Kerberos realm is the same as the domain name for the network, in uppercase letters. It's necessary if you configure DNS support for Kerberos.
- **KDCs**: The KDC is the Kerberos Key Distribution Center. The entry here should correspond either to the Fully Qualified Domain Name (FQDN) or the IP address of the actual Kerberos server.
- **Admin Servers**: The administrative server associated with the KDC is frequently located on the same system. On the Kerberos administrative server, the kadmind daemon is running.
- **Use DNS To Resolve Hosts To Realms**: Where a trusted DNS server exists for the local network, you can allow the local system to use a DNS server to find the realm. If this option is activated, the Realm text box will be blanked out.
- **Use DNS To Locate KDCs For Realms**: Where a trusted DNS server exists for the local network, you can allow the local system to use a DNS server to find the KDC and administrative server. If this option is activated, the KDCs and Admin Servers text boxes will be blanked out.

Click Apply. After a few moments, the Authentication Configuration window will close and changes will be made to the aforementioned configuration files. In addition, the sssd service will be started.

9.7. Configuring Local Authentication Settings

The Authentication Configuration Tool can also control some user settings that relate to security, such as creating home directories, setting password hash algorithms, and authorization. These settings are done independently of identity/user store settings.

For example, to create user home directories:

```
authconfig --enablemkhomedir --update
```

To set or change the hash algorithm used to encrypt user passwords:

```
authconfig --passalgo=sha512 --update
```

9.8. Configuring Fingerprint Authentication

There is one option to enable support for fingerprint readers. This option can be used alone or in conjunction with other **authconfig** settings, like LDAP user stores.

```
authconfig --enablefingerprint --update
```

9.9. Configuring Smart Card Authentication

All that is required to use smart cards with a system is to set the **--enablesmartcard** option:

```
authconfig --enablesmartcard --update
```

There are other configuration options for smart cards, such as changing the default smart card module, setting the behavior of the system when the smart card is removed, and requiring smart cards for login. For example, this tells the system to lock out a user immediately if the smart card is removed (a setting of 1 ignores it if the smart card is removed):

```
authconfig --enablesmartcard --smartcardaction=0 --update
```

*****Warning**: Do not use the **--enablerequiresmartcard** option until you have successfully authenticated to the system using a smart card. Otherwise, users may be unable to log into the system.

9.10. Managing Kickstart and Configuration Files

The **--update** option updates all of the configuration files with the configuration changes. There are a couple of alternative options with slightly different behavior:

--kickstart: writes the updated configuration to a kickstart file.
--test: prints the full configuration, with changes, to stdout but does not edit any configuration files.

Additionally, **authconfig** can be used to back up and restore previous configurations. All archives are saved to a unique subdirectory in the **/var/lib/authconfig/** directory. For example, the **--savebackup** option gives the backup directory as **2011-07-01**:

```
authconfig --savebackup=2011-07-01
```

This backs up all of the authentication configuration files beneath the **/var/lib/authconfig/backup-2011-07-01** directory.

Any of the backups can be used to restore the configuration using the **--restorebackup** option. **authconfig** automatically makes a backup of the configuration before it applies any changes (with the **--update** option). The configuration can be restored from that automatic backup using the **--restorelastbackup** option.

9.11. Using Custom Home Directories

If LDAP users have home directories that are not in **/home** and the system is configured to create home directories the first time users log in, then these directories are created with the wrong permissions.

1.- Apply the correct SELinux context and permissions from the **/home** directory to the home directory that is created on the local system. For example:

```
# semanage fcontext -a -e /home /home/locale
```

2.- Install the ***oddjob-mkhomedir*** package on the system.
This package provides the **pam_oddjob_mkhomedir.so** library, which the Authentication Configuration Tool uses to create home directories. The **pam_oddjob_mkhomedir.so** library, unlike the default **pam_mkhomedir.so** library, can create SELinux labels.

The Authentication Configuration Tool automatically uses the **pam_oddjob_mkhomedir.so** library if it is available. Otherwise, it will default to using **pam_mkhomedir.so**

3.- Make sure the **oddjobd** service is running.

4.- Re-run the Authentication Configuration Tool and enable home directories, as in Section 10.1.3, "Configuring Alternative Authentication Features".

If home directories were created before the home directory configuration was changed, then correct the permissions and SELinux contexts. For example:

```
# semanage fcontext -a -e /home /home/locale
# restorecon -R -v /home/locale
```

10. Building a chroot jail environment

A chroot environment is simply a directory inside which we can find a file system hierarchy exactly like the original operating system. You can then use the UNIX **chroot** command to open a shell in that directory so that command running under that shell sees only the chroot environment and can't mess up your system. This is very useful for many different reasons, for example if you want to build some software packages and you don't want their build dependencies to pollute your real system.

Building a **chroot** environment is not difficult at all using the right tools.

To start, log into the system and create a directory where you want to build your chroot jail:

```
# mkdir  -p  /var/tmp/chroot
```

Next step is to initialize the RPM database so that we can install all of the software we need in the chroot jail: we will need to create the directory for the database because RPM expects it to exist, and then use the RPM "**rebuilddb**" command:

```
# mkdir  -p  /var/tmp/chroot/var/lib/rpm
# rpm  --rebuilddb  --root=/var/tmp/chroot
```

In order for YUM to manage to install software into our chroot system it needs to know which kernel version to install – for this to work it needs the kernel package to be installed in the root. Doing this properly will take a lot of time and require us to manually download and install many packages – and that is boring. What we will do instead is just to download the kernel package:

```
# yumdownloader kernel-smp-2.6.9-89.35.1.EL.x86_64
```

And just install it forcefully:

```
# rpm -i --root=/var/tmp/chroot --nodeps kernel-smp-2.6.9-89.35.1.EL.x86_64
```

Finally we can call on YUM to install the rest of our system:

```
# yum --installroot=/var/tmp/chroot install -y rpm-build yum
```

The last step will take a bit of time to complete as YUM has to download all the software that goes into your chroot jail environment, but when it's ready you can run **chroot /var/tmp/chroot** to see how it looks inside.

Optional

It probably feels pretty bare inside, with a default bash prompt and no aliases. To make the chroot jail a bit more comfortable you can populate the chrooted root directory with the skeleton account files so it feels more like a proper system – before you **chroot**, copy the files from **/etc/skel** to the **chroot /root** home die:

```
# cp /var/tmp/chroot/etc/skel/.??* /var/tmp/chroot/root
```

Then run the **chroot** command and tell it to start Bash as a login shell:

```
# chroot /var/tmp/chroot /bin/bash -l
```

Now you should have a nice bash prompt just like in a real system.

The special file systems

If you try to run some stuff inside the chroot, for example **yum**, you'll see that some things are not really functional because they want access to **/proc** or **/dev**. In order to work with these you will want to mount the real **/proc** and **/dev** file systems into the **chroot**, like so (but get out of the chroot first):

```
# mount --bind /proc /var/tmp/chroot/proc
# mount --bind /dev /var/tmp/chroot/dev
```

This will let processes from inside the chroot (especially if you're running in the chroot as the root user) to see and tap into various parts of the real system even though they do not have access to the actual files – this is a security issue and if you plan to run software inside the chroot that you do not trust, then I suggest not doing a simple chroot, and instead using a virtual machine or something like User-Mode Linux.

Of course binding the **/proc** and **/dev** file systems is optional and if you don't run any software inside the chroot that needs these, you can skip that part.

Network

Network access should work fine inside the chroot, but we didn't setup name resolving – fortunately this is rather easy: just copy the **nameserver** file from your real system into the chroot:

```
# cp /etc/resolv.conf /var/tmp/chroot/etc/resolv.conf
```

And now you can use YUM to install additional software from inside the chroot jail. Of course you can always exit the jail and install from outside using the **yum --installroot** switch.

11. Accessing remote systems via 'ssh' and VNC

11.1. Configure key-based authentication

This, for example, allows us to connect to multiple systems via scripts without have to re-authenticate every time. A decent walkthrough is available at http://linuxproblem.org/art_9.html

Scenario: You are user A on host A, and you want to log onto host B as user B

As user **a** on host **A**, execute

```
# ssh-keygen -t rsa
```

Echo out the contents of **~/.ssh/id_rsa.pub** (save to clipboard or copy via ssh to host **B**)

As user **b** on host **B**, make the **.ssh** directory if it doesn't already exist

```
# mkdir ~/.ssh
# chmod 700 ~/.ssh
```

Edit the file **~/.ssh/authorized_keys** and enter the contents from **id_rsa.pub** on system **A**

```
# vi ~/.ssh/authorized_keys
# chown 600 ~/.ssh/authorized_keys
```

Assuming all went well, user **a** on host **A** should be able to run **ssh b@B** and be automatically logged in.

*** **Note**: After having configured SSH (authorized keys, etc), it may be necessary to restore permissions on root's .sash directory in order to allow root to connect without providing a password:

```
# restorecon -R -v /root/.ssh
```

11.2. Configure additional options described in documentation

N/A

11.3. SSH quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- **Packages needed to provide the service**

```
# yum install openssh-server
```

- **Configure SELinux to support the service**

```
# getsebool -a | grep ssh
```

*** **Note**: After having configured SSH (authorized keys, etc), it may be necessary to restore permissions on root's .ssh directory in order to allow root to connect without providing a password:

```
# restorecon -R -v /root/.ssh
```

- **Configure the service to start when the system is booted**

```
# chkconfig ssh on
```

- **Configure the service for basic operation**

  - Install service
  - Configure the service to start when the system is booted
  - Configure SELinux support

- Update **/etc/sysconfig/iptables**:
  Open **tcp** port **22**

- **Configure host-based and user-based security for the service**

  - Host
    Use **iptables**

  - User
    /etc/ssh/sshd_config:
    AllowUsers user@host

    OR disable shell access for a user if needed

11.4. Connecting via 'ssh'

Basic ssh access is simple:

```
# ssh  user@host
```

ssh to a custom port:

```
# ssh -p port_number user@host
```

ssh bringing X

```
# ssh -X user@host
```

ssh as another user

```
# ssh -l user host
```

Display debugging messages as it connects. Useful if you have having some issues connecting to a certain machine.

```
# ssh -v user@host
```

Those are the main options for ssh, as always "man ssh" to see all the other magic.

11.5. Configure Virtual Network Computing (VNC) to work with RHEL

Install the prerequisite required packages to enable VNC on Red Hat Enterprise Linux (RHEL6) connecting to:
tigervnc-server
tigervnc-server-module
tigervnc

Edit the **/etc/sysconfig/vncservers** file to include the users will be running VNC servers. Add a line to that file as follows:

```
VNCSERVERS="N:user"
```

Where N is the number of the display VNC server will be running and user is the username by which the server will be run as. Multiple displays and users can be specified by placing a space between them, as follows:

```
VNCSERVERS="N:user1 Y:user2"
```

Note that if the X Window System is used, display 0 cannot be used for VNC as it is already being used by X.

For each user specified, a VNC password needs to be set. VNC passwords are completely separate from the normal system password for that account. A user can set their VNC password by executing the vncpasswd command as shown below

```
# vncpasswd
```

'root' may run "**vncpasswd <username>**" command to set password for username, but this may lead to mistakes. The user should run the command **vncpasswd** by himself, which will create a directory ~username/.vnc owned by the username.

By default, VNC starts up only a simple window manager and a terminal window. For the full Red Hat environment, create ~username/.vnc/xstartup and include the following lines:

```
#!/bin/bash
unset SESSION_MANAGER
exec /etc/X11/xinit/xinitrc
```

Lastly, ensure that the resulting file has the execute bit set:

```
# chmod 755 ~username/.vnc/xstartup
```

In order to start the vncserver service immediately, run the following command:

```
# service  vncserver start
```

Now that VNC is running, **vncviewer** command can be used to connect from a remote Linux machine to the proper VNC-based X session. If the remote system is running Windows, a Windows-based VNC viewer program can be freely downloaded from the website http://www.realvnc.com or http://www.tigervnc.org

```
# vncviewer  <vncserver-address:Z>
```

replace Z with the number of the VNC-based X session where the application is running.

***** Note**: These VNC-based X sessions must be left running when users are finished with them. They can do this by simply closing the vncviewer program instead of logging out. This maintains the state of the desktop so that when they reconnect, all their programs will be in the same state in which they were left.

- If a user accidentally logs out from within a VNC-based X session they should follow these steps to get X working again:

Log in to the server as the user for whom VNC needs to be restarted. Issue following commands:

```
# vncserver -kill :Z

# vncserver :Z
```

11.6. /etc/sysconfig/vncservers

The **/etc/sysconfig/vncservers** file configures the way the Virtual Network Computing (VNC) server starts up. By default, it contains the following options:

VNCSERVERS=value

A list of space separated ***display:username*** pairs. For example: **VNCSERVERS="*2:myusername*"**

VNCSERVERARGS[*display*]=*value*

Additional arguments to be passed to the VNC server running on the specified display. For example:

```
VNCSERVERARGS[2]="-geometry 800x600 -nolisten tcp -localhost"
```

11.7. How to tunnel VNC connections over 'ssh'

To tunnel VNC connections over SSH, you must have a real system account on the machine running the VNC server. You must also know what display the VNC server is running on (which can be found in the /etc/sysconfig/vncservers file). Once you have these pieces of information, connect to the VNC server with the following SSH command:

```
# ssh -L 590X:127.0.0.1:590X -N -f -l username servername
```

Replace the X's with the display number the VNC server is running on. For example, if the VNC server was running on display 1, you would specify 5901.
In your VNC client, instead of connecting directly to the VNC server, connect to "127.0.0.1:590X ", again replacing X with the appropriate display number. This will tunnel the connection over SSH, providing greatly enhanced security.

12. Login or switch users to 'root' account

12.1. Why should one use "su" instead of login as 'root'

Several reasons exist as why a user would want to perform system tasks as "**su**" opposed to being logged in as '**root**'.

Several programs need to run as current user in order to perform specific tasks related to that user. If you are logged in as 'root' and run an exploited application your system is comprised.

Constantly using "**su**" command will more than likely keep you aware that you are in 'root' and will not blindly issue a command that will leave your system unusable, for example, removing entire filesystems.

If you work in a high traffic area and leave your workstation unlocked others will only have access to that particular user instead of an entire system.

Another option to protecting system critical files is to allow access to the "**sudo**" command. This grants particular users access to certain commands without granting entire root access to the system.

In conclusion it is a safe assumption that denying access to 'root' account entirely and only allowing certain commands to be issued by certain non-root accounts ensures system stability and the less likelihood of the system being compromised.

12.2. Allow a user to run commands as root ('sudo')

/etc/sudoers file can be edited with '**visudo**' command, tool that allows modifying and verifying syntax and errors in the file. If one tries to modify directly /etc/sudoers file, this has read-only permissions.

Basic syntax of a list:
```
XXXX_Alias   LIST_NAME = element1, element2, element3
```

Basic syntax of a rule:
```
[user | %grupo | LIST_NAME]   [host]   =        (userid to be used)       command(s)
```

Aliases and rules may also be defined. Aliases allow defining a list of commands, a list of users, a list of hosts and running commands as a different user.

Defining a command list alias:

```
Cmnd_Alias   COMMANDLIST          =        /sbin/service httpd restart, \
/usr/bin/vim /etc/httpd/conf.d/variables.conf, \
/usr/bin/vim /etc/php.ini
```

Allowing a user to run a command list:

```
username        ALL      =        COMMANDLIST
```

This allows 'username' to run all the commands in 'COMMANDLIST' from any host.

Defining a user list alias:

```
User_Alias     USERLIST       =        user1, user2, user3
```

Allowing a user list to run a command:

```
USERLIST        ALL      =        /usr/bin/vim
```

This allow users in 'USERLIST' to run /usr/bin/vim from any host

Defining a host list alias:

```
Host_Alias     HOSTLIST     =     192.168.0.1X, 192.168.0.2X, 192.168.0.3X
```

Allowing a user list to run a list of command from a host list:

```
USERLIST     HOSTLIST     =     COMMANDLIST
```

This allows users in 'USERLIST' to run commands in 'COMMANDLIST' only of they are connected from hosts in 'HOSTLIST'

12.3. Restrict the use of 'su' command

To restrict the use of **su** command, first of all it is recommended to disable root SSH login before proceeding.

You can SSH using a regular user account, then use the **su** command to obtain root access. This is true for any user that enters the **su** command and enters the root password. Root access means absolute access, thus, it is recommended to limit the usernames that can use the **su** command and get root access.

There is a group called '**wheel**' on the Linux system that we can be used for this. We can add usernames that we want to have **su** access to become a member of the wheel group and then restrict **su** so that only the members of the wheel group can use the **su** command.

To do this, follow these steps. Commands should be executed by the root user:

```
# usermod  -G  wheel  <username>
```

Edit the PAM configuration file for **su**, **/etc/pam.d/su**, and uncomment this line:

```
# auth required /lib/security/pam_wheel.so use_uid
```

Doing this will permit only members of the administrative group wheel to use the su command.

12.4. Using 'su' command

The **su** command enables you to take the identification (and the privileges) of another user on the system without logging out.

One common use of **su** command is to temporarily be the 'root' user to accomplish some task that requires root's privileges. Other common uses are to take the identification of a 'system' user.

The **su** command accepts several arguments that modify its behaviour:

The option '**-**' or '**-l**' runs a 'login shell'. This means that when the **su** command runs and the effective user ID is substituted, all the usual login scripts are processed for that user. This is very useful, as you then get the user's environment as well as the effective user ID.

Also notice the '**-c**' option. This allows the user to pass the shell a command that will be immediately run. Once the command is completed, the user is returned to the normal command line:

```
# su  -  -c  "<command + parametres>"
Password:
```

***** Note**: "**su - root**" = "**su -**"

12.5. Differences between "su -" and "su"

The main difference between "**su -**" and **su** is that the former makes the shell a login shell. This is very important especially if the user is going to **su** from a regular user account to a root (superuser) account. Normal users do not usually have /sbin/ and /usr/sbin/ in their search path. Therefore if a normal user wants to execute the command ifconfig, for example, after doing su, he usually gets the error message.

With "**su -**", on the other hand, root's .bashrc and .bash_profile and other special environment settings get sourced and this puts the sbin directories into the search path.

13. Access Control Lists (ACL)

Files and directories have permission sets for the owner of the file, the group associated with the file, and all other users for the system. However, these permission sets have limitations. For example, different permissions cannot be configured for different users. Thus, *Access Control Lists* (ACLs) were implemented.

The Red Hat Enterprise Linux kernel provides ACL support for the ext3 file system and NFS-exported file systems. ACLs are also recognized on ext3 file systems accessed via Samba.

Along with support in the kernel, the **acl** package is required to implement ACLs. It contains the utilities used to add, modify, remove, and retrieve ACL information.

The **cp** and **mv** commands copy or move any ACLs associated with files and directories.

13.1. Mounting File Systems

Before using ACLs for a file or directory, the partition for the file or directory must be mounted with ACL support. If it is a local ext3 file system, it can be mounted with the following command:

```
# mount -t ext3 -o acl device-name mountpoint
```

For example:

```
# mount -t ext3 -o acl /dev/VolGroup00/LogVol02 /work
```

Alternatively, if the partition is listed in the /etc/fstab file, the entry for the partition can include the acl option:

```
LABEL=/work    /work    ext3   acl     1 2
```

If an ext3 file system is accessed via Samba and ACLs have been enabled for it, the ACLs are recognized because Samba has been compiled with the **--with-acl-support** option. No special flags are required when accessing or mounting a Samba share.

We can use **tune2fs** to set ACLs as a default mount option:

```
# tune2fs -o acl /dev/volume/home
```

13.1.1. NFS

By default, if the file system being exported by an NFS server supports ACLs and the NFS client can read ACLs, ACLs are utilized by the client system.

To disable ACLs on NFS shares when configuring the server, include the **no_acl** option in the **/etc/exports** file. To disable ACLs on an NFS share when mounting it on a client, mount it with the **no_acl** option via the command line or the **/etc/fstab** file.

13.2. Setting Access ACLs

There are two types of ACLs: *access ACLs* and *default ACLs*. An access ACL is the access control list for a specific file or directory. A default ACL can only be associated with a directory; if a file within the directory does not have an access ACL, it uses the rules of the default ACL for the directory. Default ACLs are optional.
ACLs can be configured:

Per user
Per group
Via the effective rights mask
For users not in the user group for the file

The **setfacl** utility sets ACLs for files and directories. Use the **-m** option to add or modify the ACL of a file or directory:

```
# setfacl -m rules files
```

Rules (***rules***) must be specified in the following formats. Multiple rules can be specified in the same command if they are separated by commas.

u:*uid:perms* - Sets the access ACL for a user. The user name or UID may be specified. The user may be any valid user on the system.

g:*gid:perms* - Sets the access ACL for a group. The group name or GID may be specified. The group may be any valid group on the system.

m:*perms* - Sets the effective rights mask (nomal permissions). The mask is the union of all permissions of the owning group and all of the user and group entries.

o:*perms* - Sets the access ACL for users other than the ones in the group for the file.

Permissions (***perms***) must be a combination of the characters **r**, **w**, and **x** for read, write, and execute.
If a file or directory already has an ACL, and the **setfacl** command is used, the additional rules are added to the existing ACL or the existing rule is modified.

For example, to give read and write permissions to user ***andrius***:

```
# setfacl -m u:andrius:rw /project/somefile
```

To remove all the permissions for a user, group, or others, use the **-x** option and do not specify any permission:

```
# setfacl -x user files
```

For example, to remove all permissions from the user with UID 500:

```
# setfacl -x u:500 /project/somefile
```

13.3. Setting Default ACLs

To set a default ACL, add d: before the rule and specify a directory instead of a file name.
For example, to set the default ACL for the /share/ directory to read and execute for users not in the user group (an access ACL for an individual file can override it):

```
# setfacl -m d:o:rx /share
```

13.4. Retrieving ACLs

To determine the existing ACLs for a file or directory, use the **getfacl** command. In the example below, the **getfacl** is used to determine the existing ACLs for a file.

```
# getfacl home/john/picture.png
```

The above command returns the following output:

```
# file: home/john/picture.png
# owner: john
# group: john
user::rw-
group::r--
```

```
other::r--
```

If a directory with a default ACL is specified, the default ACL is also displayed as illustrated below. For example, **getfacl home/sales/** will display similar output:

```
# file: home/sales/
# owner: john
# group: john
user::rw-
user:barryg:r--
group::r--
mask::r--
other::r--
default:user::rwx
default:user:john:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

13.5. Archiving File Systems with ACLs

By default, the **dump** command now preserves ACLs during a backup operation. When archiving a file or file system with **tar**, use the **--acls** option to preserve ACLs. Similarly, when using **cp** to copy files with ACLs, include the **--preserve=mode** option to ensure that ACLs are copied across too. In addition, the **-a** option (equivalent to **-dR --preserve=all**) of **cp** also preserves ACLs during a backup along with other information such as timestamps, SELinux contexts, and the like.

The **star** utility is similar to the **tar** utility in that it can be used to generate archives of files; however, some of its options are different. The **star** package is required to use this utility.

Option	Description
-c	Creates an archive file.
-n	Do not extract the files; use in conjunction with **-x** to show what extracting the files does.
-r	Replaces files in the archive. The files are written to the end of the archive file, replacing any files with the same path and file name.
-t	Displays the contents of the archive file.
-u	Updates the archive file. The files are written to the end of the archive if they do not exist in the archive, or if the files are newer than the files of the same name in the archive. This option only works if the archive is a file or an unblocked tape that may backspace.
-x	Extracts the files from the archive. If used with **-U** and a file in the archive is older than the corresponding file on the file system, the file is not extracted.
-help	Displays the most important options.
-xhelp	Displays the least important options.
-/	Do not strip leading slashes from file names when extracting the files from an archive. By default, they are stripped when files are extracted.
-acl	When creating or extracting, archives or restores any ACLs associated with the files and directories.

13.6. Compatibility with older systems

If an ACL has been set on any file on a given file system, that file system has the **ext_attr** attribute. This attribute can be seen using the following command:

```
# tune2fs -l filesystem-device
```

A file system that has acquired the **ext_attr** attribute can be mounted with older kernels, but those kernels do not enforce any ACLs which have been set.

Versions of the **e2fsck** utility included in version 1.22 and higher of the **e2fsprogs** package (including the versions in Red Hat Enterprise Linux 2.1 and 4) can check a file system with the **ext_attr** attribute. Older versions refuse to check it.

14. Locating and analyzing system logs files

Most of the logs you will deal with are going to be located in **/var/log/**

Logs are written in a way that makes them easy to parse through with text processing tools like **cat**, **grep**, and **awk**.

One example would be searching for "Failed" logins in **/var/log/secure**:

```
# cat /var/log/secure | grep Failed | less
Apr  1 16:17:06 mytest sshd[19632]: Failed password for root from 84.204.56.234 port 39045 ssh2
Apr  1 16:17:09 mytest sshd[19634]: Failed password for root from 84.204.56.234 port 39351 ssh2
Apr  1 16:17:13 mytest sshd[19636]: Failed password for root from 84.204.56.234 port 39660 ssh2
Apr  1 22:13:40 mytest sshd[19741]: Failed password for bin from 200.76.17.194 port 53407 ssh2
Apr  1 22:13:43 mytest sshd[19744]: Failed password for bin from 200.76.17.194 port 40100 ssh2
Apr  1 22:13:46 mytest sshd[19747]: Failed password for bin from 200.76.17.194 port 51759 ssh2
Apr  1 22:13:49 mytest sshd[19749]: Failed password for bin from 200.76.17.194 port 45675 ssh2
Apr  1 22:13:52 mytest sshd[19751]: Failed password for bin from 200.76.17.194 port 54379 ssh2
Apr  1 22:13:55 mytest sshd[19753]: Failed password for bin from 200.76.17.194 port 12218 ssh2
Apr  2 06:05:01 mytest sshd[20102]: Failed password for root from 117.211.83.59 port 34815 ssh2
```

Lets say we just want to process the text, and get a count of how many logins were failed in this file. We can pipe the output into wc -l, which counts lines.

```
# cat /var/log/secure | grep Failed | wc -l
90
```

View and manage log files 01
View and manage log files 02
View and manage log files 03

14.1. Configure a system to log to a remote system

In prior releases of redhat, remote logging was configured via **syslogd**. In RHEL6, this is replaced with **rsyslog**.

http://www.rsyslog.com/sending-messages-to-a-remote-syslog-server has a great walkthrough on setting this up

Edit **/etc/rsyslog.conf** and enter the below line (using the appropriate IP or DNS name)

```
*.*   @@192.168.10.1:514
```

Restart the **rsyslog** daemon:

```
# service rsyslog restart
```

Test the configuration by running **logger -p warn foo**. This will log a message in the local **/var/log/messages** and should log a similar message at the same location on the remote server

14.2. Configure a system to accept logging from a remote system

The first step is to setup a remote server to receive the logging messages. http://www.rsyslog.com/receiving-messages-from-a-remote-system has a great walkthrough on setting this up.

Edit **/etc/rsyslog.conf** and clear the # before the lines allowing **syslog** reception

```
$ModLoad imudp.so
$UDPServerRun 514
$ModLoad imtcp.so
$InputTCPServerRUN 514
```

Restart the rsyslog daemon

```
# service rsyslog restart
```

Open the firewall to allow syslog connections:

```
# iptables -I INPUT -p tcp --dport 514 -j ACCEPT
# iptables -I INPUT -p udp --dport 514 -j ACCEPT
# iptables-save > /etc/sysconfig/iptables
```

15. Managing system resources

15.1. Identify CPU/memory intensive processes, adjust process priority with renice, and kill processes

A few commands to help you identify processes are **ps** and **top**. These are commands that you will actually use extensively to monitor systems in the workplace.

ps - report a snapshot of the current processes.

ps helps you to see what processes are being run, what files and commands they are being run with, who they are being run by, as well and their process ids. All the above items are crucial when troubleshooting issues on a Red Hat Enterprise Linux 6 system.

EXAMPLES

To see every process on the system using standard syntax:

```
# ps -e
# ps -ef
# ps -eF
# ps -ely
```

To see every process on the system using BSD syntax:

```
# ps ax
# ps axu
```

To print a process tree:

```
# ps -ejH
# ps axjf
```

To get info about threads:

```
# ps -eLf
# ps axms
```

To get security info:

```
# ps -eo euser,ruser,suser,fuser,f,comm,label
# ps axZ
# ps -eM
```

To see every process running as root (real & effective ID) in user format:

```
# ps -U root -u root u
```

top - display Linux tasks

At its most basic usage you can just type:

```
# top
```

renice — alter priority of running processes

As stated in the description, **renice** is a linux utility to change the priority of a process. This could obviously come in handy while trying to keep a process at bay.

```
# renice +1 987 -u daemon root -p 32
```

This would change the priority of process ID's 987 and 32, and all processes owned by users daemon and root.

kill - terminate a process

Like it states in the name, this kills processes.

The most common implementation of this is:

```
# kill 2342
```

If that doesn't kill the process you would use the -9 switch, which will take out most any process.

```
# kill -9 2342
```

***** Note**: The -9 command should be used with caution. Make sure you are killing the right pid, otherwise terrible things may transpire.

15.2. Monitoring CPU utilization

Unlike bandwidth, monitoring CPU utilization is much more straightforward. From a single percentage of CPU utilization in GNOME System Monitor, to the more in-depth statistics reported by sar, it is possible to accurately determine how much CPU power is being consumed and by what.

The most common tool is "**top**".

```
9:44pm up 2 days, 2 min, 1 user, load average: 0.14, 0.12, 0.09 90
processes: 82 sleeping, 1 running, 7 zombie, 0 stopped
CPU0 states: 0.4% user, 1.1% system, 0.0% nice, 97.4% idle
CPU1 states: 0.5% user, 1.3% system, 0.0% nice, 97.1% idle
Mem: 1288720K av, 1056260K used, 232460K free, 0K shrd, 145644K buff
Swap: 522104K av, 0K used, 522104K free 469764K cached
PID     USER    PRI     NI      SIZE    RSS     SHARE   STAT    %CPU    %MEM    TIME    COMMAND
30997   ed      16      0       1100    1100    840     R       1.7     0.0     0:00    top
1120    root    5       -10     249M    174M    71508   S       0.9     13.8    254:59  X
1260    ed      15      0       54408   53M     6864    S       0.7     4.2     12:09   terminal
888     root    15      0       2428    2428    1796    S       0.1     0.1     0:06    sendmail
1264    ed      15      0       16336   15M     9480    S       0.1     1.2     1:58    rhn-applet
1       root    15      0       476     476     424     S       0.0     0.0     0:05    init
2       root    0K      0       0       0       0       SW      0.0     0.0     0:00    mgr_CPU0
3       root    0K      0       0       0       0       SW      0.0     0.0     0:00    mgr_CPU1
4       root    15      0       0       0       0       SW      0.0     0.0     0:01    keventd
[....]
```

- The **load average** is a number corresponding to the average number of runnable processes on the system. The load average is often listed as three sets of numbers (as top does), which represent the load average for the past 1, 5, and 15 minutes.

- The next line, although not strictly related to CPU utilization, has an indirect relationship, in that it shows the **number of runnable processes**. The number of runnable processes is a good indicator of how CPU-bound a system might be.

- Next are the lines displaying the **current utilization for each of the CPUs** in the system. The utilization statistics show whether the CPU cycles were expended for user-level or system-level processing; also included is a statistic showing how much CPU time was expended by processes with altered scheduling priorities. Finally, there is an idle time statistic.

- Moving down into the process-related section of the display, we find that the **process using** the most **CPU** power is top itself; in other words, the one runnable process on this otherwise-idle system was top taking a "picture" of itself.

***** Note**: It is important to remember that the very act of running a system monitor affects the resource utilization statistics you receive. All software-based monitors do this to some extent.

To gain more detailed knowledge regarding CPU utilization, we must change tools. If we examine output from **vmstat**, we obtain a slightly different understanding of our example system:

```
procs   -----------memory----------  ---swap-- ----io--- --system--  -------cpu-------
r   b   swpd     free   buff   cache   si    so    bi    bo   in    cs     us  sy  id  wa  st
1   0   0        233276 146636 469808  0     0     7     7    14    27     10  3   87  0   0
0   0   0        233276 146636 469808  0     0     0     0    523   138    3   0   96  0   0
0   0   0        233276 146636 469808  0     0     0     0    557   385    2   1   97  0   0
0   0   0        233276 146636 469808  0     0     0     0    544   343    2   0   97  0   0
0   0   0        233276 146636 469808  0     0     0     0    517   89     2   0   98  0   0
0   0   0        233276 146636 469808  0     0     0     32   518   102    2   0   98  0   0
0   0   0        233276 146636 469808  0     0     0     0    516   91     2   1   98  0   0
```

```
0   0   0      233276 146636 469808  0    0    0    0    516  72      2   0   98  0   0
0   0   0      233276 146636 469808  0    0    0    0    516  88      2   0   97  0   0
0   0   0      233276 146636 469808  0    0    0    0    516  81      2   0   97  0   0
```

Here we have used the command **vmstat 1 10** to sample the system every second for ten times. At first, the CPU-related statistics (the **us**, **sy**, and **id** fields) seem similar to what **top** displayed, and maybe even appear a bit less detailed. However, unlike **top**, we can also gain a bit of insight into how the CPU is being used.
If we examine at the **system** fields, we notice that the CPU is handling about 500 interrupts per second on average and is switching between processes anywhere from 80 to nearly 400 times a second. If you think this seems like a lot of activity, think again, because the user-level processing (the **us** field) is only averaging 2%, while system-level processing (the **sy** field) is usually under 1%. Again, this is an idle system.

Reviewing the tools Sysstat offers, we find that **iostat** and **mpstat** provide little additional information over what we have already experienced with top and vmstat. However, **sar** produces a number of reports that can come in handy when monitoring CPU utilization.

The first report is obtained by the command **sar -q**, which displays the run queue length, total number of processes, and the load averages for the past one and five minutes. Here is a sample:

```
12:00:01 AM      runq-sz plist-sz  ldavg-1 ldavg-5
12:10:00 AM      3          122         0.07        0.28
12:20:01 AM      5          123         0.00        0.03
[...]
09:50:00 AM      5          124         0.67        0.65
Average:         4          123         0.26        0.26
```

In this example, the system is always busy (given that more than one process is runnable at any given time), but is not overly loaded (due to the fact that this particular system has more than one processor).

The next CPU-related sar report is produced by the command **sar -u**:

```
12:00:01 AM      CPU   %user      %nice      %system    %idle
12:10:00 AM      all   3.69       20.10      1.06       75.15
12:20:01 AM      all   1.73       0.22       0.80       97.25
[...]
10:00:00 AM      all   35.17      0.83       1.06       62.93
Average:         all   7.47       4.85       3.87       83.81
```

The statistics contained in this report are no different from those produced by many of the other tools. The biggest benefit here is that sar makes the data available on an ongoing basis and is therefore more useful for obtaining long-term averages, or for the production of CPU utilization graphs.

On multiprocessor systems, the **sar -U** command can produce statistics for an individual processor or for all processors. Here is an example of output from **sar -U ALL**:

```
12:00:01 AM  CPU   %user  %nice  %system     %idle
12:10:00 AM  0     3.46   21.47  1.09        73.98
12:10:00 AM  1     3.91   18.73  1.03        76.33
12:20:01 AM  0     1.63   0.25         0.78        97.34
12:20:01 AM  1     1.82   0.20         0.81        97.17
```

```
[...]
10:00:00 AM   0       39.12    0.75            1.04            59.09
10:00:00 AM   1       31.22    0.92            1.09            66.77
Average:              0        7.61     4.91           3.86            83.61
Average:              1        7.33     4.78           3.88            84.02
```

The **sar -w** command reports on the number of context switches per second, making it possible to gain additional insight in where CPU cycles are being spent:

```
12:00:01 AM          cswch/s
12:10:00 AM          537.97
12:20:01 AM          339.43
[...]
10:10:00 AM          319.42
Average:             1158.25
```

It is also possible to produce two different sar reports on interrupt activity. The first, (produced using the **sar -I SUM** *command*) displays a single "interrupts per second" statistic:

```
12:00:01 AM          INTR          intr/s
12:10:00 AM          sum           539.15
12:20:01 AM          sum           539.49
[...]
10:40:01 AM          sum           539.10
Average:             sum           541.00
```

By using the command **sar -I PROC**, it is possible to break down interrupt activity by processor (on multiprocessor systems) and by interrupt level (from 0 to 15):

```
12:00:00 AM   CPU   i000/s   i001/s   i002/s   i008/s   i009/s   i011/s   i012/s
12:10:01 AM   0     512.01   0.00     0.00     0.00     3.44     0.00     0.00
12:10:01 AM   CPU   i000/s   i001/s   i002/s   i008/s   i009/s   i011/s   i012/s
12:20:01 AM   0     512.00   0.00     0.00     0.00     3.73     0.00     0.00
[...]
10:30:01 AM   CPU   i000/s   i001/s   i002/s   i003/s   i008/s   i009/s   i010/s
10:40:02 AM   0     512.00   1.67     0.00     0.00     0.00     15.08    0.00
Average:      0     512.00   0.42     0.00     N/A      0.00     6.03     N/A
```

This report (which has been truncated horizontally to fit on the page) includes one column for each interrupt level (for example, the i002/s field illustrating the rate for interrupt level 2). If this were a multiprocessor system, there would be one line per sample period for each CPU.
Another important point to note about this report is that **sar** adds or removes specific interrupt fields if no data is collected for that field. The example report above provides an example of this, the end of the report includes interrupt levels (3 and 10) that were not present at the start of the sampling period.

***** Note**: There are two other interrupt-related **sar** reports: **sar -I ALL** and **sar -I XALL**. However, the default configuration for the **sadc** data collection utility does not collect the information necessary for these reports. This can be changed by editing the file **/etc/cron.d/sysstat**, and changing this line:

```
*/10 * * * * root /usr/lib/sa/sa1 1 1
```

to this:

```
*/10 * * * * root /usr/lib/sa/sa1 -I 1 1
```

Keep in mind this change does cause additional information to be collected by **sadc**, and results in larger data file sizes. Therefore, make sure your system configuration can support the additional space consumption.

15.3. Produce and deliver reports on system utilization (processor, memory, disk, and network)

Installed and running by default, the **sysstat** package contains tools that capture system performance throughout the day, and automatically summarizes it for us. Generating utilization reports is then a simple matter of knowing the right **sar** command to execute.

- Processor
  - Basic processor report: **sar** or **sar -u**
  - Per processor statistics: **sar -P [0,1,..;|ALL]**
  - Load average: **sar -q**
  - Power management (not enabled by default): **sar -m**

- Memory
  - Kernel paging: **sar -B**
  - Memory: **sar -r**
  - Swap space: **sar -S**
  - Swapping: **sar -W**

- Disk
  - IO and transfer rate stats: **sar -b**

  - Block devices (disks): **sar -d** (**-p** to use pretty names)

- Network
  - Network statistics: **sar -n DEV**
  - Network errors: **sar -n EDEV**

- Everything
  - All reports simultaneously: **sar -A**

When the sysstat service is started, it uses parameters shown in the **sysstat** and **sysstat.ioconf** files, in the **/etc/sysconfig** directory

The **sysstat** package includes a regular cron job. Available in the **/etc/cron.d** directory, that job collects information on system utilization and sends it to log files in the **/var/log/sa** directory. Examine the **sysstat** file in the **/etc/cron.d** directory. The first line suggests a job that's run every ten minutes, by the root administrative user.

```
*/10 * * * * root /usr/lib64/sa/sa1  1 1
```

The **sa1** command, with the **1** and **1** at the end, specifies that the job should be written once, one second after the job is started. Information from this command is collected in file named sadd in the **/var/log/sa** directory, where *dd* represents the day of the month.

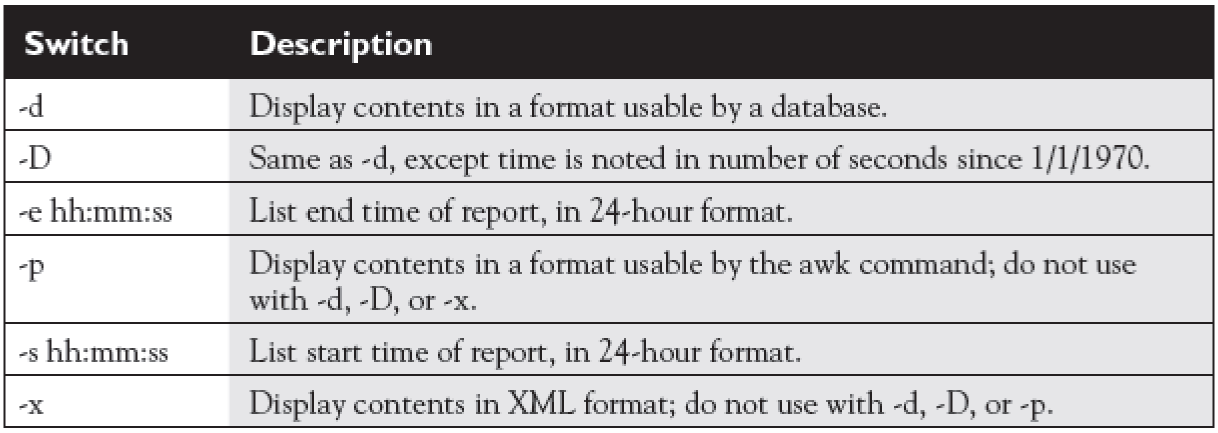
The next line is more powerful than it looks. On a daily basis, at seven minutes before midnight, with the privileges of the root administrative user, the **sa2** command writes a daily report on just about everything to the noted file in the **/var/log/sa** directory.

```
53 23 * * * root /usr/lib64/sa/sa2 -A
```

Information from this command is collected in file named **sar*dd*** in the **/var/log/sa** directory, where *dd* represents the day of the month. The **sar*dd*** files in that directory have already been processed into text files.

15.3.1. Prepare a system status report

The binary log files with names like *sa10* (for the tenth day of the month) can be processed in a number of ways by the **sadf** command.
Some of the more important **sadf** switches are:



Switch	Description
-d	Display contents in a format usable by a database.
-D	Same as -d, except time is noted in number of seconds since 1/1/1970.
-e hh:mm:ss	List end time of report, in 24-hour format.
-p	Display contents in a format usable by the awk command; do not use with -d, -D, or -x.
-s hh:mm:ss	List start time of report, in 24-hour format.
-x	Display contents in XML format; do not use with -d, -D, or -p.

For example, the following command sets up a report with data between the start and end of the tenth of the month:

```
# sadf -s 00:00:01 -e 23:59:59 /var/log/sa/sa10 > <activity10>
```

The data is redirected to the *activity10* file, for later processing. But the power of the **sysstat** package comes from the way it interacts with other command, **sar.** But only some of the switches to the **sar** command work with **sadf.** As suggested in the **sadf** man page, the following command prepares a report based on "memory, swap space, and network statistics" from the /var/log/sa/sa21 file:

```
# sadf -d /var/log/sa/sa21 -- -r -n DEV
```

While the **-d** is associated with the **sadf** command, the double-dash (**--**) points to options associated with the **sar** command. So the **-r** reports memory usage, and the **-n DEV** reports statistics from network devices.

You might modify the previous **sadf** command to meet all four of the items listed in the related RHCE objective:

```
# sadf -d /var/log/sa/sa21 -- -u -r -dp -n DEV
```

In other words, the **sadf** command specifies output usable by a database (**-d**) from the database file in the **/var/log/sa** directory associated with the twenty-first of the month. The double dash (**--**) points to **sar** command switches, with CPU utilization (**-u**); RAM utilization (**-r**); activity by block device; presented in more familiar block files such as sda (**-p**); with statistics from network devices (**-n DEV**).

15.4. Kill a remote session without killing processes started inside it

When a remote login session disconnects, the process started inside the remote login session will receive a SIGHUP signal and be killed. The following examples show how you can keep a process running in the background when the remote login session quits.

Method 1 - Using the **nohup**

nohup: run a command immune to hangups, with output to a non-tty.

Log on to the remote server.

```
# ssh root@<192.168.100.103>
```

Using **nohup** to run tasks on remote server via ssh session.

```
# nohup ping www.redhat.com &
[1] 5509
# nohup: appending output to `nohup.out'
The output of the task will be writed in the file /root/nohup.out
```

Run the **ps** command to verify that the ping process is still running.

```
# ps -elf | grep ping
4 S root 5509 5423 0 75 0 - 477 - 09:14 pts/1 00:00:00 ping www.redhat.com
```

Logout of the ssh login session, and login again to the remote server. Run the **ps** command again.

```
# ps -elf | grep ping
4 S root 5509 1 0 75 0 - 477 - 09:14 ? 00:00:00 ping www.redhat.com
```

The ping process is still running.

Method 2 - Using the **setsid**

setsid: run a program in a new session

Login in a remote server.

```
# ssh root@<192.168.100.103>
```

Use **setsid** to run tasks on the remote server via ssh session.

```
# setsid ping www.redhat.com
```

Run the ps command to verify that the ping process is still running.

```
# ps -elf | grep ping
4 S root 7653 1 0 75 0 - 477 - 09:40 ? 00:00:00 ping www.redhat.com
```

Logout of the ssh login session, and login again to the remote server. Run the **ps** again.

```
# ps -elf | grep ping
4 S root 7653 1 0 75 0 - 477 - 09:40 ? 00:00:00 ping www.redhat.com
```

The ping process is still running too.

15.5. Killing a zombie process or process in the "D" state with the parent ID of 1

A process with the **parent ID of 1** means that it is a child process of init. The process **init** is protected from even the command **kill -9 init** (which is a command that usually means "kill this process, regardless of anything else").

These processes cannot be killed without a reboot. To kill a zombie child process of init, the computer must be rebooted.

15.6. Resource monitoring

http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/4/html/Introduction_To_System_Administration/ch-resource.html

16. Configure a system as an iSCSI initiator that persistently mounts an iSCSI target

iSCSI initiator is the client who initiates the connection, iSCSI target is the server providing the storage.

The first thing to do is setup an iSCSI target (this means, assign a device - disks; this is not a role provided by Red Hat out of the box).

The easiest method of setting up a target is to use Openfiler, http://www.openfiler.com, it's a quick install and a fairly easy configuration. There is a nice walkthrough at http://www.techhead.co.uk/how-to-configure-openfiler-v23-iscsi-storage-for-use-with-vmware-esx that details how to setup the filer for iscsi.

Once setup, we now need to configure the iSCSI initiator. There is a great article on doing this at http://www.cyberciti.biz/tips/rhel-centos-fedora-linux-iscsi-howto.html

1.- Install necessary packages

```
# yum install iscsi-initiator-utils
# service iscsi start
```

2.- Configure initiator

Execute **iscsiadm -m discoverydb -t sendtargets -p <192.168.10.1> -D** to perform the discovery. Use **fdisk** to view all partitions and identify the new disk **fdisk -l** (should be something like /dev/sdb)

3.- Make the disk mount persistent

Execute **chkconfig iscsi on**

Because the device name can change between reboots, Red Hat suggests mounting the partition by using the UUID, executing **ls -l /dev/disk/by-uuid**, to find the uuid of the new disk

Edit **/etc/fstab** to configure the disk to mount on startup (pay attention to **_netdev** option that will prevent F.S. from being mounted before network services are started - in any other case system will fail to start up correctly)

```
UUID=93a0429d-0318-45c0-8320-9676ebf1ca79   /toto   ext3   _netdev         0  0
```

17. Partitions ("parted")

The utility **parted** allows users to:

- View the existing partition table
- Change the size of existing partitions
- Add partitions from free space or additional hard drives

By default, the **parted** package is included when installing Red Hat. To start **parted**, log in as root and type the command **parted** ***/dev/sda*** at a shell prompt (where ***/dev/sda*** is the device name for the drive you want to configure).

If you want to remove or resize a partition, the device on which that partition resides must not be in use. Creating a new partition on a device which is in use (while possible) is not recommended.
For a device to not be in use, none of the partitions on the device can be mounted, and any swap space on the device must not be enabled.

As well, the partition table should not be modified while it is in use because the kernel may not properly recognize the changes. If the partition table does not match the actual state of the mounted partitions, information could be written to the wrong partition, resulting in lost and overwritten data. The easiest way to achieve this is to boot your system in rescue mode. When prompted to mount the file system, select **Skip**.

Alternately, if the drive does not contain any partitions in use (system processes that use or lock the file system from being unmounted), you can **unmount** them with the umount command and turn off all the swap space on the hard drive with the **swapoff** command.

parted commands

Command	Description
`check` ***minor-num***	Perform a simple check of the file system
`cp` ***from to***	Copy file system from one partition to another; ***from*** and ***to*** are the minor numbers of the partitions
`help`	Display list of available commands
`mklabel` ***label***	Create a disk label for the partition table
`mkfs` ***minor-num file-system-type***	Create a file system of type ***file-system-type***
`mkpart` ***part-type fs-type start-mb end-mb***	Make a partition without creating a new file system
`mkpartfs` ***part-type fs-type start-mb end-mb***	Make a partition and create the specified file system
`move` ***minor-num start-mb end-mb***	Move the partition
`name` ***minor-num name***	Name the partition for Mac and PC98 disklabels only
`print`	Display the partition table
`quit`	Quit **parted**
`rescue` ***start-mb end-mb***	Rescue a lost partition from ***start-mb*** to ***end-mb***
`resize` ***minor-num start-mb end-mb***	Resize the partition from ***start-mb*** to ***end-mb***
`rm` ***minor-num***	Remove the partition
`select` ***device***	Select a different device to configure
`set` ***minor-num flag state***	Set the flag on a partition; ***state*** is either on or off
`toggle` [***NUMBER*** [***FLAG***]	Toggle the state of ***FLAG*** on partition ***NUMBER***
`unit` ***UNIT***	Set the default unit to ***UNIT***

17.1. Viewing the partition table

After starting **parted**, use the command **print** to view the partition table. A table similar to the following appears:

```
Model: ATA ST3160812AS (scsi)
Disk /dev/sda: 160GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Number  Start   End     Size    Type      File system  Flags
 1      32.3kB  107MB   107MB   primary   ext3         boot
 2      107MB   105GB   105GB   primary   ext3
 3      105GB   107GB   2147MB  primary   linux-swap
 4      107GB   160GB   52.9GB  extended                      root
 5      107GB   133GB   26.2GB  logical   ext3
 6      133GB   133GB   107MB   logical   ext3
 7      133GB   160GB   26.6GB  logical                lvm
```

In the partition table, the Minor number is the partition number. For example, the partition with minor number 1 corresponds to /dev/sda1. The Start and End values are in megabytes. Valid Type are metadata, free, primary, extended, or logical. The Filesystem is the file system type, which can be any of the following:

ext2
ext3
fat16
fat32
hfs
jfs
linux-swap
ntfs
reiserfs
hp-ufs
sun-ufs
xfs

If a **Filesystem** of a device shows no value, this means that its file system type is unknown.
The **Flags** column lists the flags set for the partition. Available flags are boot, root, swap, hidden, raid, lvm, or lba.

17.2. Creating a partition

Before creating a partition, boot into rescue mode (or unmount any partitions on the device and turn off any swap space on the device).

Start **parted**, where ***/dev/sda*** is the device on which to create the partition:

```
# parted /dev/sda
```

View the current partition table to determine if there is enough free space:

```
print
```

17.2.1. Making the partition

If there is not enough free space, you can resize an existing partition.

From the partition table, determine the start and end points of the new partition and what partition type it should be. You can only have four primary partitions (with no extended partition) on a device. If you need more than four partitions, you can have three primary partitions, one extended partition, and multiple logical partitions within the extended.

For example, to create a primary partition with an ext3 file system from 1024 megabytes until 2048 megabytes on a hard drive type the following command:

```
mkpart  primary  ext3  1024  2048
```

The changes start taking place as soon as you press **Enter**, so review the command before executing to it.
After creating the partition, use the **print** command to confirm that it is in the partition table with the correct partition type, file system type, and size. Also remember the minor number of the new partition so that you can label any file systems on it. You should also view the output of **cat /proc/partitions** to make sure the kernel recognizes the new partition.

17.2.2. Formatting and labeling the partition

The partition still does not have a file system. Create the file system:

```
# /sbin/mkfs -t ext3 /dev/sda6
```

***** Warning**: Formatting the partition permanently destroys any data that currently exists on the partition.

Next, give the file system on the partition a label. For example, if the file system on the new partition is ***/dev/sda6*** and you want to label it ***/work***, use:

```
# e2label /dev/sda6 /work
```

By default, the installation program uses the mount point of the partition as the label to make sure the label is unique. You can use any label you want.

17.2.3. Add to /etc/fstab

As root, edit the **/etc/fstab** file to include the new partition using the partition's UUID. Use the **blkid -L *label*** command to retrieve the partition's UUID. The new line should look similar to the following:

```
UUID=93a0429d-0318-45c0-8320-9676ebf1ca79    /work    ext3    defaults 1  2
```

The first column should contain **UUID=** followed by the file system's UUID. The second column should contain the mount point for the new partition, and the next column should be the file system type (for example, ext3 or swap).

If the fourth column is the word **defaults**, the partition is mounted at boot time. To mount the partition without rebooting, as root, type the command:

```
# mount  /work
```

17.3. Removing a partition

Before removing a partition, boot into rescue mode (or unmount any partitions on the device and turn off any swap space on the device).

Start **parted**, where ***/dev/sda*** is the device on which to remove the partition:

```
# parted /dev/sda
```

View the current partition table to determine the minor number of the partition to remove:

```
print
```

Remove the partition with the command **rm**. For example, to remove the partition with minor number 3:

```
rm  3
```

The changes start taking place as soon as you press **Enter**, so review the command before committing to it.

After removing the partition, use the **print** command to confirm that it is removed from the partition table. You should also view the output of

```
# cat /proc/partitions
```

to make sure the kernel knows the partition is removed.

The last step is to remove it from the **/etc/fstab** file. Find the line that declares the removed partition, and remove it from the file.

17.4. Resizing a partition

Before resizing a partition, boot into rescue mode (or unmount any partitions on the device and turn off any swap space on the device).

Start **parted**, where ***/dev/sda*** is the device on which to resize the partition:

```
# parted /dev/sda
```

View the current partition table to determine the minor number of the partition to resize as well as the start and end points for the partition:

```
print
```

To resize the partition, use the **resize** command followed by the minor number for the partition, the starting place in megabytes, and the end place in megabytes. For example:

```
resize  3  1024  2048
```

***** Warning**: A partition cannot be made larger than the space available on the device.

After resizing the partition, use the **print** command to confirm that the partition has been resized correctly, is the correct partition type, and is the correct file system type.

After rebooting the system into normal mode, use the command **df** to make sure the partition was mounted and is recognized with the new size.

17.5. How to use UUID or label to mount partitions

List all UUIDs, using **blkid** command-line utility to locate/print block device attributes:

```
# blkid

        /dev/sda1: TYPE="ntfs" UUID="A0F0582EF0580CC2"
        /dev/sda2: UUID="8c2da865-13f4-47a2-9c92-2f31738469e8" TYPE="ext3"
        /dev/sda3: TYPE="swap" UUID="5641913f-9bcc-4d8a-8bcb-ddfc3159e70f"
        /dev/sda5: UUID="FAB008D6B0089AF1" TYPE="ntfs"
        /dev/sdb1: UUID="32c61b65-f2f8-4041-a5d5-3d5ef4182723" TYPE="ext3"
        /dev/sdb2: UUID="41c22818-fbad-4da6-8196-c816df0b7aa8" TYPE="ext3"

# vi /etc/fstab

        UUID=8c2da865-13f4-47a2-9c92-2f31738469e8  /disk/mount_point  ext3 defaults 0 1

# mount -a
```

Now to mount a filesystem via label requires another step, to label the filesystem. Luckily this is done in one easy step using **e2label**. I am going to label the filesystem luksdrive

```
# e2label /dev/mapper/mynew_data luksdrive
```

Now we can unmount the filesystem, change our fstab to use a label, and run mount a again to see it mounted via label instead.

```
# umount /mynew_data/
```

Then edit **/etc/fstab** this time using **LABEL=*luksdrive*** in place of **UUID**. So the line should look like:

```
LABEL=luksdrive       /mynew_data          ext4    defaults       1 2
```

Run mount -a and mount to confirm:

```
# mount -a
# mount
[.....]
/dev/mapper/mynew_data on /mynew_data type ext4 (rw)
```

18. Managing logical volumes/volume groups/physical volumes

18.1. Understanging LVM (basics)

LVM (Logical Volume Management) partitions provide a number of advantages over standard partitions. LVM partitions are formatted as physical volumes. One or more physical volumes are combined to form a volume group. Each volume group's total storage is then divided into one or more logical volumes. The logical volumes function much like standard partitions. They have a file system type, such as **ext4**, and a mount point.

The /boot partition and LVM

On most architectures, the boot loader cannot read LVM volumes. You must make a standard, non-LVM disk partition for your **/boot** partition.

However, on System z, the **zipl** boot loader supports **/boot** on LVM logical volumes with linear mapping.

To understand LVM better, imagine the physical volume as a pile of blocks. A block is simply a storage unit used to store data. Several piles of blocks can be combined to make a much larger pile, just as physical volumes are combined to make a volume group. The resulting pile can be subdivided into several smaller piles of arbitrary size, just as a volume group is allocated to several logical volumes.

An administrator may grow or shrink logical volumes without destroying data, unlike standard disk partitions. If the physical volumes in a volume group are on separate drives or RAID arrays then administrators may also spread a logical volume across the storage devices.

You may lose data if you shrink a logical volume to a smaller capacity than the data on the volume requires. To ensure maximum flexibility, create logical volumes to meet your current needs, and leave excess storage capacity unallocated. You may safely grow logical volumes to use unallocated space, as your needs dictate.

LVM is a tool for logical volume management which includes allocating disks, striping, mirroring and resizing logical volumes.

With LVM, a hard drive or set of hard drives is allocated to one or more physical volumes. LVM physical volumes can be placed on other block devices which might span two or more disks.

The **physical volumes** are combined into **logical volumes**, with the exception of the /boot partition. The /boot partition cannot be on a logical volume group because the boot loader cannot read it. If the root (/) partition is on a logical volume, create a separate /boot partition which is not a part of a volume group.

Since a physical volume cannot span over multiple drives, to span over more than one drive, create one or more physical volumes per drive.

The volume groups can be divided into **logical volumes**, which are assigned mount points, such as /home and / and file system types, such as ext2 or ext3. When "partitions" reach their full capacity, free space from the volume group can be added to the logical volume to increase the size of the partition. When a new hard drive is added to the system, it can be added to the volume group, and partitions that are logical volumes can be increased in size.

On the other hand, if a system is partitioned with the ext3 file system, the hard drive is divided into partitions of defined sizes. If a partition becomes full, it is not easy to expand the size of the partition. Even if the partition is moved to another hard drive, the original hard drive space has to be reallocated as a different partition or not used.

18.2. Create and remove physical volumes, assign physical volumes to volume groups, create and delete logical volumes

Create and remove physical volumes

Creating a physical volume in LVM is the first step in the LVM setup. It's the part where you actually tell Red Hat Enterprise Linux that you want a disk to be used for LVM.

pvcreate is the command used to add the physical volumes, or physical partitions:

```
# pvcreate /dev/sdb
Physical volume "/dev/sdb" successfully created
```

pvremove is used to disassociate the volume from LVM:

```
# pvremove /dev/sdb
Labels on physical volume "/dev/sdb" successfully wiped
```

Assign physical volumes to volume groups

Once that physical volume has been created we can add it to a volume group with the **vgcreate** or **vgextend** (if the volume group has already been created).

If the volume group does not exist, you can create it and add physical volumes in one shot:

```
# vgextend MyVolGroup /dev/sdc
No physical volume label read from /dev/sdc
Physical volume "/dev/sdc" successfully created
Volume group "MyVolGroup" successfully extended
```

To assign a new physical volume to an existing volume group we use **vgextend**:

```
# vgextend MyVolGroup /dev/sdc
Volume group "MyVolGroup" successfully extended
```

Similarly if we want to remove /dev/sdc from that group we would run **vgreduce**:

```
# vgreduce MyVolGroup /dev/sdc
Removed "/dev/sdc" from volume group "MyVolGroup"
```

Create and delete logical volumes

Logical Volumes have similar commands to create and delete as Volume Groups and Physical Volumes.

To create a new logical volume:

```
# lvcreate -L 100M -n lvol00 MyVolGroup
Logical volume "lvol00" created
```

To display the volume after for confirmation:

```
# lvdisplay /dev/MyVolGroup/lvol00
--- Logical volume ---
LV Name                /dev/MyVolGroup/lvol00
VG Name                MyVolGroup
LV UUID                zwLMev-i63w-7Jpk-XuqZ-VGl7-89Ov-WpoewP
LV Write Access        read/write
LV Status              available
# open                 0
LV Size                100.00 MiB
Current LE             25
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           253:2
```

To delete the logical volume you would use the LV Name listed in the results of **lvdisplay**:

```
# lvremove /dev/MyVolGroup/lvol00
Do you really want to remove active logical volume lvol00? [y/n]: y
Logical volume "lvol00" successfully removed
```

18.3. Adding, removing and moving Swap space

Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM. Swap space is located on hard drives, which have a slower access time than physical memory.
Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap files.
If:
M = Amount of RAM in GB, and S = Amount of swap in GB, then

```
If M < 2
    S = M *2
Else
    If M < 32
        S = M + 2
    Else
        S = M
```

18.3.1. Adding Swap on an LVM2 Logical Volume

Identify swap partition to enlarge:

```
# cat /etc/fstab | grep swap
/dev/VolGroup00/LogVol01   swap            swap   defaults      0 0
```

Disable swapping for the associated logical volume:

```
# swapoff -v /dev/VolGroup00/LogVol01
```

Resize the LVM2 logical volume by 256 MB:

```
# lvresize /dev/VolGroup00/LogVol01 -L +256M
```

Format the new swap space:

```
# mkswap /dev/VolGroup00/LogVol01
```

Enable the extended logical volume:

```
# swapon -v /dev/VolGroup00/LogVol01
```

18.3.2. Creating an LVM2 Logical Volume for Swap

Create the LVM2 logical volume of size 256 MB:

```
# lvcreate VolGroup00 -n LogVol02 -L 256M
```

Format the new swap space:

```
# mkswap /dev/VolGroup00/LogVol02
```

Add the following entry to the /etc/fstab file:

```
/dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

Enable the extended logical volume:

```
# swapon -v /dev/VolGroup00/LogVol02
```

18.3.3. Creating a Swap file

Determine the size of the new swap file in megabytes and multiply by 1024 to determine the number of blocks. For example, the block size of a 64 MB swap file is 65536.

Type the following command with count being equal to the desired block size:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

Setup the swap file with the command:

```
# mkswap /swapfile
```

To enable the swap file immediately but not automatically at boot time:

```
# swapon /swapfile
```

To enable it at boot time, edit /etc/fstab to include the following entry:

```
/swapfile swap swap defaults 0 0
```

18.4. Add new partitions, logical volumes and swap to a system non-destructively

Add new partitions

Adding new partitions is fairly straightforward using tools like **fdisk** or **parted**. We will be using **fdisk** for this objective.

First step is to open the device that we want to make the partition on.

```
# fdisk -cu /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x110ea9fa.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
```

You generally want to print the current partition table on the device with the p command

```
Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x110ea9fa

Device Boot      Start         End      Blocks   Id  System
```

If nothing shows up under the list of devices, then there are no partitions. To make the first partition we will use the '**n**' command, for new. Then follow the prompts to create the new partition. Here we are making a 50MB partition.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-4194303, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-4194303, default 4194303): +50M
```

Once you've created the partition it's a good idea to print the partition table. Once you are happy with the changes, write them to the disk with the w command.

```
Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x110ea9fa

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      104447       51200   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Add new logical volumes

To add a new logical volume you first need to create a new partition and set it to type Linux LVM.

```
# fdisk -cu /dev/sdb

Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x110ea9fa

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n
Command action
   e   extended
```

```
  p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-4194303, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-4194303, default 4194303): +50M


Command (m for help): t
Selected partition 1
Hex code (type L to list codes): L

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           39  Plan 9          82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      41  PPC PReP Boot   85  Linux extended  c7  Syrinx
 5  Extended        42  SFS             86  NTFS volume set da  Non-FS data
 6  FAT16           4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS       4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
 8  AIX             4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    50  OnTrack DM      93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       52  CP/M            9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
 e  W95 FAT16 (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  GPT
 f  W95 Ext'd (LBA) 55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 61  SpeedStor       a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    64  Novell Netware  af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 65  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  70  DiskSecure Mult b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 75  PC/IX           bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 80  Old Minix       be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1
Hex code (type L to list codes): 8e
Changed system type of partition 1 to 8e (Linux LVM)

Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x110ea9fa

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      104447       51200   8e  Linux LVM

Command (m for help): w
```

```
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Once you create the partition as Linux LVM type you can now create a physical volume with that partition. Keep in mind we are using small partitions in order to get the concept across, in real life these would be entire disks being added as physical volumes.

To create a physical volume we use pvcreate:

```
# pvcreate /dev/sdb1 physicalvol1
Physical volume "/dev/sdb1" successfully created

# pvdisplay

  "/dev/sdb1" is a new physical volume of "50.00 MiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               50.00 MiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               ysMMtz-jYei-tNph-tvSU-3sSy-vemf-sjWxbO
```

Next we have to create the volume group and add the physical volume to it. We can do this with the **vgcreate** command.

```
# vgcreate MyNewVolGroup /dev/sdb1
 Volume group "MyNewVolGroup" successfully created
```

Now that we have a physical volume, and its part of a volume group, so we can make a logical volume. To create a logical volume named ***LogVol1*** from the volume group ***MyNewVolGroup*** we would use the following command.

```
# lvcreate -L 28M -n LogVol1 MyNewVolGroup
  Logical volume "LogVol1" created
```

Now there is a logical volume named "***LogVol1***" that is a part of "***MyNewVolGroup***"

Add new swap partition

Adding a swap partition is like adding any other partition, just changing his type to Linux Swap.

```
# fdisk -cu /dev/sdb
```

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First sector (104448-4194303, default 104448):
Using default value 104448
Last sector, +sectors or +size{K,M,G} (104448-4194303, default 4194303): +50M

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes
128 heads, 57 sectors/track, 574 cylinders, total 4194304 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x110ea9fa

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      104447       51200   8e  Linux LVM
/dev/sdb2          104448      206847       51200   82  Linux swap / Solaris

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

To activate a swap partion

```
# swapon -v /dev/sdb2
swapon on /dev/sdb2
swapon: /dev/sdb2: found swap signature: version 1, page-size 4, same byte order
swapon: /dev/sdb2: pagesize=4096, swapsize=52428800, devsize=52428800
```

Confirm that the swap was added

```
# swapon -s
Filename                                Type            Size    Used    Priority
/dev/dm-1                               partition       1015800 0       -1
/dev/sdb2                               partition       51192 0         -2
```

18.5. Extend existing unencrypted ext4-formatted logical volumes

To extend a logical volume you can use the **lvextend** command, with the **-L** switch to specify the size. There are two ways to do this, make sure you are careful with the syntax.

First lets display our logical volumes

```
# lvdisplay
 --- Logical volume ---
 LV Name                /dev/MyNewVolgroup1/MyNewLogVol1
 VG Name                MyNewVolgroup1
 LV UUID                ZyZZLu-oQYu-ifti-ww28-VzQ7-xV5w-wOvBim
 LV Write Access        read/write
 LV Status              available
 # open                 0
 LV Size                200.00 MiB
 Current LE             50
 Segments               1
 Allocation             inherit
 Read ahead sectors     auto
 - currently set to     256
 Block device           253:2
```

Next we will extend the logical volume, adding 100M. See how we do this by putting a + sign in front of the size?

```
# lvextend -L +100M /dev/MyNewVolgroup1/MyNewLogVol1
Extending logical volume MyNewLogVol1 to 300.00 MiB
Logical volume MyNewLogVol1 successfully resized
```

Now we extend the volume TO BE 400M. So here we are specifying the end size, as opposed to adding that amount. This is an important distinction to make, especially during testing. Last thing you want to do is destroy a filesystem because you had to shrink it in order to meet the test requirements.

```
# lvextend -L 400M /dev/MyNewVolgroup1/MyNewLogVol1
Extending logical volume MyNewLogVol1 to 400.00 MiB
Logical volume MyNewLogVol1 successfully resized
```

19. Create and configure LUKS-encrypted partitions and logical volumes

19.1. Disk encryption

Block device encryption protects the data on a block device by encrypting it. To access the device's decrypted contents, a user must provide a passphrase or key as authentication. This provides additional security beyond existing OS security mechanisms in that it protects the device's contents even if it has been physically removed from the system.

19.2. Encrypting block devices using dm-crypt/LUKS

Linux Unified Key Setup (LUKS) is a specification for block device encryption. It establishes an on-disk format for the data, as well as a passphrase/key management policy.

LUKS uses the kernel device mapper subsystem via the **dm-crypt** module. This arrangement provides a low-level mapping that handles encryption and decryption of the device's data. User-level operations, such as creating and accessing encrypted devices, are accomplished through the use of the **cryptsetup** utility.

19.2.1. How to access the encrypted devices after installation? (System Startup)

During system startup you will be presented with a passphrase prompt. After the correct passphrase has been provided the system will continue to boot normally. If you used different passphrases for multiple encrypted devices you may need to enter more than one passphrase during the startup.

19.3. Creating encrypted block devices in Anaconda

One can create encrypted devices during system installation. This allows to easily configuring a system with encrypted partitions.

To enable block device encryption, check the "Encrypt System" checkbox when selecting automatic partitioning or the "Encrypt" checkbox when creating an individual partition, software RAID array, or logical volume. After you finish partitioning, you will be prompted for an encryption passphrase. This passphrase will be required to access the encrypted devices. If you have pre-existing LUKS devices and provided correct passphrases for them earlier in the install process the passphrase entry dialog will also contain a checkbox. Checking this checkbox indicates that you would like the new passphrase to be added to an available slot in each of the pre-existing encrypted block devices.

*****Tip**: Checking the "Encrypt System" checkbox on the "Automatic Partitioning" screen and then choosing "Create custom layout" does not cause any block devices to be encrypted automatically.

Most types of block devices can be encrypted using LUKS. From anaconda you can encrypt partitions, LVM physical volumes, LVM logical volumes, and software RAID arrays.

19.4. Creating encrypted block devices on the installed system after installation

Encrypted block devices can be created and configured after installation.

19.4.1. Optional: Fill the device with random data

Filling <device> (eg: /dev/sda3) with random data before encrypting it greatly increases the strength of the encryption. The downside is that it can take a very long time.

The best way, which provides high quality random data but takes a long time (several minutes per gigabyte on most systems):

```
# dd if=/dev/urandom of=<device>
```

Fastest way, which provides lower quality random data:

```
# badblocks -c 10240 -s -w -t random -v <device>
```

19.4.2. Format the device as a dm-crypt/LUKS encrypted device

```
# cryptsetup luksFormat <device>
```

After supplying the passphrase twice the device will be formatted for use. To verify, use the following command:

```
# cryptsetup isLuks <device> && echo "Ok"
```

To see a summary of the encryption information for the device, use the following command:

```
# cryptsetup luksDump <device>
```

19.4.3. Create a mapping to allow access to the device's decrypted contents

To access the device's decrypted contents, a mapping must be established using the kernel **device-mapper**.

It is useful to choose a meaningful name for this mapping. LUKS provides a UUID (Universally Unique Identifier) for each device. This, unlike the device name (eg: **/dev/sda3**), is guaranteed to remain constant as long as the LUKS header remains intact. To find a LUKS device's UUID, run the following command:

```
# cryptsetup luksUUID <device>
```

An example of a reliable, informative and unique mapping name would be **luks-<uuid>**, where <uuid> is replaced with the device's LUKS UUID (eg: luks-50ec957a-5b5a-47ee-85e6-f8085bbc97a8). This naming convention might seem unwieldy but is it not necessary to type it often.

```
# cryptsetup luksOpen <device> <name>
```

There should now be a device node, /dev/mapper/<name>, which represents the decrypted device. This block device can be read from and written to like any other unencrypted block device.

To see some information about the mapped device, use the following command:

```
# dmsetup info <name>
```

To remove de device node, /dev/mapper/<name>, run following command:

```
# cryptsetup luksClose /dev/mapper/keyfile
```

19.4.4. Create filesystems on the mapped device, or continue to build complex storage structures using the mapped device

Use the mapped device node **(/dev/mapper/<name>**) as any other block device. To create an **ext2** filesystem on the mapped device, use the following command:

```
# mke2fs  /dev/mapper/<name>
```

19.4.5. Add the mapping information to /etc/crypttab and /etc/fstab

In order for the system to set up a mapping for the device, an entry must be present in the **/etc/crypttab** file. If the file doesn't exist, create it and change the owner and group to root (**root:root**) and change the mode to **0744**. Add a line to the file with the following format:

```
<name>  <device>  none
```

The <device> field should be given in the form "**UUID=<luks_uuid>**", where <luks_uuid> is the LUKS uuid as given by the command **cryptsetup luksUUID <device>**. This ensures the correct device will be identified and used even if the device node (eg: /dev/sda5) changes.

```
lukspartitionname   UUID=d7a6fa65-6c37-4e8d-8cfa-4d6fea764abe     none
```

***** Note**: This way of opening LUKS partitions will ask for a password to open the device. If we want the device to be opened automatically we can create and add a password file and modify the corresponding entry in the **/etc/crypttab** file with following format:

```
<name>  <device>  <absolute_path_to_file>
```

Add an entry to /etc/fstab. This is only necessary if you want to establish a persistent association between the device and a mountpoint. Use the decrypted device, **/dev/mapper/<name>** in the **/etc/fstab** file.

In many cases it is desirable to list devices in **/etc/fstab** by UUID or by a filesystem label. The main purpose of this is to provide a constant identifier in the event that the device name (eg: /dev/sda4) changes. LUKS device names in the form of **/dev/mapper/luks-<luks_uuid>** are based only on the device's LUKS UUID, and are therefore guaranteed to remain constant. This fact makes them suitable for use in **/etc/fstab**.

If we wanted to reference by UUID we could get the UUID with the **blkid** command.

```
# blkid
/dev/mapper/newlukspart: UUID="51b401e1-2120-4f52-88ff-d6c80c379276" TYPE="ext4"
```

then in **fstab**:

```
UUID="51b401e1-2120-4f52-88ff-d6c80c379276"   /mountpoint    ext4   defaults   0  0
```

19.5. Disk Encryption Common Post-Installation Tasks

Set a randomly generated key as an additional way to access an encrypted block device

- Generate a key. This will generate a 256-bit key in the file $HOME/keyfile.

```
# dd  if=/dev/urandom  of=$HOME/keyfile  bs=32  count=1
# chmod  600  $HOME/keyfile
```

- Add the key to an available keyslot on the encrypted device

```
# cryptsetup  luksAddKey  <device>  <~/keyfile>
```

After being prompted for any one of the existing passphrases for authentication, key in keyfile will be added to the device.

- This key file may be used to open LUKS device with following command:

```
# cryptsetup luksOpen /dev/sda3   lukspartition  --key-file  keyfile
```

- Remove a key from a device

```
# cryptsetup  luksRemoveKey  /dev/sda3  keyfile
```

You will be prompted for any one of the remaining passphrases for authentication.

Add an additional passphrase as an additional way to access an encrypted block device

- Add a new passphrase to an existing device

```
# cryptsetup  luksAddKey  <device>
```

After being prompted for any one of the existing passphrases for authentication, you will be prompted to enter the new passphrase.

- Remove a passphrase from a device

```
# cryptsetup  luksRemoveKey  <device>
```

You will be prompted for the passphrase you wish to remove and then for any one of the remaining passphrases for authentication.

20. The ext4 FileSystem

The **ext4** file system is a scalable extension of the **ext3** file system. ext4 can support files and file systems of up to **16 terabytes in size**. It also supports an **unlimited number of sub-directories** (the ext3 file system only supports up to 32,000). Further, ext4 is backward compatible with ext3 and ext2, allowing these older versions to be mounted with the ext4 driver.

Main Features

ext4 uses **extents** (as opposed to the traditional block mapping scheme used by ext2 and ext3), which improves performance when using large files and reduces metadata overhead for large files. In addition, ext4 also labels unallocated block groups and inode table sections accordingly, which allows them to be skipped during a file system check. This makes for quicker file system checks, which becomes more beneficial as the file system grows in size.

20.1. Creating an ext4 FileSystem

The default filesystem with Red Hat Enterprise Linux 6 is ext4. The **mkfs** command can be used with the filesystem type as shown below. We are making a filesystem on a newly created partition, /dev/sdc1

To create an ext4 file system, use the **mkfs.ext4** command. In general, the default options are optimal for most usage scenarios, as in:

```
# mkfs.ext4  /dev/<device>
```

If we wanted to make an ext3 filesystem we would just run, **mkfs.ext3** /dev/<device>

For striped block devices (e.g. RAID5 arrays), the stripe geometry can be specified at the time of file system creation. Using proper stripe geometry greatly enhances performance of an ext4 file system.

When creating file systems on lvm or md volumes, **mkfs.ext4** chooses an optimal geometry. This may also be true on some hardware RAIDs which export geometry information to the operating system.

To specify stripe geometry, use the **-E** option of **mkfs.ext4** (i.e. extended file system options) with the following sub-options:

stride=value
Specifies the RAID chunk size.

stripe-width=value
Specifies the number of data disks in a RAID device, or the number of stripe units in the stripe.

For both sub-options, **value** must be specified in file system block units. For example, to create a file system with a 64k stride (i.e. 16 x 4096) on a 4k-block file system, use the following commmand:

```
# mkfs.ext4  -E  stride=16,stripe-width=64  /dev/<device>
```

20.2. Converting an ext3 File System to ext4

Many of the ext4 file system enhancements over ext3 result from modified metadata structures configured on the disk during file system creation. However, an existing ext3 file system can be upgraded to take advantage of other improvements in ext4.

Whenever possible, create a new ext4 file system and migrate your data to it instead of converting from ext3 to ext4. This ensures a better metadata layout, allowing for the enhanced performance natively provided by ext4.

To enable ext4 features on an existing ext3 file system, begin by using the **tune2fs** command in the following manner:

```
# tune2fs -O extents,uninit_bg /dev/<device>
```

The **-O** option sets, clears, or initializes a comma-delimited list of file system features. With the **extents** parameter, the file system will now use extents instead of the indirect block scheme for storing data blocks in an inode (but only for files created after activating extents feature). The **uninit_bg** parameter allows the kernel to mark unused block groups accordingly.

After using **tune2fs** to modify the file system, perform a file system check using the following command:

```
# e2fsck -f /dev/<device>
```

Note that without the file system check, the converted file system cannot be mounted. During the course of conversion, **e2fsck** may print the following warning:

One or more block group descriptor checksums are invalid

This warning is generally benign, as **e2fsck** will repair any invalid block group descriptors it encounters during the conversion process.

An ext3 file system converted to ext4 in the manner described in this section can no longer be mounted as ext3.

In addition, an ext2 file system cannot be converted directly to ext4; it should be converted to ext3, at which point it can be converted to ext4 (or mounted using the ext4 driver).

20.3. Mounting an ext4 File System

An ext4 file system can be mounted with no extra options. For example:

```
# mount /dev/<device> </mount/point>
```

The ext4 file system also supports several mount options to influence behavior. For example, the **acl** parameter enables access control lists, while the **user_xattr** parameter enables user extended attributes. To enable both options, use their respective parameters with **-o**, as in:

```
# mount -o acl,user_xattr /dev/<device> </mount/point>
```

20.3.1. Write Barriers

By default, ext4 uses write barriers to ensure file system integrity even when power is lost to a device with write caches enabled. For devices without write caches, or with battery-backed write caches, disable barriers using the **nobarrier** option, as in:

```
# mount -o nobarrier /dev/<device> </mount/point>
```

20.3.2. Mounting an ext3 File System as ext4

An ext3 file system can also be mounted as ext4 without changing the format, allowing it to be mounted as ext3 again in the future. To do so, run the following command (where *device* is an ext3 file system):

```
# mount -t ext4 /dev/<device> </mount/point>
```

Doing so will only allow the ext3 file system to use ext4-specific features that do not require a file format conversion. These features include delayed allocation and multi-block allocation, and exclude features such as extent mapping.

20.4. Resizing an ext4 Filesystem

Before growing an ext4 file system, ensure that the underlying block device is of an appropriate size to hold the file system later. Use the appropriate resizing methods for the affected block device.

An ext4 file system may be grown while mounted using the resize2fs command, as in:

```
# resize2fs /mount/point <size>
```

The resize2fs command can also decrease the size of an unmounted ext4 file system, as in:

```
# resize2fs /dev/device <size>
```

When resizing an ext4 file system, the **resize2fs** utility reads the size in units of file system block size, unless a suffix indicating a specific unit is used. The following suffixes indicate specific units:

s — 512kb sectors
K — kilobytes
M — megabytes
G — gigabytes

20.5. Other ext4 File System Utilities

e2fsck
Used to repair an ext4 file system. This tool checks and repairs an ext4 file system more efficiently than ext3, thanks to updates in the ext4 disk structure.

e2label
Changes the label on an ext4 file system. This tool can also works on ext2 and ext3 file systems.

quota
Controls and reports on disk space (blocks) and file (inode) usage by users and groups on an ext4 file system.

tune2fs
Can adjust configurable file system parameters for ext2, ext3, and ext4 file systems.

debugfs
Debugs ext2, ext3, or ext4 file systems.

e2image
Saves critical ext2, ext3, or ext4 file system metadata to a file.

21. Network interfaces

21.1. Network configuration files

The primary network configuration files are as follows:

/etc/hosts: The main purpose of this file is to resolve hostnames that cannot be resolved any other way. It can also be used to resolve hostnames on small networks with no DNS server. Regardless of the type of network the computer is on, this file should contain a line specifying the IP address of the loopback device (**127.0.0.1**) as **localhost.localdomain**.

/etc/resolv.conf: This file specifies the IP addresses of DNS servers and the search domain. Unless configured to do otherwise, the network initialization scripts populate this file:

```
search <domain1.domain1> <domain2.domain2>
nameserver XXX.XXX.XXX.XXX
nameserver XXX.XXX.XXX.XXX
```

/etc/sysconfig/network: This file specifies routing and host information for all network interfaces:

```
NETWORKING=yes
HOSTNAME=<hostname>
GATEWAY=<10.70.72.1>
NETWORKDELAY=<5>
```

/etc/sysconfig/network-scripts/ifcfg-interface-name: For each network interface, there is a corresponding interface configuration script. Each of these files provide information specific to a particular network interface.

*** **Note**: The **/etc/sysconfig/networking/** directory is used by the Network Administration Tool (**system-config-network**) and its contents should not be edited manually. Using only one method for network configuration is strongly encouraged, due to the risk of configuration deletion.

21.2. Interface configuration files

Interface configuration files control the software interfaces for individual network devices. As the system boots, it uses these files to determine what interfaces to bring up and how to configure them. These files are usually named **ifcfg-*name***, where ***name*** refers to the name of the device that the configuration file controls.

21.2.1. Ethernet interfaces

One of the most common interface files is **ifcfg-eth0**, which controls the first Ethernet network interface card or NIC in the system. In a system with multiple NICs, there are multiple **ifcfg-eth*X*** files (where ***X*** is a unique number corresponding to a specific interface). Because each device has its own configuration file, an administrator can control how each interface functions individually.

The following is a sample ifcfg-eth0 file for a system using a fixed IP address:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=10.0.1.27
USERCTL=no
```

The values required in an interface configuration file can change based on other values. For example, the **ifcfg-eth0** file for an interface using DHCP looks different because IP information is provided by the DHCP server:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

Below is a listing of the configurable parameters in an Ethernet interface configuration file:

BONDING_OPTS=*parameters*

Sets the configuration parameters for the bonding device, and is used in **/etc/sysconfig/network-scripts/ifcfg-bond<N>**

This configuration method is used so that multiple bonding devices can have different configurations. It is highly recommended to place all of your bonding options after the **BONDING_OPTS** directive in **ifcfg-*name***. Do not specify options for the bonding device in **/etc/modprobe.d/bonding.conf**, or in the deprecated **/etc/modprobe.conf** file.

BOOTPROTO=*protocol*

Where ***protocol*** is one of the following:

- **none** — No boot-time protocol should be used.
- **bootp** — The BOOTP protocol should be used.
- **dhcp** — The DHCP protocol should be used.

BROADCAST=*address*

Where ***address*** is the broadcast address. This directive is deprecated, as the value is calculated automatically with **ipcalc**.

DEVICE=*name*

Where ***name*** is the name of the physical device (except for dynamically-allocated PPP devices where it is the logical name).

DHCP_HOSTNAME=*name*

Where ***name*** is a short hostname to be sent to the DHCP server. Use this option only if the DHCP server requires the client to specify a hostname before receiving an IP address.

DNS{1,2}=*address*

Where ***address*** is a name server address to be placed in **/etc/resolv.conf** if the **PEERDNS** directive is set to **yes**.

ETHTOOL_OPTS=*options*

Where ***options*** are any device-specific options supported by **ethtool**. For example, if you wanted to force 100Mb, full duplex:

ETHTOOL_OPTS="autoneg off speed 100 duplex full"

Instead of a custom initscript, use **ETHTOOL_OPTS** to set the interface speed and duplex settings. Custom initscripts run outside of the network init script lead to unpredictable results during a post-boot network service restart.

*** **Note**: Set "autoneg off" before changing speed or duplex settings

Changing speed or duplex settings almost always requires disabling autonegotiation with the **autoneg off** option. This needs to be stated first, as the option entries are order-dependent.

GATEWAY=*address*

Where ***address*** is the IP address of the network router or gateway device (if any).

HOTPLUG=*answer*

Where ***answer*** is one of the following:

yes — This device should be activated when it is hot-plugged (this is the default option).

no — This device should not be activated when it is hot-plugged.

The **HOTPLUG=no** option can be used to prevent a channel bonding interface from being activated when a bonding kernel module is loaded.

HWADDR=*MAC-address*

Where ***MAC-address*** is the hardware address of the Ethernet device in the form **AA:BB:CC:DD:EE:FF**. This directive must be used in machines containing more than one NIC to ensure that the interfaces are assigned the correct device names regardless of the configured load order for each NIC's module. This directive should not be used in conjunction with **MACADDR**.

IPADDR=*address*

Where ***address*** is the IP address.

LINKDELAY=*time*

Where ***time*** is the number of seconds to wait for link negotiation before configuring the device.

MACADDR=*MAC-address*

Where ***MAC-address*** is the hardware address of the Ethernet device in the form **AA:BB:CC:DD:EE:FF**. This directive is used to assign a MAC address to an interface, overriding the one assigned to the physical NIC. This directive should not be used in conjunction with **HWADDR**.

MASTER=*bond-interface*

Where ***bond-interface*** is the channel bonding interface to which the Ethernet interface is linked.

This directive is used in conjunction with the **SLAVE** directive.

NETMASK=*mask*

Where ***mask*** is the netmask value.

NETWORK=*address*
Where ***address*** is the network address. This directive is deprecated, as the value is calculated automatically with ipcalc.

ONBOOT=*answer*
Where ***answer*** is one of the following:
- **yes** — This device should be activated at boot-time.
- **no** — This device should not be activated at boot-time.

PEERDNS=*answer*
Where ***answer*** is one of the following:
- **yes** — Modify **/etc/resolv.conf** if the DNS directive is set. If using DHCP, then **yes** is the default.
- **no** — Do not modify **/etc/resolv.conf**.

SLAVE=*answer*
Where ***answer*** is one of the following:
- **yes** — This device is controlled by the channel bonding interface specified in the **MASTER** directive.
- **no** — This device is not controlled by the channel bonding interface specified in the **MASTER** directive.

This directive is used in conjunction with the **MASTER** directive.

SRCADDR=*address*
Where ***address*** is the specified source IP address for outgoing packets.

USERCTL=*answer*
Where **answer** is one of the following:
- **yes** — Non-root users are allowed to control this device.
- **no** — Non-root users are not allowed to control this device.

Other common interface configuration files include the following:

ifcfg-lo: A local *loopback interface* is often used in testing, as well as being used in a variety of applications that require an IP address pointing back to the same system. Any data sent to the loopback device is immediately returned to the host's network layer.

21.2.2. Channel bonding interfaces

Red Hat Enterprise Linux allows administrators to bind multiple network interfaces together into a single channel using the **bonding** kernel module and a special network interface called a *channel bonding interface*. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

To create a channel bonding interface, create a file in the **/etc/sysconfig/network-scripts/** directory called **ifcfg-bond*N***, replacing ***N*** with the number for the interface, such as **0**.

The contents of the file can be identical to whatever type of interface is getting bonded, such as an Ethernet interface. The only difference is that the **DEVICE** directive must be **bond*N***, replacing ***N*** with the number for the interface.

The following is a sample channel bonding configuration file:

```
DEVICE=bond0
IPADDR=192.168.1.1
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
BONDING_OPTS="bonding parameters separated by spaces"
```

After the channel bonding interface is created, the network interfaces to be bound together must be configured by adding the **MASTER** and **SLAVE** directives to their configuration files. The configuration files for each of the channel-bonded interfaces can be nearly identical.

For example, if two Ethernet interfaces are being channel bonded, both eth0 and eth1 may look like the following example:

```
DEVICE=ethN
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
```

For a channel bonding interface to be valid, the kernel module must be loaded. To ensure that the module is loaded when the channel bonding interface is brought up, create a new file as root named ***bonding.conf*** in the **/etc/modprobe.d/** directory. Note that you can name this file anything you like as long as it ends with a **.conf** extension. Insert the following line in this new file:

```
alias  bondN  bonding
```

*** **Note**: Put all bonding module parameters in ifcfg-bondN files

Parameters for the bonding kernel module must be specified as a space-separated list in the **BONDING_OPTS="*bonding parameters*"** directive in the **ifcfg-bond*N*** interface file. Do not specify options for the bonding device in **/etc/modprobe.d/*bonding.conf***, or in the deprecated **/etc/modprobe.conf** file.

21.2.3. Alias and clone files

Alias interface configuration files, which are used to bind multiple addresses to a single interface, use the **ifcfg-*if-name:alias-value*** naming scheme.

For example, an **ifcfg-eth0:0** file could be configured to specify **DEVICE=eth0:0** and a static IP address of **10.0.0.2**, serving as an alias of an Ethernet interface already configured to receive its IP information via DHCP in **ifcfg-eth0**. Under this configuration, eth0 is bound to a dynamic IP address, but the same physical network card can receive requests via the fixed, 10.0.0.2 IP address.

***** Warning**: **Alias interfaces do not support DHCP**.

A clone interface configuration file should use the following naming convention: **ifcfg-*if-name-clone-name***. While an alias file allows multiple addresses for an existing interface, a clone file is used to specify additional options for an interface. For example, a standard DHCP Ethernet interface called eth0, may look similar to this:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Since the default value for the **USERCTL** directive is **no** if it is not specified, users cannot bring this interface up and down. To give users the ability to control the interface, create a clone by copying **ifcfg-eth0** to **ifcfg-eth0-*user*** and add the following line to **ifcfg-eth0-*user***:

```
USERCTL=yes
```

This way a user can bring up the eth0 interface using the **/sbin/ifup eth0-*user*** command because the configuration options from **ifcfg-eth0** and **ifcfg-eth0-*user*** are combined. While this is a very basic example, this method can be used with a variety of options and interfaces.

21.3. Interface control scripts

The interface control scripts activate and deactivate system interfaces. There are two primary interface control scripts that call on control scripts located in the **/etc/sysconfig/network-scripts/** directory: **/sbin/ifdown** and **/sbin/ifup**.
The **ifup** and **ifdown** interface scripts are symbolic links to scripts in the **/sbin/** directory. When either of these scripts are called, they require the value of the interface to be specified, such as:

```
# ifup  eth0
```

Two files used to perform a variety of network initialization tasks during the process of bringing up a network interface are **/etc/rc.d/init.d/functions** and **/etc/sysconfig/network-scripts/network-functions**.

After verifying that an interface has been specified and that the user executing the request is allowed to control the interface, the correct script brings the interface up or down. The following are common interface control scripts found within the **/etc/sysconfig/network-scripts/** directory:

ifup-aliases: Configures IP aliases from interface configuration files when more than one IP address is associated with an interface.

ifup-ippp and **ifdown-ippp**: Brings ISDN interfaces up and down.

ifup-ipv6 and **ifdown-ipv6**: Brings IPv6 interfaces up and down.

ifup-plip: Brings up a PLIP interface.

ifup-plusb: Brings up a USB interface for network connections.

ifup-post and **ifdown-post**: Contains commands to be executed after an interface is brought up or down.

ifup-ppp and **ifdown-ppp**: Brings a PPP interface up or down.

ifup-routes: Adds static routes for a device as its interface is brought up.

ifdown-sit and **ifup-sit**: Contains function calls related to bringing up and down an IPv6 tunnel within an IPv4 connection.

ifup-wireless: Brings up a wireless interface.

The easiest way to manipulate all network scripts simultaneously is to use the **/sbin/service** command on the network service (**/etc/rc.d/init.d/network**), as illustrated the following command:

```
# /sbin/service network <action>
```

Here, action can be either **start**, **stop**, or **restart**.

To view a list of configured devices and currently active network interfaces, use the following command:

```
# /sbin/service network status
```

21.4. Configuring static routes

If static routes are required, they can be configured for each interface. This can be useful if you have multiple interfaces in different subnets. Use the **route** command to display the IP routing table.

Static route configuration is stored in a **/etc/sysconfig/network-scripts/route-*interface*** file.

The **route-*interface*** file has two formats: IP command arguments and network/netmask directives.

IP Command Arguments Format

Define a default gateway on the first line. This is only required if the default gateway is not set via DHCP:

default via *X.X.X.X* dev *interface*

X.X.X.X is the IP address of the default gateway. The ***interface*** is the interface that is connected to, or can reach, the default gateway.

Define a static route. Each line is parsed as an individual route:

```
X.X.X.X/X via X.X.X.X dev interface
```

X.X.X.X/X is the network number and netmask for the static route. ***X.X.X.X*** and ***interface*** are the IP address and interface for the default gateway respectively. The ***X.X.X.X*** address does not have to be the default gateway IP address. In most cases, ***X.X.X.X*** will be an IP address in a different subnet, and **interface** will be the interface that is connected to, or can reach, that subnet. Add as many static routes as required.

The following is a sample **route-eth0** file using the IP command arguments format. The default gateway is 192.168.0.1, interface eth0. The two static routes are for the 10.10.10.0/24 and 172.16.1.0/24 networks:

```
default via 192.168.0.1 dev eth0
10.10.10.0/24 via 192.168.0.1 dev eth0
172.16.1.0/24 via 192.168.0.1 dev eth0
```

Static routes should only be configured for other subnets. The above example is not necessary, since packets going to the 10.10.10.0/24 and 172.16.1.0/24 networks will use the default gateway anyway. Below is an example of setting static routes to a different subnet, on a machine in a 192.168.0.0/24 subnet. The example machine has an eth0 interface in the 192.168.0.0/24 subnet, and an eth1 interface (10.10.10.1) in the 10.10.10.0/24 subnet:

```
10.10.10.0/24 via 10.10.10.1 dev eth1
```

Network/Netmask Directives Format

You can also use the network/netmask directives format for **route-*interface*** files. The following is a template for the network/netmask format, with instructions following afterwards:

```
ADDRESS0=X.X.X.X NETMASK0=X.X.X.X GATEWAY0=X.X.X.X
```

ADDRESS0=*X.X.X.X* is the network number for the static route.
NETMASK0=*X.X.X.X* is the netmask for the network number defined with **ADDRESS0=*X.X.X.X***
GATEWAY0=*X.X.X.X* is the default gateway, or an IP address that can be used to reach **ADDRESS0=*X.X.X.X***

The following is a sample **route-eth0** file using the network/netmask directives format. The default gateway is 192.168.0.1, interface eth0. The two static routes are for the 10.10.10.0/24 and 172.16.1.0/24 networks. However, as mentioned before, this example is not necessary as the 10.10.10.0/24 and 172.16.1.0/24 networks would use the default gateway anyway:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
ADDRESS1=172.16.1.0
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.1
```

Subsequent static routes must be numbered sequentially, and must not skip any values. For example, **ADDRESS0**, **ADDRESS1**, **ADDRESS2**, and so on.

Below is an example of setting static routes to a different subnet, on a machine in the 192.168.0.0/24 subnet. The example machine has an eth0 interface in the 192.168.0.0/24 subnet, and an eth1 interface (10.10.10.1) in the 10.10.10.0/24 subnet:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=10.10.10.1
```

Note that if DHCP is used, it can assign these settings automatically.

21.5. Network function files

Red Hat Enterprise Linux makes use of several files that contain important common functions used to bring interfaces up and down. Rather than forcing each interface control file to contain these functions, they are grouped together in a few files that are called upon when necessary.

The **/etc/sysconfig/network-scripts/network-functions** file contains the most commonly used IPv4 functions, which are useful to many interface control scripts. These functions include contacting running programs that have requested information about changes in the status of an interface, setting hostnames, finding a gateway device, verifying whether or not a particular device is down, and adding a default route.

As the functions required for IPv6 interfaces are different from IPv4 interfaces, a **/etc/sysconfig/network-scripts/network-functions-ipv6** file exists specifically to hold this information. The functions in this file configure and delete static IPv6 routes, create and remove tunnels, add and remove IPv6 addresses to an interface, and test for the existence of an IPv6 address on an interface.

21.6. Additional resources

/usr/share/doc/initscripts-version/sysconfig.txt
A guide to available options for network configuration files, including IPv6 options.

/usr/share/doc/iproute-version/ip-cref.ps
This file contains a wealth of information about the ip command, which can be used to manipulate routing tables, among other things. Use the **ggv** or **kghostview** application to view this file.

21.7. Persistent network routes

To set static routes, use the **/etc/sysconfig/network-scripts/route-*interface*** files which read at network service initialization time. For example to add static route for eth0, create a file **/etc/sysconfig/network-scripts/route-eth0** and add the routes as explained below.

There are two possible formats for this file. The first is with **ip** command arguments and the second is with network/netmask directives.

Format 1:

For **ip** commands, the **ifup-route** script supplies **ip route add** and the contents of the file are all parameters necessary to set up the route. For example, to set up a default route, the file would contain the following:

```
default via X.X.X.X dev bond0
10.10.10.0/24 via X.X.X.X dev bond0
```

In the above, ***X.X.X.X*** is the gateway IP address. The second line adds another static route where ***X.X.X.X*** is the gateway IP address. Multiple lines will be parsed as individual routes.

Format 2:

The alternative format is as follows:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=X.X.X.X
```

In the above example, **X.X.X.X** is the gateway IP address. Subsequent entries must be numbered sequentially (for example ADDRESS1=, NETMASK1=, GATEWAY0=). Note, that multiple entries must be sequentially numbered and must not skip a value (0 must be followed by 1, not a number greater than 1).

21.8. Adding and deleting static routes from IP routing table

The **ip** command is part of the **iproute** RPM and is more powerful than other utilities available to configure routes, network addresses, and other networking settings from the command line. The **iproute** RPM is installed by default on Red Hat Enterprise Linux systems.

To add or delete a route, there are four things you need to know:

- The network/subnet you wish to reach.
- The netmask of the subnet, in CIDR (Classless Internet Domain Representation), /xx notation.
- The interface you wish this route to be added to, i.e., which interface to use to reach the subnet.
- The address of a router in a local subnet which can reach this network. This is often called a gateway address, but this should not to be confused with a default gateway. (Default gateways are used when no other matching route exists for a given destination).

Example

Assume, for this example, that you have a system with two network interfaces in it, **eth0** and **eth1**. In our theoretical example, eth0 is connected to 192.168.0.0/24, and eth1 is connected to 192.168.1.0/24. However, there is another network, 10.20.30.0/24, that is reachable via a router on the 192.168.1.0/24 network, 192.168.1.254.

This is the current routing table for our example system:

```
# ip route show
        192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.7
        192.168.1.0/24 dev eth1 proto kernel scope link src 192.168.1.7
        default via 192.168.0.1 dev eth0
```

The following ip route command would add the desired route to the kernel routing table:

```
# ip route add 10.20.30.0/24 via 192.168.1.254 dev eth1
```

and can be confirmed by viewing the routing table again:

```
# ip route show
        10.20.30.0/24 via 192.168.1.254 dev eth1
        192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.7
        192.168.1.0/24 dev eth1 proto kernel scope link src 192.168.1.7
        default via 192.168.0.1 dev eth0
```

For further information regarding the ip command and other utilities in the **iproute** RPM, please refer to the **iproute** documentation. On Red Hat Enterprise Linux systems with the **iproute** RPM installed, a copy of the manual should be available (in Postscript format) at **/usr/share/doc/iproute-<version>/ip-cref.ps**

22. Network troubleshooting

NOT COVERED IN THIS GUIDE

23. Network services

23.1. Using the 'chkconfig' utility

The **chkconfig** utility is a command line application to configure which services are to be started in selected runlevels. It also allows you to list all available services along with their current setting. Note that with the exception of listing, you must have superuser privileges to use this command.

23.1.1. Listing the services

To display a list of system services (i.e., both the services from the **/etc/rc.d/init.d/** directory, and the services controlled by **xinetd**), either type **chkconfig --list**, or use **chkconfig** with no additional arguments. You should be presented with an output similar to this:

```
# chkconfig --list
NetworkManager  0:off   1:off   2:on    3:on    4:on    5:on    6:off
abrtd           0:off   1:off   2:off   3:on    4:off   5:on    6:off
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
anamon          0:off   1:off   2:off   3:off   4:off   5:off   6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
avahi-daemon    0:off   1:off   2:off   3:on    4:on    5:on    6:off
... several lines omitted ...
wpa_supplicant  0:off   1:off   2:off   3:off   4:off   5:off   6:off

xinetd based services:
        chargen-dgram:  off
        chargen-stream: off
        cvs:            off
        daytime-dgram:  off
        daytime-stream: off
        discard-dgram:  off
... several lines omitted ...
        time-stream:    off
```

As you can see, each line consists of the name of the service followed by its status (on or off) for each of the seven numbered runlevels. For example, in the listing above, ***NetworkManager*** is enabled for runlevel 2, 3, 4, and 5, while ***abrtd*** runs in runlevel 3 and 5. The ***xinetd*** based services are listed at the end, being either ***on***, or ***off***.

To display the current settings for selected service only, use **chkconfig --list** followed by the name of the service.

You can also use **chkconfig --list *service*** to display the status of a service that is managed by **xinetd**. In that case, the output will simply contain the information whether the service is enabled or disabled.

23.1.2. Enabling/running a service

To enable the service for runlevels 2, 3, 4, and 5 at the same time, type **chkconfig *service* on**. For instance:

```
# chkconfig httpd on
```

To enable the service for certain runlevels only, add the **--level** option followed by the string of numbers from 0 to 6 representing each runlevel in which you want the service to run. For example, to enable the **abrtd** for runlevels 3 and 5, type:

```
# chkconfig abrtd on --level 35
```

The service will be started the next time you enter one of these runlevels. If you need to start the service immediately, use the **service** command:

```
# service abrtd start
Starting abrtd:                                    [  OK  ]
```

To enable the service that is managed by **xinetd**, use **chkconfig** ***service*** on only, as the **--level** option is not allowed:

```
# chkconfig rsync on
```

If the **xinetd** daemon is running, the service is immediately enabled without having to restart the daemon manually.

23.1.3. Disabling/stopping a service

To disable the service for runlevels 2, 3, 4, and 5 at the same time, type **chkconfig** ***service*** **off**. For instance:

```
# chkconfig httpd off
```

To disable the service for certain runlevels only, add the **--level** option followed by the string of numbers from 0 to 6 representing each runlevel in which you do not want the service to run. For example, to disable the **abrtd** for runlevels 2 and 4, type:

```
# chkconfig abrtd off --level 24
```

The service will be stopped the next time you enter one of these runlevels. If you need to stop the service immediately, use the **service** command:

```
# service httpd stop
Stopping httpd:                                  [  OK  ]
```

To disable the service that is managed by **xinetd**, use **chkconfig** ***service*** **off** only, as the **--level** option is not allowed:

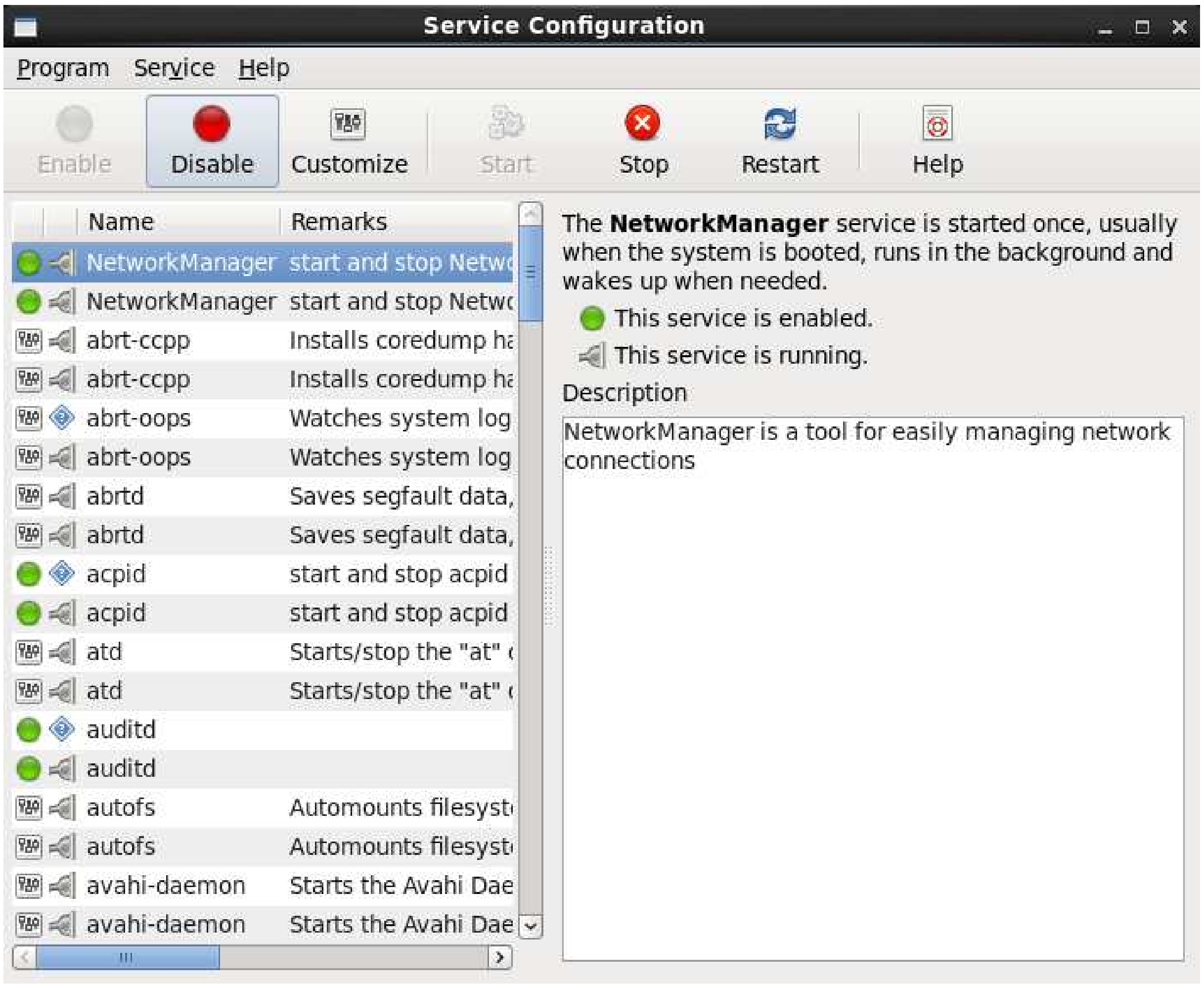
```
# chkconfig rsync off
```

If the **xinetd** daemon is running, the service is immediately disabled without having to restart the daemon manually.

23.2. Using the Service Configuration Utility

The **Service Configuration** utility is a graphical application developed by Red Hat to configure which services are started in a particular runlevel, as well as to start, stop, and restart them from the menu.

To start the utility, select **System** → **Administration** → **Services** from the panel, or type the command **system-config-services** at a shell prompt (e.g., xterm or GNOME Terminal).



		Name	Remarks
🟢	◁	NetworkManager	start and stop Netw
🟢	◁	NetworkManager	start and stop Netw
▦	◁	abrt-ccpp	Installs coredump ha
▦	◁	abrt-ccpp	Installs coredump ha
▦	◈	abrt-oops	Watches system log
▦	◁	abrt-oops	Watches system log
▦	◁	abrtd	Saves segfault data,
▦	◁	abrtd	Saves segfault data,
🟢	◈	acpid	start and stop acpid
🟢	◁	acpid	start and stop acpid
▦	◁	atd	Starts/stop the "at"
▦	◁	atd	Starts/stop the "at"
🟢	◈	auditd	
🟢	◁	auditd	
▦	◁	autofs	Automounts filesyst
▦	◁	autofs	Automounts filesyst
▦	◁	avahi-daemon	Starts the Avahi Dae
▦	◁	avahi-daemon	Starts the Avahi Dae

The **NetworkManager** service is started once, usually when the system is booted, runs in the background and wakes up when needed.

🟢 This service is enabled.

◁ This service is running.

Description

NetworkManager is a tool for easily managing network connections

Icon	Description
🟢	The service is enabled.
🔴	The service is disabled.
▦	The service is enabled for selected runlevels only.
◁	The service is running.
⊂	The service is stopped.
⚠	There is something wrong with the service.
◈	The status of the service is unknown.

23.3. Controlling access to services

There are several different methods for managing access to system services. Decide which method of management to use based on the service, your system's configuration, and your level of Linux expertise.

The easiest way to deny access to a service is to turn it off. Both the services managed by **xinetd** and the services in the **/etc/rc.d/init.d** hierarchy (also known as SysV services) can be configured to start or stop using three different applications:

- **Services Configuration Tool** — a graphical application that displays a description of each service, displays whether each service is started at boot time (for runlevels 3, 4, and 5), and allows services to be started, stopped, and restarted.
- **ntsysv** — a text-based application that allows you to configure which services are started at boot time for each runlevel. **Non-xinetd** services cannot be started, stopped, or restarted using this program.
- **chkconfig** — a command line utility that allows you to turn services on and off for the different runlevels. **Non-xinetd** services cannot be started, stopped, or restarted using this utility.

You may find that these tools are easier to use than the alternatives — editing the numerous symbolic links located in the directories below **/etc/rc.d** by hand or editing the **xinetd** configuration files in **/etc/xinetd.d**

Another way to manage access to system services is by using **iptables** to configure an IP firewall.

Alternatively, if you are looking for a utility to set general access rules for your home machine, and/or if you are new to Linux, try the **Security Level Configuration Tool (system-config-securitylevel)**, which allows you to select the security level for your system, similar to the **Firewall Configuration** screen in the installation program.

23.3.1. TCP wrappers

Many UNIX system administrators are accustomed to using TCP wrappers to manage access to certain network services. Any network services managed by **xinetd** (as well as any program with built-in support for **libwrap**) can use TCP wrappers to manage access. **xinetd** can use the **/etc/hosts.allow** and **/etc/hosts.deny** files to configure access to system services. As the names imply, **hosts.allow** contains a list of rules that allow clients to access the network services controlled by **xinetd**, and **hosts.deny** contains rules to deny access. The **hosts.allow** file takes precedence over the **hosts.deny** file. Permissions to grant or deny access can be based on individual IP address (or hostnames) or on a pattern of clients.

xinetd

To control access to Internet services, use **xinetd**, which is a secure replacement for **inetd**. The **xinetd** daemon conserves system resources, provides access control and logging, and can be used to start special-purpose servers. **xinetd** can be used to provide access only to particular hosts, to deny access to particular hosts, to provide access to a service at certain times, to limit the rate of incoming connections and/or the load created by connections, and more

xinetd runs constantly and listens on all ports for the services it manages. When a connection request arrives for one of its managed services, **xinetd** starts up the appropriate server for that service.

The configuration file for **xinetd** is **/etc/xinetd.conf**, but the file only contains a few defaults and an instruction to include the **/etc/xinetd.d/**directory. To enable or disable an **xinetd** service, edit its configuration file in the **/etc/xinetd.d/**directory. If the **disable** attribute is set to **yes**, the service is disabled. If the **disable** attribute is set to **no**, the service is enabled. You can edit any of the **xinetd** configuration files or change its enabled status using the **Services Configuration Tool**, **ntsysv**, or **chkconfig**. For a list of network services controlled by **xinetd**, review the contents of the **/etc/xinetd.d**/ directory

23.3.2. Services Configuration Tool

The **Services Configuration Tool** is a graphical application developed by Red Hat to configure which **SysV** services in the **/etc/rc.d/init.d** directory are started at boot time (for runlevels 3, 4, and 5) and which **xinetd** services are enabled. It also allows you to start, stop, and restart SysV services as well as restart **xinetd**.

To start the **Services Configuration Tool** from the desktop, go to the **Main Menu Button** (on the Panel) => **System Settings** => **Server Settings** => **Services** or type the command **system-config-services** at a shell prompt (for example, in an **XTerm** or a **GNOME terminal**).

23.3.3. ntsysv

The **ntsysv** utility provides a simple interface for activating or deactivating services. You can use **ntsysv** to turn an **xinetd**-managed service on or off. You can also use **ntsysv** to configure runlevels. By default, only the current runlevel is configured. To configure a different runlevel, specify one or more runlevels with the **--level** option. For example, the command **ntsysv --level 345** configures runlevels 3, 4, and 5.

***** Warning**: Services managed by **xinetd** are immediately affected by **ntsysv**. For all other services, changes do not take effect immediately. You must stop or start the individual service with the command **service *daemon* stop**. In the previous example, replace ***daemon*** with the name of the service you want to stop; for example, **httpd**. Replace **stop** with **start** or **restart** to start or restart the service.

23.3.4. chkconfig

The **chkconfig** command can also be used to activate and deactivate services. The **chkconfig --list** command displays a list of system services and whether they are started (**on**) or stopped (**off**) in runlevels 0-6.

If the **chkconfig --list** command is used to query a service managed by **xinetd**, it displays whether the **xinetd** service is enabled (**on**) or disabled (**off**). For example, the command **chkconfig --list finger** returns the following output:

```
finger      on
```

As shown, **finger** is enabled as an **xinetd** service. If **xinetd** is running, **finger** is enabled.

If you use **chkconfig --list** to query a service in **/etc/rc.d**, service's settings for each runlevel are displayed. For example, the command **chkconfig --list httpd** returns the following output:

```
httpd       0:off   1:off   2:on   3:on   4:on   5:on   6:off
```

chkconfig can also be used to configure a service to be started (or not) in a specific runlevel. For example, to turn **nscd** off in runlevels 3, 4, and 5, use the following command:

```
# chkconfig --level 345 nscd off
```

***** Warning**: Services managed by **xinetd** are immediately affected by **chkconfig**. For example, if **xinetd** is running, **finger** is disabled, and the command **chkconfig finger on** is executed, **finger** is immediately enabled without having to restart **xinetd** manually. Changes for other services do not take effect immediately after using **chkconfig**. You must stop or start the individual service with the command **service *daemon* [stop|start|restart]**.

23.4. Listing services running

23.4.1. Listing network services running or listening

```
# netstat -atup
```

```
# netstat -atup | grep LISTEN
```

Where,

-a : Display all listening and non-listening sockets.
-t : Select all TCP services
-u : Select all UDP services
-p : Display the PID and name of the program to which each socket belongs

23.4.2. Listing running services

```
# service --status-all
```

24. Autofs

The **automount** utility can mount and unmount NFS file systems automatically (on-demand mounting), therefore saving system resources. It can be used to mount other file systems including AFS, SMBFS, CIFS, and local file systems.

***** Note**: The **nfs-utils** package is now a part of both the 'NFS file server' and the 'Network File System Client' groups. As such it is no longer installed by default with the Base group. Ensure that nfs-utils is installed on the system first before attempting to automount an NFS share. Note that autofs is also part of the 'Network File System Client' group.

autofs uses **/etc/auto.master** (master map) as its default primary configuration file. This can be changed to use another supported network source and name using the **autofs** configuration (in **/etc/sysconfig/autofs**) in conjunction with the Name Service Switch (NSS) mechanism. An instance of the **autofs** version 4 daemon was run for each mount point configured in the master map and so it could be run manually from the command line for any given mount point. This is not possible with **autofs** version 5, because it uses a single daemon to manage all configured mount points; as such, all automounts must be configured in the master map. This is in line with the usual requirements of other industry standard automounters. Mount point, hostname, exported directory, and options can all be specified in a set of files (or other supported network sources) rather than configuring them manually for each host.

24.1. Autofs configuration

The primary configuration file for the automounter is **/etc/auto.master**, also referred to as the master map. The master map lists **autofs**-controlled mount points on the system, and their corresponding configuration files or network sources known as automount maps. The format of the master map is as follows:

mount-point map-name options

mount-point: The autofs mount point e.g /home.
map-name: The name of a map source which contains a list of mount points, and the file system location from which those mount points should be mounted. The syntax for a map entry is described below.
options: If supplied, these will apply to all entries in the given map provided they don't themselves have options specified. This behaviour is different from autofs version 4 where options where cumulative. This has been changed to implement mixed environment compatibility.

* Sample line from /etc/auto.master file
/home /etc/auto.home

The general format of maps is similar to the master map, however the "options" appear between the mount point and the location instead of at the end of the entry as in the master map:

mount-point [options] location

mount-point: This refers to the autofs mount point. This can be a single directory name for an indirect mount or the full path of the mount point for direct mounts. Each direct and indirect map entry key (mount-point above) may be followed by a space separated list of offset directories (sub directory names each beginning with a "/") making them what is known as a multi-mount entry.

options: Whenever supplied, these are the mount options for the map entries that do not specify their own options.

location: This refers to the file system location such as a local file system path (preceded with the Sun map format escape character ":" for map names beginning with "/"), an NFS file system or other valid file system location.

The following is a sample of contents from a map file (i.e. **/etc/auto.home**):

```
payroll  -fstype=nfs  personnel:/dev/hda3
sales  -fstype=ext3  personnel:/dev/hda4
```

The first column in a map file indicates the **autofs** mount point (**sales** and **payroll** from the server called **personnel**). The second column indicates the options for the **autofs** mount while the third column indicates the source of the mount. Following the above configuration, the autofs mount points will be **/home/payroll** and **/home/sales**. The **-fstype=** option is often omitted and is generally not needed for correct operation.
The automounter will create the directories if they do not exist. If the directories exist before the automounter was started, the automounter will not remove them when it exits. You can start or restart the automount daemon by issuing either of the following two commands:

```
# service autofs start
# service autofs restart
```

Using the above configuration, if a process requires access to an **autofs** unmounted directory such as **/home/payroll/2006/July.sxc**, the automount daemon automatically mounts the directory. If a timeout is specified, the directory will automatically be unmounted if the directory is not accessed for the timeout period.
You can view the status of the automount daemon by issuing the following command:

```
# service autofs status
```

24.2. Overriding or augmenting site configuration files

It can be useful to override site defaults for a specific mount point on a client system. For example, consider the following conditions:

- Automounter maps are stored in NIS and the **/etc/nsswitch.conf** file has the following directive:

```
automount:    files nis
```

- The **auto.master** file contains the following:

```
+auto.master
```

- The NIS **auto.master** map file contains the following:

```
/home auto.home
```

- The NIS auto.home map contains the following:

```
beth       fileserver.example.com:/export/home/beth
joe       fileserver.example.com:/export/home/joe
*      fileserver.example.com:/export/home/&
```

- The file map **/etc/auto.home** does not exist.

Given these conditions, let's assume that the client system needs to override the NIS map **auto.home** and mount home directories from a different server. In this case, the client will need to use the following **/etc/auto.master** map:

```
/home -/etc/auto.home
+auto.master
```

And the **/etc/auto.home** map contains the entry:

```
*    labserver.example.com:/export/home/&
```

Because the automounter only processes the first occurrence of a mount point, **/home** will contain the contents of **/etc/auto.home** instead of the NIS **auto.home** map.

Alternatively, if you just want to augment the site-wide **auto.home** map with a few entries, create a **/etc/auto.home** file map, and in it put your new entries and at the end, include the NIS **auto.home** map. Then the **/etc/auto.home** file map might look similar to:

```
mydir someserver:/export/mydir
```

```
+auto.home
```

Given the NIS **auto.home** map listed above, **ls /home** would now output:

```
beth  joe  mydir
```

This last example works as expected because **autofs** knows not to include the contents of a file map of the same name as the one it is reading. As such, **autofs** moves on to the next map source in the **nsswitch** configuration.

24.3. Using LDAP to store automounter maps

LDAP client libraries must be installed on all systems configured to retrieve automounter maps from LDAP. On Red Hat Enterprise Linux, the **openldap** package should be installed automatically as a dependency of the **automounter**. To configure LDAP access, modify **/etc/openldap/ldap.conf**. Ensure that BASE, URI, and schema are set appropriately for your site.

The most recently established schema for storing automount maps in LDAP is described by **rfc2307bis**. To use this schema it is necessary to set it in the **autofs** configuration (**/etc/sysconfig/autofs**) by removing the comment characters from the schema definition. For example:

Setting autofs configuration

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

Ensure that these are the only schema entries not commented in the configuration. Note that the **automountKey** replaces the **cn** attribute in the **rfc2307bis** schema. An **LDIF** of a sample configuration is described below:

LDF configuration

```
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (&(objectclass=automountMap)(automountMapName=auto.master))
# requesting: ALL
#

# auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: top
objectClass: automountMap
```

```
automountMapName: auto.master

# extended LDIF
#
# LDAPv3
# base <automountMapName=auto.master,dc=example,dc=com> with scope subtree
# filter: (objectclass=automount)
# requesting: ALL
#

# /home, auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: automount
cn: /home

automountKey: /home
automountInformation: auto.home

# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (&(objectclass=automountMap)(automountMapName=auto.home))
# requesting: ALL
#

# auto.home, example.com
dn: automountMapName=auto.home,dc=example,dc=com
objectClass: automountMap
automountMapName: auto.home

# extended LDIF
#
# LDAPv3
# base <automountMapName=auto.home,dc=example,dc=com> with scope subtree
# filter: (objectclass=automount)
# requesting: ALL
#

# foo, auto.home, example.com
dn: automountKey=foo,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: foo
automountInformation: filer.example.com:/export/foo

# /, auto.home, example.com
```

```
dn: automountKey=/,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: /
automountInformation: filer.example.com:/export/&
```

25. NFS & Samba shares

25.1. NFS

Currently, there are three versions of NFS. NFS version 2 (**NFSv2**) is older and is widely supported. NFS version 3 (**NFSv3**) supports safe asynchronous writes and a more robust error handling than NFSv2; it also **supports 64-bit file sizes** and offsets, allowing clients to access more than 2Gb of file data. NFS version 4 (**NFSv4**) **works through firewalls and on the Internet, no longer requires an rpcbind** service, supports ACLs, and utilizes stateful operations. When mounting a file system via NFS, NFSv4 is used by default, if the server supports it.

TCP is the default transport protocol for NFS version 2 and 3 under Fedora. UDP can be used for compatibility purposes as needed, but is not recommended for wide usage. NFSv4 requires TCP.
All the RPC/NFS daemon have a '**-p**' command line option that can set the port, making firewall configuration easier.

The mounting and locking protocols have been incorporated into the NFSv4 protocol. The server also listens on the well-known TCP port 2049. As such, NFSv4 does not need to interact with **rpcbind**, **rpc.lockd**, and **rpc.statd** daemons. The **rpc.mountd** daemon is still required on the NFS server so set up the exports, but is not involved in any over-the-wire operations.

In order for NFS to work with a default installation of Fedora with a firewall enabled, configure IPTables with the default TCP port 2049. Without proper IPTables configuration, NFS will not function properly.
The NFS initialization script and **rpc.nfsd** process now allow binding to any specified port during system start up. However, this can be error-prone if the port is unavailable, or if it conflicts with another daemon.

25.2. NFS required services

nfs: '**service nfs start**' starts the NFS server and the appropriate RPC processes to service requests for shared NFS file systems.

nfslock: '**service nfslock start**' activates a mandatory service that starts the appropriate RPC processes which allow NFS clients to lock files on the server.

rpcbind: accepts port reservations from local RPC services. These ports are then made available (or advertised) so the corresponding remote RPC services can access them. rpcbind responds to requests for RPC services and sets up connections to the requested RPC service. **This is not used with NFSv4**.

The following RPC processes facilitate NFS services:

rpc.mountd: This process receives mount requests from NFS clients and verifies that the requested file system is currently exported. This process is started automatically by the nfs service and does not require user configuration.

rpc.nfsd: Allows explicit NFS versions and protocols the server advertises to be defined. It works with the Linux kernel to meet the dynamic demands of NFS clients, such as providing server threads each time an NFS client connects. This process corresponds to the nfs service.

rpc.lockd: Allows NFS clients to lock files on the server. If rpc.lockd is not started, file locking will fail. rpc.lockd implements the Network Lock Manager (NLM) protocol. This process corresponds to the nfslock service. This is not used with NFSv4.

rpc.statd: This process implements the Network Status Monitor (NSM) RPC protocol, which notifies NFS clients when an NFS server is restarted without being gracefully brought down. rpc.statd is started automatically by the nfslock service, and does not require user configuration. This is not used with NFSv4.

rpc.rquotad: This process provides user quota information for remote users. rpc.rquotad is started automatically by the nfs service and does not require user configuration.

rpc.idmapd: Provides NFSv4 client and server upcalls, which map between on-the-wire NFSv4 names (which are strings in the form of *user@domain*) and local UIDs and GIDs. For idmapd to function with NFSv4, the /etc/idmapd.conf must be configured. This service is required for use with NFSv4, although not when all hosts share the same DNS domain name.

25.3. Starting and stopping NFS

To run an NFS server, the **rpcbind** service must be running. To verify that **rpcbind** is active, use the following command:

```
# service rpcbind status
```

If the **rpcbind** service is running, then the **nfs** service can be started. To start an NFS server, use the following command as root:

```
# service nfs start
```

*** **Note**: **nfslock** must also be started for both the NFS client and server to work properly.
To start NFS locking, use the following command:

```
# service nfslock start
```

If NFS is set to start at boot, ensure that **nfslock** also starts by running **chkconfig --list nfslock**. If **nfslock** is not set to **on**, this implies that you will need to manually run the **service nfslock start** each time the computer starts.
To set **nfslock** to automatically start on boot, use **chkconfig nfslock on**.
nfslock is only needed for NFSv2 and NFSv3.

To stop the server, use:

```
# service nfs stop
```

The **restart** option is a shorthand way of stopping and then starting NFS. This is the most efficient way to make configuration changes take effect after editing the configuration file for NFS. To restart the server, as root, type:

```
# service nfs restart
```

The **condrestart** (conditional restart) option only starts **nfs** if it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running. To conditionally restart the server, as root, type:

```
# service nfs condrestart
```

To reload the NFS server configuration file without restarting the service, as root, type:

```
# service nfs reload
```

25.4. NFS Server configuration

25.4.1. The /etc/exports configuration file

The **/etc/exports** file controls which file systems are exported to remote hosts and specifies options. It follows the following syntax rules:

- Blank lines are ignored.
- To add a comment, start a line with the hash mark (**#**).
- You can wrap long lines with a backslash (****).
- Each exported file system should be on its own individual line.
- Any lists of authorized hosts placed after an exported file system must be separated by space characters.
- Options for each of the hosts must be placed in parentheses directly after the host identifier, without any spaces separating the host and the first parenthesis.

Each entry for an exported file system has the following structure:

export host(options)

export: The directory being exported

host: The host or network to which the export is being shared

options: The options to be used for host

Multiple hosts, along with specific options for each host, can be specified. To do so, list them on the same line as a space-delimited list, with each hostname followed by its respective options (in parentheses), as in:

export host1(options1) host2(options2) host3(options3)

The host(s) can be in the following forms:

Single machine: A fully-qualified domain name (that can be resolved by the server), hostname (that can be resolved by the server), or an IP address.

Series of machines specified via wildcards: Use the ***** or **?** character to specify a string match. Wildcards are not to be used with IP addresses; however, they may accidentally work if reverse DNS lookups fail. When specifying wildcards in fully qualified domain names, dots (.) are not included in the wildcard.

IP networks: Use ***a.b.c.d/z***, where **a.b.c.d** is the network and **z** is the number of bits in the netmask (for example 192.168.0.0/24). Another acceptable format is ***a.b.c.d/netmask***, where ***a.b.c.d*** is the network and ***netmask*** is the netmask (for example, 192.168.100.8/255.255.255.0).

Netgroups
Use the format **@*group-name***, where ***group-name*** is the NIS netgroup name.

In its simplest form, the **/etc/exports** file only specifies the exported directory and the hosts permitted to access it, as in the following example:

/exported/directory bob.example.com

Here, **bob.example.com** can mount **/exported/directory/** from the NFS server. Because no options are specified in this example, NFS will use default settings, which are:

ro: The exported file system is read-only. Remote hosts cannot change the data shared on the file system. To allow hosts to make changes to the file system (i.e. read/write), specify the **rw** option.

sync: The NFS server will not reply to requests before changes made by previous requests are written to disk. To enable asynchronous writes instead, specify the option **async**.

wdelay: The NFS server will delay writing to the disk if it suspects another write request is imminent. This can improve performance as it reduces the number of times the disk must be accesses by separate write commands, thereby reducing write overhead. To disable this, specify the **no_wdelay**; note that **no_wdelay** is only available if the default **sync** option is also specified.

root_squash: This prevents root users connected remotely from having root privileges; instead, the NFS server will assign them the user ID **nfsnobody.** This effectively "squashes" the power of the remote root user to the lowest local user, preventing possible unauthorized writes on the remote server. To disable root squashing, specify **no_root_squash**.
To squash every remote user (including root), use **all_squash**. To specify the user and group IDs that the NFS server should assign to remote users from a particular host, use the **anonuid** and **anongid** options, respectively, as in:

***export host*(anonuid=*uid*,anongid=*gid*)**

Here, ***uid*** and ***gid*** are user ID number and group ID number, respectively. The **anonuid** and **anongid** options allow you to create a special user/group account for remote NFS users to share.
Important

***** Note**: By default, access control lists (ACLs) are supported by NFS. To disable this feature, specify the **no_acl** option when exporting the file system.

25.4.2. The exportfs command

Every file system being exported to remote users via NFS, as well as the access level for those file systems, are listed in the **/etc/exports** file. When the **nfs** service starts, the **/usr/sbin/exportfs** command launches and reads this file, passes control to **rpc.mountd** (if NFSv2 or NFSv3) for the actual mounting process, then to **rpc.nfsd** where the file systems are then available to remote users.

When issued manually, the **/usr/sbin/exportfs** command allows the root user to selectively export or unexport directories without restarting the NFS service. When given the proper options, the **/usr/sbin/exportfs** command writes the exported file systems to **/var/lib/nfs/xtab**. Since **rpc.mountd** refers to the **xtab** file when deciding access privileges to a file system, changes to the list of exported file systems take effect immediately.

The following is a list of commonly-used options available for **/usr/sbin/exportfs**:

-r: Causes all directories listed in **/etc/exports** to be exported by constructing a new export list in **/etc/lib/nfs/xtab**. This option effectively refreshes the export list with any changes made to **/etc/exports**.

-a: Causes all directories to be exported or unexported, depending on what other options are passed to **/usr/sbin/exportfs**. If no other options are specified, **/usr/sbin/exportfs** exports all file systems specified in /etc/exports.

-o *file-systems*: Specifies directories to be exported that are not listed in /**etc/exports**. Replace ***file-systems*** with additional file systems to be exported. These file systems must be formatted in the same way they are specified in **/etc/exports**. This option is often used to test an exported file system before adding it permanently to the list of file systems to be exported.

-i: Ignores **/etc/exports**; only options given from the command line are used to define exported file systems.

-u: Unexports all shared directories. The command **/usr/sbin/exportfs -ua** suspends NFS file sharing while keeping all NFS daemons up. To re-enable NFS sharing, use **exportfs -r**.

-v: Verbose operation, where the file systems being exported or unexported are displayed in greater detail when the **exportfs** command is executed.

If no options are passed to the **exportfs** command, it displays a list of currently exported file systems.

25.4.3. Running NFS behind a firewall

NFS requires **rpcbind**, which dynamically assigns ports for RPC services and can cause problems for configuring firewall rules. To allow clients to access NFS shares behind a firewall, edit the **/etc/sysconfig/nfs** configuration file to control which ports the required RPC services run on.

The **/etc/sysconfig/nfs** may not exist by default on all systems. If it does not exist, create it and add the following variables, replacing ***port*** with an unused port number (alternatively, if the file exists, un-comment and change the default entries as required):

MOUNTD_PORT=port
Controls which TCP and UDP port **mountd** (**rpc.mountd**) uses.

STATD_PORT=port
Controls which TCP and UDP port status (**rpc.statd**) uses.
LOCKD_TCPPORT=port
Controls which TCP port **nlockmgr** (**rpc.lockd**) uses.
LOCKD_UDPPORT=port
Controls which UDP port **nlockmgr** (**rpc.lockd**) uses.

If NFS fails to start, check **/var/log/messages**. Normally, NFS will fail to start if you specify a port number that is already in use. After editing **/etc/sysconfig/nfs**, restart the NFS service using **service nfs restart**. Run the **rpcinfo -p** command to confirm the changes.

To configure a firewall to allow NFS, perform the following steps:

- Allow TCP port 2049 for NFS.
- Allow TCP port 111 (**rpcbind/sunrpc**).
- Allow the TCP and UDP port specified **with MOUNTD_PORT="*port*"**
- Allow the TCP and UDP port specified **with STATD_PORT="*port*"**
- Allow the TCP port specified with **LOCKD_TCPPORT="*port*"**
- Allow the UDP port specified with **LOCKD_UDPPORT="*port*"**

25.5. NFS client configuration

The mount command mounts NFS shares on the client side. Format:

```
# mount -t nfs -o <options> host:/remote/export /local/directory
```

25.5.1. Mounting NFS filesystems using /etc/fstab

The general syntax for the line in /etc/fstab is as follows:

```
server:/remote/export   /local/directory   nfs   options   0   0
```

25.6. Common NFS mount options

Beyond mounting a file system via NFS on a remote host, you can also specify other options at mount time to make the mounted share easier to use. These options can be used with manual **mount** commands, **/etc/fstab** settings, and **autofs**.

The following are options commonly used for NFS mounts:

intr: Allows NFS requests to be interrupted if the server goes down or cannot be reached.

lookupcache=*mode*: Specifies how the kernel should manage its cache of directory entries for a given mount point. Valid arguments for ***mode*** are **all**, **none**, or **pos/positive**.

nfsvers=*version*: Specifies which version of the NFS protocol to use, where ***version*** is 2, 3, or 4.. This is useful for hosts that run multiple NFS servers. If no version is specified, NFS uses the highest version supported by the kernel and **mount** command.
The option ***vers*** is identical to ***nfsvers***, and is included in this release for compatibility reasons.

noacl: Turns off all ACL processing. This may be needed when interfacing with older versions of Fedora, Red Hat Enterprise Linux, Red Hat Linux, or Solaris, since the most recent ACL technology is not compatible with older systems.

nolock: Disables file locking. This setting is occasionally required when connecting to older NFS servers.

noexec: Prevents execution of binaries on mounted file systems. This is useful if the system is mounting a non-Linux file system containing incompatible binaries.

nosuid: Disables **set-user-identifier** or **set-group-identifier** bits. This prevents remote users from gaining higher privileges by running a **setuid** program.

port=*num*: **port=num** — Specifies the numeric value of the NFS server port. If ***num*** is ***0*** (the default), then **mount** queries the remote host's **rpcbind** service for the port number to use. If the remote host's NFS daemon is not registered with its **rpcbind** service, the standard NFS port number of TCP 2049 is used instead.

rsize=*num* and **wsize=*num***: These settings speed up NFS communication for reads (**rsize**) and writes (**wsize**) by setting a larger data block size (***num***, in bytes), to be transferred at one time. Be careful when changing these values; some older Linux kernels and network cards do not work well with larger block sizes. For NFSv2 or NFSv3, the default values for both parameters is set to 8192. For NFSv4, the default values for both parameters is set to 32768.

sec=*mode*: Specifies the type of security to utilize when authenticating an NFS connection. Its default setting is **sec=*sys***, which uses local UNIX UIDs and GIDs by using AUTH_SYS to authenticate NFS operations.

sec=krb5 uses Kerberos V5 instead of local UNIX UIDs and GIDs to authenticate users.
sec=krb5i uses Kerberos V5 for user authentication and performs integrity checking of NFS operations using secure checksums to prevent data tampering.
sec=krb5p uses Kerberos V5 for user authentication, integrity checking, and encrypts NFS traffic to prevent traffic sniffing. This is the most secure setting, but it also involves the most performance overhead.

tcp: Instructs the NFS mount to use the TCP protocol.

udp: Instructs the NFS mount to use the UDP protocol.

25.7. Provide NFS network shares to specific clients

Restricting access to NFS shares can be done by restricting firewall access (**iptables**), or by configuring the **/etc/exports** file. The **/etc/exports** file can restrict access to a single machine, a wildcard, or an IP network.

It all starts with installing and starting NFS

```
# yum install nfs-utils rpcbind
# service nfs start
# service rpcbind start
```

- Restricting to a single machine (can be exported to an IP or hostname)

Edit **/etc/exports**

Configure the export command like the following

```
/media  192.168.10.10(rw,no_root_squash)
```

Restart the service - **service nfs restart**

- Restricting to a wildcard -- this allows exporting to a name or IP address with wildcards

Edit **/etc/exports**

Configure the export command like the following

```
/media  *.example.com(rw,no_root_squash)
or   /media  192.168.*10(rw,no_root_squash)
```

Restart the service - **service nfs restart**

- Restricting to an IP network -- this allows exporting to an entire subnet, or group of addresses

Edit **/etc/exports**

Configure the export command like the following

```
/media  192.168.10.0/24(rw,no_root_squash)
```

Restart the service - **service nfs restart**

25.8. Provide NFS network shares suitable for group collaboration

Provide network shares as shown in previous point and:

- Create a sharegroup
- Add users to sharegroup
- Create shared directory and set gid on it

25.9. NFS quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- Packages needed to provide the service

```
# yum  install  nfs-utils  rpcbind
```

- **Configure SELinux to support the service**

```
# getsebool -a | grep nfs
```

At firs, it is not necessary to change any SELinux boolean

- **Configure the service to start when the system is booted**

```
# chkconfig nfs on
```

- **Configure the service for basic operation**

  - Install service
  - "Provide network shares to specific clients"
  - Configure the service to start when the system is booted
  - Configure SELinux support
  - Configure static lockd, statd, mountd, rquotad ports in **/etc/sysconfig/nfs**
  - Update **/etc/sysconfig/iptables**:
    Open **tcp** port **2049**

- **Configure host-based and user-based security for the service**

  - Host
    Define host permissions in **/etc/exports**

  - User
    Use filesystem permissions

25.10. Connecting to a Samba share

To query the network for Samba servers, use the **findsmb** command. For each server found, it displays its IP address, NetBIOS name, workgroup name, operating system, and SMB server version. To connect to a Samba share from a shell prompt, type the following command:

```
# smbclient //<hostname>/<sharename> -U <username>
```

Replace <***hostname***> with the hostname or IP address of the Samba server you want to connect to, <***sharename***> with the name of the shared directory you want to browse, and <***username***> with the Samba username for the system. Enter the correct password or press **Enter** if no password is required for the user.

If you see the **smb:\>** prompt, you have successfully logged in. Once you are logged in, type **help** for a list of commands. If you wish to browse the contents of your home directory, replace ***sharename*** with your username. If the **-U** switch is not used, the username of the current user is passed to the Samba server.

To exit **smbclient**, type **exit** at the **smb:\>** prompt.

25.11. Mounting the share

Sometimes it is useful to mount a Samba share to a directory so that the files in the directory can be treated as if they are part of the local file system.
To mount a Samba share to a directory, create the directory if it does not already exist, and execute the following command as root:

```
# mount -t smbfs -o username=<username> //<servername>/<sharename> /mnt/point
```

This command mounts <***sharename***> from <***servername***> in the local directory ***/mnt/point***

25.12. Mount and unmount CIFS and NFS network file systems

Mounting network shares of type NFS and CIFS on Red Hat Enterprise Linux 6 is done with the mount command.

Mount CIFS share

Mounting samba/windows shares requires the **-t cifs** option followed by the ***//server***.

```
# mount -t cifs //server/share /mnt --verbose -o user=username
```

Unmount CIFS share

```
# umount /mnt
```

Mount NFS share

Mounting a nfs share would be done with the **-t nfs** option, followed but the ***server***:/***mount***

```
# mount -o rw -t nfs hostname:/mountpoint /mnt
```

Unmount NFS share

```
# umount /mnt
```

25.13. Samba quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- Packages needed to provide the service

```
# yum install samba-client samba-common samba
```

- Configure SELinux to support the service

```
# getsebool -a | grep samba
```

To allow users to connect their own home directories following boolean must be enable

```
# setsebool -P samba_enable_home_dirs on
```

Normally, Samba can only share those files and directories labeled with the **samba_share_t** file type. It is true, the **samba_share_t** file type is not required if the **samba_export_all_ro** or **samba_export_all_rw** booleans are enabled. However, that would be a security risk. So in most cases, you'll want to enable directories and files with the noted file type with a command like the following:

```
# chcon -R -t samba_share_t /share
```

In addition, to make sure the changes survive a relabel of SELinux, you'll want to set up the file_contexts.local file in /etc/selinux/targeted/contexts/files directory with a command such as the following:

```
# semanage fcontext -a -t samba_share_t "/share(/.*)?"
# restorecon -R -v /share    # Not necessary if chcon command has been run previously
```

- **Configure the service to start when the system is booted**

```
# chkconfig smb start
```

- **Configure the service for basic operation**

  - Install service
  - Create a share
  - Configure the service to start when the system is booted
  - Configure SELinux support
  - Update **/etc/sysconfig/iptables**
    open **tcp** ports **139** and **445**

- **Configure host-based and user-based security for the service**

  - Host
    Use **iptables**
    /etc/samba/smb.conf can also be used with "**hosts allow**"/"**hosts deny**" property

  - User
    Configure user's permissions in **/etc/samba/smb.conf**
    To allow users to connect to their own home directories:

```
[homes]
        comment = Home Directories
        browseable = no
        writable = yes
```

Don't forget to set samba passwords for users - **smbpasswd -a <user>**

25.14. Provide Samba network shares to specific clients

Here are the steps needed to add an SMB share to specific clients.

Install samba

```
# yum install samba-client samba-common samba
```

Configure the **/etc/samba/samba.conf** file; Find the line workgroup and set the correct workgroup name

```
workgroup = WORKGROUP
```

At the end of the file, create a new directory block using the same syntax as the others. This example will create a share named "foo" that is only accessible by user "foo"

The following default would limit access to the networks with the **192.168.12.0** and **192.168.13.0** network IP addresses, as well as the local computer (**127.**):

```
hosts allow = 127.   192.168.12.   192.168.13.
```

It’s possible to configure a hosts deny directive in a similar fashion. With such directives, you can set up host-based security for Samba. In the global section, such security would apply server-wide. You can also use the hosts allow and hosts deny directives in the definitions for individual shared directories.

```
# foo
[foo]
comment = Public share
hosts allow = 127.   192.168.12.   192.168.13.
path = /foo
# public = yes
writeable = yes
browseable = yes
valid users = foo
# invalid users = bar
```

Save the file and restart the services -- **service smb restart**, **service nmb restart**

Make sure the user "**foo**" exists, and set the samba password - **smbpasswd -a foo**

25.15. Provide Samba network shares suitable for group collaboration

Configuring SMB shares for groups is very similar to sharing for an individual. The only gotcha here is making sure security on the folder are set properly.

Scenario: You have a group named **group1**, users **foo** and **bar** are members of this group. You need to share a directory named **/group1** to these users only.

Install samba

```
# yum install samba-client samba-common samba
```

Configure the **/etc/samba/samba.conf** file; Find the line workgroup and set the correct workgroup name

At the end of the file, create a new directory block using the same syntax as the others. Note the use of the **+group1** for valid users, this identifies it as a group instead of a user

```
# group1
[group1]
comment = Public share
hosts allow = 127.   192.168.12.   192.168.13.
path = /group1
# public = yes
writeable = yes
browseable = yes
valid users = +group1
# invalid users = foo
```

Save the file and restart the services -- **service smb restart**, **service nmb restart**

Ensure the folder being shared is owned by the group

```
# chown root:group1 /group1
```

Ensure the file permissions allow the group to read/write

```
# chmod 775 /group1 -R
```

Make sure that users "**foo**" and "**bar**" exist, and set the samba password - **smbpasswd -a <user>**

25.16. Samba additional configuration

browseable: indicates if share is visible or not in network environnment (for instance, when scannning share from a client with command **smbclient -L <samba_server> -U <user>**)
create mask = *0770* : Permissions when creating files in share
directory mask = *2770* : Permissions when creating folders in share

The security directive may be a bit confusing. The standard value of the directive, as shown here, means that connections check the local password database. It is appropriate when configuring this computer as a Domain Controller (DC), specifically a Primary Domain Controller (PDC).

security = user

Alternatively, to configure this computer as a member server on a domain, use a password database from a DC. Strangely enough, in that case, you would substitute the following command:

security = domain

To set up a Linux system as a workstation that happens to share directories on a Microsoft domain, you'll need to set up the computer as a member server on that domain.

To configure a system as a member server on an Active Directory network, substitute the following command:

security = ads

Alternatively, to use a database from another computer that is not a DC, you'd substitute the following command:

security = server

Finally, to configure a system on a peer-to-peer workgroup that does not require usernames, substitute the following command:

security = share

To summarize, there are five basic authentication options: **share, user, server, domain, and ads**.

Now, refocus this directive on the authentication database. The default is **security = user**; in this case, make sure the Samba usernames and passwords that you create match those on individual Windows NT/2000/XP/Vista systems on the network.

If the database is local, it could be either

passdb backend = smbpasswd

or

passdb backend = tdbsam

The **smbpasswd** database is local, stored in the local **/var/lib/samba/private** directory. The **tdbsam** option, short for the Trivial Database Security Accounts Manager, sets up a local account database in the **/var/lib/samba/private** directory.

Alternatively, for a remote database such as LDAP, you could activate the following directive. If the LDAP server is located on a remote system, that Uniform Resource Identifier (URI) address can be included here.

passdb backend = ldapsam

If you've set up **security = server** or **security = domain**, you'll also want to activate the following directive with the name or IP address of the password server.

Alternatively, you could replace <NT-Server-Name> with a * to have Samba search for the password server.

; password server = <NT-Server-Name>

If you've set up **security = ads**, you'll also want to activate the following directive to specify the Active Directory (AD) realm, substituting the actual AD realm for MY_REALM:

```
; realm = MY_REALM
```

To list users in samba database run following command:

```
# pdbedit -L
```

Name Resolution

The following section allows you to set up a Samba server with a database of NetBIOS names and IP addresses, starting with the following comment:

```
#----------- Name Resolution --------------
```

The Windows Internet Name Service (WINS) is functionally equivalent to DNS on Microsoft-based networks such as Samba. If you activate the following command, Samba activates a WINS server on the local computer:

```
; wins support = yes
```

Alternatively, you can point the local computer to a remote WINS server on the network; of course, you'd have to substitute the IP address for w.x.y.z. Do not activate both the wins support and wins server directives on the same system, as they are incompatible.

```
; wins server = w.x.y.z
```

Samba servers may not installed on every Linux system. In that case, you could enable the following directive to allow access from such systems with only Samba client software:

```
; wins proxy = yes
```

If the answer to a name resolution request is not in a WINS server, the following directive would allow the same search through configured DNS servers:

```
; dns proxy = yes
```

25.17. Samba as a client

To browse shared directories from a Linux computer, you should know how to use the **smbclient** command. This can test connectivity to any SMB host on a Windows or Samba-based Linux/Unix computer. Assuming it's allowed by a firewall, you can use smbclient to check the shared directories and printers from other systems on at least the local network. For example, the following smbclient command checks shared directories and printers:

```
# smbclient -L server1.example.com -U <anna>
```

I've specified two arguments with the smbclient command: the **-L** specifies the name of the Samba server, and the **-U** specifies a username on the remote computer.

When the command reaches the Samba server, you're prompted for the appropriate password.

You can use the **smbclient** command to make a client connection similar to an FTP connection with the following command:

```
# smbclient //<server1.example.com/public> -U <user>
```

Of course, most administrators would prefer to mount that share on a local directory. That's where options to the mount command are helpful.

Shares can be mounted by the root administrative user. The standard is with the **mount.cifs** command, functionally equivalent to the **mount -t cifs** command.

For example, the following command mounts the share named public on the local /home/shared directory:

```
# mount.cifs //server1.example.com/public /home/shared -o username=anna
```

This command prompts for user anna's password on the remote server. That password should be part of the Samba user authentication database on the server1.example.com system, normally different from the standard Linux authentication database. Of course, that user 'anna' could also mount her remote home directory in a similar fashion, with a command like the following:

```
# mount.cifs //server1.example.com/anna /home/donna/remote -o username=anna
```

As it certainly takes a few extra steps to set up a shared directory, it would be useful to automate the process. The standard method is through the **/etc/fstab** configuration file. You could set up the public share in **/etc/fstab** by adding the following line (which is wrapped):

```
//server1.example.com/public /home/shared cifs rw,username=anna,password=pass, 0 0
```

But that can be a risk, as the **/etc/fstab** file is world-readable. To that end, you can configure a dedicated credentials file with the username and password, as follows:

```
//server/pub /share cifs rw,credentials=/etc/samba/smbanna 0 0
```

You can then set up the username and password in the /etc/samba/smbanna file:

```
username=anna
password=annaspassword
```

While the contents of that file must still exist in clear text, you can configure the /etc/samba/smbanna file as readable only by the root administrative user. It's also possible to configure the automounter with similar information.

26. SMTP

26.1. Configure a mail transfer agent (MTA) to accept inbound email from other systems

By default postfix will accept only mail originating locally, and all it takes is updating a config file to change that.

Install the necessary packages

```
# yum install postfix
```

Edit the **/etc/postfix/main.cf** file; Find the line **inet_interfaces = localhost** and change it to **inet_interfaces = all**

Restart the service

```
# service postfix restart
```

Open the firewall

```
# iptables -I INPUT -p tcp --dport 25 -j ACCEPT
```

You should be able to test this by telnetting from a remote computer. If you receive a connection, you're good to go.

26.2. Configure an MTA to forward (relay) email through a smart host

This one is similar to defaulting to accept mail locally; by default postfix will only send mail to local recipients.

Install the necessary packages

```
# yum install postfix
```

Edit the **/etc/postfix/main.cf** file; Find the **relayhost** section and add a line **relayhost = 192.168.10.1**

Restart the service

```
# service postfix restart
```

You should be able to test this by sending an email to a remote user.

26.3. SMTP quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- **Packages needed to provide the service**

```
# yum install postfix
```

- **Configure SELinux to support the service**

```
# getsebool -a | grep postfix
```

- **Configure the service to start when the system is booted**

```
# chkconfig postfix on
```

- **Configure the service for basic operation**

  - Install service
  - Configure the service to start when the system is booted
  - Configure SELinux support
  - Update **/etc/sysconfig/iptables**:
    open **tcp** ports **25**

- **Configure host-based and user-based security for the service**

  - Host
    Use iptables

  - User
    /etc/postfix/main.cf:

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_recipient_restrictions =\
permit_sasl_authenticated, permit_mynetworks, reject_unauth_destination
```

```
# service saslauthd start
# chkconfig saslauthd on
```

27. Network Time Protocol (NTP) setup

You can synchronize the system clock with a remote server over the Network Time Protocol (NTP). For the one-time synchronization only, use the **ntpdate** command:

1.- Firstly, check whether the selected NTP server is accessible:

```
# ntpdate -q server_address
```

For example:

```
# ntpdate -q 0.rhel.pool.ntp.org
```

2.- When you find a satisfactory server, run the **ntpdate** command followed by one or more server addresses:

```
# ntpdate <server_address1  ...  server_addressX>
```

For instance:

```
# ntpdate 0.rhel.pool.ntp.org 1.rhel.pool.ntp.org
```

Unless an error message is displayed, the system time should now be set. You can check the current by setting typing **date** without any additional arguments.

3.- In most cases, these steps are sufficient. Only if you really need one or more system services to always use the correct time, enable running the **ntpdate** at boot time:

```
# chkconfig ntpdate on
```

***** Note**: If the synchronization with the time server at boot time keeps failing, i.e., you find a relevant error message in the **/var/log/boot.log** system log, try to add the following line to **/etc/sysconfig/network**:

```
NETWORKWAIT=1
```

However, the more convenient way is to set the **ntpd** daemon to synchronize the time at boot time automatically:

1.- Open the NTP configuration file **/etc/ntp.conf** in a text editor, or create a new one if it does not already exist:

```
# vi  /etc/ntp.conf
```

2.- Add or edit the list of public NTP servers. If you are using Red Hat Enterprise Linux 6, the file should already contain the following lines, but feel free to change or expand these according to your needs:

```
server 0.rhel.pool.ntp.org
server 1.rhel.pool.ntp.org
server 2.rhel.pool.ntp.org
```

***** Tip**: To speed the initial synchronization up, add the **iburst** directive at the end of each line:

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
```

3.- Once you have the list of servers complete, in the same file, set the proper permissions, giving the unrestricted access to localhost only:

```
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict -6 ::1
```

4.- Save all changes, exit the editor, and restart the NTP daemon:

```
# service ntpd restart
```

5.- Make sure that ntpd daemon is started at boot time:

```
# chkconfig ntpd on
```

27.1. Synchronize time using other NTP peers

1.- The easiest way to configure NTP is to use the GUI:

On the **top bar**, right-click the time and select **Preferences** - Click **Time Settings** and the **Set System Time** - Check the box **Synchronize date and time over the network** - Edit the list of **NTP servers** and click **OK**

2.- Alternatively, you can execute **system-config-date** to go directly to step 3.

3.- To perform the same via command line:

Edit **/etc/ntp.conf**

Configure 1 or more server lines like below

```
server 0.rhel.pool.ntp.org
or server 192.168.10.1
```

Start the service

```
# service ntpd start
```

When all finished, make sure ntpd is set to start automatically for next reboot **chkconfig ntpd on**.
You can also perform a one-off sync by running **ntpdate 192.168.10.1** (this only works if ntpd isn't running)

27.2. NTP quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- **Packages needed to provide the service**

```
# yum install ntp
```

- **Configure SELinux to support the service**

N/A

- **Configure the service to start when the system is booted**

```
# chkconfig ntpd on
```

- **Configure the service for basic operation**

  - Install service
  - Configure the service to start when the system is booted
  - (If NTP is configured as a server) Update **/etc/sysconfig/iptables**:
    Open **tcp** and **udp** port **123**

- **Configure host-based and user-based security for the service**

  - Host
    (If NTP is configured as a server) Use **iptables**

  - User
    N/A

28. Configure a default configuration HTTP server

Installing apache via yum on Red Hat Enterprise Linux 6 does most of the setup for you.

```
# yum install httpd

# service httpd start
```

Now if you try to visit the main ip or domain of the server, you may run into an issue getting to the site. Whenever you enable a network service like a web server, you also have to allow the outside to use that service. We have to add an entry into iptables.

```
# iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

This would add an entry into iptables, but to survive a reboot we would have to save this.

```
# service iptables save
```

Now an easier way of doing this, is to use system-config-firewall, which is the gui/tui tool to configure a firewall.

```
# system-config-firewall
```

This may not make things perfect, but it can definitely give you a jump start to molding rules.

***** Note**: it may not be installed by default.

Once you have the firewall open, and the web server is running, you should be seeing the apache page in a browser. Should you place an html file in /var/www/html/, you would be able to see the contents of it through your browser.

28.1. The Apache HTTP server

See documentation (pag. 283)

28.2. HTTP quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- **Packages needed to provide the service**

```
# yum install httpd
```

or

```
# rpm -ivh httpd*
```

- **Configure SELinux to support the service:** (for a basic configuration, at first, it is not necessary to change any boolean)

- Add the following line to **/etc/sysconfig/iptables** file to allow the http traffic (think about **tcp** port **443** if HTTPS is being used):

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
```

- **Configure the service to start when the system is booted**

```
# chkconfig httpd on
```

- **Configure the service for basic operation**

  - Install service
  - Configure the service to start when the system is booted
  - Configure SELinux support (for a basic configuration it is not necessary to change any boolean)
  - Update **/etc/sysconfig/iptables**:
    Open **tcp** port **80** and, eventually, **tcp** port **443**

- **Configure host-based and user-based security for the service**

  - Host
    - use iptables

      and/or

- **/etc/httpd/conf/httpd.conf**:
```
<Directory "/var/www/html/help">
        Order deny,allow
        Allow from <good_ip>
        Deny from all
</Directory>
```

- User
  - **/etc/httpd/conf/httpd.conf**:
```
<Directory "/var/www/html/help">
        AuthType Basic
        AuthName "Password Protected Test"
        AuthUserFile /etc/httpd/passwordfile
        Require user <user1>
</Directory>
```
```
# htpasswd -c /etc/httpd/passwordfile <user1>
```

- Group
  - **/etc/httpd/conf/httpd.conf**:
```
<Directory "/var/www/html/help">
        AuthType Basic
        AuthName "Password Protected Test"
        AuthUserFile /etc/httpd/userpasswords
        AuthGroupFile /etc/httpd/grouppasswords
        Require group <usergroup>
</Directory>
```
```
# vi /etc/httpd/grouppasswords
        web:    user1

# htpasswd -c /etc/httpd/passwordfile <user1>
```

28.3. Configure a Virtual Host

HTTP virtual hosts allow a single web server to act like multiple web servers, either by publishing to multiple IPs and ports, or by publishing multiple sites and identifying them by name. This feature allows you to publish *http://foo.example.com* and *http://bar.example.com* from the same host with a single address, and the server returns the appropriate information based on the site name the customer has typed in.

- Create directories to hold the content
```
# cd /var/www
# mkdir <virtualhost1>
# mkdir <virtualhost2>
# echo "This is the server 1" > virtualhost1/index.html
# echo "This is the server 2" > virtualhost2/index.html
```

STEP 2: Create virtual directories

```
# vi /etc/http/conf/httpd.conf
```

- Uncomment the line **NameVirtualHost *:80**
- Copy the last 7 lines twice, remove the # at the beginning
- Edit the **DocumentRoot** and **ServerName** lines to match your new directories
- Save the file and execute **service httpd restart**

```
<VirtualHost *:80>
    ServerAdmin root@vhost1.example.com
    DocumentRoot /var/www/html/virtualhost1
    ServerName vhost1.example.com
    ErrorLog logs/vhost1.example.com-error_log
    CustomLog logs/vhost1.example.com-access_log common
</VirtualHost>


<VirtualHost *:80>
    ServerAdmin root@vhost2.example.com
    DocumentRoot /var/www/html/virtualhost2
    ServerName vhost2.example.com
    ErrorLog logs/vhost2.example.com-error_log
    CustomLog logs/vhost2.example.com-access_log common
</VirtualHost>
```

Uncomment **NameVirtualHost** directive in order to avoid error messages like this:
Starting httpd: [warn] _default_ VirtualHost overlap on port 80, the first has precedence

```
NameVirtualHost *:80
```

28.3.1. Securing Virtual Hosts

If you're configuring a secure web server that conforms to the HTTPS protocol, Red Hat provides a different configuration file for this purpose: **ssl.conf** in the **/etc/httpd/conf.d** directory. If this file isn't available, you need to install the **mod_ssl** RPM. Before editing this file, back it up.

Make sure that following two lines are commented out in **httpd.conf** in order to SSL work properly:

```
LoadModule ssl_module modules/mod_ssl.so
        ...
Include conf.d/*.conf
```

Now, in **/etc/httpd/conf.d/ssl.conf**, verify that following Listen directive is active:

```
    Listen 443
```

As suggested by the title, this configuration file includes a number of passphrase dialogues. Generally, no mayor changes are required to configure a new Virtual Host:

```
    # <VirtualHost -default-:443>
    <VirtualHost *:443>
        DocumentRoot "/var/www/html/host1.example.com"
        ServerName vhost1.example.com

        ErrorLog logs/ssl_error_log
        TransferLog logs/ssl_access_log
        LogLevel warn

        SSLEngine on
        SSLProtocol all -SSLv2
        SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
        SSLCertificateFile /etc/pki/tls/certs/vhost1.example.com.crt  <-- New certificate!
        SSLCertificateKeyFile /etc/pki/tls/private/vhost1.example.com.key

        <Files ~ "\.(cgi|shtml|phtml|php3?)$">
            SSLOptions +StdEnvVars
        </Files>
        <Directory "/var/www/cgi-bin">
            SSLOptions +StdEnvVars
        </Directory>

        SetEnvIf User-Agent ".*MSIE.*" \
            nokeepalive ssl-unclean-shutdown \
            downgrade-1.0 force-response-1.0

        CustomLog logs/ssl_request_log \
            "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

***** Tip**: A new certificate can be created by running '**genkey**' command in this way (**do not encrypt certificate; otherwise the password will be required in order to start web server**):

```
    # cd /etc/pki/tls/certs
    # genkey vhost1.example.com
```

28.3.2. Redirecting all traffic to secure HTTPS

```
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

28.4. Configure private directories

At first glance, this objective can mean 2 things: allowing users to setup **public_html** directories, or securing directories with configuration files or **.htaccess.** A great page detailing the setup of HTTP can be found at http://www.brennan.id.au/13-Apache_Web_Server.html

- Allowing users to setup public_html directories

Edit the **/etc/httpd/conf/httpd.conf** and find the line **UserDir disabled**. Comment out this line, and uncomment the line **UserDir public_html**.

Uncomment following lines, too, to activate access to users' home directories:

```
<Directory /home/*/public_html>
        AllowOverride FileInfo AuthConfig Limit
        Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec

        <Limit GET POST OPTIONS>
                Order allow,deny
                Allow from all
        </Limit>
        <LimitExcept GET POST OPTIONS>
                Order deny,allow
                Deny from all
        </LimitExcept>
</Directory>
```

Finally we need to grant access to directories by adding the right permissions on the user's home for the user owning httpd daemon, 'apache':

```
# mkdir /home/<user>/public_html
# chown <user:group> /home/<user>/public_html
# chmod 701 /home/<user>
# chmod 701 /home/<user>/public_html
# setfacl -m u:apache:rx /home/luissma
# setfacl -m u:apache:rx /home/luissma/public_html
```

Restart the web server - **service httpd restart**

Configure SELinux to grant access to home directories:

```
# setsebool -P httpd_enable_homedirs true
```

*** **Note**: There may be multiple layers of security blocking access including folder, file and selinux restrictions. Specifically, ensure the apache user has access to the home and **public_html** directories, as well as all files under the **public_html** directory.

- Securing directories

Edit the **/etc/httpd/conf/httpd.conf** file

```
<Directory "/var/www/html/help">
        AuthType Basic
        AuthName "Password Protected Test"
        AuthUserFile /etc/httpd/conf/passwordfile
        [ AuthGroupFile /etc/httpd/conf/groupfile ]
        [ Require valid-user | Require user <userX> | Require group <groupY> ]
</Directory>
```

Add users to the **authusers** file - **htpasswd [-c] /etc/httpd/conf/authusers <username>**
Restart the web server - **service httpd restart**
Access should now be restricted to ***username***

- .htaccess + .htauthusers [+.htauthgroup]

This is traditionally used to restrict access to **public_html** directories since the average user doesn't have access to edit the **httpd.conf** file.

In the target folder, touch following files: **.htaccess**, **.htauthusers** and, eventualy, **.htauthgroup**

Edit **.htaccess** and enter the following (note the **AuthUserFile** appears to need a fully qualified path)

```
AuthType Basic
AuthName "Private Area"
AuthUserFile /home/username/public_html/private/.htauthusers
[ AuthGroupFile /home/username/public_html/private/.htauthgroup ]
[ Require valid-user | Require user <userX> | Require group <groupY> ]
```

Execute **htpasswd .htauthusers <username>**

*** **Note**: Do not foget to add following directive to corresponding section in main configuration file in order to activate authentication:

```
AllowOverride AuthConfig
```

Access should now be restricted to user ***username***

28.5. Deploy a basic CGI application

The default apache configuration allows execution of CGI scripts in the **/var/www/cgi-bin/** directory. This is controlled by the **ExecCGI** option for a specified directory.

To allow Apache to read CGI files, the **httpd.conf** file includes the **LoadModule cgi_module** directive. To make it easier for clients to find CGI files through their browsers, Apache includes the **ScriptAlias** directive. For example, the following **ScriptAlias** directive links the cgi-bin subdirectory to the default **/var/www/cgi-bin** directory:

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin"
```

With this **ScriptAlias** directive, if the web site is ***server1.example.com***, scripts can be found in the **http://server1.example.com/cgi-bin/** URL.

Alternatively, you can set up CGI scripts in a directory other than **/var/www/cgi-bin** and change the reference accordingly.

As suggested in the Apache web server documentation available from the **httpdmanual** package, you'd need to make changes to allow CGI scripts to actually be executable by the Apache server:

```
<Directory /var/www/cgi-bin>
        AllowOverride None
        Options ExecCGI
        AddHandler cgi-script .pl
        Order allow,deny
        Allow from all
</Directory>
```

The **AddHandler** directive associates **CGI scripts** with files with the **.pl** extension.

If CGI scripts are required for one of the previously configured virtual hosts, you'll need to set up a different **ScriptAlias** and a corresponding **<Directory>** container. For the **vhost1.example.com** site described previously, I add the following directive:

```
ScriptAlias /cgi-bin/ /var/www/html/virtualhost1/cgi-bin/
```

Once a script is included in the target directory, it will begin to respond as an executable. Sample cgi:

```
# vi /var/www/html/virtualhost1/cgi-bin/helloworld.pl

#!/usr/bin/perl
print "Content-Type: text/plain", "\n\n";
print "Hello World in Perl", "\n";
```

Do not forget to set execute permission on the script:

```
# chmod 755 /var/www/html/virtualhost1/cgi-bin/helloworld.pl
```

And verify that scripts have the **httpd_sys_script_exec_t** SELinux file type. If necessary:

```
# semanage fcontext -a -t httpd_sys_script_exec_t /var/www/html/virtualhost1/cgi-bin(/.*)?"

# restorecon -R -v /var/www/html/virtualhost1/cgi-bin/
```

28.6. Configure group-managed content

If desired, a directory with specific permissions can be created:

```
# groupadd -g <gid> web
Add users to new group
# mkdir /www/var/html/<webdesign>
# chgrp apache.web /www/var/html/webdesign
# chmod 775 /www/var/html/webdesign
# chmod g+s /www/var/html/webdesign


# vi /etc/httpd/conf/httpd.conf
        <Directory "/var/www/html/webdesign">
                AuthType Basic
                AuthName "Password Protected Directory"
                AuthUserFile /etc/httpd/conf/webdesigners_userpasswords
                AuthGroupFile /etc/httpd/conf/webdesigners_group
                Require group <webdesigners>
        </Directory>

# vi /etc/httpd/conf/webdesigners_group
        webdesigners:   user1   user2

# htpasswd -c /etc/httpd/conf/webdesigners_userpasswords <user1>
# htpasswd /etc/httpd/conf/webdesigners_userpasswords <user2>
```

29. DNS

29.1. Configure a caching-only name server

- Verify that /etc/resolv.conf contains a valid DNS IP:

```
# cat /etc/resolv.conf
nameserver <192.168.1.1>
```

- Install the needed packages:

```
# yum install bind-*
```

This will install following packages:

bind
bind-chroot
bind-devel
bind-dyndb-ldap
bind-libs
bind-sdb
bind-utils

- Edit **/etc/named.conf** (main configuration file for DNS) and modify following directives:

```
listen-on port 53 { 127.0.0.1; };
allow-query     { localhost; };
```

to permit listening through specific network interfaces (IP separated by a blank space) or through all network interfaces on the local server:

```
listen-on port 53 { 127.0.0.1; 192.168.10.41; xxx.xxx.xxx.xxx; };
listen-on port 53 { any; };
```

and to allow querying to specific client(s) or to all clients connected to the server:

```
allow-query     { 192.168.10.21; xxx.xxx.xxx.xxx; };
allow-query     { any; };
```

- Configure DNS service to start automatically after a reboot of the server:

```
# chkconfig --level 345 named on
```

- Restart DNS services:

```
# service named restart
```

- Do not forget to open iptables for **port 53** (both **tcp** and **udp** traffic)

```
# iptables -A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
# iptables -A INPUT -p udp -m udp --dport 53 -j ACCEPT
```

29.2. Configure a caching-only name server to forward DNS queries

- Verify that /etc/resolv.conf contains a valid DNS IP:

```
# cat /etc/resolv.conf
```

```
nameserver <192.168.1.1>
```

- Install the needed packages:

```
# yum install bind-*
```

This will install following packages:

```
bind
bind-chroot
bind-devel
bind-dyndb-ldap
bind-libs
bind-sdb
bind-utils
```

- Edit **/etc/named.conf** (main configuration file for DNS) and modify "options" paragraph with only following directives (set a valid DNS server in "forwarders" directive):

```
options {
        listen-on port 53 { any; };
        directory     "/var/named";
        forward only;
        forwarders { <192.168.1.1>; };     <--- DNS server
};
```

"**listen-on port**" directive can be configured as for a simple caching only name server, this is:

```
listen-on port 53 { 127.0.0.1; 192.168.10.41; xxx.xxx.xxx.xxx; };
listen-on port 53 { any; };
```

- Configure DNS service to start automatically after a reboot of the server:

```
# chkconfig --level 345 named on
```

- Restart DNS services:

```
# service named restart
```

- Do not forget to open iptables for **port 53** (both **tcp** and **udp** traffic)

```
# iptables -A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
# iptables -A INPUT -p udp -m udp --dport 53 -j ACCEPT
```

29.3. DNS quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- **Packages needed to provide the service**

```
# yum install bind-*
```

- **Configure SELinux to support the service**

(At first, no SELinux boolean change is necessary)

```
# getsebool -a | grep named
```

- **Configure the service to start when the system is booted**

```
# chkconfig named on
```

- **Configure the service for basic operation**
    - Install service
    - "Configure a caching-only name server"
    - Configure the service to start when the system is booted
    - Configure SELinux support
    - Update **/etc/sysconfig/iptables**:
      Open **tcp** and **udp** port **53**

- **Configure host-based and user-based security for the service**

    - Host
      Open **tcp** and **udp** port **53** with **iptables**

    - User
      N/A

30. Configure a default FTP server

vsftpd installs with a default configuration that works for this requirement. So a basic:

```
# yum install vsftpd
```

```
# chkconfig vsftpd on
```

This will get your default server up and running. But what about firewall and selinux?

Iptables:

For iptables you want to open up port 20 and 21, to allow ftp requests in.

```
# iptables -I INPUT 5 -p tcp -m tcp --dport 20 -j ACCEPT

# iptables -I INPUT 5 -p tcp -m tcp --dport 21 -j ACCEPT
```

Then remember to always save your iptables rules so they survive a reboot.

SELinux:

*** **Tip**: All this information is stored in man pages, so rather than memorizing, use the resources available. If you search for _selinux, then all services that have information on how to be configured with SELinux will show up. To search the man pages use:

```
# man -k _selinux | grep ftp
ftpd_selinux       (8)  - Security-Enhanced Linux policy for ftp daemons
ftpdctl_selinux    (8)  - Security Enhanced Linux Policy for the ftpdctl processes
tftpd_selinux      (8)  - Security Enhanced Linux Policy for the tftpd processes

# man ftpd_selinux
```

To make a ftp server's content available run the following:

```
# semanage fcontext -a -t public_content_t "/var/ftp(/.*)?"
# restorecon -F -R -v /var/ftp
```

That's it. As long as you can install the application **vsftpd**, open the correct ports in **iptables**, and set context in **SELinux** you are good to go on this objective.

30.1. Configure anonymous-only download

Enabling anonymous-only download appears to be enabled by default. Below are the steps needed in case it isn't in the future.

- Install packages

```
# yum install vsftpd ftp
```

- Edit **/etc/vsftpd/vsftpd.conf**
    Find the line **anonymous_enabled** and set it to **YES**
    Ensure that **anon_upload_enabled** is set to **NO**

- Use **public_content_t file context** for content:

```
# semanage fcontext -a -t public_content_t "/var/ftp(/.*)?"
# restorecon -F -R -v /var/ftp
```

- Restart ftp

```
# service vsftpd restart
```

30.2. Enable anonymous upload

- Install packages

```
# yum install vsftpd ftp
```

- Edit **/etc/vsftpd/vsftpd.conf**
  Find the line **anonymous_enabled** and set it to **YES**
  Ensure that **anon_upload_enabled** is set to **YES**

- Configure SELinux to support the service

```
# getsebool -a | grep ftpd
# setsebool -P allow_ftpd_anon_write on
```

- Make /var/ftp contents available and /var/ftp/pub, for instance, writable:

```
# semanage fcontext -a -t public_content_t "/var/ftp(/.*)?"
# restorecon -F -R -v /var/ftp

# chown ftp:ftp /var/ftp/pub
# semanage fcontext -a -t public_content_rw_t "/var/ftp/pub(/.*)?"
# restorecon -F -R -v /var/ftp/pub
```

- Restart ftp

```
# service vsftpd restart
```

30.3. FTP quick configuration procedure (packages, SELinux, auto start, basic operation, host/user-based security)

- Packages needed to provide the service

```
# yum install vsftpd
```

- Configure SELinux to support the service (only if anonymous write permissions are needed)

```
# getsebool -a | grep ftpd
# setsebool -P allow_ftpd_anon_write on
```

- **Make /var/ftp contents available**:

```
# semanage fcontext -a -t public_content_t "/var/ftp(/.*)?"
# restorecon -F -R -v /var/ftp
```

- **Configure the service to start when the system is booted**

```
# chkconfig vsftpd on
```

- **Configure the service for basic operation**
  - Install service
  - "Configure anonymous-only download"
  - Configure the service to start when the system is booted
  - Configure SELinux support
  - Update **/etc/sysconfig/iptables**:
    open **tcp** ports **20** & **21**

- **Configure host-based and user-based security for the service**

  - Host
    Open **tcp** ports **20** & **21** with iptables

  - User
    /etc/vsftpd/vsftpd.conf:
    local_enable=[YES|NO]
    and/or
    userlist_enable=[YES|NO]
    This last directive must be used in conjunction with **/etc/vsftpd/ftpusers** and **/etc/vsftpd/user_list** files.

    Enable following boolean if users are meant to be able to write on their own home directories:

```
# setsebool -P ftp_home_dir 1
```

31. Firewalls

Standard security implementations usually employ some form of dedicated mechanism to control access privileges and restrict network resources to users who are authorized, identifiable, and traceable. Red Hat Enterprise Linux includes several tools to assist administrators and security engineers with network-level access control issues.

Common types of firewalls and how they function:

Method	Description	Advantages	Disadvantages
NAT	*Network Address Translation* (NAT) places private IP subnetworks behind one or a small pool of public IP addresses, masquerading all requests to one source rather than several. The Linux kernel has builtin NAT functionality through the Netfilter kernel subsys-tem.	· Can be configured transpa-rently to machines on a LAN · Protection of many machines and services behind one or more external IP addresses simplifies administration duties · Restriction of user access to and from the LAN can be configured by opening and closing ports on the NAT firewall/gateway	· Cannot prevent malicious activity once users connect to a service outside of the firewall
Packet Filter	A packet filtering firewall reads each data packet that passes through a LAN. It can read and process pac-kets by header information & filters the packet based on sets of programmable rules imp-lemented by the firewall administrator. The Linux kernel has builtin packet filtering functiona-lity through the Netfilter kernel subsystem.	· Customizable through the `iptables` frontend utility · Does not require any customi-zation on the client side, as all network activity is filtered at the router level rather than the application level · Since packets are not transmit-ted through a proxy, network performance is faster due to direct connection from client to remote host	· Cannot filter packets for content like proxy firewalls · Processes packets at the protocol layer, but cannot filter packets at an applica-tion layer · Complex network archi-tectures can make establi-shing packet filtering rules difficult, especially if coup-led with *IP masquerading* or local subnets and DMZ networks
Proxy	Proxy firewalls filter all requests of a certain proto-col or type from LAN clients to a proxy machine, which then makes those requests to the Internet on behalf of the local client. A proxy machine acts as a buffer between malicious remote users and the internal network client machines.	· Gives administrators control over what applications and protocols function outside of the LAN · Some proxy servers can cache frequently-accessed data locally rather than having to use the Internet connection to request it. This helps to reduce band-width consumption · Proxy services can be logged and monitored closely, allowing tighter control over resource utilization on the network	· Proxies are often applicati-on-specific (HTTP, Telnet, etc.), or protocol-restricted (most proxies work with TCP-connect. services only) · Application services cannot run behind a proxy, so your application servers must use a separate form of network security · Proxies can become a net-work bottleneck, as all requests&transmissions are passed through one source rather than directly from a client to a remote service

31.1. Netfilter and IPTables

The Linux kernel features a powerful networking subsystem called *Netfilter*. The Netfilter subsystem provides stateful or stateless packet filtering as well as NAT and IP masquerading services. Netfilter also has the ability to *mangle* IP header information for advanced routing and connection state management. The power and flexibility of Netfilter is implemented and controled using the **iptables** tool.

iptables uses the Netfilter subsystem to enhance network connection, inspection, and processing. **iptables** features advanced logging, pre- and post-routing actions, network address translation, and port forwarding, all in one command line interface.

31.2. Basic Firewall configuration

A firewall attempts to prevent malicious software from spreading to your computer. It also helps to prevent unauthorized users from accessing your computer.

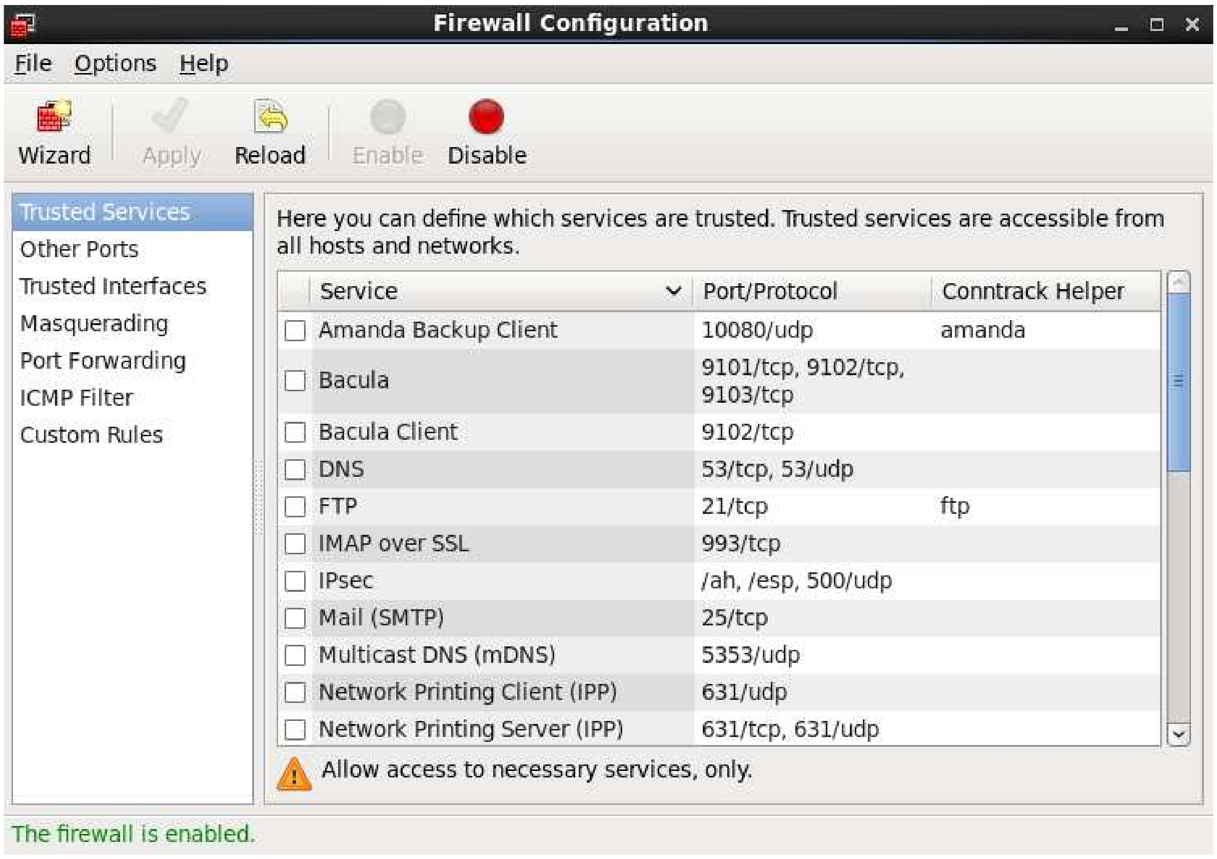
In a default Red Hat Enterprise Linux installation, a firewall exists between your computer or network and any untrusted networks, for example the Internet. It determines which services on your computer remote users can access. A properly configured firewall can greatly increase the security of your system. It is recommended that you configure a firewall for any Red Hat Enterprise Linux system with an Internet connection.

31.2.1. Firewall configuration tool

During the **Firewall Configuration** screen of the Red Hat Enterprise Linux installation, you were given the option to enable a basic firewall as well as to allow specific devices, incoming services, and ports.

After installation, you can change this preference by using the **Firewall Configuration Tool**.

To start this application, either select **System** → **Administration** → **Firewall** from the panel, or type **system-config-firewall** at a shell prompt.



Here you can define which services are trusted. Trusted services are accessible from all hosts and networks.

	Service	Port/Protocol	Conntrack Helper
☐	Amanda Backup Client	10080/udp	amanda
☐	Bacula	9101/tcp, 9102/tcp, 9103/tcp	
☐	Bacula Client	9102/tcp	
☐	DNS	53/tcp, 53/udp	
☐	FTP	21/tcp	ftp
☐	IMAP over SSL	993/tcp	
☐	IPsec	/ah, /esp, 500/udp	
☐	Mail (SMTP)	25/tcp	
☐	Multicast DNS (mDNS)	5353/udp	
☐	Network Printing Client (IPP)	631/udp	
☐	Network Printing Server (IPP)	631/tcp, 631/udp	

Allow access to necessary services, only.

The firewall is enabled.

31.2.2. Enabling and disabling the firewall

Select one of the following options for the firewall:

- **Disabled**: Disabling the firewall provides complete access to your system and does no security checking. This should only be selected if you are running on a trusted network (not the Internet) or need to configure a custom firewall using the **iptables** command line tool.

*** **Warning**: Firewall configurations and any customized firewall rules are stored in the **/etc/sysconfig/iptables** file. If you choose **Disabled** and click **OK**, these configurations and firewall rules will be lost.

- **Enabled**: This option configures the system to reject incoming connections that are not in response to outbound requests, such as DNS replies or DHCP requests. If access to services running on this machine is needed, you can choose to allow specific services through the firewall.
If you are connecting your system to the Internet, but do not plan to run a server, this is the safest choice.

31.2.3. Trusted services

Enabling options in the Trusted services list allows the specified service to pass through the firewall.

WWW (**HTTP**): The HTTP protocol is used by Apache (and by other Web servers) to serve web pages. If you plan on making your Web server publicly available, select this service. This option is not required for viewing pages locally or for developing web pages. This service requires that the **httpd** package be installed.
Enabling **WWW** (**HTTP**) will not open a port for HTTPS, the SSL version of HTTP. If this service is required, select the **Secure WWW** (**HTTPS**) service.

FTP: The FTP protocol is used to transfer files between machines on a network. If you plan on making your FTP server publicly available, select this service. This service requires that the **vsftpd** package be installed.

SSH: Secure Shell (SSH) is a suite of tools for logging into and executing commands on a remote machine. To allow remote access to the machine via **ssh**, select this service. This service requires that the **openssh-server** package be installed.

Telnet: Telnet is a protocol for logging into remote machines. Telnet communications are unencrypted and provide no security from network snooping. Allowing incoming Telnet access is not recommended. To allow remote access to the machine via telnet, select this service. This service requires that the **telnet-server** package be installed.

Mail (**SMTP**): SMTP is a protocol that allows remote hosts to connect directly to your machine to deliver mail. You do not need to enable this service if you collect your mail from your ISP's server using POP3 or IMAP, or if you use a tool such as **fetchmail**. To allow delivery of mail to your machine, select this service. Note that an improperly configured SMTP server can allow remote machines to use your server to send spam.

NFS4: The Network File System (NFS) is a file sharing protocol commonly used on *NIX systems. Version 4 of this protocol is more secure than its predecessors. If you want to share files or directories on your system with other network users, select this service.

Samba: Samba is an implementation of Microsoft's proprietary SMB networking protocol. If you need to share files, directories, or locally-connected printers with Microsoft Windows machines, select this service.

31.2.4. Other ports

The **Firewall Configuration Tool** includes an **Other ports** section for specifying custom IP ports as being trusted by **iptables**. For example, to allow IRC and Internet printing protocol (IPP) to pass through the firewall, add the following to the **Other ports** section:

```
194:tcp,631:tcp
```

31.2.5. Saving the settings

Click **OK** to save the changes and enable or disable the firewall. If **Enable firewall** was selected, the options selected are translated to **iptables** commands and written to the **/etc/sysconfig/iptables** file. The **iptables** service is also started so that the firewall is activated immediately after saving the selected options. If **Disable firewall** was selected, the **/etc/sysconfig/iptables** file is removed and the **iptables** service is stopped immediately.

The selected options are also written to the **/etc/sysconfig/system-config-firewall** file so that the settings can be restored the next time the application is started. Do not edit this file by hand.
Even though the firewall is activated immediately, the **iptables** service is not configured to start automatically at boot time.

31.2.6. Activating the IPTables service

The firewall rules are only active if the **iptables** service is running. To manually start the service, use the following command as the root user:

```
# service  iptables  restart
iptables: Applying firewall rules:                [  OK  ]
```

To ensure that iptables starts when the system is booted, use the following command:

```
# chkconfig  --level  345  iptables  on
```

31.3. Using IPTables

http://wiki.centos.org/HowTos/Network/IPTables

The first step in using **iptables** is to start the **iptables** service. Use the following command as the root user to start the **iptables** service:

```
# service iptables restart
iptables: Applying firewall rules:                [  OK  ]
```

*** **Note**: The **ip6tables** service can be turned off if you intend to use the **iptables** service only. If you deactivate the **ip6tables** service, remember to deactivate the IPv6 network also. Never leave a network device active without the matching firewall.

To force **iptables** to start by default when the system is booted, use the following command as the root user:

```
# chkconfig --level 345 iptables on
```

This forces **iptables** to start whenever the system is booted into runlevel 3, 4, or 5.

31.3.1. IPTables command syntax

The following sample **iptables** command illustrates the basic command syntax:

```
# iptables -A <chain> -j <target>
```

The **-A** option specifies that the rule be appended to <*chain*>. Each chain is comprised of one or more *rules*, and is therefore also known as a *ruleset*.

The three built-in chains are **INPUT**, **OUTPUT**, and **FORWARD**. These chains are permanent and cannot be deleted. The chain specifies the point at which a packet is manipulated.

The **-j** <***target***> option specifies the target of the rule; i.e., what to do if the packet matches the rule. Examples of built-in targets are ACCEPT, DROP, and REJECT.

31.3.2. Basic firewall policies

Establishing basic firewall policies creates a foundation for building more detailed, user-defined rules.

Each **iptables** chain is comprised of a default policy, and zero or more rules which work in concert with the default policy to define the overall ruleset for the firewall.

The default policy for a chain can be either DROP or ACCEPT. Security-minded administrators typically implement a default policy of DROP, and only allow specific packets on a case-by-case basis. For example, the following policies block all incoming and outgoing packets on a network gateway:

```
# iptables -P INPUT DROP
# iptables -P OUTPUT DROP
```

It is also recommended that any *forwarded* packets — network traffic that is to be routed from the firewall to its destination node — be denied as well, to restrict internal clients from inadvertent exposure to the Internet. To do this, use the following rule:

```
# iptables -P FORWARD DROP
```

When you have established the default policies for each chain, you can create and save further rules for your particular network and security requirements.

31.3.3. Saving and restoring IPTables rules

Changes to **iptables** are transitory; if the system is rebooted or if the **iptables** service is restarted, the rules are automatically flushed and reset. To save the rules so that they are loaded when the **iptables** service is started, use the following command as the root user:

```
# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

The rules are stored in the file **/etc/sysconfig/iptables** and are applied whenever the service is started or the machine is rebooted.

31.4. Common IPTables filtering

Preventing remote attackers from accessing a LAN is one of the most important aspects of network security. The integrity of a LAN should be protected from malicious remote users through the use of stringent firewall rules.

However, with a default policy set to block all incoming, outgoing, and forwarded packets, it is impossible for the firewall/gateway and internal LAN users to communicate with each other or with external resources.

To allow users to perform network-related functions and to use networking applications, administrators must open certain ports for communication.

For example, to allow access to port 80 on the firewall, append the following rule:

```
# iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

This allows users to browse websites that communicate using the standard port 80. To allow access to secure websites (for example, https://www.example.com/), you also need to provide access to port 443, as follows:

```
# iptables -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

*** **Note**: When creating an **iptables** ruleset, order is important. If a rule specifies that any packets from the 192.168.100.0/24 subnet should be dropped, and this is followed by a rule that allows packets from 192.168.100.13 (which is within the dropped subnet), then the second rule is ignored. The rule to allow packets from 192.168.100.13 must precede the rule that drops the remainder of the subnet.

To insert a rule in a specific location in an existing chain, use the -I option. For example:

```
# iptables -I INPUT 1 -i lo -p all -j ACCEPT
```

This rule is inserted as the first rule in the INPUT chain to allow local loopback device traffic.

There may be times when you require remote access to the LAN. Secure services, for example SSH, can be used for encrypted remote connection to LAN services.

Administrators with PPP-based resources (such as modem banks or bulk ISP accounts), dial-up access can be used to securely circumvent firewall barriers. Because they are direct connections, modem connections are typically behind a firewall/gateway.

For remote users with broadband connections, however, special cases can be made. You can configure **iptables** to accept connections from remote SSH clients. For example, the following rules allow remote SSH access:

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

These rules allow incoming and outbound access for an individual system, such as a single PC directly connected to the Internet or a firewall/gateway. However, they do not allow nodes behind the firewall/gateway to access these services. To allow LAN access to these services, you can use Network Address Translation (NAT) with **iptables** filtering rules.

31.5. Forward and NAT rules

Most ISPs provide only a limited number of publicly routable IP addresses to the organizations they serve.

Administrators must, therefore, find alternative ways to share access to Internet services without giving public IP addresses to every node on the LAN. Using private IP addresses is the most common way of allowing all nodes on a LAN to properly access internal and external network services.

Edge routers (such as firewalls) can receive incoming transmissions from the Internet and route the packets to the intended LAN node. At the same time, firewalls/gateways can also route outgoing requests from a LAN node to the remote Internet service.

This forwarding of network traffic can become dangerous at times, especially with the availability of modern cracking tools that can spoof internal IP addresses and make the remote attacker's machine act as a node on your LAN.

To prevent this, **iptables** provides routing and forwarding policies that can be implemented to prevent abnormal usage of network resources.

The **FORWARD** chain allows an administrator to control where packets can be routed within a LAN. For example, to allow forwarding for the entire LAN (assuming the firewall/gateway is assigned an internal IP address on eth1), use the following rules:

```
# iptables -A FORWARD -i eth1 -j ACCEPT
# iptables -A FORWARD -o eth1 -j ACCEPT
```

This rule gives systems behind the firewall/gateway access to the internal network. The gateway routes packets from one LAN node to its intended destination node, passing all packets through its eth1 device.

***** Note**: By default, the IPv4 policy in Red Hat Enterprise Linux kernels disables support for IP forwarding. This prevents machines that run Red Hat Enterprise Linux from functioning as dedicated edge routers. To enable IP forwarding, use the following command as the root user:

```
# sysctl -w net.ipv4.ip_forward=1
```

```
net.ipv4.ip_forward = 1
```

or

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

This configuration change is only valid for the current session; it does not persist beyond a reboot or network service restart. To permanently set IP forwarding, edit the **/etc/sysctl.conf** file as follows: Locate the following line:

```
net.ipv4.ip_forward = 0
```

Edit it to read as follows:

```
net.ipv4.ip_forward = 1
```

As the root user, run the following command to enable the change to the **sysctl.conf** file:

```
# sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
[output truncated]
```

31.5.1. Postrouting and IP masquerading

Accepting forwarded packets via the firewall's internal IP device allows LAN nodes to communicate with each other; however they still cannot communicate externally to the Internet.

To allow LAN nodes with private IP addresses to communicate with external public networks, configure the firewall for IP masquerading, which masks requests from LAN nodes with the IP address of the firewall's external device (in this case, eth0):

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

This rule uses the NAT packet matching table (**-t nat**) and specifies the built-in POSTROUTING chain for NAT (**-A POSTROUTING**) on the firewall's external networking device (**-o eth0**).

POSTROUTING allows packets to be altered as they are leaving the firewall's external device.

The **-j MASQUERADE** target is specified to mask the private IP address of a node with the external IP address of the firewall/gateway.

* Don't forget to save **iptables** rules

31.5.2. Prerouting

If you have a server on your internal network that you want make available externally, you can use the **-j DNAT** target of the PREROUTING chain in NAT to specify a destination IP address and port where incoming packets requesting a connection to your internal service can be forwarded.

For example, if you want to forward incoming HTTP requests to your dedicated Apache HTTP Server at 172.31.0.23, use the following command as the root user:

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to 172.31.0.23:80
```

This rule specifies that the nat table use the built-in PREROUTING chain to forward incoming HTTP requests exclusively to the listed destination IP address of 172.31.0.23.

***** Note**: If you have a default policy of DROP in your FORWARD chain, you must append a rule to forward all incoming HTTP requests so that destination NAT routing is possible. To do this, use the following command as the root user:

```
# iptables -A FORWARD -i eth0 -p tcp --dport 80 -d 172.31.0.23 -j ACCEPT
```

This rule forwards all incoming HTTP requests from the firewall to the intended destination; the Apache HTTP Server behind the firewall.

31.5.3. DMZs and IPTables

You can create **iptables** rules to route traffic to certain machines, such as a dedicated HTTP or FTP server, in a demilitarized zone (DMZ). A DMZ is a special local subnetwork dedicated to providing services on a public carrier, such as the Internet.

For example, to set a rule for routing incoming HTTP requests to a dedicated HTTP server at 10.0.4.2 (outside of the 192.168.1.0/24 range of the LAN), NAT uses the **PREROUTING** table to forward the packets to the appropriate destination:

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to-destination 10.0.4.2:80
```

With this command, all HTTP connections to port 80 from outside of the LAN are routed to the HTTP server on a network separate from the rest of the internal network. This form of network segmentation can prove safer than allowing HTTP connections to a machine on the network.

If the HTTP server is configured to accept secure connections, then port 443 must be forwarded as well.

31.6. Malicious Software and Spoofed IP Addresses

More elaborate rules can be created that control access to specific subnets, or even specific nodes, within a LAN. You can also restrict certain dubious applications or programs such as trojans, worms, and other client/server viruses from contacting their server.

For example, some trojans scan networks for services on ports from 31337 to 31340 (called the elite ports in cracking terminology).

Since there are no legitimate services that communicate via these non-standard ports, blocking them can effectively diminish the chances that potentially infected nodes on your network independently communicate with their remote master servers.

The following rules drop all TCP traffic that attempts to use port 31337:

```
# iptables -A OUTPUT -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
# iptables -A FORWARD -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
```

You can also block outside connections that attempt to spoof private IP address ranges to infiltrate your LAN.

For example, if your LAN uses the 192.168.1.0/24 range, you can design a rule that instructs the Internet-facing network device (for example, eth0) to drop any packets to that device with an address in your LAN IP range.

Because it is recommended to reject forwarded packets as a default policy, any other spoofed IP address to the external-facing device (eth0) is rejected automatically.

```
# iptables -A FORWARD -s 192.168.1.0/24 -i eth0 -j DROP
```

*** **Note**: There is a distinction between the **DROP** and **REJECT** targets when dealing with appended rules. The **REJECT** target denies access and returns a **connection refused** error to users who attempt to connect to the service. The **DROP** target, as the name implies, drops the packet without any warning.
Administrators can use their own discretion when using these targets.

31.7. IPTables and connection tracking

You can inspect and restrict connections to services based on their connection state. A module within **iptables** uses a method called connection tracking to store information about incoming connections. You can allow or deny access based on the following connection states:

NEW — A packet requesting a new connection, such as an HTTP request.
ESTABLISHED — A packet that is part of an existing connection.
RELATED — A packet that is requesting a new connection but is part of an existing connection. For example, FTP uses port 21 to establish a connection, but data is transferred on a different port (typically port 20).
INVALID — A packet that is not part of any connections in the connection tracking table.

You can use the stateful functionality of **iptables** connection tracking with any network protocol, even if the protocol itself is stateless (such as UDP). The following example shows a rule that uses connection tracking to forward only the packets that are associated with an established connection:

```
# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32. IPTables

http://wiki.centos.org/HowTos/Network/IPTables

*** **Note**: The default firewall mechanism in the 2.4 and later kernels is **iptables**, but **iptables** cannot be used if **ipchains** is already running. If **ipchains** is present at boot time, the kernel issues an error and fails to start **iptables**.
The functionality of **ipchains** is not affected by these errors.

32.1. Packet filtering

The Linux kernel uses the **Netfilter** facility to filter packets, allowing some of them to be received by or pass through the system while stopping others. This facility is built in to the Linux kernel, and has three built-in tables or rules lists, as follows:

- **filter** — The default table for handling network packets.
- **nat** — Used to alter packets that create a new connection and used for Network Address Translation (NAT).
- **mangle** — Used for specific types of packet alteration.

Each table has a group of built-in chains, which correspond to the actions performed on the packet by **netfilter**.

The built-in chains for the filter table are as follows:

INPUT - Applies to network packets that are targeted for the host.
OUTPUT - Applies to locally-generated network packets.
FORWARD - Applies to network packets routed through the host.

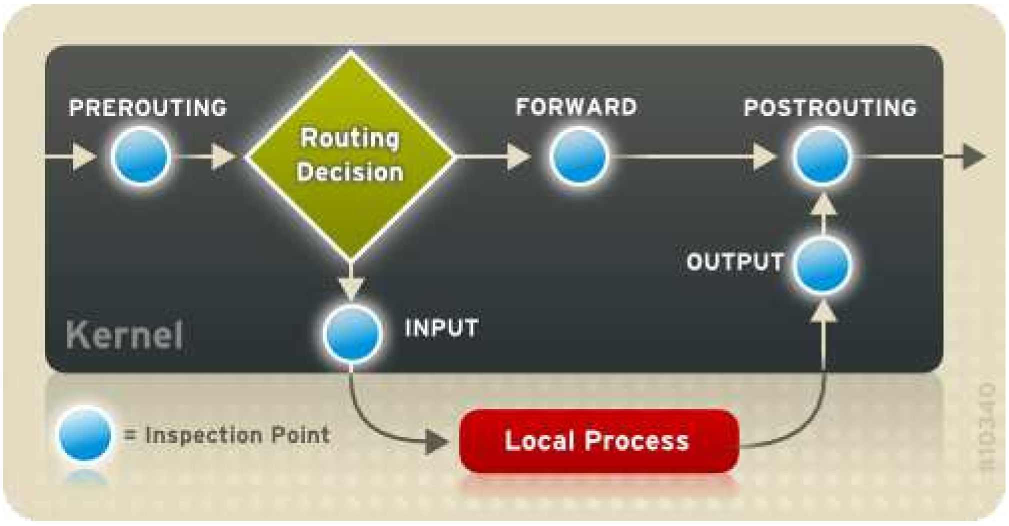
The built-in chains for the nat table are as follows:

PREROUTING - Alters network packets when they arrive.
OUTPUT - Alters locally-generated network packets before they are sent out.
POSTROUTING - Alters network packets before they are sent out.

The built-in chains for the mangle table are as follows:

INPUT — Alters network packets targeted for the host.
OUTPUT — Alters locally-generated network packets before they are sent out.
FORWARD — Alters network packets routed through the host.
PREROUTING — Alters incoming network packets before they are routed.
POSTROUTING — Alters network packets before they are sent out.

Every network packet received by or sent from a Linux system is subject to at least one table. However, a packet may be subjected to multiple rules within each table before emerging at the end of the chain. The structure and purpose of these rules may vary, but they usually seek to identify a packet coming from or going to a particular IP address, or set of addresses, when using a particular protocol and network service. The following image outlines how the flow of packets is examined by the iptables subsystem:



*** **Note**: By default, firewall rules are saved in the **/etc/sysconfig/iptables** or **/etc/sysconfig/ip6tables** files.
The **iptables** service starts before any DNS-related services when a Linux system is booted. This means that firewall rules can only reference numeric IP addresses (for example, 192.168.0.1). Domain names (for example, host.example.com) in such rules produce errors.

Regardless of their destination, when packets match a particular rule in one of the tables, a target or action is applied to them. If the rule specifies an **ACCEPT** target for a matching packet, the packet skips the rest of the rule checks and is allowed to continue to its destination. If a rule specifies a **DROP** target, that packet is refused access to the system and nothing is sent back to the host that sent the packet. If a rule specifies a **QUEUE** target, the packet is passed to user-space. If a rule specifies the optional **REJECT** target, the packet is dropped, but an error packet is sent to the packet's originator.
Every chain has a default policy to **ACCEPT**, **DROP**, **REJECT**, or **QUEUE**. If none of the rules in the chain apply to the packet, then the packet is dealt with in accordance with the default policy.
The **iptables** command configures these tables, as well as sets up new tables if necessary.

32.2. Command options for IPTables

Rules for filtering packets are created using the **iptables** command. The following aspects of the packet are most often used as criteria:

Packet Type - Specifies the type of packets the command filters.
Packet Source/Destination - Specifies which packets the command filters based on the source or destination of the packet.
Target - Specifies what action is taken on packets matching the above criteria.

The options used with specific **iptables** rules must be grouped logically, based on the purpose and conditions of the overall rule, for the rule to be valid. The remainder of this section explains commonly-used options for the **iptables** command.

32.2.1. Structure of the IPTables command options

Many **iptables** commands have the following structure:

```
iptables [-t <table-name>] <command> <chain-name> \
         <parameter-1> <option-1> \
         <parameter-n> <option-n>
```

<***table-name***> - Specifies which table the rule applies to. If omitted, the **filter** table is used.
<***command***> - Specifies the action to perform, such as appending or deleting a rule.
<***chain-name***> - Specifies the chain to edit, create, or delete.
<***parameter***>-<***option***> pairs - Parameters and associated options that specify how to process a packet that matches the rule.

The length and complexity of an **iptables** command can change significantly, based on its purpose.

For example, a command to remove a rule from a chain can be very short:

```
# iptables -D <chain-name> <line-number>
```

In contrast, a command that adds a rule which filters packets from a particular subnet using a variety of specific parameters and options can be rather long. When constructing **iptables** commands, it is important to remember that some parameters and options require further parameters and options to construct a valid rule. This can produce a cascading effect, with the further parameters requiring yet more parameters. Until every parameter and option that requires another set of options is satisfied, the rule is not valid.
Type **iptables -h** to view a comprehensive list of **iptables** command structures.

32.2.2. Command options

Command options instruct **iptables** to perform a specific action. Only one command option is allowed per **iptables** command. With the exception of the help command, all commands are written in upper-case characters.

The **iptables** commands are as follows:

-A - Appends the rule to the end of the specified chain. Unlike the -I option described below, it does not take an integer argument. It always appends the rule to the end of the specified chain.

-D <**integer**> | <**rule**> - Deletes a rule in a particular chain by number (such as **5** for the fifth rule in a chain), or by rule specification. The rule specification must exactly match an existing rule.

-E - Renames a user-defined chain. A user-defined chain is any chain other than the default, pre-existing chains. (Refer to the **-N** option, below, for information on creating user-defined chains.) This is a cosmetic change and does not affect the structure of the table.

*** **Note**: If you attempt to rename one of the default chains, the system reports a **Match not found** error. You cannot rename the default chains.

-F - Flushes the selected chain, which effectively deletes every rule in the chain. If no chain is specified, this command flushes every rule from every chain.

-h - Provides a list of command structures, as well as a quick summary of command parameters and options.

-I [<**integer**>] - Inserts the rule in the specified chain at a point specified by a user-defined integer argument. If no argument is specified, the rule is inserted at the top of the chain.

*** **Note**: As noted above, the order of rules in a chain determines which rules apply to which packets. This is important to remember when adding rules using either the **-A** or **-I** option.

This is especially important when adding rules using the **-I** with an integer argument. If you specify an existing number when adding a rule to a chain, **iptables** adds the new rule before (or above) the existing rule.

-L - Lists all of the rules in the chain specified after the command. To list all rules in all chains in the default filter table, do not specify a chain or table. Otherwise, the following syntax should be used to list the rules in a specific chain in a particular table:

```
# iptables -L <chain-name> -t <table-name>
```

-N - Creates a new chain with a user-specified name. The chain name must be unique, otherwise an error message is displayed.

-P - Sets the default policy for the specified chain, so that when packets traverse an entire chain without matching a rule, they are sent to the specified target, such as ACCEPT or DROP.

-R - Replaces a rule in the specified chain. The rule's number must be specified after the chain's name. The first rule in a chain corresponds to rule number one.

-X - Deletes a user-specified chain. You cannot delete a built-in chain.

-Z - Sets the byte and packet counters in all chains for a table to zero.

32.2.3. IPTables parameter options

Certain **iptables** commands, including those used to add, append, delete, insert, or replace rules within a particular chain, require various parameters to construct a packet filtering rule.

-c - Resets the counters for a particular rule. This parameter accepts the **PKTS** and **BYTES** options to specify which counter to reset.

-d - Sets the destination IP address or network of a packet that matches the rule. When matching a network, the following IP address/netmask formats are supported:

- **N.N.N.N/M.M.M.M** - Where **N.N.N.N** is the IP address range and **M.M.M.M** is the netmask.
- **N.N.N.N/M** - Where **N.N.N.N** is the IP address range and **M** is the bitmask.

-f - Applies this rule only to fragmented packets. You can use the exclamation point character (**!**) option before this parameter to specify that only unfragmented packets are matched.

*** **Note**: Distinguishing between fragmented and unfragmented packets is desirable, despite fragmented packets being a standard part of the IP protocol.

Originally designed to allow IP packets to travel over networks with differing frame sizes, these days fragmentation is more commonly used to generate DoS attacks using mal-formed packets. It's also worth noting that IPv6 disallows fragmentation entirely.

-i - Sets the incoming network interface, such as **eth0** or **ppp0**. With **iptables**, this optional parameter may only be used with the INPUT and FORWARD chains when used with the **filter** table and the PREROUTING chain with the **nat** and **mangle** tables.

This parameter also supports the following special options:

- Exclamation point character (**!**) - Reverses the directive, meaning any specified interfaces are excluded from this rule.
- Plus character (+) - A wildcard character used to match all interfaces that match the specified string. For example, the parameter **-i eth+** would apply this rule to any Ethernet interfaces but exclude any other interfaces, such as **ppp0**.

If the **-i** parameter is used but no interface is specified, then every interface is affected by the rule.

-j - Jumps to the specified target when a packet matches a particular rule. The standard targets are **ACCEPT**, **DROP**, **QUEUE**, and **RETURN**.

Extended options are also available through modules loaded by default with the Red Hat Enterprise Linux **iptables** RPM package. Valid targets in these modules include **LOG**, **MARK**, and **REJECT**, among others. Refer to the **iptables** man page for more information about these and other targets.

This option can also be used to direct a packet matching a particular rule to a user-defined chain outside of the current chain so that other rules can be applied to the packet.

If no target is specified, the packet moves past the rule with no action taken. The counter for this rule, however, increases by one.

-o - Sets the outgoing network interface for a rule. This option is only valid for the OUTPUT and FORWARD chains in the **filter** table, and the POSTROUTING chain in the **nat** and **mangle** tables. This parameter accepts the same options as the incoming network interface parameter (**-i**).

-p <**protocol**> - Sets the IP protocol affected by the rule. This can be either **icmp**, **tcp**, **udp**, or all, or it can be a numeric value, representing one of these or a different protocol. You can also use any protocols listed in the **/etc/protocols** file.

The "**all**" protocol means the rule applies to every supported protocol. If no protocol is listed with this rule, it defaults to "**all**".

-s - Sets the source for a particular packet using the same syntax as the destination (**-d**) parameter.

32.2.4. IPTables match options

Different network protocols provide specialized matching options which can be configured to match a particular packet using that protocol. However, the protocol must first be specified in the **iptables** command. For example, **-p** <**protocol-name**> enables options for the specified protocol. Note that you can also use the protocol ID, instead of the protocol name. Refer to the following examples, each of which has the same effect:

```
# iptables -A INPUT -p icmp --icmp-type any -j ACCEPT
# iptables -A INPUT -p 5813 --icmp-type any -j ACCEPT
```

Service definitions are provided in the **/etc/services** file. For readability, it is recommended that you use the service names rather than the port numbers.

*****Warning**: Secure the **/etc/services** file to prevent unauthorized editing. If this file is editable, crackers can use it to enable ports on your machine you have otherwise closed. To secure this file, run the following commands as root:

```
# chown root.root /etc/services
# chmod 0644 /etc/services
# chattr +i /etc/services
```

32.2.5. Target options

When a packet has matched a particular rule, the rule can direct the packet to a number of different targets which determine the appropriate action. Each chain has a default target, which is used if none of the rules on that chain match a packet or if none of the rules which match the packet specify a target.

The following are the standard targets:

<**user-defined-chain**> — A user-defined chain within the table. User-defined chain names must be unique. This target passes the packet to the specified chain.

ACCEPT - Allows the packet through to its destination or to another chain.

DROP - Drops the packet without responding to the requester. The system that sent the packet is not notified of the failure.

QUEUE - The packet is queued for handling by a user-space application.

RETURN - Stops checking the packet against rules in the current chain. If the packet with a **RETURN** target matches a rule in a chain called from another chain, the packet is returned to the first chain to resume rule checking where it left off. If the **RETURN** rule is used on a built-in chain and the packet cannot move up to its previous chain, the default target for the current chain is used.

In addition, extensions are available which allow other targets to be specified. These extensions are called target modules or match option modules and most only apply to specific tables and situations.

Many extended target modules exist, most of which only apply to specific tables or situations. Some of the most popular target modules included by default in Red Hat Enterprise Linux are:

LOG - Logs all packets that match this rule. Because the packets are logged by the kernel, the **/etc/syslog.conf** file determines where these log entries are written. By default, they are placed in the **/var/log/messages** file.

Additional options can be used after the **LOG** target to specify the way in which logging occurs:

--log-level — Sets the priority level of a logging event. Refer to the syslog.conf man page for a list of priority levels.

--log-ip-options — Logs any options set in the header of an IP packet.

--log-prefix — Places a string of up to 29 characters before the log line when it is written. This is useful for writing syslog filters for use in conjunction with packet logging.

***** Note**: Due to an issue with this option, you should add a trailing space to the **log-prefix** value.

--log-tcp-options — Logs any options set in the header of a TCP packet.

--log-tcp-sequence — Writes the TCP sequence number for the packet in the log.

REJECT — Sends an error packet back to the remote system and drops the packet.

The **REJECT** target accepts **--reject-with** <***type***> (where <***type***> is the rejection type) allowing more detailed information to be returned with the error packet. The message **port-unreachable** is the default error type given if no other option is used. Refer to the **iptables** man page for a full list of <**type**> options.

32.2.6. Listing options

The default list command, **iptables -L** [<**chain-name**>], provides a very basic overview of the default filter table's current chains. Additional options provide more information:

-v — Displays verbose output, such as the number of packets and bytes each chain has processed, the number of packets and bytes each rule has matched, and which interfaces apply to a particular rule.

-x — Expands numbers into their exact values. On a busy system, the number of packets and bytes processed by a particular chain or rule may be abbreviated to **Kilobytes**, **Megabytes**, or **Gigabytes**. This option forces the full number to be displayed.

-n — Displays IP addresses and port numbers in numeric format, rather than the default hostname and network service format.

--line-numbers — Lists rules in each chain next to their numeric order in the chain. This option is useful when attempting to delete the specific rule in a chain or to locate where to insert a rule within a chain.

-t <**table-name**> — Specifies a table name. If omitted, defaults to the filter table.

32.3. Saving IPTables rules

Rules created with the **iptables** command are stored in memory. If the system is restarted before saving the **iptables** rule set, all rules are lost. For **netfilter** rules to persist through a system reboot, they need to be saved. To save **netfilter** rules, type the following command as root:

```
# /sbin/service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

This executes the **iptables** init script, which runs the **/sbin/iptables-save** program and writes the current **iptables** configuration to **/etc/sysconfig/iptables**. The existing **/etc/sysconfig/iptables** file is saved as **/etc/sysconfig/iptables.save**.

The next time the system boots, the **iptables** init script reapplies the rules saved in **/etc/sysconfig/iptables** by using **the /sbin/iptables-restore** command.

While it is always a good idea to test a new **iptables** rule before committing it to the **/etc/sysconfig/iptables** file, it is possible to copy **iptables** rules into this file from another system's version of this file. This provides a quick way to distribute sets of **iptables** rules to multiple machines.

You can also save the **iptables** rules to a separate file for distribution, backup, or other purposes. To do so, run the following command as root:

```
# iptables-save > <filename>
```

*** **Note**: If distributing the **/etc/sysconfig/iptables** file to other machines, type **/sbin/service iptables restart** for the new rules to take effect.

*** **Note**: Note the difference between the **iptables** command (**/sbin/iptables**), which is used to manipulate the tables and chains that constitute the **iptables** functionality, and the **iptables** service (**/sbin/service iptables**), which is used to enable and disable the **iptables** service itself.

32.4. IPTables control scripts

There are two basic methods for controlling **iptables** in Red Hat Enterprise Linux:

1.- **Firewall Configuration Tool** (**system-config-firewall**) — A graphical interface for creating, activating, and saving basic firewall rules.

2.- **/sbin/service iptables** <**option**> — Used to manipulate various functions of **iptables** using its initscript. The following options are available:

- **start** — If a firewall is configured (that is, **/etc/sysconfig/iptables** exists), all running **iptables** are stopped completely and then started using the **/sbin/iptables-restore** command. This option only works if the **ipchains** kernel module is not loaded. To check if this module is loaded, type the following command as root:

```
# lsmod | grep ipchains
```

If this command returns no output, it means the module is not loaded. If necessary, use the **/sbin/rmmod** command to remove the module.

- **stop** — If a firewall is running, the firewall rules in memory are flushed, and all iptables modules and helpers are unloaded.

If the **IPTABLES_SAVE_ON_STOP** directive in the **/etc/sysconfig/iptables-config** configuration file is changed from its default value to **yes**, current rules are saved to **/etc/sysconfig/iptables** and any existing rules are moved to the file **/etc/sysconfig/iptables.save**.

- **restart** — If a firewall is running, the firewall rules in memory are flushed, and the firewall is started again if it is configured in **/etc/sysconfig/iptables**. This option only works if the **ipchains** kernel module is not loaded.

If the **IPTABLES_SAVE_ON_RESTART** directive in the **/etc/sysconfig/iptables-config** configuration file is changed from its default value to yes, current rules are saved to **/etc/sysconfig/iptables** and any existing rules are moved to the file **/etc/sysconfig/iptables.save**.

- **status** — Displays the status of the firewall and lists all active rules.

The default configuration for this option displays IP addresses in each rule. To display domain and hostname information, edit the **/etc/sysconfig/iptables-config** file and change the value of **IPTABLES_STATUS_NUMERIC** to no.

- **panic** — Flushes all firewall rules. The policy of all configured tables is set to **DROP**.

This option could be useful if a server is known to be compromised. Rather than physically disconnecting from the network or shutting down the system, you can use this option to stop all further network traffic but leave the machine in a state ready for analysis or other forensics.

- **save** — Saves firewall rules to **/etc/sysconfig/iptables** using **iptables-save**.

32.4.1. IPTables control scripts configuration files

The behavior of the **iptables** initscripts is controlled by the **/etc/sysconfig/iptables-config** configuration file. The following is a list of directives contained in this file:

IPTABLES_MODULES - Specifies a space-separated list of additional **iptables** modules to load when a firewall is activated. These can include connection tracking and NAT helpers.

IPTABLES_MODULES_UNLOAD - Unloads modules on restart and stop. This directive accepts the following values:

yes - The default value. This option must be set to achieve a correct state for a firewall restart or stop.

no - This option should only be set if there are problems unloading the netfilter modules.

IPTABLES_SAVE_ON_STOP - Saves current firewall rules to **/etc/sysconfig/iptables** when the firewall is stopped. This directive accepts the following values:

yes - Saves existing rules to **/etc/sysconfig/iptables** when the firewall is stopped, moving the previous version to the **/etc/sysconfig/iptables.save** file.

no - The default value. Does not save existing rules when the firewall is stopped.

IPTABLES_SAVE_ON_RESTART - Saves current firewall rules when the firewall is restarted. This directive accepts the following values:

yes - Saves existing rules to **/etc/sysconfig/iptables** when the firewall is restarted, moving the previous version to the **/etc/sysconfig/iptables.save** file.

no - The default value. Does not save existing rules when the firewall is restarted.

IPTABLES_SAVE_COUNTER - Saves and restores all packet and byte counters in all chains and rules. This directive accepts the following values:

yes - Saves the counter values.

no - The default value. Does not save the counter values.

IPTABLES_STATUS_NUMERIC — Outputs IP addresses in numeric form instead of domain or hostnames. This directive accepts the following values:

yes - The default value. Returns only IP addresses within a status output.

no - Returns domain or hostnames within a status output.

33. Most common port numbers

Port#/Protocol	**Description**
20/tcp	FTP File Transfer Protocol - data
21/tcp	FTP File Transfer Protocol - control
22/tcp	SSH, scp, sftp
25/tcp	SMTP Simple Mail Transfer Protocol
53/tcp	DNS Domain Name System
53/udp	DNS Domain Name System
80/tcp	HTTP HyperText Transfer Protocol, WWW
443/tcp	HTTPS, SSL
123/tcp	NTP Network Time Protocol
123/udp	NTP Network Time Protocol
139/tcp	SMB
445/tcp	Microsoft-DS file sharing
514/tcp	rsyslog
514/udp	rsyslog
2049/tcp	NFS
3260/tcp	iscsi

34. Managing SELinux security

SELinux has two major components on the system. There's the kernel mechanism which is enforcing a bunch of access rules which apply to processes and files. And secondly, there's file labels: every file on your system has extra labels attached to it which tie-in with those access rules. Run ls -Z and you'll see what I mean.

34.1. Disabling/Re-enabling SELinux

Be aware that by disabling SELinux you will be removing a security mechanism on your system. Think about this carefully and, if your system is on the Internet and accessed by the public, then think about it some more. From a secure point of view applications should be fixed to work with SELinux, rather than disabling the OS security mechanism.

You need to decide if you want to disable SELinux temporarily to test the problem, or permanently switch it off. It may also be a better option to make changes to the policy to permit the operations that are being blocked. For the operating system as a whole, there are two kinds of disabling:

- **Permissive** - switch the SELinux kernel into a mode where every operation is allowed. Operations that would be denied are allowed and a message is logged identifying that it would be denied. The mechanism that defines labels for files which are being created/changed is still active.

- **Disabled** - SELinux is completely switched off in the kernel. This allows all operations to be permitted, and also disables the process which decides what to label files & processes with.

Disabling SELinux could lead to problems if you want to re-enable it again later. When the system runs with file labeling disable it will create files with no label - which could cause problems if the system is booted into Enforcement mode. A full re-labeling of the file system will be necessary.

The third mode of SELinux is being active:

- **Enforcing** - SELinux policy is enforced. SELinux denies access based on SELinux policy rules.

34.1.1. Viewing the status of SELinux

The **sestatus** command provides a configurable view into the status of SELinux. The simplest form of this command shows the following information:

```
# sestatus
SELinux status:            enabled
SELinuxfs mount:           /selinux
Current mode:              enforcing
Mode from config file:     enforcing
Policy version:            21
Policy from config file:   targeted
```

The **-v** option includes information about the security contexts of a series of files that are specified in **/etc/sestatus.conf**:

```
# sestatus -v
[files]
/etc/passwd
/etc/shadow
```

```
/bin/bash
/bin/login
/bin/sh
/sbin/agetty
/sbin/init
/sbin/mingetty
/usr/sbin/sshd
/lib/libc.so.6
/lib/ld-linux.so.2
/lib/ld.so.1

[process]
/sbin/mingetty
/sbin/agetty
/usr/sbin/sshd
```

The **-b** displays the current state of booleans. You can use this in combination with grep or other tools to determine the status of particular booleans:

```
# sestatus -b | grep httpd | grep on$
httpd_builtin_scripting        on
httpd_disable_trans            on
httpd_enable_cgi               on
httpd_enable_homedirs          on
httpd_unified                  on
```

34.1.2. Temporarily switch off enforcement

You can switch the system into permissive mode with the following command:

```
# echo  0  > /selinux/enforce
```

You'll need to be logged in as root, and in the sysadm_r role:

```
# newrole  -r  sysadm_r
```

To switch back into enforcing mode:

```
# echo  1  > /selinux/enforce
```

Use the **setenforce** command to change between permissive and enforcing modes at runtime. Use **setenforce 0** to enter permissive mode; use **setenforce 1** to enter enforcing mode.

To check what mode the system is in,

```
# cat  /selinux/enforce
```

which will print a "0" or "1" for permissive or enforcing - probably printed at the beginning of the line of the command prompt.

```
# sestatus | grep -i mode
Current mode:          enforcing
Mode from config file:  permissive
```

34.1.3. Permanently permissive

If you want the system to always start in permissive mode, do the following.

Edit **/etc/selinux/config** and you will see some lines like this:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
SELINUXTYPE=targeted
```

Just change **SELINUX=enforcing** to **SELINUX=permissive**, and you're done. A reboot is necessary to enable the desired mode.

For the other Linuxes which don't have the **/etc/selinux/config** file, you just need to edit the kernel boot line, usually in **/boot/grub/grub.conf** if you're using the GRUB boot loader. On the kernel line, add **enforcing=0** at the end. For example:

```
title SE-Linux Test System
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.20-selinux-2003040709 ro root=/dev/hda1 nousb selinux=1 enforcing=0
    #initrd /boot/initrd-2.4.20-selinux-2003040709.img
```

You will have to reboot to change SELinux mode, you just can't do it while the system is running.

34.1.4. Fully disabling SELinux

Fully disabling SELinux goes one step further than just switching into permissive mode. Disabling will completely disable all SELinux functions including file and process labeling.

Edit **/etc/selinux/config** and change the SELINUX line to **SELINUX=disabled**:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
```

```
SELINUXTYPE=targeted
```

And then reboot the system.

For the other Linuxes which don't have the **/etc/selinux/config** file, you just need to edit the kernel boot line, usually in **/boot/grub/grub.conf**, if you're using the GRUB boot loader. On the kernel line, add **selinux=0** at the end. For example:

```
title SE-Linux Test System
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.20-selinux-2003040709 ro root=/dev/hda1 nousb selinux=0
        #initrd /boot/initrd-2.4.20-selinux-2003040709.img
```

You will have to reboot to disable SELinux, you just can't do it while the system is running.

34.1.5. Re-enabling SELinux

Use the **rpm -qa | grep selinux, rpm -q policycoreutils**, and **rpm -qa setroubleshoot** commands to confirm that the SELinux packages are installed. This assumes the following packages are installed: ***selinux-policy-targeted***, ***selinux-policy***, ***libselinux***, ***libselinux-python***, ***libselinux-utils***, ***policycoreutils***, ***policycoreutils-python***, ***setroubleshoot***, ***setroubleshoot-server***, ***setroubleshoot-plugins***. The following are optional: ***policycoreutils-gui***, ***setroubleshoot***, ***selinux-policy-devel***, and ***mcstrans***.

If you've disabled SELinux as in the section above, and you want to enable it again then you've got a bit of work to do. The problem will be that files created or changed when SELinux was disabled won't have the correct file labels on them; if you just reboot in enforcing mode then a lot of stuff won't work properly.

What you need to do is to enable SELinux by editing **/etc/selinux/config** or by adding **selinux=1** to the kernel boot line, then change into permissive mode, then relabel everything, and then reboot into (or simply switch to) enforcing mode.

After booting into permissive mode, run:

```
# fixfiles  relabel
```

Alternatively, you can run **touch /.autorelabel** and reboot or put **autorelabel** on the boot command line - in both cases the file system gets a full relabel early in the boot process. Note that this can take quite some time for systems with a large number of files.

After relabeling the filesystem, you can switch to enforcing mode and your system should be fully enforcing again.

34.2. Targeted policy

Targeted policy is the default SELinux policy used in Red Hat Enterprise Linux. When using targeted policy, processes that are targeted run in a confined domain, and processes that are not targeted run in an unconfined domain. For example, by default, logged in users and system processes started by init run in the **initrc_t** domain run in the **unconfined_t** domain - both of these domains are unconfined.

Unconfined domains (as well as confined domains) are subject to executable and writeable memory checks. By default, subjects running in an unconfined domain can not allocate writeable memory and execute it. This reduces vulnerability to **buffer overflow attacks**. These memory checks are disabled by setting Booleans, which allow the SELinux policy to be modified at runtime.

34.2.1. Confined/Unconfined processes

Almost every service that listens on a network, such as *sshd* or *httpd*, is confined in Red Hat Enterprise Linux. Also, most processes that run as the Linux root user and perform tasks for users, such as the **passwd** application, are confined. When a process is confined, it runs in its own domain, such as the *httpd* process running in the **httpd_t** domain. If a confined process is compromised by an attacker, depending on SELinux policy configuration, an attacker's access to resources and the possible damage they can do is limited.

Unconfined processes run in unconfined domains, for example, init programs run in the unconfined **initrc_t** domain, unconfined kernel processes run in the **kernel_t** domain, and unconfined Linux users run in the **unconfined_t** domain. For unconfined processes, SELinux policy rules are applied, but policy rules exist that allow processes running in unconfined domains almost all access. Processes running in unconfined domains fall back to using DAC rules exclusively. If an unconfined process is compromised, SELinux does not prevent an attacker from gaining access to system resources and data, but of course, DAC rules are still used. SELinux is a security enhancement on top of DAC rules - it does not replace them.

34.2.2. Confined/Unconfined users

Each Linux user is mapped to a SELinux user via SELinux policy. This allows Linux users to inherit the restrictions on SELinux users. This Linux user mapping is seen by running the **semanage login -l** command as the Linux root user:

```
# /usr/sbin/semanage login -l
Login Name          SELinux User        MLS/MCS Range
__default__         unconfined_u        s0-s0:c0.c1023
root                unconfined_u        s0-s0:c0.c1023
system_u            system_u            s0-s0:c0.c1023
```

In Red Hat Enterprise Linux 6, Linux users are mapped to the SELinux **__default__** login by default (which is mapped to the SELinux **unconfined_u** user). The following defines the default-mapping:

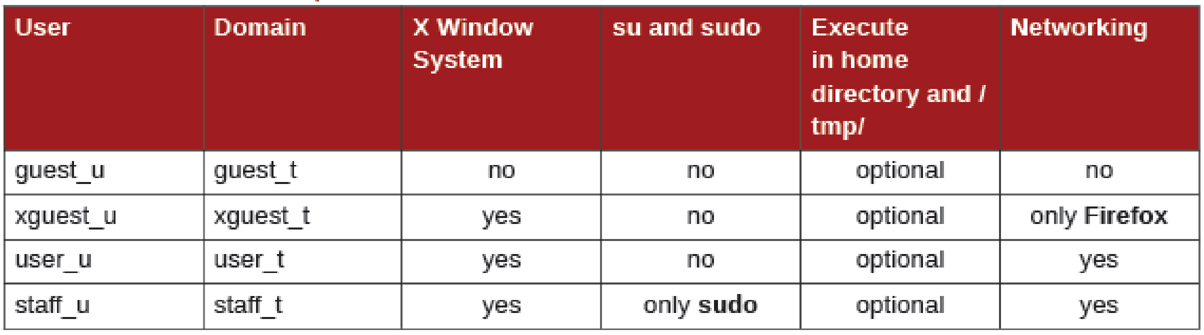
```
__default__         unconfined_u        s0-s0:c0.c1023
```

Confined and unconfined Linux users are subject to executable and writeable memory checks, and are also restricted by MCS (and MLS, if the MLS policy is used). If unconfined Linux users execute an application that SELinux policy defines can transition from the **unconfined_t** domain to its own confined domain, unconfined Linux users are still subject to the restrictions of that confined domain.

The security benefit of this is that, even though a Linux user is running unconfined, the application remains confined, and therefore, the exploitation of a flaw in the application can be limited by policy.

***** Note**: This does not protect the system from the user. Instead, the user and the system are being protected from possible damage caused by a flaw in the application.

The following confined SELinux users are available in Red Hat Enterprise Linux 6:



User	Domain	X Window System	su and sudo	Execute in home directory and /tmp/	Networking
guest_u	guest_t	no	no	optional	no
xguest_u	xguest_t	yes	no	optional	only **Firefox**
user_u	user_t	yes	no	optional	yes
staff_u	staff_t	yes	only **sudo**	optional	yes

- Linux users in the **guest_t**, **xguest_t**, and **user_t** domains can only run set user ID (setuid) applications if SELinux policy permits it (such as **passwd**). They can not run the **su** and **/usr/bin/sudo** setuid applications, and therefore, can not use these applications to become the Linux root user.

- Linux users in the **guest_t** domain have no network access, and can only log in via a terminal (including **ssh**; they can log in via **ssh**, but can not use **ssh** to connect to another system).

- The only network access Linux users in the **xguest_t** domain have is Firefox connecting to web pages.

- Linux users in the **xguest_t**, **user_t** and **staff_t** domains can log in via the X Window System and a terminal.

- By default, Linux users in the **staff_t** domain do not have permissions to execute applications with **/usr/bin/sudo**. These permissions must be configured by an administrator.

By default, Linux users in the **guest_t** and **xguest_t** domains can not execute applications in their home directories or **/tmp/**, preventing them from executing applications (which inherit users' permissions) in directories they have write access to. This helps prevent flawed or malicious applications from modifying files users' own.

By default, Linux users in the **user_t** and **staff_t** domains can execute applications in their home directories and **/tmp/**.

34.3. Use boolean settings to modify system SELinux settings

Booleans allow parts of SELinux policy to be changed at runtime, without any knowledge of SELinux policy writing. This allows changes, such as allowing services access to NFS file systems, without reloading or recompiling SELinux policy.

34.3.1. Listing booleans

For a list of Booleans, an explanation of what each one is, and whether they are on or off, run the **semanage boolean -l** command as the Linux root user. The following example does not list all Booleans:

```
# /usr/sbin/semanage  boolean  -l
SELinux                 boolean          Description
ftp_home_dir            (off  ,  off)    Allow ftp to read & write files in the user home dirs.
xen_use_nfs             (off  ,  off)    Allow xen to manage nfs files
[.....]
```

The **SELinux boolean** column lists Boolean names. The **Description** column lists whether the Booleans are on or off, and what they do.

In the following example, the **ftp_home_dir** Boolean is off, preventing the FTP daemon (vsftpd) from reading and writing to files in user home directories:

```
ftp_home_dir -> off Allow ftp to read and write files in the user home directories
```

The **getsebool -a** command lists Booleans, whether they are on or off, but does not give a description of each one. The following example does not list all Booleans:

```
# /usr/sbin/getsebool -a
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
```

Run the **getsebool *boolean-name*** command to only list the status of the boolean-name Boolean:

```
# /usr/sbin/getsebool  allow_console_login
allow_console_login --> off
```

Use a space-separated list to list multiple Booleans:

```
# /usr/sbin/getsebool  allow_console_login  allow_cvs_read_shadow
allow_console_login --> off
allow_cvs_read_shadow --> off
```

34.3.2. Configuring booleans

The **setsebool *boolean-name x*** command turns Booleans on or off, where *boolean-name* is a Boolean name, and x is either **on** to turn the Boolean on, or **off** to turn it off.

The following example demonstrates configuring the **httpd_can_network_connect_db** Boolean:

1. By default, the **httpd_can_network_connect_db** Boolean is off, preventing Apache HTTP Server scripts and modules from connecting to database servers:

```
# /usr/sbin/getsebool  httpd_can_network_connect_db
httpd_can_network_connect_db --> off
```

2. To temporarily enable Apache HTTP Server scripts and modules to connect to database servers, run the **setsebool httpd_can_network_connect_db on** command as the Linux root user.

3. Use the **getsebool httpd_can_network_connect_db** command to verify the Boolean is turned on:

```
# /usr/sbin/getsebool  httpd_can_network_connect_db
httpd_can_network_connect_db --> on
```

This allows Apache HTTP Server scripts and modules to connect to database servers.

4. This change is not persistent across reboots. To make changes persistent across reboots, run the **setsebool -P *boolean-name* on** command as the Linux root user:

```
# /usr/sbin/setsebool -P httpd_can_network_connect_db on
```

5. To temporarily revert to the default behavior, as the Linux root user, run the **setsebool httpd_can_network_connect_db off** command. For changes that persist across reboots, run the **setsebool -P httpd_can_network_connect_db off** command.

34.3.3. Booleans for NFS and CIFS

By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **nfs_t** type. Also, by default, Samba shares mounted on the client side are labeled with a default context defined by policy. In common policies, this default context uses the **cifs_t** type.

Depending on policy configuration, services may not be able to read files labeled with the **nfs_t** or **cifs_t** types. This may prevent file systems labeled with these types from being mounted and then read or exported by other services. Booleans can be turned on or off to control which services are allowed to access the **nfs_t** and **cifs_t** types.

The **setsebool** and **semanage** commands must be run as the Linux root user. The **setsebool -P** command makes persistent changes. Do not use the **-P** option if you do not want changes to persist across reboots:

Apache HTTP Server

To allow access to NFS file systems (files labeled with the **nfs_t** type):

```
# /usr/sbin/setsebool -P httpd_use_nfs on
```

To allow access to Samba file systems (files labeled with the **cifs_t** type):

```
# /usr/sbin/setsebool -P httpd_use_cifs on
```

Samba

To export NFS file systems:

```
# /usr/sbin/setsebool -P samba_share_nfs on
```

FTP (vsftpd)

To allow access to NFS file systems:

```
# /usr/sbin/setsebool -P allow_ftpd_use_nfs on
```

To allow access to Samba file systems:

```
# /usr/sbin/setsebool -P allow_ftpd_use_cifs on
```

Other Services

For a list of NFS related Booleans for other services:

```
# /usr/sbin/semanage boolean -l | grep nfs
```

For a list of Samba related Booleans for other services:

```
# /usr/sbin/semanage boolean -l | grep cifs
```

*** **Note**: These Booleans exist in SELinux policy as shipped with Red Hat Enterprise Linux 6. They may not exist in policy shipped with other versions of Red Hat Enterprise Linux or other operating systems.

34.4. Relabeling a File System

You may never need to relabel an entire file system. This usually occurs only when labeling a file system for SELinux for the first time, or when switching between different types of policy, such as changing from the targeted to the strict policy.

Relabeling a File System Using init

The recommended method for relabeling a file system is to reboot the machine. This allows the **init** process to perform the relabeling, ensuring that applications have the correct labels when they are started and that they are started in the right order. If you relabel a file system without rebooting, some processes may continue running with an incorrect context. Manually ensuring that all the daemons are restarted and running in the correct context can be difficult.

Use the following procedure to relabel a file system using this method.

```
# touch /.autorelabel
# reboot
```

At boot time, **init.rc** checks for the existence of **/.autorelabel**. If this file exists, SELinux performs a complete file system relabel (using the **/sbin/fixfiles -f -F relabel** command), and then deletes **/.autorelabel**

Relabeling a File System Using fixfiles

It is possible to relabel a file system using the fixfiles command, or to relabel based on the RPM database:
Use the following command to relabel a file system only using the **fixfiles** command:

```
# fixfiles relabel
```

Use the following command to relabel a file list based on the RPM database:

```
# fixfiles  -R  <packagename>  restore
```

Using **fixfiles** to restore contexts from packages is safer and quicker.

*** **Warning**: Running **fixfiles** on the entire file system without rebooting may make the system unstable. If the relabeling operation applies a new policy that is different from the policy that was in place when the system booted, existing processes may be running in incorrect and insecure domains. For example, a process could be in a domain that is not an allowed transition for that process in the new policy, granting unexpected permissions to that process alone.
In addition, one of the options to **fixfiles relabel** prompts for approval to empty **/tmp/** because it is not possible to reliably relabel **/tmp/**. Since **fixfiles** is run as root, temporary files that applications are relying upon are erased. This could make the system unstable or behave unexpectedly.

34.5. Labeling problems

On systems running SELinux, all processes and files are labeled with a label that contains security-relevant information. This information is called the SELinux context. If these labels are wrong, access may be denied. If an application is labeled incorrectly, the process it transitions to may not have the correct label, possibly causing SELinux to deny access, and the process being able to create mislabeled files.

A common cause of labeling problems is when a non-standard directory is used for a service. For example, instead of using **/var/www/html/** for a website, an administrator wants to use **/srv/myweb/**. On Red Hat Enterprise Linux 6, the **/srv/** directory is labeled with the **var_t** type. Files and directories created and **/srv/** inherit this type. Also, newly-created top-level directories (such as **/myserver/**) may be labeled with the **default_t** type. SELinux prevents the Apache HTTP Server (**httpd**) from accessing both of these types. To allow access, SELinux must know that the files in **/srv/myweb/** are to be accessible to httpd:

```
# /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/srv/myweb(/.*)?"
```

This **semanage** command adds the context for the **/srv/myweb/** directory (and all files and directories under it) to the SELinux file-context configuration. The **semanage** command does not change the context. As the Linux root user, run the **restorecon** command to apply the changes:

```
# /sbin/restorecon -R -v /srv/myweb
```

34.5.1. What is the correct context?

The **matchpathcon** command checks the context of a file path and compares it to the default label for that path. The following example demonstrates using **matchpathcon** on a directory that contains incorrectly labeled files:

```
# /usr/sbin/matchpathcon -V /var/www/html/*
/var/www/html/index.html has context unconfined_u:object_r:user_home_t:s0,
should be system_u:object_r:httpd_sys_content_t:s0
/var/www/html/page1.html has context unconfined_u:object_r:user_home_t:s0,
should be system_u:object_r:httpd_sys_content_t:s0
```

In this example, the **index.html** and **page1.html** files are labeled with the **user_home_t** type. This type is used for files in user home directories. Using the **mv** command to move files from your home directory may result in files being labeled with the **user_home_t** type. This type should not exist outside of home directories. Use the **restorecon** command to restore such files to their correct type:

```
# /sbin/restorecon -v /var/www/html/index.html
restorecon reset /var/www/html/index.html context
unconfined_u:object_r:user_home_t:s0->system_u:object_r:httpd_sys_content_t:s0
```

To restore the context for all files under a directory, use the **-R** option:

```
# /sbin/restorecon -R -v /var/www/html/
restorecon reset /var/www/html/page1.html context
unconfined_u:object_r:samba_share_t:s0->system_u:object_r:httpd_sys_content_t:s0
restorecon reset /var/www/html/index.html context
unconfined_u:object_r:samba_share_t:s0->system_u:object_r:httpd_sys_content_t:s0
```

34.6. SELinux file context

On systems running SELinux, all processes and files are labeled in a way that represents security-relevant information. This information is called the SELinux context. For files, this is viewed using the **ls -Z** command:

```
# ls -Z file1
-rw-rw-r--  user1  group1  unconfined_u:object_r:user_home_t:s0  file1
```

In this example, SELinux provides a user (**unconfined_u**), a role (**object_r**), a type (**user_home_t**), and a level (**s0**). This information is used to make access control decisions. On DAC systems, access is controlled based on Linux user and group IDs. SELinux policy rules are checked after DAC rules. SELinux policy rules are not used if DAC rules deny access first.

There are multiple commands for managing the SELinux context for files, such as **chcon**, **semanage fcontext**, and **restorecon**.

34.6.1. Temporary changes: chcon

The **chcon** command changes the SELinux context for files. However, changes made with the **chcon** command do not survive a file system relabel, or the execution of the **/sbin/restorecon** command.

SELinux policy controls whether users are able to modify the SELinux context for any given file. When using **chcon**, users provide all or part of the SELinux context to change. An incorrect file type is a common cause of SELinux denying access.

- Run the **chcon -t *type file-name*** command to change the file type, where *type* is a type, such as **httpd_sys_content_t**, and *file-name* is a file or directory name.

- Run the **chcon -R -t *type directory-name*** command to change the type of the directory and its contents, where *type* is a type, such as **httpd_sys_content_t**, and *directory-name* is a directory name.

34.6.2. Restoring default file contexts

Use the **/sbin/restorecon -v /path/file1** command to restore the SELinux context for the **/path/file1** file. Use the **-v** option to view what changes:

```
# /sbin/restorecon -v /path/file1
restorecon reset /path/file1 context unconfined_u:object_r:samba_share_t:s0->system_u:object_r:user_home_t:s0
```

In this example, the previous type, **samba_share_t**, is restored to the correct, **user_home_t** type. When using targeted policy (the default SELinux policy in Red Hat Enterprise Linux 6) the **/sbin/restorecon** command reads the files in the **/etc/selinux/targeted/contexts/files/** directory to see which SELinux context files should have.
The example in this section works the same for directories, for example, if file1 was a directory.

Use the **/sbin/restorecon -R -v dir1** command to restore the SELinux context for the directory **dir1** and all its contents. Use the **-v** option to view what changes.

34.6.3. Persistent changes: semanage fcontext

The **/usr/sbin/semanage fcontext** command changes the SELinux context for files or directories. When using targeted policy, changes made with this command are added to the **/etc/selinux/targeted/contexts/files/file_contexts** file if the changes are to files that exists in **file_contexts** or are added to **file_contexts.local** for new files and directories such as creating a **/web/** directory. **setfiles**, which is used when a file system is relabeled, and **/sbin/restorecon**, which restores the default SELinux contexts, read these files. This means that changes made by **/usr/sbin/semanage fcontext** are persistent, even if the file system is relabeled. SELinux policy controls whether users are able to modify the SELinux context for any given file.

To make SELinux context changes that survive a file system relabel for a file:

- Run the **/usr/sbin/semanage fcontext -a *options file-name|directory-name*** command, remembering to use the full path to the file or directory. This command will add a new entry to the file **/etc/selinux/targeted/contexts/files/file_contexts.local** with the specified user:role:type for the given file. This commands itself doesn't make effective the changes on the file

- Run the **/sbin/restorecon -v *file-name|directory-name*** command to apply the context changes.

*** **Note**: To remove a context line for a given file **file1** from the contexts file **/etc/selinux/targeted/contexts/files/file_contexts.local** run the **/usr/sbin/semanage fcontext -d /etc/file1**
When the context is removed, running **restorecon** changes the type to the default one.

To make SELinux context changes that survive a file system relabel for a directory and all its contents:

- Run the **/usr/sbin/semanage fcontext -a *options* "/*dir*(/.*)?"** command, remembering to use the full path to the directory. This command will add a new entry to the file **/etc/selinux/targeted/contexts/files/file_contexts.local** with the specified user:role:type for the given directory. This commands itself doesn't make effective the changes on the directory and its contents. It adds following line to contexts file:

```
/dir(/.*)?  system_u:object_r:httpd_sys_content_t:s0
```

- Run the **/sbin/restorecon -R -v *directory-name*** command to apply the context changes.

*** **Note**: To remove a context line for a given directory and its contents As the Linux root user, run the **/usr/sbin/semanage fcontext -d "/*dir*(/.*)?"**
After that run the **/sbin/restorecon -R -v /*dir*** command to restore the default SELinux contexts.

*** **Warning**: When changing the SELinux context with **/usr/sbin/semanage fcontext -a**, use the full path to the file or directory to avoid files being mislabeled after a file system relabel, or after the **/sbin/restorecon** command is run.

34.7. SELinux process context

On systems running SELinux, all processes and files are labeled in a way that represents security-relevant information. This information is called the SELinux context. For processes, this is viewed using the **ps -eZ | grep *process*** command:

```
# ps -eZ | grep process_name

# ps -eZ
[.....]
system_u:system_r:dhcpc_t:s0 1869 ? 00:00:00 dhclient
system_u:system_r:sshd_t:s0-s0:c0.c1023 1882 ? 00:00:00 sshd
system_u:system_r:gpm_t:s0 1964 ? 00:00:00 gpm
system_u:system_r:crond_t:s0-s0:c0.c1023 1973 ? 00:00:00 crond
system_u:system_r:kerneloops_t:s0 1983 ? 00:00:05 kerneloops
system_u:system_r:crond_t:s0-s0:c0.c1023 1991 ? 00:00:00 atd
[.....]
```

* The **system_r** role is used for system processes, such as daemons. Type Enforcement then separates each domain.

34.8. Mounting File Systems

By default, when a file system that supports extended attributes is mounted, the security context for each file is obtained from the *security.selinux* extended attribute of the file. Files in file systems that do not support extended attributes are assigned a single, default security context from the policy configuration, based on file system type.

Use the **mount -o context** command to override existing extended attributes, or to specify a different, default context for file systems that do not support extended attributes. This is useful if you do not trust a file system to supply the correct attributes, for example, removable media used in multiple systems. The **mount -o context** command can also be used to support labeling for file systems that do not support extended attributes, such as File Allocation Table (FAT) or NFS file systems. The context specified with the **context** is not written to disk: the original contexts are preserved, and are seen when mounting without a **context** option (if the file system had extended attributes in the first place).

34.8.1. Context mounts

To mount a file system with the specified context, overriding existing contexts if they exist, or to specify a different, default context for a file system that does not support extended attributes, as the Linux root user, use the **mount -o context=*SELinux_user:role:type:level*** command when mounting the desired file system. Context changes are not written to disk. By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **nfs_t** type. Without additional mount options, this may prevent sharing NFS file systems via other services, such as the Apache HTTP Server. The following example mounts an NFS file system so that it can be shared via the Apache HTTP Server:

```
# mount server:/export /local/mount/point -o context="system_u:object_r:httpd_sys_content_t:s0"
```

Newly-created files and directories on this file system appear to have the SELinux context specified with **-o context**; however, since context changes are not written to disk for these situations, the context specified with the **context** option is only retained if the **context** option is used on the next mount, and if the same context is specified.

Type Enforcement is the main permission control used in SELinux targeted policy. For the most part, SELinux users and roles can be ignored, so, when overriding the SELinux context with **-o context**, use the SELinux **system_u** user and **object_r** role, and concentrate on the type. If you are not using the MLS policy or multi-category security, use the **s0** level.

*** **Note**: When a file system is mounted with a **context** option, context changes (by users and processes) are prohibited. For example, running **chcon** on a file system mounted with a context option results in a ***Operation not supported*** error.

34.8.2. Changing the default context

On file systems that support extended attributes, when a file that lacks an SELinux context on disk is accessed, it is treated as if it had a default context as defined by SELinux policy. In common policies, this default context uses the **file_t** type. If it is desirable to use a different default context, mount the file system with the **defcontext** option.

The following example mounts a newly-created file system (on **/dev/sda2**) to the newly-created **/test/** directory. It assumes that there are no rules in **/etc/selinux/targeted/contexts/files/** that define a context for the **/test/** directory:

```
# mount /dev/sda2 /test/ -o defcontext="system_u:object_r:samba_share_t:s0"
```

In this example:

- the **defcontext** option defines that **system_u:object_r:samba_share_t:s0** is "the default security context for unlabeled files".
- when mounted, the root directory (**/test/**) of the file system is treated as if it is labeled with the context specified by **defcontext** (this label is not stored on disk). This affects the labeling for files created under **/test/**: new files inherit the **samba_share_t** type, and these labels are stored on disk.
- files created under **/test/** while the file system was mounted with a **defcontext** option retain their labels.

34.8.3. Mounting an NFS File System

By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS file systems. In common policies, this default context uses the **nfs_t** type. Depending on policy configuration, services, such as Apache HTTP Server and MySQL, may not be able to read files labeled with the **nfs_t** type. This may prevent file systems labeled with this type from being mounted and then read or exported by other services.

If you would like to mount an NFS file system and read or export that file system with another service, use the **context** option when mounting to override the **nfs_t** type. Use the following context option to mount NFS file systems so that they can be shared via the Apache HTTP Server:

```
# mount server:/export /local/mount/point -o context="system_u:object_r:httpd_sys_content_t:s0"
```

Since context changes are not written to disk for these situations, the context specified with the **context** option is only retained if the context option is used on the next mount, and if the same context is specified.

As an alternative to mounting file systems with **context** options, Booleans can be turned on to allow services access to file systems labeled with the **nfs_t** type.

34.8.4. Multiple NFS mounts

When mounting multiple mounts from the same NFS export, attempting to override the SELinux context of each mount with a different context, results in subsequent mount commands failing. In the following example, the NFS server has a single export, **/export**, which has two subdirectories, **web/** and **database/**. The following commands attempt two mounts from a single NFS export, and try to override the context for each one:

```
# mount server:/export/web /local/web  -o  context="system_u:object_r:httpd_sys_content_t:s0"

# mount server:/export/database /local/database  -o  context="system_u:object_r:mysqld_db_t:s0"
```

The second mount command fails, and the following is logged to **/var/log/messages**:

kernel: SELinux: mount invalid. Same superblock, different security settings for (dev 0:15, type nfs)

To mount multiple mounts from a single NFS export, with each mount having a different context, use the **-o nosharecache,context** options. The following example mounts multiple mounts from a single NFS export, with a different context for each mount (allowing a single service access to each one):

```
# mount server:/export/web /local/web \
        -o  nosharecache,context="system_u:object_r:httpd_sys_content_t:s0"

# mount server:/export/database /local/database \
        -o  nosharecache,context="system_u:object_r:mysqld_db_t:s0"
```

In this example, **server:/export/web** is mounted locally to **/local/web/**, with all files being labeled with the **httpd_sys_content_t** type, allowing Apache HTTP Server access. **server:/export/database** is mounted locally to **/local/database**, with all files being labeled with the **mysqld_db_t** type, allowing MySQL access. These type changes are not written to disk.

***** Note**: The **nosharecache** options allows you to mount the same subdirectory of an export multiple times with different contexts (for example, mounting **/export/web** multiple times). Do not mount the same subdirectory from an export multiple times with different contexts, as this creates an overlapping mount, where files are accessible under two different contexts.

34.8.5. Making context mounts persistent

To make context mounts persistent across remounting and reboots, add entries for the file systems in **/etc/fstab** or an automounter map, and use the desired context as a mount option. The following example adds an entry to **/etc/fstab** for an NFS context mount:

```
server:/export /local/mount/ nfs context="system_u:object_r:httpd_sys_content_t:s0" 0 0
```

34.9. Maintaining SELinux labels

Preserving SELinux Contexts When Copying

Use the **cp --preserve=context** command to preserve contexts when copying:

```
# touch file1
# ls -Z file1
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
# ls -dZ /var/www/html/
drwxr-xr-x   root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
# cp --preserve=context file1 /var/www/html/
# ls -Z /var/www/html/file1
-rw-r--r--  root root unconfined_u:object_r:user_home_t:s0 /var/www/html/file1
```

Copying and Changing the Context

Use the **cp -Z** command to change the destination copy's context. The following example was performed in the user's home directory:

```
# touch file1
# cp -Z system_u:object_r:samba_share_t:s0 file1 file2
# ls -Z file1 file2
-rw-rw-r--  user1 group1 unconfined_u:object_r:user_home_t:s0 file1
-rw-rw-r--  user1 group1 system_u:object_r:samba_share_t:s0 file2
# rm file1 file2
```

In this example, the context is defined with the **-Z** option. Without the **-Z** option, **file2** would be labeled with the **unconfined_u:object_r:user_home_t** context.

Copying a File Over an Existing File

When a file is copied over an existing file, the existing file's context is preserved (unless an option is used to preserve contexts). For example:

```
# touch /etc/file1
# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
# touch /tmp/file2
# ls -Z /tmp/file2
-rw-r--r--  root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
# cp /tmp/file2 /etc/file1
# ls -Z /etc/file1
-rw-r--r--  root root unconfined_u:object_r:etc_t:s0   /etc/file1
```

Moving files and directories

File and directories keep their current SELinux context when they are moved. In many cases, this is incorrect for the location they are being moved to.

***** Warning**: Moving files and directories with the **mv** command may result in the wrong SELinux context, preventing processes, such as the Apache HTTP Server and Samba, from accessing such files and directories.

Archiving files with **tar**

tar does not retain extended attributes by default. Since SELinux contexts are stored in extended attributes, contexts can be lost when archiving files. **Use tar --selinux** to create archives that retain contexts. If a Tar archive contains files without extended attributes, or if you want the extended attributes to match the system defaults, run the archive through **/sbin/restorecon**:

```
# tar -xvf archive.tar | /sbin/restorecon -f -
```

Depending on the directory, you may need to be the Linux root user to run the **/sbin/restorecon** command.

Archiving files with **star**

star does not retain extended attributes by default. Since SELinux contexts are stored in extended attributes, contexts can be lost when archiving files. Use **star -xattr -H=exustar** to create archives that retain contexts. The **star** package is not installed by default.

34.10. Diagnose and address routine SELinux policy violations

In Red Hat Enterprise Linux 6, the ***dbus***, ***setroubleshoot-server*** and ***audit*** packages are installed if packages are not removed from the default package selection.

SELinux denial messages, such as the following, are written to **/var/log/audit/audit.log** by default:

```
type=AVC msg=audit(1223024155.684:49): avc:  denied  { getattr } for  pid=2000 comm="httpd" path="/var/www/html/file1" dev=dm-0 ino=399185 scontext=unconfined_u:system_r:httpd_t:s0 tcontext=system_u:object_r:samba_share_t:s0 tclass=file
```

```
May  7 18:55:56 localhost setroubleshoot: SELinux is preventing httpd (httpd_t) "getattr" to /var/www/html/file1 (samba_share_t). For complete SELinux messages. run sealert -l de7e30d6-5488-466d-a606-92c9f40d316d
```

In Red Hat Enterprise Linux 6, **setroubleshootd** no longer constantly runs as a service, however it is still used to analyze the AVC messages. Two new programs act as a method to start setroubleshoot when needed: **sedispatch** and **seapplet**. **sedispatch** runs as part of the audit subsystem, and via **dbus**, sends a message when an AVC denial occurs, which will go straight to **setroubleshootd** if it is already running, or it will start **setroubleshootd** if it is not running. **seapplet** is a tool which runs in the system's toolbar, waiting for dbus messages in **setroubleshootd**, and will launch the notification bubble, allowing the user to review the denial.

Starting Daemons Automatically

To configure the auditd and rsyslogd daemons to automatically start at boot, run the following commands as the Linux root user:

```
# /sbin/chkconfig --levels 2345 auditd on
# /sbin/chkconfig --levels 2345 rsyslog on
```

Use the **service** ***service-name*** **status** command to check if these services are running, for example:

```
# /sbin/service  auditd  status
auditd (pid  1318) is running...
```

If the above services are not running (***service-name*** **is stopped**), use the **service** ***service-name*** **start** command as the Linux root user to start them. For example:

```
# /sbin/service auditd start
Starting auditd:                    [  OK  ]
```

34.11. Other tools/commands

avcstat

This command provides a short output of the access vector cache statistics since boot. You can watch the statistics in real time by specifying a time interval in seconds. This provides updated statistics since the initial output. The statistics file used is **/selinux/avc/cache_stats**, and you can specify a different cache file with the **-f /path/to/file** option.

```
# avcstat
lookups      hits     misses    allocs   reclaims     frees
47517410  47504630     12780     12780     12176     12275
```

seinfo

This utility is useful in describing the break-down of a policy, such as the number of classes, types, Booleans, allow rules, etc. **seinfo** is a command line utility that uses either the **policy.conf** file or a binary policy file as input.

```
# seinfo
Statistics for policy file: /etc/selinux/targeted/policy/policy.24
Policy Version  & Type: v.24 (binary, mls)
```

```
Classes:            77          Permissions:        229
Sensitivities:      1           Categories:         1024
Types:              3001        Attributes:         244
Users:              9           Roles:              13
Booleans:           158         Cond. Expr.:        193
Allow:              262796      Neverallow:         0
Auditallow:         44          Dontaudit:          156710
Type_trans:         10760       Type_change:        38
Type_member:        44          Role allow:         20
Role_trans:         237         Range_trans:        2546
Constraints:        62          Validatetrans:      0
Initial SIDs:       27          Fs_use:             22
Genfscon:           82          Portcon:            373
Netifcon:           0           Nodecon:            0
Permissives:        22          Polcap:             2
```

The **seinfo** command can also list the number of types with the domain attribute, giving an estimate of the number of different confined processes:

```
# seinfo -adomain -x |  wc -l
550
```

Not all domain types are confined. To look at the number of unconfined domains, use the **unconfined_domain** attribute:

```
# seinfo -aunconfined_domain_type -x | wc -l
52
```

Permissive domains can be counted with the **--permissive** option.

```
# seinfo --permissive -x | wc -l
31
```

sesearch

You can use the **sesearch** command to search for a particular type in the policy. You can search either policy source files or the binary file. For example:

```
# sesearch --role_allow -t httpd_sys_content_t /etc/selinux/targeted/policy/policy.24
Found 20 role allow rules:
  allow system_r sysadm_r;
  allow sysadm_r system_r;
  allow sysadm_r staff_r;
  allow sysadm_r user_r;
  allow system_r git_shell_r;
  allow system_r guest_r;
  allow logadm_r system_r;
  allow system_r logadm_r;
  allow system_r nx_server_r;
  allow system_r staff_r;
  allow staff_r logadm_r;
  allow staff_r sysadm_r;
  allow staff_r unconfined_r;
  allow staff_r webadm_r;
  allow unconfined_r system_r;
  allow system_r unconfined_r;
  allow system_r user_r;
  allow webadm_r system_r;
  allow system_r webadm_r;
  allow system_r xguest_r;
```

The sesearch command can provide the number of allow rules:

```
# sesearch --allow | wc -l
262798
```

And the number of dontaudit rules:

```
# sesearch --dontaudit | wc -l
156712
```

34.12. SELinux packages

In Red Hat Enterprise Linux, the SELinux packages are installed by default, in a full installation, unless they are manually excluded during installation. If performing a minimal installation in text mode, the *policycoreutils-python* package will not be installed by default. Also, by default, SELinux targeted policy is used, and SELinux runs in enforcing mode. The following is a brief description of the main SELinux packages:

policycoreutils-python: provides utilities such as **semanage**, **audit2allow**, **audit2why** and **chcat**, for operating and managing SELinux.

policycoreutils: provides utilities such as **restorecon**, **secon**, **setfiles**, **semodule**, **load_policy**, and **setsebool**, for operating and managing SELinux.

policycoreutils-gui: provides **system-config-selinux**, a graphical tool for managing SELinux.

selinux-policy: provides the SELinux Reference Policy. The SELinux Reference Policy is a complete SELinux policy, and is used as a basis for other policies, such as the SELinux targeted policy. The **selinux-policy-devel** package provides development tools, such as **/usr/share/selinux/devel/policygentool** and **/usr/share/selinux/devel/policyhelp**, as well as example policy files.

selinux-policy-policy: provides SELinux policies. For targeted policy, install **selinux-policy-targeted**. For MLS, install **selinux-policy-mls**.

setroubleshoot-server: translates denial messages, produced when access is denied by SELinux, into detailed descriptions that are viewed with **sealert** (which is provided by this package).

setools-console: this package provides the *Tresys Technology SETools distribution*, a number of tools and libraries for analyzing and querying policy, audit log monitoring and reporting, and file context management. The **setools** package is a meta-package for SETools. The **setools-gui** package provides the **apol**, **seaudit**, and **sediffx** tools. The **setools-console** package provides the **seauditreport**, **sechecker**, **sediff**, **seinfo**, **sesearch**, **findcon**, **replcon**, and **indexcon** command line tools.

libselinux-utils: provides the **avcstat**, **getenforce**, **getsebool**, **matchpathcon**, **selinuxconlist**, **selinuxdefcon**, **selinuxenabled**, **setenforce**, **togglesebool** tools.

mcstrans: translates levels, such as **s0-s0:c0.c1023**, to an easier to read form, such as **SystemLow-SystemHigh**. This package is not installed by default.

35. Automating system tasks

In Linux, tasks, which are also known as *jobs*, can be configured to run automatically within a specified period of time, on a specified date, or when the system load average is below a specified number. Red Hat Enterprise Linux is pre-configured to run important system tasks to keep the system updated. For example, the slocate database used by the **locate** command is updated daily. A system administrator can use automated tasks to perform periodic backups, monitor the system, run custom scripts, and more.

Red Hat Enterprise Linux comes with several automated tasks utilities: **cron**, **at**, and **batch**.

35.1. Cron and Anacron

Both, Cron and Anacron, are daemons that can be used to schedule the execution of recurring tasks according to a combination of the time, day of the month, month, day of the week, and week.

Cron assumes that the system is on continuously. If the system is not on when a job is scheduled, it is not executed. Cron allows jobs to be run as often as every minute. Anacron does not assume the system is always on, remembers every scheduled job, and executes it the next time the system is up. However, Anacron can only run a job once a day.

To use the cron service, the **cronie** RPM package must be installed and the **crond** service must be running. **anacron** is a sub-package of **cronie**. To determine if these packages are installed, use the **rpm -q cronie cronie-anacron** command.

35.1.1. Starting/stopping the service

To determine if the service is running, use the command **/sbin/service crond status**. To start the cron service, use the command **/sbin/service crond start**. To stop the service, use the command **/sbin/service crond stop**. It is recommended that you start the service at boot time.

35.1.2. Configuring Anacron jobs

The main configuration file to schedule jobs is **/etc/anacrontab** (only root is allowed to modify this file), which contains the following lines:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
```

```
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1                 5                  cron.daily       nice run-parts /etc/cron.daily
7                 25                 cron.weekly      nice run-parts /etc/cron.weekly
@monthly          45                 cron.monthly     nice run-parts /etc/cron.monthly
```

The first three lines are variables used to configure the environment in which the anacron tasks are run. The **SHELL** variable tells the system which shell environment to use (in this example the bash shell). The **PATH** variable defines the path used to execute commands. The output of the anacron jobs are emailed to the username defined with the **MAILTO** variable. If the **MAILTO** variable is not defined, (i.e. is empty, **MAILTO=**), email is not sent.

The next two lines are variables that modify the time for each scheduled job. The **RANDOM_DELAY** variable denotes the maximum number of minutes that will be added to the **delay in minutes** variable which is specified for each job. The minimum delay value is set, by default, to 6 minutes. A **RANDOM_DELAY** set to 12 would therefore add, randomly, between 6 and 12 minutes to the **delay in minutes** for each job in that particular anacrontab. **RANDOM_DELAY** can also be set to a value below 6, or even 0. When set to 0, no random delay is added. This proves to be useful when, for example, more computers that share one network connection need to download the same data every day. The **START_HOURS_RANGE** variable defines an interval (in hours) when scheduled jobs can be run. In case this time interval is missed, for example, due to a power down, then scheduled jobs are not executed that day.

The rest of the lines in the **/etc/anacrontab** file represent scheduled jobs and have the following format:

period in days   delay in minutes   job-identifier   command

period in days: specifies the frequency of execution of a job in days. This variable can be represented by an integer or a macro (**@daily**, **@weekly**, **@monthly**), where **@daily** denotes the same value as the integer 1, **@weekly** the same as 7, and **@monthly** specifies that the job is run once a month, independent on the length of the month.
delay in minutes: specifies the number of minutes anacron waits, if necessary, before executing a job. This variable is represented by an integer where 0 means no delay.
job-identifier: specifies a unique name of a job which is used in the log files.
command: specifies the command to execute. The command can either be a command such as **ls /proc >> /tmp/proc** or a command to execute a custom script.

35.1.3. Configuring Cron jobs

The configuration file to configure cron jobs, **/etc/crontab** (only root is allowed to modify this file), contains the following lines:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# For details see man 4 crontabs
# Example of job definition:
```

```
# .---------------- minute (0 - 59)
# | .------------- hour (0 - 23)
# | | .---------- day of month (1 - 31)
# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user command to be executed
```

The first three lines contain the same variables as an **anacrontab** file, **SHELL**, **PATH** and **MAILTO**. The fourth line contains the **HOME** variable. The **HOME** variable can be used to set the home directory to use when executing commands or scripts.

The rest of the lines in the **/etc/crontab** file represent scheduled jobs and have the following format:

minute hour day month day of week user command

minute: any integer from 0 to 59
hour: any integer from 0 to 23
day: any integer from 1 to 31 (must be a valid day if a month is specified)
month: any integer from 1 to 12 (or the short name of the month such as jan or feb)
day of week: any integer from 0 to 7, where 0 or 7 represents Sunday (or the short name of the week such as sun or mon)
user: specifies the user under which the jobs are run
command: the command to execute (the command can either be a command such as ls /proc >> /tmp/proc or the command to execute a custom script)

For any of the above values, an asterisk (*) can be used to specify all valid values. For example, an asterisk for the month value means execute the command every month within the constraints of the other values.

A hyphen (-) between integers specifies a range of integers. For example, 1-4 means the integers 1, 2, 3, and 4.

A list of values separated by commas (,) specifies a list. For example, 3, 4, 6, 8 indicates those four specific integers.

The forward slash (/) can be used to specify step values. The value of an integer can be skipped within a range by following the range with **/<integer>**. For example, **0-59/2** can be used to define every other minute in the minute field. Step values can also be used with an asterisk. For instance, the value ***/3** can be used in the month field to run the task every third month.

Any lines that begin with a hash sign (#) are comments and are not processed.

Users other than root can configure cron tasks by using the **crontab** utility. All user-defined crontabs are stored in the **/var/spool/cron/** directory and are executed using the usernames of the users that created them. To create a crontab as a user, login as that user and type the command **crontab -e** to edit the user's crontab using the editor specified by the **VISUAL** or **EDITOR** environment variable. The file uses the same format as **/etc/crontab**. When the changes to the crontab are saved, the crontab is stored according to username and written to the file **/var/spool/cron/username**. To list the contents of your own personal crontab file, use the **crontab -l** command.

*** **Note**: Do not specify a user; When using the crontab utility, there is no need to specify a user when defining a job.

The **/etc/cron.d/** directory contains files that have the same syntax as the **/etc/crontab** file. Only root is allowed to create and modify files in this directory.

*** **Note**: Do not restart the daemon to apply the changes; The cron daemon checks the **/etc/anacrontab** file, the **/etc/crontab** file, the **/etc/cron.d/** directory, and the **/var/spool/cron/** directory every minute for any changes. If any changes are found, they are loaded into memory. Thus, the daemon does not need to be restarted if an anacrontab or a crontab file is changed.

35.1.4. Controlling access to Cron

The **/etc/cron.allow** and **/etc/cron.deny** files are used to restrict access to cron. The format of both access control files is one username on each line. Whitespace is not permitted in either file. The cron daemon (**crond**) does not have to be restarted if the access control files are modified. The access control files are checked each time a user tries to add or delete a cron job.

The root user can always use cron, regardless of the usernames listed in the access control files.

If the file **cron.allow** exists, only users listed in it are allowed to use cron, and the **cron.deny** file is ignored.

If **cron.allow** does not exist, users listed in **cron.deny** are not allowed to use cron.

Access can also be controlled through Pluggable Authentication Modules (PAM). These settings are stored in **/etc/security/access.conf**. For example, adding the following line in this file forbids creating crontabs for all users except the root user:

```
-:ALL  EXCEPT  root  :cron
```

The forbidden jobs are logged in an appropriate log file or, when using "crontab -e", returned to the standard output.

35.1.5. Black/White listing of Cron jobs

Black/White listing of jobs is used to omit parts of the defined jobs that do not need to be executed. When calling the **run-parts** script on a cron folder, such as **/etc/cron.daily**, we can define which of the programs in this folder will not be executed by **run-parts**.

To define a black list, create a **jobs.deny** file in the folder that **run-parts** will be executing from. For example, if we need to omit a particular program from /etc/cron.daily, then, a file **/etc/cron.daily/jobs.deny** has to be created. In this file, specify the names of the omitted programs from the same directory. These will not be executed when a command, such as **run-parts /etc/cron.daily**, is executed by a specific job.

To define a white list, create a **jobs.allow** file.

The principles of **jobs.deny** and **jobs.allow** are the same as those of **cron.deny** and **cron.allow**.

35.2. At and Batch

While cron is used to schedule recurring tasks, the **at** command is used to schedule a one-time task at a specific time and the **batch** command is used to schedule a one-time task to be executed when the systems load average drops below 0.8.

To use **at** or **batch**, the **at** RPM package must be installed, and the **atd** service must be running. To determine if the package is installed, use the **rpm -q at** command. To determine if the service is running, use the command **/sbin/service atd status**.

35.2.1. Configuring At jobs

To schedule a one-time job at a specific time, type the command **at *time***, where ***time*** is the time to execute the command.
The argument **time** can be one of the following:

HH:MM format: For example, 04:00 specifies 4:00 a.m. If the time is already past, it is executed at the specified time the next day.
midnight: Specifies 12:00 a.m.
noon: Specifies 12:00 p.m.
teatime: Specifies 4:00 p.m.
month-name day year format: For example, January 15 2002 specifies the 15th day of January in the year 2002. The year is optional.
MMDDYY, **MM/DD/YY**, or **MM.DD.YY** formats: For example, 011502 for the 15th day of January in the year 2002.
now + time: time is in minutes, hours, days, or weeks. For example, now + 5 days specifies that the command should be executed at the same time five days from now.

The time must be specified first, followed by the optional date. For more information about the time format, read the **/usr/share/doc/at-<version>/timespec** text file.

After typing the **at** command with the time argument, the **at>** prompt is displayed. Type the command to execute, press **Enter**, and press **Ctrl+D**. Multiple commands can be specified by typing each command followed by the **Enter** key. After typing all the commands, press **Enter** to go to a blank line and press **Ctrl+D**. Alternatively, a shell script can be entered at the prompt, pressing **Enter** after each line in the script, and pressing **Ctrl+D** on a blank line to exit. If a script is entered, the shell used is the shell set in the user's **SHELL** environment, the user's login shell, or **/bin/sh** (whichever is found first).

If the set of commands or script tries to display information to standard output, the output is emailed to the user.
Use the command **atq** to view pending jobs.

35.2.2. Configuring Batch jobs

To execute a one-time task when the load average is below 0.8, use the **batch** command.

After typing the **batch** command, the **at>** prompt is displayed. Type the command to execute, press **Enter**, and press **Ctrl+D**. Multiple commands can be specified by typing each command followed by the **Enter** key. After typing all the commands, press **Enter** to go to a blank line and press **Ctrl+D**. Alternatively, a shell script can be entered at the prompt, pressing **Enter** after each line in the script, and pressing **Ctrl+D** on a blank line to exit. If a script is entered, the shell used is the shell set in the user's **SHELL** environment, the user's login shell, or **/bin/sh** (whichever is found first). As soon as the load average is below 0.8, the set of commands or script is executed.

If the set of commands or script tries to display information to standard out, the output is emailed to the user.
Use the command **atq** to view pending jobs.

35.2.3. Viewing pending jobs

To view pending **at** and **batch** jobs, use the **atq** command. The **atq** command displays a list of pending jobs, with each job on a line. Each line follows the job number, date, hour, job class, and username format. Users can only view their own jobs. If the root user executes the **atq** command, all jobs for all users are displayed.

35.2.4. Additional command line options

Additional command line options for **at** and **batch** include:

Option	Description
`-f`	Read the commands or shell script from a file instead of specifying them at the prompt.
`-m`	Send email to the user when the job has been completed.
`-v`	Display the time that the job is executed.

35.2.5. Controlling Access to At and Batch

The **/etc/at.allow** and **/etc/at.deny** files can be used to restrict access to the **at** and **batch** commands. The format of both access control files is one username on each line. Whitespace is not permitted in either file. The **at** daemon (**atd**) does not have to be restarted if the access control files are modified. The access control files are read each time a user tries to execute the **at** or **batch** commands.

The root user can always execute **at** and **batch** commands, regardless of the access control files.

If the file **at.allow** exists, only users listed in it are allowed to use **at** or **batch**, and the **at.deny** file is ignored.

If **at.allow** does not exist, users listed in **at.deny** are not allowed to use **at** or **batch**.

35.2.6. Starting/Stopping the service

To start the **at** service, use the command **/sbin/service atd start**. To stop the service, use the command **/sbin/service atd stop**. It is recommended that you start the service at boot time.

36. Kickstart installations

http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/ch-kickstart2.html

36.1. Install Red Hat Enterprise Linux automatically using Kickstart

Installing a system via Kickstart comes in pretty useful in real life.

There are a few ways to create a kickstart file that would be used in the automatic installation of a Redhat Enterprise Linux 6 system. There's always writing the thing from scratch, which while always an option, is not so efficient. Besides that there is:

- **system-config-kickstart**

- using the **anaconda-ks.cfg** that was created during an installation.

Working with a premade kickstart file
Let's say we have this info:

> **kickstart file = http://192.168.111.23/pub/ks/redhat6.kfg**
> **ip of new install = 192.168.111.222 (same subnet)**
> **netmask = 255.255.255.0**

First we would boot the system with some sort of boot media, most likely the RHEL 6 CD-ROM #1 and at the boot prompt (when it asks you what you want to do) you would type a command like this, substituting your own info:

```
linux ks=http://192.168.111.23/pub/ks/redhat6.kfg append ip=192.168.111.222 netmask=255.255.255.0
```

As long as everything is configured correctly and the installation media is where is should be, then this should install pretty hands off.

37. RHEL-based virtualization

37.1. Configure a physical machine to host virtual guests

A default RHEL 6 system should come prepared to host virtual guests, minus the packages. In RHEL5 you had to make sure that you were running the **xen kernel**, which would require installing and booting into that kernel. RHEL 6 is simple, if it Virtualization is not installed, install it.

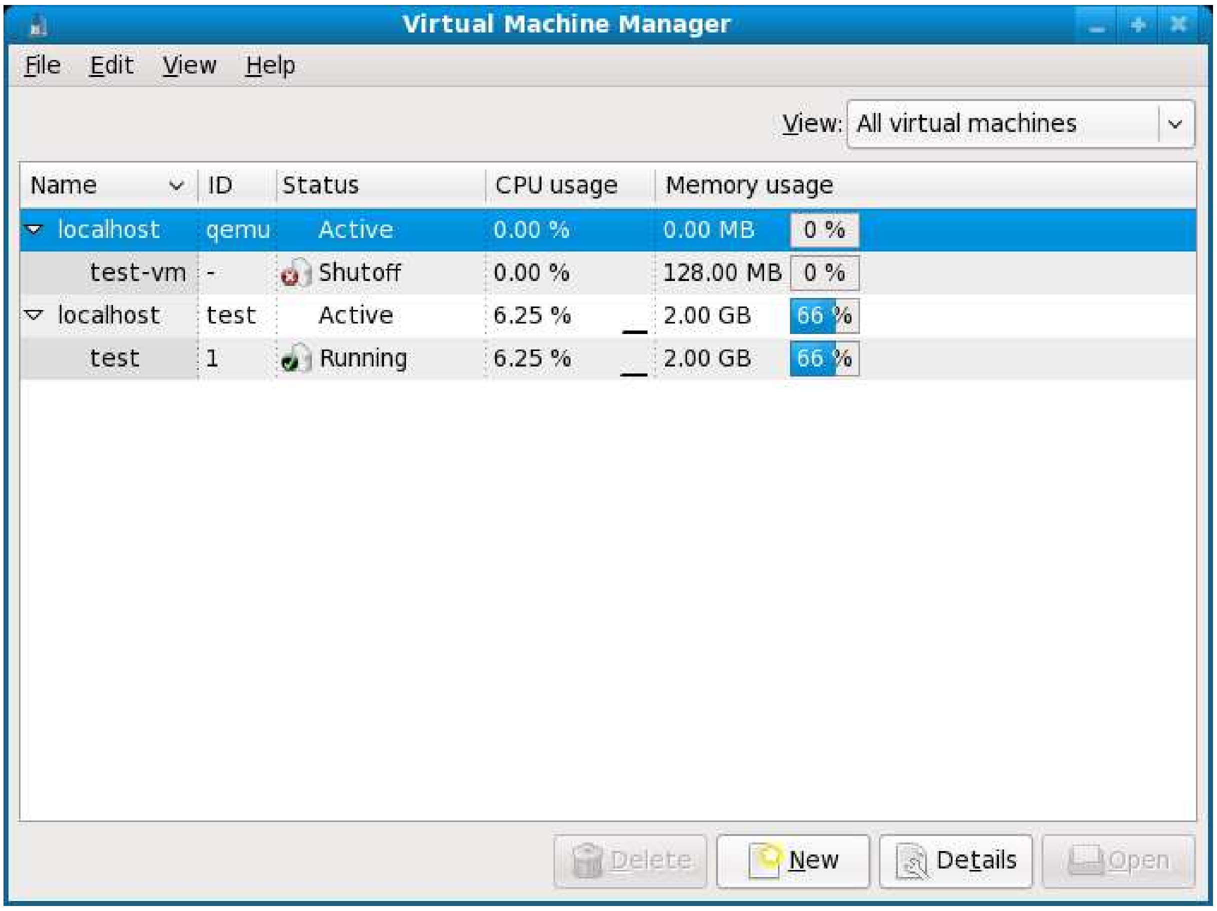
```
# yum groupinstall "Virtualization"
```

That will install everything needed to run virtual guests on RHEL 6. Easy huh?

37.2. Accessing a virtual machine's console

There are two ways to pull it up, one from the gui menu on the desktop, and the other with the following command in terminal:

```
# virt-manager
```



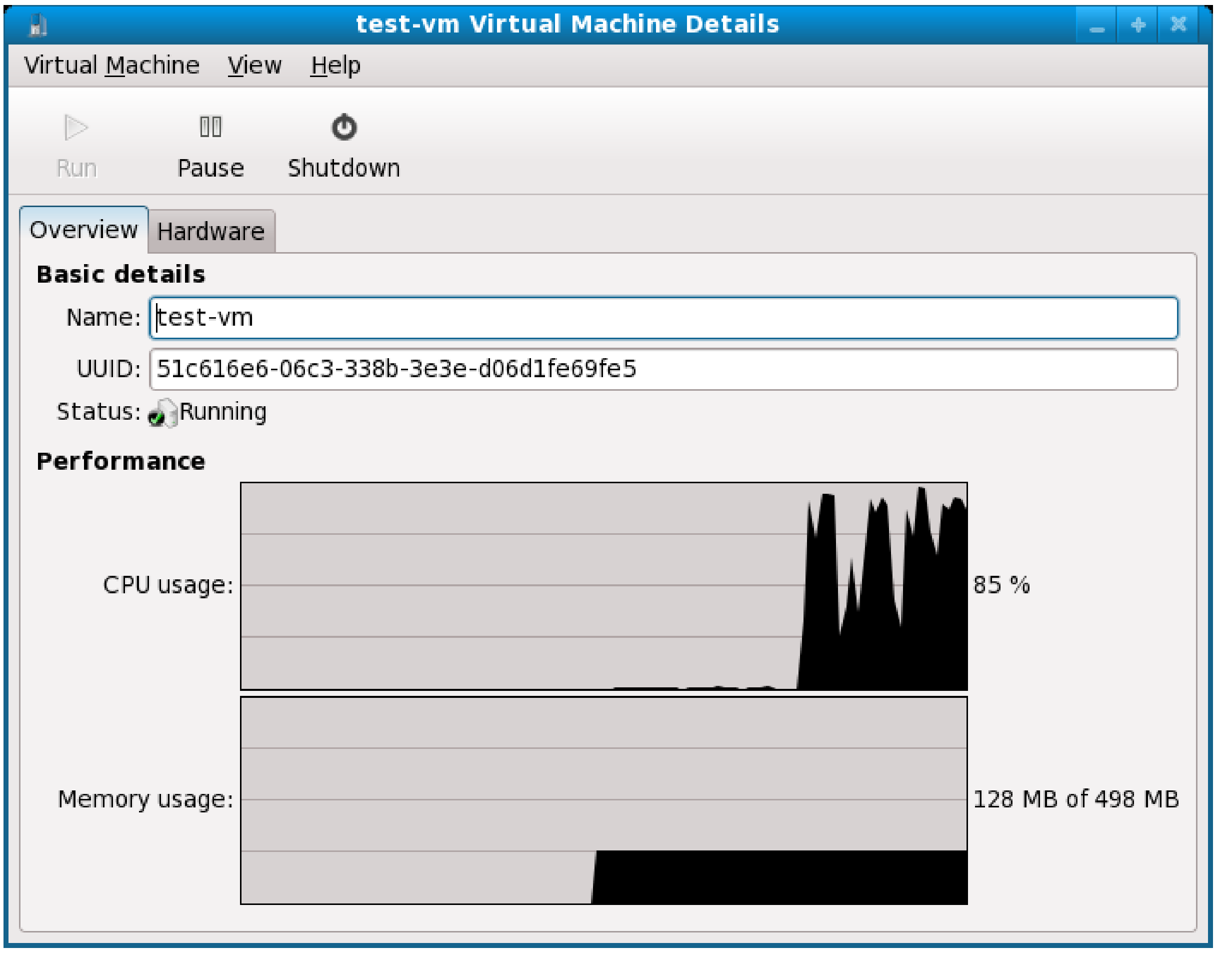
Name	ID	Status	CPU usage	Memory usage	
localhost	qemu	Active	0.00 %	0.00 MB	0 %
test-vm	-	Shutoff	0.00 %	128.00 MB	0 %
localhost	test	Active	6.25 %	2.00 GB	66 %
test	1	Running	6.25 %	2.00 GB	66 %

37.3. Start and stop virtual machines

Starting and stopping virtual machines is just like starting and stopping real machines.
The only main difference is that you can start and stop them right from the Virtual Machine Manager (**virt-manager**).

The Red Hat Enterprise Linux 6 Virtual Machine Manager has a fairly intuitive graphical user interface, with obvious start and stop buttons for each machine. If you haven't used it at all, it would be good to download a trial copy of Red Hat, to get an idea of how to navigate the program.

You can see in the screenshot the Run, Pause, and Shutdown buttons:



Basic details

Name: test-vm

UUID: 51c616e6-06c3-338b-3e3e-d06d1fe69fe5

Status: Running

Performance

CPU usage: 85 %

Memory usage: 128 MB of 498 MB

37.4. Configure systems to launch virtual machines at boot

The **virsh** command is the command line utility, and the quickest way to accomplish this. But you always to have the option of messing around with the GUI, in case you forget this.

To start a vm type:

```
# virsh start machninename01
```

To shutdown:

```
# virsh shutdown machninename01
```

To destroy is actually to bring the machine hard down, like pulling the plug on it. Not deleting it.

```
# virsh destroy machninename01
```

To gain access to a virtual console:

```
# virsh console machninename01
```

To configure the machine to launch at boot, or autostart the virsh server:

```
# virsh autostart machninename01
```

38. Package management with RPM

RPM has five basic modes of operation (not counting package building): installing, uninstalling, upgrading, querying, and verifying. This section contains an overview of each mode. For complete details and options, try **rpm --help**.

38.1. Install and update software packages from Red Hat Network, a remote repository, or from the local filesystem

To work with Red Hat, first system has to be registered:

```
# rhn_register
```

And follow the instructions.

If we don't have a subscription to Red Hat Network, then we'll use a local repository or a remote one.

Most common situation is having a remote repository that you need to pull packages from. Usually you be given a url to connect to looking something like this: **http://myremote.com/repo/i386/** The yum repo files are located in **/etc/yum.repos.d/** and end with a **.repo** extension. The format is simple to setup a repo on the fly:

```
[myremote]
name=myremote
baseurl=http://myremote.com/repo/i386/
enabled=1
gpgcheck=0
```

Those are the essential elements to pull packages via **yum** from that repo.

Setting up a local repo with a disk is almost the same with a few steps before. First the installation disk needs to be mounted, and the packages copied from **Packages/** into another directory on the server. In this case we will use **file:///directory/path/to/repo/** as the url, where **/directory/path/to/repo/** is the directory that contains the **rpm** files.

Next the package **creatrepo** needs to be installed. Once copied files into the directory, run:

```
# createrepo .
```

Now that you have a repo setup, yum needs to know about it. Create a file named **mylocal.repo** in the **/etc/yum.repos.d/** directory:

```
[mylocal]
name=mylocal
baseurl=file:///directory/path/to/repo/
enabled=1
gpgcheck=0
```

Run a **yum** command to test, and it should be pulling information about packages from the local repository:

```
# yum list httpd
```

38.2. Using RPM

38.2.1. Finding RPM packages

Before using an RPM, you must know where to find them. An Internet search returns many RPM repositories, but if you are looking for RPM packages built by Red Hat, Inc, they can be found at the following locations:

- The Red Hat Enterprise Linux CD-ROMs
- The Red Hat Errata Page available at http://www.redhat.com/apps/support/errata/
- A Red Hat FTP Mirror Site available at http://www.redhat.com/download/mirror.html
- Red Hat Network

38.2.2. Installing

RPM packages typically have file names like **foo-1.0-1.i386.rpm**. The file name includes the package name (**foo**), version (**1.0**), release (**1**), and architecture (**i386**). To install a package, log in as root and type the following command at a shell prompt:

```
# rpm -Uvh foo-1.0-1.i386.rpm
```

If installation is successful, the following output is displayed:

```
Preparing...          ################################## [100%]
  1:foo               ################################## [100%]
```

The signature of a package is checked automatically when installing or upgrading a package. The signature confirms that the package was signed by an authorized party. For example, if the verification of the signature fails, an error message such as the following is displayed:

```
error: V3 DSA signature: BAD, key ID 0352860f
```

If it is a new, header-only, signature, an error message such as the following is displayed:

```
error: Header V3 DSA signature: BAD, key ID 0352860f
```

If you do not have the appropriate key installed to verify the signature, the message contains the word **NOKEY** such as:

```
warning: V3 DSA signature: NOKEY, key ID 0352860f
```

***** Warning**: If you are installing a kernel package, you should use **rpm -ivh** instead. Installing packages is designed to be simple, but you may sometimes see errors.

Package Already Installed

If the package of the same version is already installed, the following is displayed:

```
Preparing...        ################################## [100%]
package foo-1.0-1 is already installed
```

If the same version you are trying to install is already installed, and you want to install the package anyway, you can use the **--replacepkgs** option, which tells RPM to ignore the error:

```
# rpm -ivh --replacepkgs foo-1.0-1.i386.rpm
```

This option is helpful if files installed from the RPM were deleted or if you want the original configuration files from the RPM to be installed.

Conflicting files

If you attempt to install a package that contains a file which has already been installed by another package or an earlier version of the same package, the following is displayed:

```
Preparing...        ################################## [100%]
file /usr/bin/foo from install of foo-1.0-1 conflicts with file from package bar-2.0.20
```

To make RPM ignore this error, use the **--replacefiles** option:

```
# rpm -ivh --replacefiles foo-1.0-1.i386.rpm
```

Unresolved dependencies

RPM packages can, essentially, depend on other packages, which mean that they require other packages to be installed to run properly. If you try to install a package which has an unresolved dependency, output similar to the following is displayed:

```
error: Failed dependencies:
    bar.so.2 is needed by foo-1.0-1
  Suggested resolutions:
    bar-2.0.20-3.i386.rpm
```

If you are installing a package from the Red Hat Enterprise Linux CD-ROM set, it usually suggests the package(s) needed to resolve the dependency. Find the suggested package(s) on the Red Hat Enterprise Linux CD-ROMs or from the Red Hat FTP site (or mirror), and add it to the command:

```
# rpm  -ivh  foo-1.0-1.i386.rpm  bar-2.0.20-3.i386.rpm
```

If installation of both packages is successful, output similar to the following is displayed:

```
Preparing...       ################################## [100%]
1:foo              ################################## [ 50%]
2:bar              ################################## [100%]
```

If it does not suggest a package to resolve the dependency, you can try the **--redhatprovides** option to determine which package contains the required file. You need the **rpmdb-redhat** package installed to use this option

```
# rpm -q --redhatprovides bar.so.2
```

If the package that contains **bar.so.2** is in the installed database from the **rpmdb-redhat** package, the name of the package is displayed:

```
# bar-2.0.20-3.i386.rpm
```

To force the installation anyway (which is not recommended since the package may not run correctly), use the **--nodeps** option.

38.2.3. Uninstalling

Uninstalling a package is just as simple as installing one. Type the following command at a shell prompt:

```
# rpm -e foo-1.0-1.i386.rpm
```

You can encounter a dependency error when uninstalling a package if another installed package depends on the one you are trying to remove. For example:

```
error: Failed dependencies:
    foo is needed by (installed) bar-2.0.20-3.i386.rpm
```

To cause RPM to ignore this error and uninstall the package anyway, which may break the package depending on it, use the **--nodeps** option.

38.2.4. Upgrading

Upgrading a package is similar to installing one. Type the following command at a shell prompt:

```
# rpm -Uvh foo-2.0-1.i386.rpm
```

As part of upgrading a package, RPM automatically uninstalls any old versions of the **foo** package. In fact, you may want to always use **-U** to install packages which works even when there are no previous versions of the package installed.

*** **Tip**: You don't want to use the **-U** option for installing kernel packages because RPM replaces the previous kernel package. This does not affect a running system, but if the new kernel is unable to boot during your next restart, there would be no other kernel to boot instead.
Using the **-i** option adds the kernel to your GRUB boot menu (**/etc/grub.conf**). Similarly, removing an old, unneeded kernel removes the kernel from GRUB.

Because RPM performs intelligent upgrading of packages with configuration files, you may see a message like the following:

```
saving /etc/foo.conf as /etc/foo.conf.rpmsave
```

This message means that your changes to the configuration file may not be forward compatible with the new configuration file in the package, so RPM saved your original file and installed a new one. You should investigate the differences between the two configuration files and resolve them as soon as possible, to ensure that your system continues to function properly.
Upgrading is really a combination of uninstalling and installing, so during an RPM upgrade you can encounter uninstalling and installing errors, plus one more. If RPM thinks you are trying to upgrade to a package with an older version number, the output is similar to the following:

```
package foo-2.0-1 (which is newer than foo-1.0-1) is already installed
```

To force RPM to upgrade anyway, use the **--oldpackage** option:

```
# rpm -Uvh --oldpackage foo-1.0-1.i386.rpm
```

38.2.5. Freshening

Freshening a package is similar to upgrading one. Type the following command at a shell prompt:

```
# rpm -Fvh foo-1.2-1.i386.rpm
```

RPM's freshen option checks the versions of the packages specified on the command line against the versions of packages that have already been installed on your system. When a newer version of an already-installed package is processed by RPM's freshen option, it is upgraded to the newer version.

However, RPM's freshen option does not install a package if no previously-installed package of the same name exists. This differs from RPM's upgrade option, as an upgrade does install packages, whether or not an older version of the package was already installed.

RPM's freshen option works for single packages or package groups. If you have just downloaded a large number of different packages, and you only want to upgrade those packages that are already installed on your system, freshening does the job. If you use freshening, you do not have to delete any unwanted packages from the group that you downloaded before using RPM.

In this case, issue the following command:

```
# rpm -Fvh *.rpm
```

RPM automatically upgrades only those packages that are already installed.

38.2.6. Querying

Use the **rpm -q** command to query the database of installed packages. The **rpm -q foo** command displays the package name, version, and release number of the installed package **foo**:

foo-2.0-1

Instead of specifying the package name, use the following options with **-q** to specify the package(s) you want to query. These are called Package Selection Options.

-a: queries all currently installed packages.
-f <***file***>: queries the package which owns <***file***>. When specifying a file, you must specify the full path of the file (for example, **/bin/ls**).
-p <***packagefile***>: queries the package <***packagefile***>.

There are a number of ways to specify what information to display about queried packages. The following options are used to select the type of information for which you are searching. These are called Information Query Options.

-i: displays package information including name, description, release, size, build date, install date, vendor, and other miscellaneous information.
-l: displays the list of files that the package contains.
-s: displays the state of all the files in the package.
-d: displays a list of files marked as documentation (man pages, info pages, READMEs, etc.).
-c: displays a list of files marked as configuration files. These are the files you change after installation to adapt the package to your system (for example, **sendmail.cf**, **passwd**, **inittab**, etc.).

For the options that display lists of files, add **-v** to the command to display the lists in a familiar **ls -l** format.

38.2.7. Verifying

Verifying a package compares information about files installed from a package with the same information from the original package. Among other things, verifying compares the size, MD5 sum, permissions, type, owner, and group of each file.

The command **rpm -V** verifies a package. You can use any of the Package Verify Options listed for querying to specify the packages you wish to verify. A simple use of verifying is **rpm -V foo**, which verifies that all the files in the foo package are as they were when they were originally installed. For example:

- To verify a package containing a particular file:

```
# rpm -Vf /usr/bin/vim
```

- To verify ALL installed packages:

```
# rpm -Va
```

- To verify an installed package against an RPM package file:

```
# rpm -Vp foo-1.0-1.i386.rpm
```

This command can be useful if you suspect that your RPM databases are corrupt.

If everything verified properly, there is no output. If there are any discrepancies, they are displayed. The format of the output is a string of eight characters (a c denotes a configuration file) and then the file name. Each of the eight characters denotes the result of a comparison of one attribute of the file to the value of that attribute recorded in the RPM database. A single period (.) means the test passed.

The following characters denote failure of certain tests:

- **5** — MD5 checksum
- **S** — file size
- **L** — symbolic link
- **T** — file modification time
- **D** — device
- **U** — user
- **G** — group
- **M** — mode (includes permissions and file type)
- **?** — unreadable file

If you see any output, use your best judgment to determine if you should remove or reinstall the package, or fix the problem in another way.

38.3. Checking a package's signature

If you wish to verify that a package has not been corrupted or tampered with, examine only the md5sum by typing the following command at a shell prompt: (<***rpm-file***> with file name of the RPM package):

```
# rpm -K --nosignature <rpm-file>
```

The message <***rpm-file***>: **md5 OK** is displayed. This brief message means that the file was not corrupted by the download. To see a more verbose message, replace **-K** with **-Kvv** in the command.

On the other hand, how trustworthy is the developer who created the package? If the package is signed with the developer's GnuPG key, you know that the developer really is who they say they are.

An RPM package can be signed using Gnu Privacy Guard (or GnuPG), to help you make certain your downloaded package is trustworthy.

GnuPG is a tool for secure communication; it is a complete and free replacement for the encryption technology of PGP, an electronic privacy program. With GnuPG, you can authenticate the validity of documents and encrypt/decrypt data to and from other recipients. GnuPG is capable of decrypting and verifying PGP 5.x files as well.

During installation, GnuPG is installed by default. That way you can immediately start using GnuPG to verify any packages that you receive from Red Hat. First, you must import Red Hat's public key.

38.3.1. Importing keys

To verify Red Hat packages, you must import the Red Hat GPG key. To do so, execute the following command at a shell prompt:

```
# rpm --import /usr/share/rhn/RPM-GPG-KEY
```

To display a list of all keys installed for RPM verification, execute the command:

```
# rpm -qa gpg-pubkey*
```

For the Red Hat key, the output includes:

```
gpg-pubkey-db42a60e-37ea5438
```

To display details about a specific key, use rpm **-qi** followed by the output from the previous command:

```
# rpm -qi gpg-pubkey-db42a60e-37ea5438
```

38.3.2. Verifying signature of packages

To check the GnuPG signature of an RPM file after importing the builder's GnuPG key, use the following command (replace <***rpm-file***> with filename of the RPM package):

```
# rpm -K <rpm-file>
```

If all goes well, the following message is displayed: **md5 gpg OK**. That means that the signature of the package has been verified and that it is not corrupt.

38.4. Diagnosing and fixing problems with RPM

Perhaps you have deleted some files by accident, but you are not sure what you deleted. To verify your entire system and see what might be missing, you could try the following command:

```
# rpm -Va
```

If some files are missing or appear to have been corrupted, you should probably either re-install the package or uninstall and then re-install the package.

At some point, you might see a file that you do not recognize. To find out which package owns it, enter:

```
# rpm -qf /usr/bin/ggv
```

The output would look like the following:

```
ggv-2.6.0-2
```

We can combine the above two examples in the following scenario. Say you are having problems with **/usr/bin/paste**. You would like to verify the package that owns that program, but you do not know which package owns paste. Enter the following command,

```
# rpm -Vf /usr/bin/paste
```

and the appropriate package is verified.

Do you want to find out more information about a particular program? You can try the following command to locate the documentation which came with the package that owns that program:

```
# rpm -qdf /usr/bin/free
```

The output would be similar to the following:

```
/usr/share/doc/procps-3.2.3/BUGS
/usr/share/doc/procps-3.2.3/FAQ
/usr/share/doc/procps-3.2.3/NEWS
[.....]
/usr/share/man/man5/sysctl.conf.5.gz
/usr/share/man/man8/sysctl.8.gz
/usr/share/man/man8/vmstat.8.gz
```

You may find a new RPM, but you do not know what it does. To find information about it, use the following command:

```
# rpm -qip crontabs-1.10-7.noarch.rpm
```

The output would be similar to the following:

```
Name       : crontabs                  Relocations: (not relocatable)
Version    : 1.10                       Vendor: Red Hat, Inc
Release    : 7                     Build Date: Mon 20 Sep 2004 05:58:10 PM EDT
Install Date: (not installed)            Build Host: tweety.build.redhat.com
Group      : System Environment/Base      Source RPM: crontabs-1.10-7.src.rpm
Size       : 1004                      License: Public Domain
Signature  : DSA/SHA1, Wed 05 Jan 2005 06:05:25 PM EST, Key ID 80cddb42a60e
Packager   : Red Hat, Inc <http://bugzilla.redhat.com/bugzilla>
Summary    : Root crontab files used to schedule the execution of programs.
Description :
The crontabs package contains root crontab files. Crontab is the
program used to install, uninstall, or list the tables used to drive the
cron daemon. The cron daemon checks the crontab files to see when
particular commands are scheduled to be executed. If commands are
scheduled, then it executes them.
```

Perhaps you now want to see what files the crontabs RPM installs. You would enter the following:

```
# rpm -qlp crontabs-1.10-5.noarch.rpm
```

The output is similar to the following:

```
/etc/cron.daily
```

/etc/cron.hourly
/etc/cron.monthly
/etc/cron.weekly
/etc/crontab
/usr/bin/run-parts

38.5. Build a simple RPM that packages a single file

The **rpmbuild** command is used to build a binary RPM from source code, as configured with a **.spec** file. As the command is not available by default, you'll have to install it from the **rpm-build** package. You should also install the **rpmdevtools** package, which will be useful.

- Install the necessary packages: **rpm-build** and **rpmdevtools**:

```
# yum install rpm-build  rpmdevtools
```

If possible avoid creating packages as 'root' as this is quite dangerous because we could delete or overwrite OS existng files.

- Setup the folder structure. Placed on the HOME folder of a user without admin privileges, run following command:

```
# rpmdev-setuptree
```

This will create rpmbuild structure where we will work on building packages

rpmbuild/SRPMS
rpmbuild/SPECS
rpmbuild/SOURCES
rpmbuild/RPMS
rpmbuild/BUILD

This can be alternatively done by running '**mkdir**' command:

```
# mkdir -pv rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
```

- Under **rpmbuild/SOURCES** create a folder with source file(s). **Do not forget** to provide a revision number. For instance, our source file will be a script named '**HelloWorld.sh**'

```
# cd rpmbuild/SOURCES
# mkdir <HelloWorld-1.0>
# vi <HelloWorld-1.0/HelloWorld.sh>
# chmod 755 <HelloWorld-1.0/HelloWorld.sh>
# touch <HelloWorld-1.0/configure>
# chmod 755 <HelloWorld-1.0/configure>
# tar  czvf  <HelloWorld-1.0.tar.gz>  <HelloWorld-1.0>
```

- Create the spec file: This is the hard part - configuring the RPM on what to build, install, and configure. A sample spec file can be created by running:

```
# cd
# rpmdev-newspec  rpmbuild/SPECS/<HelloWorld-1.0.spec>
```

But there is still a lot to add and remove to make this work. This is a spec file created using the sample file:

```
Name:           HelloWorld
Version:        1.0                 <--- Version must be the same as the name of
                                         the folder containing the package file(s)
Release:        1%{?dist}           <--- Release number, added to package's name
Summary:        Hello World Script

Group:          Miscellaneous       <--- For general purposes use 'Miscellaneous'
License:        License text
# URL:
Source0:        HelloWorld-1.0.tar.gz      <--- Pay attention to the name of tar file
BuildArch:      noarch          <--- If nothing specified, current architecture is used
                                    [ i386 | x86_64 | noarch ]
BuildRoot:      %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)

# BuildRequires:                    <--- Required packages to build this one
# Requires:                         <--- Required packages for installing this one

%description
This is a text describing what the Package is meant for

%prep
%setup -q

%build
# %configure                        <--- We have nothing to configure or compile
# make %{?_smp_mflags}                   so we comment lines out

%install
rm -rf $RPM_BUILD_ROOT
# make install DESTDIR=$RPM_BUILD_ROOT          <--- We have nothing to compile
install -d -m 0755 $RPM_BUILD_ROOT/opt/HelloWorld
install -m 0755 HelloWorld.sh $RPM_BUILD_ROOT/opt/HelloWorld/HelloWorld.sh
^ These two directives indicate where folders and files will be created/installed
      and their effective rights

%clean
rm -rf $RPM_BUILD_ROOT
```

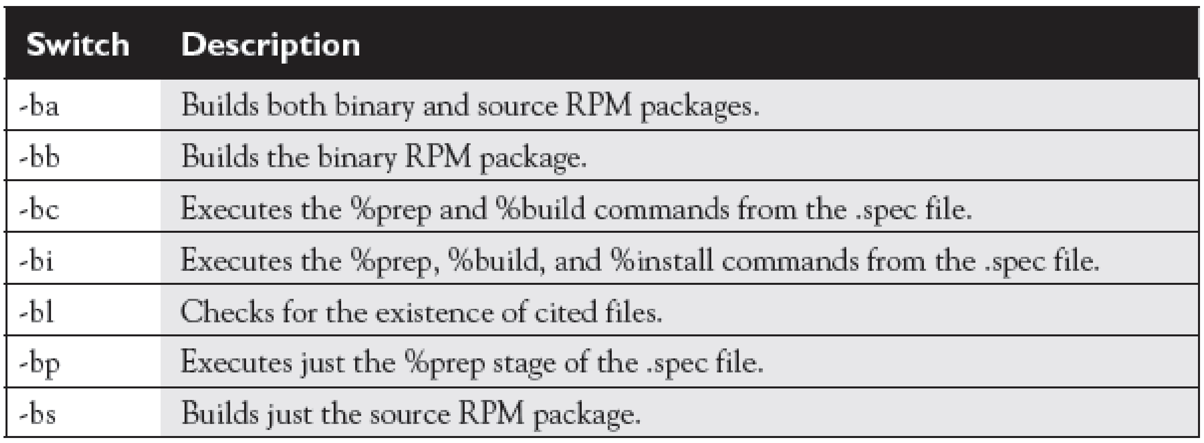
```
%files
%defattr(-,root,root,-)
# %doc
/opt/HelloWorld/HelloWorld.sh     <--- We confirm the file(s) to install

%changelog
```

- Build the RPM using **rpmbuild** command. Most documentation assumes the **rpmbuild** command uses a spec file in the **rpmbuild/SPECS** directory. In that case, you'd use the **-b** switch. Once the spec file is complete, build the package with:

```
# rpmbuild -ba SPECS/<HelloWorld-1.0.spec>
```

Assuming no errors occurred, new package is under the RPMS folder ready to be installed.

rpmbuild options:



Switch	Description
-ba	Builds both binary and source RPM packages.
-bb	Builds the binary RPM package.
-bc	Executes the %prep and %build commands from the .spec file.
-bi	Executes the %prep, %build, and %install commands from the .spec file.
-bl	Checks for the existence of cited files.
-bp	Executes just the %prep stage of the .spec file.
-bs	Builds just the source RPM package.

39. Manually upgrading the kernel

Red Hat Enterprise Linux kernels are packaged in the RPM format so that they are easy to upgrade and verify using the **Yum** or **PackageKit** package managers. **PackageKit** automatically queries the Red Hat Network servers and informs you of packages with available updates, including kernel packages.

This chapter is therefore only useful for users who need to manually update a kernel package using the **rpm** command instead of **yum**.

*****Warning: Building a custom kernel is not supported**

Building a custom kernel is not supported by the Red Hat Global Services Support team,

39.1. Overview of kernel packages

Red Hat Enterprise Linux contains the following kernel packages:

kernel — Contains the kernel for single, multicore and multiprocessor systems.

kernel-debug — Contains a kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.

kernel-debug-devel — Contains the development version of the kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.

kernel-devel — Contains the kernel headers and makefiles sufficient to build modules against the kernel package.

kernel-doc — Documentation files from the kernel source. Various portions of the Linux kernel and the device drivers shipped with it are documented in these files. Installation of this package provides a reference to the options that can be passed to Linux kernel modules at load time.

By default, these files are placed in the **/usr/share/doc/kernel-doc-<kernel_version>/** directory.

kernel-firmware — Contains all of the firmware files that are required by various devices to operate.

kernel-headers — Includes the C header files that specify the interface between the Linux kernel and user-space libraries and programs. The header files define structures and constants that are needed for building most standard programs.

perf — This package contains supporting scripts and documentation for the perf tool shipped in each kernel image subpackage.

39.2. Preparing to upgrade

Before upgrading the kernel, it is recommended that you take some precautionary steps.

First, ensure that working boot media exists for the system in case a problem occurs. If the boot loader is not configured properly to boot the new kernel, the system cannot be booted into Red Hat Enterprise Linux without working boot media.

USB media often comes in the form of flash devices sometimes called pen drives, thumb disks, or keys, or as an externally-connected hard disk device. Almost all media of this type is formatted as a **VFAT** file system. You can create bootable USB media on media formatted as **ext2**, **ext3**, or **VFAT**.

You can transfer a distribution image file or a minimal boot media image file to USB media. Make sure that sufficient free space is available on the device. Around **4 GB** is required for a distribution DVD image, around **700 MB** for a distribution CD image, or around **10 MB** for a minimal boot media image.

You must have a copy of the **boot.iso** file from a Red Hat Enterprise Linux installation DVD, or installation CD-ROM#1, and you need a USB storage device formatted with the **VFAT** file system and around **16 MB** of free space. The following procedure will not affect existing files on the USB storage device unless they have the same path names as the files that you copy onto it. To create USB boot media, perform the following commands as the root user:

1.- Install the SYSLINUX bootloader on the USB storage device:

```
# syslinux /dev/sdX1
```

...where ***sdX*** is the device name.

2.- Create mount points for boot.iso and the USB storage device:

```
# mkdir /mnt/isoboot /mnt/diskboot
```

3.- Mount boot.iso:

```
# mount -o loop boot.iso /mnt/isoboot
```

4.- Mount the USB storage device:

```
# mount /dev/<sdX1> /mnt/diskboot
```

5.- Copy the ISOLINUX files from the boot.iso to the USB storage device:

```
# cp /mnt/isoboot/isolinux/* /mnt/diskboot
```

6.- Use the **isolinux.cfg** file from **boot.iso** as the **syslinux.cfg** file for the USB device:

```
# grep -v local /mnt/isoboot/isolinux/isolinux.cfg > /mnt/diskboot/syslinux.cfg
```

7.- Unmount boot.iso and the USB storage device:

```
# umount /mnt/isoboot /mnt/diskboot
```

8.- You should reboot the machine with the boot media and verify that you are able to boot with it before continuing.

Alternatively, on systems with a floppy drive, you can create a boot diskette by installing the mkbootdisk package and running the **mkbootdisk** command as root. Refer to man mkbootdisk man page after installing the package for usage information.

To determine which kernel packages are installed, execute the command **yum list installed "kernel-*"** at a shell prompt. The output will comprise some or all of the following packages, depending on the system's architecture, and the version numbers may differ:

```
# yum list installed "kernel-*"
kernel.x86_64                2.6.32-17.el6        installed
kernel-doc.noarch            2.6.32-17.el6        installed
kernel-firmware.noarch       2.6.32-17.el6        installed
kernel-headers.x86_64        2.6.32-17.el6        installed
```

From the output, determine which packages need to be downloaded for the kernel upgrade. For a single processor system, the only required package is the kernel package.

39.3. Performing the upgrade

After retrieving all of the necessary packages, it is time to upgrade the existing kernel.

*****Warning**: It is strongly recommended to keep the old kernel in case there are problems with the new kernel.

At a shell prompt, change to the directory that contains the kernel RPM packages. Use **-i** argument with the **rpm** command to keep the old kernel. Do not use the **-U** option, since it overwrites the currently installed kernel, which creates boot loader problems. For example:

```
# rpm -ivh kernel-<kernel_version>.<arch>.rpm
```

The next step is to verify that the initial RAM disk image has been created.

39.4. Verifying the initial RAM Disk image

The job of the initial RAM disk image is to preload the block device modules, such as for IDE, SCSI or RAID, so that the root file system, on which those modules normally reside, can then be accessed and mounted. On Red Hat Enterprise Linux 6 systems, whenever a new kernel is installed using either the **Yum**, **PackageKit** or **RPM** package manager, the **Dracut** utility is always called by the installation scripts to create an ***initramfs*** (initial RAM disk image).

On all architectures other than IBM eServer System i you can create an **initramfs** by running the **dracut** command. However, you usually don't need to create an **initramfs** manually: this step is automatically performed if the kernel and its associated packages are installed or upgraded from RPM packages distributed by Red Hat.

You can verify that an **initramfs** corresponding to your current kernel version exists and is specified correctly in the **grub.conf** configuration file by following this procedure:

Verifying the Initial RAM Disk Image

1.- As root, list the contents in the **/boot/** directory and find the kernel (**vmlinuz-<*kernel_version*>**) and **initramfs**-<***kernel_version***> with the latest (most recent) version number:

Ensuring that the kernel and initramfs versions match:

```
# ls /boot/
config-2.6.32-17.el6.x86_64          lost+found
config-2.6.32-19.el6.x86_64          symvers-2.6.32-17.el6.x86_64.gz
config-2.6.32-22.el6.x86_64          symvers-2.6.32-19.el6.x86_64.gz
efi                                  symvers-2.6.32-22.el6.x86_64.gz
grub                                 System.map-2.6.32-17.el6.x86_64
initramfs-2.6.32-17.el6.x86_64.img   System.map-2.6.32-19.el6.x86_64
initramfs-2.6.32-19.el6.x86_64.img   System.map-2.6.32-22.el6.x86_64
initramfs-2.6.32-22.el6.x86_64.img   vmlinuz-2.6.32-17.el6.x86_64
initrd-2.6.32-17.el6.x86_64kdump.img vmlinuz-2.6.32-19.el6.x86_64
initrd-2.6.32-19.el6.x86_64kdump.img vmlinuz-2.6.32-22.el6.x86_64
initrd-2.6.32-22.el6.x86_64kdump.img
```

This shows that:
- we have three kernels installed (or, more correctly, three kernel files are present in /boot/),
- the latest kernel **is vmlinuz-2.6.32-22.el6.x86_64**, and
- an initramfs file matching our kernel version,
initramfs-2.6.32-22.el6.x86_64kdump.img, also exists.

***** Note**: initrd files in the **/boot** directory are not the same as initramfs files:
In the **/boot/** directory you may find several **initrd**-<***version***>**kdump.img** files. These are special files created by the **Kdump** mechanism for kernel debugging purposes, are not used to boot the system, and can safely be ignored.

2.- (Optional) If your **initramfs**-<***kernel_version***> file does not match the version of the latest kernel in **/boot/**, or, in certain other situations, you may need to generate an **initramfs** file with the **Dracut** utility. Simply invoking **dracut** as root without options causes it to generate an **initramfs** file in the **/boot/** directory for the latest kernel present in that directory:

```
# dracut
```

You must use the **--force** option if you want **dracut** to overwrite an existing **initramfs** (for example, if your **initramfs** has become corrupt). Otherwise **dracut** will refuse to overwrite the existing **initramfs** file:

```
# dracut
Will not override existing initramfs (/boot/initramfs-2.6.32-22.el6.x86_64.img)
without --force
```

You can create an initramfs in the current directory by calling **dracut** <***initramfs_name***> <***kernel_version***>:

```
# dracut "initramfs-$(uname -r).img" $(uname -r)
```

If you need to specify specific kernel modules to be preloaded, add the names of those modules (minus any file name suffixes such as **.ko**) inside the parentheses of the **add_dracutmodules**="<***module***> [<***more_modules***>]" directive of the **/etc/dracut.conf** configuration file. You can list the file contents of an **initramfs** image file created by dracut by using the **lsinitrd** <***initramfs_file***> command:

```
# lsinitrd  initramfs-2.6.32-22.el6.x86_64.img
initramfs-2.6.32-22.el6.x86_64.img:
========================================================
dracut-004-17.el6
========================================================
drwxr-xr-x  23 root     root         0 May  3 22:34 .
drwxr-xr-x   2 root     root         0 May  3 22:33 proc
-rwxr-xr-x   1 root     root      7575 Mar 25 19:53 init
drwxr-xr-x   7 root     root         0 May  3 22:34 etc
drwxr-xr-x   2 root     root         0 May  3 22:34 etc/modprobe.d
[output truncated]
```

3.- Examine the **grub.conf** configuration file in the **/boot/grub/** directory to ensure that an **initrd initramfs**-<***kernel_version***>**.img** exists for the kernel version you are booting.

39.5. Verifying the Boot Loader

When you install a kernel using **rpm**, the kernel package creates an entry in the boot loader configuration file for that new kernel. However, **rpm** does not configure the new kernel to boot as the default kernel. You must do this manually when installing a new kernel with **rpm**.
It is always recommended to double-check the boot loader configuration file after installing a new kernel with **rpm** to ensure that the configuration is correct. Otherwise, the system may not be able to boot into Red Hat Enterprise Linux properly. If this happens, boot the system with the boot media created earlier and re-configure the boot loader.

39.5.1. Configuring the GRUB Boot Loader

GRUB's configuration file, **/boot/grub/grub.conf**, contains a few lines with directives, such as **default**, **timeout**, **splashimage** and **hiddenmenu** (the last directive has no argument). The remainder of the file contains 4-line stanzas that each refer to an installed kernel. These stanzas always start with a **title** entry, after which the associated **root**, **kernel** and **initrd** directives should always be indented. Ensure that each stanza starts with a **title** that contains a version number (in parentheses) that matches the version number in the **kernel /vmlinuz**-<***version_number***> line of the same stanza.

```
# cat /boot/grub/grub.conf

# grub.conf generated by anaconda
[comments omitted]
default=1
timeout=0
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu

title Red Hat Enterprise Linux (2.6.32-22.el6.x86_64)
  root (hd0,0)
  kernel /vmlinuz-2.6.32-22.el6.x86_64 ro root=/dev/mapper/vg_vm6b-lv_root
        rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS rd_NO_MD rd_NO_DM
        LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc
        KEYTABLE=us rhgb quiet crashkernel=auto
  initrd /initramfs-2.6.32-22.el6.x86_64.img

title Red Hat Enterprise Linux (2.6.32-19.el6.x86_64)
  root (hd0,0)
  kernel /vmlinuz-2.6.32-19.el6.x86_64 ro root=/dev/mapper/vg_vm6b-lv_root
        rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS rd_NO_MD rd_NO_DM
        LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc
        KEYTABLE=us rhgb quiet crashkernel=auto
  initrd /initramfs-2.6.32-19.el6.x86_64.img

title Red Hat Enterprise Linux 6 (2.6.32-17.el6.x86_64)
  root (hd0,0)
  kernel /vmlinuz-2.6.32-17.el6.x86_64 ro root=/dev/mapper/vg_vm6b-lv_root
        rd_LVM_LV=vg_vm6b/lv_root rd_NO_LUKS rd_NO_MD rd_NO_DM
        LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc
        KEYTABLE=us rhgb quiet
  initrd /initramfs-2.6.32-17.el6.x86_64.img
```

If a separate **/boot/** partition was created, the paths to the kernel and the **initramfs** image are relative to **/boot/**. This is the case in example above. Therefore the **initrd /initramfs-2.6.32-22.el6.x86_64.img** line in the first kernel stanza means that the **initramfs** image is actually located at **/boot/initramfs-2.6.32-22.el6.x86_64.img** when the root file system is mounted, and likewise for the kernel path (for example: **kernel /vmlinuz-2.6.32-22.el6.x86_64**) in each stanza of **grub.conf**.

***** Tip**: The initrd directive in grub.conf refers to an initramfs image. In kernel boot stanzas in **grub.conf**, the **initrd** directive must point to the location (relative to the **/boot/** directory if it is on a separate partition) of the ***initramfs*** file corresponding to the same kernel version. This directive is called **initrd** because the previous tool which created initial RAM disk images, **mkinitrd**, created what were known as **initrd** files. Thus the **grub.conf** directive remains **initrd** to maintain compatibility with other tools. The file-naming convention of systems using the **dracut** utility to create the initial RAM disk image is: **initramfs-<*kernel_version*>.img**
Dracut is a new utility available in Red Hat Enterprise Linux 6, and much-improved over **mkinitrd**.

You should ensure that the kernel version number as given on **the kernel /vmlinuz-<*kernel_version*>** line matches the version number of the **initramfs** image given on the **initrd /initramfs-<*kernel_version*>.img** line of each stanza.

The **default=** directive tells GRUB which kernel to boot by default. Each **title** in **grub.conf** represents a bootable kernel. GRUB counts the **title**d stanzas representing bootable kernels starting with **0**. In the example above the line **default=1** indicates that GRUB will boot, by default, the second kernel entry, i.e. **title Red Hat Enterprise Linux (2.6.32-19.el6.x86_64)**.

In the example GRUB is therefore configured to boot an older kernel, when we compare by version numbers. In order to boot the newer kernel, which is the first **title** entry in **grub.conf**, we would need to change the **default** value to **0**.

After installing a new kernel with **rpm**, verify that **/boot/grub/grub.conf** is correct, change the **default=** value to the new kernel (while remembering to count from **0**), and reboot the computer into the new kernel. Ensure your hardware is detected by watching the boot process output.

If GRUB presents an error and is unable to boot into the default kernel, it is often easiest to try to boot into an alternative or older kernel so that you can fix the problem.

***** Note**: Causing the GRUB boot menu to display. If you set the **timeout** directive in **grub.conf** to **0**, GRUB will not display its list of bootable kernels when the system starts up. In order to display this list when booting, press and hold any alphanumeric key while and immediately after BIOS information is displayed, and GRUB will present you with the GRUB menu.

Alternatively, use the boot media you created earlier to boot the system.

40. Kernel parameters

40.1. Using /proc/sys and sysctl to modify and set kernel run-time parameters

1.- In its simplest form, this is a fairly straight forward task. If you view the file **/etc/sysctl.conf**, you will see several attributes with their appropriate values, these are the values applied at startup. This file can be edited directly with a text editor, then the values reloaded by executing **sysctl -p**

2.- These values can also be changed at runtime by modifying the values under **/proc/sys**. For instance, in **/etc/sysctl.conf** there is an attribute named **net.ipv4.ip_forward**, this attribute can also be viewed or modified as **/proc/sys/net/ipv4/ip_forward**.

To view the running value, run:

```
# cat /proc/sys/net/ipv4/ip_forward
```

To change the running value, run:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

The kernel attributes and values available to change can be found by either browsing the **/proc/sys** folders, or by running **sysctl -a**

3.- Lastly, values can be hot modified by running **sysctl -w** command:

```
# /sbin/sysctl -w kernel.domainname="example.com"
```

***** Note**: Take into account that values that are not in **/etc/sysctl.conf** won't be persistent through a system reboot.

41. Other commands & abilities

41.1. 'dmsetup'

dmsetup - low level logical volume management. It manages logical devices that use the device-mapper driver. Devices are created by loading a table that specifies a target for each sector (512 bytes) in the logical device.

```
# dmsetup info <device_name>
```

Outputs some brief information about the device in the form:
State: SUSPENDED|ACTIVE, READ-ONLY
Tables present: LIVE and/or INACTIVE
Open reference count
Last event sequence number (used by wait)
Major and minor device number
Number of targets in the live table
UUID

41.2. Hard and soft links

Hard Links

A hard link is a link where two files are really the same file.

Watch how when we create a file, and link to it with a hard link, the inodes (exact location on the harddisk) are the same.

```
# touch file.txt
# ln file.txt file1.txt
# ls -li file*
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file1.txt
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file.txt
```

When we create a third file linking it to the original, we see the same thing. They all are have an inode of 524594.

```
# ln file.txt file2.txt
# ls -li file*
524594 -rw-r--r--. 3 root root 0 Mar 21 12:54 file1.txt
524594 -rw-r--r--. 3 root root 0 Mar 21 12:54 file2.txt
524594 -rw-r--r--. 3 root root 0 Mar 21 12:54 file.txt
```

What happens if we delete the original file?

```
# rm file.txt
rm: remove regular empty file `file.txt'? y
# ls -li file*
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file1.txt
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file2.txt
```

As you can see, the other two files are intact and have not been removed, even though the original file is gone. That is because they are all the same file, when you make a hard link to it you are just putting another reference to it with a different name. Until the last file with that inode gets deleted, that file lives on.

Let's put some text in file2.txt and see what happens

```
# echo "things" >> file2.txt
# ls -li file*
524594 -rw-r--r--. 2 root root 7 Mar 21 13:01 file1.txt
524594 -rw-r--r--. 2 root root 7 Mar 21 13:01 file2.txt
# cat file1.txt
things
# cat file2.txt
things
```

As you can see, the files both grew to 7 bytes, and when we look inside each one, they both have the same text. That's because they are the same.

Soft Links

A soft link is much different from a hard link. Most people relate hard links to shortcuts in Windows. When you put a shortcut on your Desktop, it is just a link to the something on your computer. If you delete it no biggie, it's just a link. Soft links are the same way.

```
# touch testfile.txt
# ln -s testfile.txt testfile1.txt
# ls -li testfile*
524726 lrwxrwxrwx. 1 root root 12 Mar 21 13:11 testfile1.txt -> testfile.txt
524725 -rw-r--r--. 1 root root  0 Mar 21 13:11 testfile.txt
```

Here we created a file, testfile.txt, and then ran **ln -s** to create a soft link. When we ran **ls -li** we see that now the inodes are different, and testfile1.txt shows highlighted with an arrow to testfile.txt.

OK, so now let's repeat what we did above for hard links. I will make another soft link, linking to the original file testfile.txt and call it testfile2.txt. Then delete the original and **ls -li**

```
# ln -s testfile.txt testfile2.txt
# ls -li testfile*
524726 lrwxrwxrwx. 1 root root 12 Mar 21 13:11 testfile1.txt -> testfile.txt
524727 lrwxrwxrwx. 1 root root 12 Mar 21 13:15 testfile2.txt -> testfile.txt
524725 -rw-r--r--. 1 root root  0 Mar 21 13:11 testfile.txt
# rm testfile.txt
rm: remove regular empty file `testfile.txt'? y
# ls -li testfile*
524726 lrwxrwxrwx. 1 root root 12 Mar 21 13:11 testfile1.txt -> testfile.txt
        (these two lines have a background of red,
524727 lrwxrwxrwx. 1 root root 12 Mar 21 13:15 testfile2.txt -> testfile.txt
        signifying that they are broken)
```

If we try and cat the two files, to see the contents, we get an error. We can no longer access these files, they are broken links.

```
# cat testfile*
cat: testfile1.txt: No such file or directory
cat: testfile2.txt: No such file or directory
```

41.3. Using "tar", "gzip", and "bzip2" to archive and compress files

41.3.1. tar

Create a tar file from a folder called test1

```
# tar cvf test1.tar test1
```

c = create
v = verbose
f = file

Extract test1.tar

```
# tar xvf test1.tar
```

x = extract
v = verbose
f = file

List contents of tar archive

```
# tar tf test1.tar
```

41.3.2. gzip

This is most commonly used in combination with **tar**, using the **z** switch. **tar** itself does not compress, it just packs.

```
# tar cvzf test1.tar.gz test1
```

Although it can be used by itself

```
# gzip test1
```

```
# gunzip test1.gz
```

*** **Note**: This does not preserve the .gz file, it extracts it and removes it.

41.3.3. Reading "gzip" compressed text files

```
# zcat <messages>.gz
# zmore <messages>.gz
# zless <messages>.gz
```

41.3.4. bzip2

bzip2 uses a different algorithm to compress files than the other tools, but very similar options

Create a bzip2 file

```
# bzip2 test1
```

```
# bunzip2 test1.bz2
```

*** **Note**: This does not preserve the .gz file, it extracts it and removes it. Also does not compress directories, only files.

41.3.5. Archiving file systems with ACLs

The "**star**" utility is similar to the **tar** utility in that it can be used to generate archives of files; however, some of its options are different:

Option	Description
-c	Creates an archive file.
-n	Do not extract the files; use in conjunction with -x to show what extracting the files does.
-r	Replaces files in the archive. The files are written to the end of the archive file, replacing any files with the same path and file name.
-t	Displays the contents of the archive file.
-u	Updates the archive file. The files are written to the end of the archive if they do not exist in the archive or if the files are newer than the files of the same name in the archive. This option only work if the archive is a file or an unblocked tape that may backspace.
-x	Extracts the files from the archive. If used with -U and a file in the archive is older than the corresponding file on the file system, the file is not extracted.
-**help**	Displays the most important options.
-**xhelp**	Displays the least important options.
-/	Do not strip leading slashes from file names when extracting the files from an archive. By default, they are striped when files are extracted.
-**acl**	**When creating or extracting, archive or restore any ACLs associated with the files and directories.**

41.4. 'dig'

dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than dig.

```
# dig redhat.com

; <<>> DiG 9.3.6-P1-RedHat-9.3.6-16.P1.el5 <<>> redhat.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36491
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 0

;; QUESTION SECTION:
;redhat.com.                IN    A

;; ANSWER SECTION:
redhat.com.        60     IN    A     209.132.183.181

;; AUTHORITY SECTION:
redhat.com.        600    IN    NS    ns4.redhat.com.
redhat.com.        600    IN    NS    ns1.redhat.com.
redhat.com.        600    IN    NS    ns3.redhat.com.
redhat.com.        600    IN    NS    ns2.redhat.com.
```

```
;; Query time: 31 msec
;; SERVER: 193.57.239.150#53(193.57.239.150)
;; WHEN: Mon Apr  2 13:05:18 2012
;; MSG SIZE  rcvd: 116
```

41.5. 'pidof'

pidof -- find the process ID of a running program:

```
# ps -ef | grep -v grep | grep syslog
root     28842     1  0 Mar30 ?        00:00:00 supervising syslog-ng
root     28843 28842  0 Mar30 ?        00:00:00 /opt/syslog-ng/sbin/syslog-ng --no-caps

# pidof syslog-ng
28843 28842
```

41.6. 'less'

less is a program similar to **more**, but which allows backward movement in the file as well as forward movement. Also, **less** does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like **vi**.

41.7. 'watch'

The **watch** command issues another command every two seconds, updating by clearing the screen and writing the new output to the same screen location.

```
watch  <command>
```

You can control the delay between updates by using the -n option, and can cause any changes between updates to be highlighted by using the -d option, as in the following command:

```
watch -n 1 -d <command>
```

41.8. 'newrole'

newrole - run a shell with a new SELinux role.

```
# newrole [-r|--role] ROLE [-t|--type] TYPE [-l|--level] LEVEL [-- [ARGS]...]
```

Run a new shell in a new context. The new context is derived from the old context in which **newrole** is originally executed. If the **-r** or **--role** option is specified, then the new context will have the role specified by **ROLE**. If the **-t** or **--type** option is specified, then the new context will have the type (domain) specified by **TYPE**. If a role is specified, but no type is specified, the default type is derived from the specified role. If the **-l** or **--level** option is specified, then the new context will have the sensitivity level specified by **LEVEL**. If **LEVEL** is a range, the new context will have the sensitivity level and clearance specified by that range.

41.9. 'chattr'/'lsattr'

chattr changes the file attributes on a Linux file system.

```
# chattr [ -RVf ] [ -v version ] [ mode ] files...
```

The format of a symbolic mode is **+-=[acdeijstuADST]**.

The operator '**+**' causes the selected attributes to be added to the existing attributes of the files; '**-**' causes them to be removed; and '**=**' causes them to be the only attributes that the files have.

The letters 'acdeijstuADST' select the new attributes for the files: append only (**a**), compressed (**c**), no dump (**d**), extent format (**e**), immutable (**i**), data journalling (**j**), secure deletion (**s**), no tail-merging (**t**), undeletable (**u**), no atime updates (**A**), synchronous directory updates (**D**), synchronous updates (**S**), and top of directory hierarchy (**T**).

The following attributes are read-only, and may be listed by **lsattr** but not modified by chattr: huge file (**h**), compression error (**E**), indexed directory (**I**), compression raw access (**X**), and compressed dirty file (**Z**).

-R - Recursively change attributes of directories and their contents.
-V - Be verbose with chattr's output and print the program version.
-f - Suppress most error messages.
-v ***version*** - Set the file's version/generation number.

41.10. 'lsmod'/'modinfo'/'modprobe'/'insmod'/'rmmod'

lsmod - program to show the status of modules in the Linux Kernel

modinfo - program to show information about a Linux Kernel module

```
# modinfo [-0]  [-F field]  [modulename|filename ...]
```

```
# modinfo -V
```

```
# modinfo -h
```

modprobe - program to add and remove modules from the Linux Kernel

insmod - simple program to insert a module into the Linux Kernel

```
# insmod [filename]  [module options ...]
```

rmmod - simple program to remove a module from the Linux Kernel

```
# rmmod [-f]  [-w]  [-s]  [-v]  [modulename]
```

41.11. 'nano'

nano - Nano's ANOther editor, an enhanced free Pico clone

```
# nano [OPTIONS] [[+LINE,COLUMN] FILE]...
```

nano is a small, free and friendly editor which aims to replace **Pico**, the default editor included in the non-free Pine package. Rather than just copying Pico's look and feel, **nano** also implements some missing (or disabled by default) features in Pico, such as "search and replace" and "go to line and column number.

41.12. 'apropos'/'which'/'whatis'/'makewhatis'

apropos - search the whatis database for strings

```
# apropos keyword1  keyword2 ...
```

which - shows the full path of (shell) commands. If found under $PATH

```
# which [options] [--] programname [...]
```

whatis - search the whatis database for complete words.

```
# whatis keyword1  keyword2 ...
```

makewhatis - Create the whatis database

```
# makewhatis [-u] [-v] [-w] [-s sections ] [-c [catpath]] [manpath]
```

makewhatis reads all the manual pages contained in the given sections of **manpath** or the preformatted pages contained in the given sections of **catpath**. For each page, it writes a line in the **whatis** database; each line consists of the name of the page and a short description, separated by a dash. The description is extracted using the content of the NAME section of the manual page.

41.13. 'w'/'who'/'whoami'/'id

w - Show who is logged on and what they are doing.

who - show who is logged on

id - print user identity for USERNAME, or the current user.

whoami - Print the user name associated with the current effective user ID. Same as **id -un**

42. System documentation: man, info, and in /usr/share/doc

The most commonly used help pages are **man pages**.

```
# man vim
```

This will give you the manual pages, with descriptions of the options, examples, and information about the application. If you don't remember exactly what man page you need, but you know what utility you are trying to use you can search man pages.

For example, lets find all man pages relating to LVM:

```
# man -k lvm
README.vmesa [perlvmesa] (1)  - building and installing Perl for VM/ESA
lvm                 (8)  - LVM2 tools
lvm.conf [lvm]      (5)  - Configuration file for LVM2
lvmchange           (8)  - change attributes of the logical volume manager
lvmconf             (8)  - LVM configuration modifier
lvmdiskscan         (8)  - scan for all devices visible to LVM2
lvmdump             (8)  - create lvm2 information dumps for diagnostic purposes
lvmsadc             (8)  - LVM system activity data collector
lvmsar              (8)  - LVM system activity reporter
perlvms             (1)  - VMS-specific documentation for Perl
pvcreate            (8)  - initialize a disk or partition for use by LVM
pvresize            (8)  - resize a disk or partition in use by LVM2
```

As you can see you get an output of results from the search. Really helpful in situations that you forgot the name of a certain utility.

man -k is equivalent to **apropos** command.

info is nearly identical, referencing the info docs. Its not quite as nice to use, and therefore is not as popular.

You could also get information from the **/usr/share/doc**. Here you can find other information about the program itself, or that particular version. The following output is a typical doc directory.

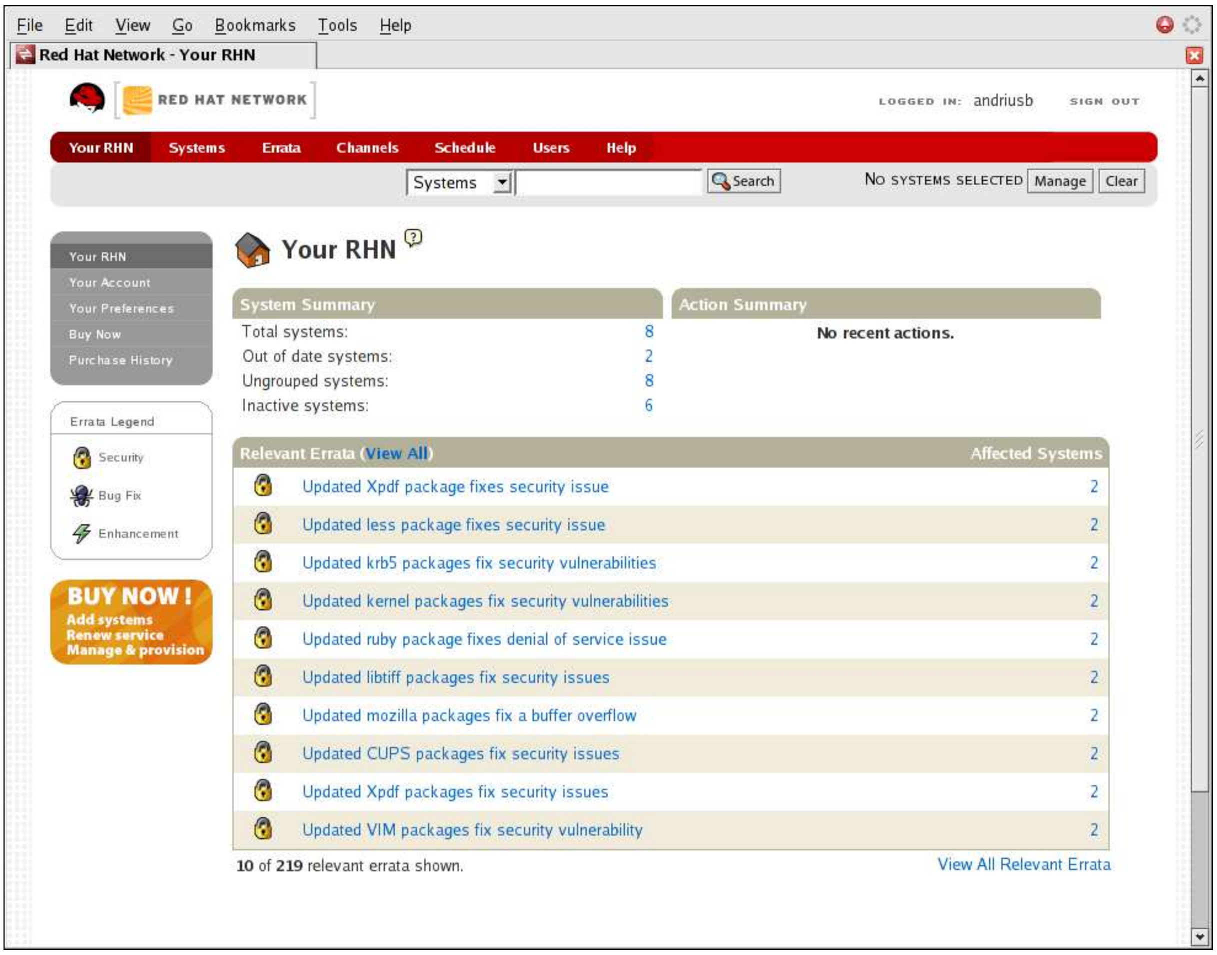
```
# pwd
/usr/share/doc/yum-3.2.27
# ls
AUTHORS  ChangeLog  COPYING  INSTALL  README  TODO
```

As you can see it's very different, simply text files with license, readme, install instructions, etc.

*****Tip**: If you don't get any output from man pages, try running **makewhatis** & **which** will build the man pages.

43. Red Hat Network

Red Hat Network is an Internet solution for managing one or more Red Hat Enterprise Linux systems. All Security Alerts, Bug Fix Alerts, and Enhancement Alerts (collectively known as Errata Alerts) can be downloaded directly from Red Hat using the **Package Updater** standalone application or through the RHN website available at https://rhn.redhat.com/



Your RHN

System Summary	
Total systems:	8
Out of date systems:	2
Ungrouped systems:	8
Inactive systems:	6

Action Summary

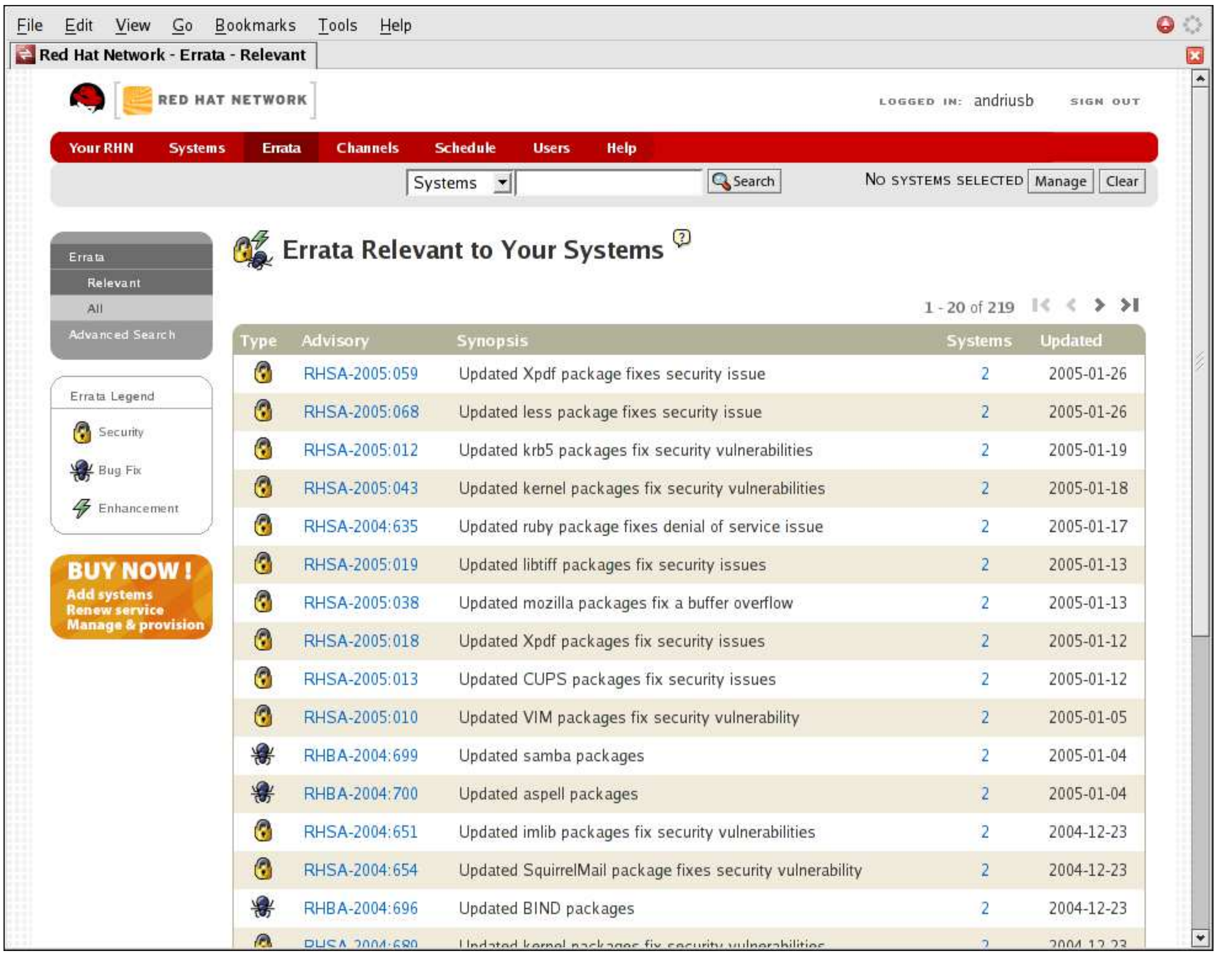
No recent actions.

Relevant Errata (View All)	Affected Systems
Updated Xpdf package fixes security issue	2
Updated less package fixes security issue	2
Updated krb5 packages fix security vulnerabilities	2
Updated kernel packages fix security vulnerabilities	2
Updated ruby package fixes denial of service issue	2
Updated libtiff packages fix security issues	2
Updated mozilla packages fix a buffer overflow	2
Updated CUPS packages fix security issues	2
Updated Xpdf packages fix security issues	2
Updated VIM packages fix security vulnerability	2

10 of 219 relevant errata shown. View All Relevant Errata

Red Hat Network saves you time because you receive email when updated packages are released. You do not have to search the Web for updated packages or security alerts. By default, Red Hat Network installs the packages as well. You do not have to learn how to use RPM or worry about resolving software package dependencies; RHN does it all.
Red Hat Network features include:

Errata Alerts — learn when Security Alerts, Bug Fix Alerts, and Enhancement Alerts are issued for all the systems in your network



Errata Relevant to Your Systems

1 - 20 of 219

Type	Advisory	Synopsis	Systems	Updated
🔒	RHSA-2005:059	Updated Xpdf package fixes security issue	2	2005-01-26
🔒	RHSA-2005:068	Updated less package fixes security issue	2	2005-01-26
🔒	RHSA-2005:012	Updated krb5 packages fix security vulnerabilities	2	2005-01-19
🔒	RHSA-2005:043	Updated kernel packages fix security vulnerabilities	2	2005-01-18
🔒	RHSA-2004:635	Updated ruby package fixes denial of service issue	2	2005-01-17
🔒	RHSA-2005:019	Updated libtiff packages fix security issues	2	2005-01-13
🔒	RHSA-2005:038	Updated mozilla packages fix a buffer overflow	2	2005-01-13
🔒	RHSA-2005:018	Updated Xpdf packages fix security issues	2	2005-01-12
🔒	RHSA-2005:013	Updated CUPS packages fix security issues	2	2005-01-12
🔒	RHSA-2005:010	Updated VIM packages fix security vulnerability	2	2005-01-05
🐞	RHBA-2004:699	Updated samba packages	2	2005-01-04
🐞	RHBA-2004:700	Updated aspell packages	2	2005-01-04
🔒	RHSA-2004:651	Updated imlib packages fix security vulnerabilities	2	2004-12-23
🔒	RHSA-2004:654	Updated SquirrelMail package fixes security vulnerability	2	2004-12-23
🐞	RHBA-2004:696	Updated BIND packages	2	2004-12-23
🔒	RHSA-2004:689	Updated kernel packages fix security vulnerabilities	2	2004-12-23

Automatic email notifications — Receive an email notification when an Errata Alert is issued for your system(s)
Scheduled Errata Updates — Schedule delivery of Errata Updates
Package installation — Schedule package installation on one or more systems with the click of a button
Package Updater — Use the Package Updater to download the latest software packages for your system (with optional package installation)
Red Hat Network website — Manage multiple systems, downloaded individual packages, and schedule actions such as Errata Updates through a secure Web browser connection from any computer

*** **Warning**: You must activate your Red Hat Enterprise Linux product before registering your system with Red Hat Network to make sure your system is entitled to the correct services. To activate your product, go to: http://www.redhat.com/apps/activate/

After registering, use one of the following methods to start receiving updates:

- Select **Applications** (the main menu on the panel) => **System Tools** => **Package Updater** on your desktop
- Execute the command **yum** from a shell prompt
- Use the RHN website at https://rhn.redhat.com/
- Click on the package icon when it appears in the panel to launch the **Package Updater**.

43.1. Install and update software packages from Red Hat Network, a remote repository, or from the local filesystem

```
# rhn_register
```

And follow the instructions. Now, if you aren't lucky enough to have a subscription to this wonderful service, then you will most likely using a repo that you create, or that is given to you.

Most common situation is having a remote repository that you need to pull packages from. Usually you be given a url to connect to looking something like this: **http://myremote.com/repo/i386/**. The **yum** repo files are located in **/etc/yum.repos.d/** and end with a **.repo** extension. The format is simple to setup a repo on the fly.

```
[myremote]
name=myremote
baseurl=http://myremote.com/repo/i386/
enabled=1
gpgcheck=0
```

Those are the essential elements to pull packages via **yum** from that repo.

Setting up a local repo with a disk is almost the same with a few steps before. First the disk needs to be mounted, and the packages copied from Packages/ into another directory on the server. In this case we will use **file:///directory/path/to/repo/** as the url, where **/directory/path/to/repo/** is the directory that contains the rpm files.

Next the package **creatrepo** needs to be installed. Once installed cd into the directory and run:

```
# createrepo .
```

Now that you have a repo setup, **yum** needs to know about it. Create a file named **mylocal.repo** in the **/etc/yum.repos.d/** directory.

```
[mylocal]
name=mylocal
baseurl=file:///directory/path/to/repo/
enabled=1
gpgcheck=0
```

Run a yum command to test, and it should be pulling information about packages from the local repo.

```
# yum list httpd
```