5

The Boot Process

CERTIFICATION OBJECTIVES

5.01	The BIOS and the UEFI	5.05	Network Configuration
5.02	Bootloaders and GRUB	5.06	Time Synchronization
5.03	Between GRUB and Login	✓	Two-Minute Drill
5.04	Control by Runlevel	Q&A	Self Test

This chapter is focused on what happens from the moment a system is powered up to the time a login prompt is available. Those are the fundamentals of the boot process. When RHEL 6 is properly installed, the BIOS/UEFI points to the a specific media device. Assuming it's a local hard drive, the master boot record (MBR) of that device points to the GRUB bootloader. Once an option to boot RHEL 6 is selected in GRUB, the associated commands point to and initializes the Linux kernel. which then starts **init**, the first Linux process. The **init** process then initializes the system and moves into appropriate runlevels. When Linux boots into a specific runlevel, it starts a series of services, including the client associated with the Network Time Protocol (NTP). You can customize this process.

INSIDE THE EXAM

Understanding the Boot Process

Objectives related to the boot process have been consolidated into the RHCSA exam. Perhaps the most basic skill related to the boot process is an understanding of the commands that start and stop the boot process, such as **shutdown** and **reboot**:

- Boot, reboot, and shut down a system normally

Of course, that starts with the way a system is powered up. In this chapter, you'll review the standard Linux runlevels. From the standard RHEL 6 boot menu, you need to know how to:

- Boot systems into different runlevels manually

Closely related to this objective is this one:

- Use single-user mode to gain access to a system

If you are already familiar with single-user mode, you should understand that "access" in single user mode is password-free access to the root administrative account. An early release of the RHCSA objectives included the following phrase: "for which the root password is not known."

Also closely related is this objective, focused on the configuration file associated with different runlevels:

- Configure systems to boot into a specific runlevel automatically

As Linux is a network operating system, and as most users can't do much without networking, it's important to know how to

- Configure network services to start automatically at boot

With the focus on the boot process, you'll also learn how to

- Modify the system bootloader.

Closely related to these objectives, and part of the boot process, are objectives related to how filesystems are mounted, covered in Chapter 6.

CERTIFICATION OBJECTIVE 5.01

The BIOS and the UEFI

While not officially a Red Hat exam prerequisite or requirement, a basic understanding of the BIOS and the UEFI is a fundamental skill for all serious computer users. The UEFI has replaced the BIOS on many modern systems and can do so much more. But as the UEFI supports changes to boot media in similar ways, the functionality for our purposes is the same.

Because of the variety of BIOS/UEFI software available, this discussion is general. It's not possible to provide any sort of step-by-step instructions for modifying the wide array of available BIOS/UEFI menus. In any case, such instructions are not directly relevant either to the administration of Linux or to any of the Red Hat exams. However, these skills can help you boot from different Linux installation media, access default virtualization settings, and more.

Basic System Configuration

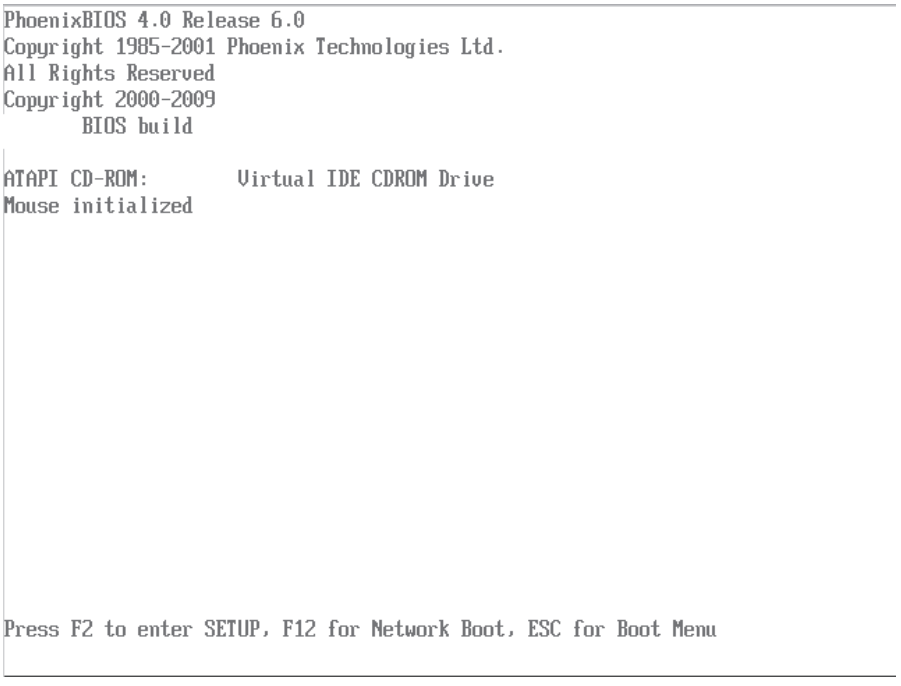
When a computer is powered up, the first thing that starts is the BIOS/UEFI. Based on settings stored in stable, read-only memory, the BIOS/UEFI system performs a series of diagnostics to detect and connect the CPU and key controllers. This is known as the Power On Self Test (POST). If you hear beeps during this process, there may be a hardware problem such as an improperly connected hard drive controller. The BIOS/UEFI system then looks for attached devices such as the graphics card. After the graphics hardware is detected, you may see a screen similar to Figure 5-1, which displays other hardware as detected, tested, and verified.

If your system has an UEFI menu, it may include a Trusted Platform Module (TPM). While it's built to enhance security on a system, it has caused controversy within the open-source community, due to privacy issues. Many open-source professionals are working to minimize any such problems through the Open Trusted Computing (OpenTC) group of the European Union. RHEL 6 takes advantage of TPM hardware features to enhance system security.

Once complete, the BIOS/UEFI passes control to the MBR of the boot device, normally the first hard drive. The first stage of the GRUB bootloader is normally copied to the MBR. It serves as a pointer to the other information from the GRUB menu. At that point, you should see a bootloader screen.

FIGURE 5-1

The BIOS Initialization menu.



Startup Menus

Generally, the only reason to go into the BIOS/UEFI menu during the Red Hat exams is to boot from different media, such as a CD, floppy, or USB key. In many cases, you can bypass this process. Return to Figure 5-1. The options for this particular system are shown in the bottom of the screen. In this case, pressing F2 enters SETUP, the BIOS menu; pressing F12 boots directly from a network device; and pressing ESC starts a boot menu. The actual keys vary by system.

In many cases, all you see after POST is a blank screen. The BIOS/UEFI is often configured in this way. In that case, you'll need to do some guessing based on your experience on how to reveal the screen shown in Figure 5-1 and access the boot or BIOS menu.

In many cases, boot menus are directly accessible by pressing a key such as ESC, DEL, F1, F2, or F12. Such boot menus may have entries similar to:

```
      Boot Menu
1. Removable Devices
2. Hard Drive
```

```
3. CD-ROM Drive
4. USB Drive
5. Built-In LAN
```

From that or similar menus, you should be able to select the desired boot device using the arrow and ENTER keys. If that doesn't work, you'll have to use the BIOS /UEFI menu to boot from the desired drive.

Access to Linux Bootloaders

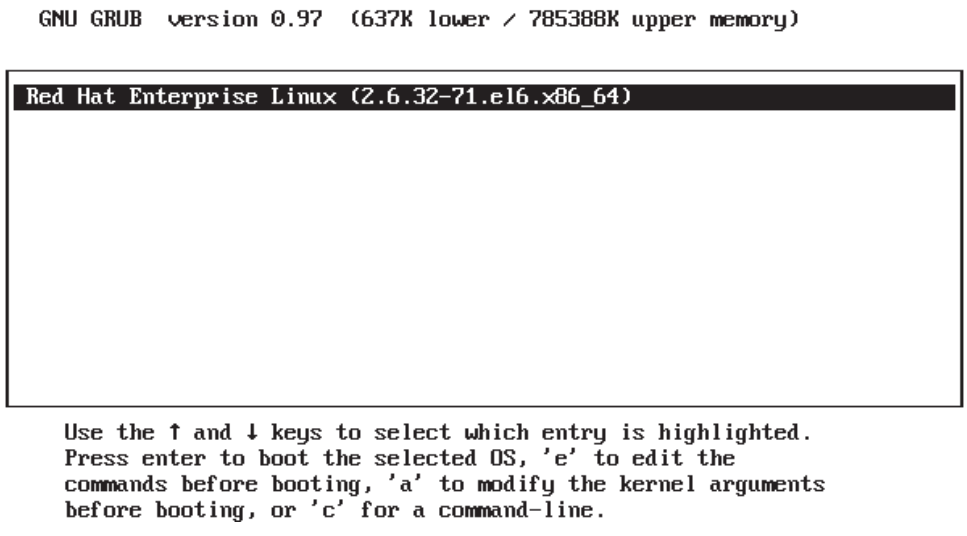
As noted in Chapter 2, the default bootloader is GRUB, and the first part of it (known as stage 1) is installed in the MBR of the default drive. Normally, the BIOS should automatically start the bootloader, with a message similar to:

```
Booting Red Hat Enterprise Linux Server (2.6.32-71.el6) in 5
seconds...
```

Alternatively, if you press a key before those five seconds are complete, GRUB will present a menu similar to that shown in Figure 5-2.

If the system includes more than one Linux kernel, or more than one operating system, there may be multiple choices available, which you can highlight with the UP ARROW and DOWN ARROW keys. To boot Linux from the highlighted option, press ENTER.

FIGURE 5-2 The GRUB menu



On old PCs (pre-twenty-first century), some BIOSes can't find your bootloader unless it's located within the first 1024 cylinders of the hard disk. For that reason, the partition where the /boot directory is configured is normally the first available primary partition.

On systems with multiple hard drives, there is one more caveat. On PATA (Primary Advanced Technology Attachment) hard drives, the /boot directory must be on a hard drive attached to the primary PATA controller. If local drives are all SCSI (Small Computer Systems Interface) hard drives, the /boot directory must be located on a hard drive with SCSI ID 0 or ID 1. For systems with a mix of hard drives, the /boot directory must be located on either the first IDE drive or a SCSI drive with ID 0.

CERTIFICATION OBJECTIVE 5.02

Bootloaders and GRUB

The standard bootloader associated with Red Hat Enterprise Linux (RHEL) is GRUB, the GRand Unified Bootloader. Red Hat has not supported LILO, the Linux Loader, for years. As suggested by the Red Hat exam requirements, for the RHCSA exam, you need to know how to use the GRUB menu to boot into different runlevels, and diagnose and correct boot failures arising from bootloader errors. RHEL 6 uses the traditional version of GRUB, 0.97. The associated configuration file is relatively easy to understand and customize, as you'll see later in this chapter. Some Linux distributions such as Ubuntu have moved to GRUB 2.0. While the look and feel of the GRUB 2.0 menu is similar to what's seen on RHEL 6, the steps required to configure that bootloader are quite different.

If you want to take full advantage of the security features associated with TPM, consider TrustedGRUB as a bootloader. TrustedGRUB can become part of the "chain of trust" from the TPM through RHEL 6, verifying the integrity of the local operating system. For more information, see http://sourceforge.net/projects/trustedgrub/.

GRUB, the GRand Unified Bootloader

The Red Hat has implemented GRUB as the only bootloader for its Linux distributions. It's normally configured to boot into a default kernel, presumably one associated with RHEL 6. The selected GRUB option finds the kernel in the /boot directory and finds the GRUB menu, which will look similar to Figure 5-2. You can use the GRUB menu to boot any operating system detected during the Linux installation process, or any other operating system added to appropriate configuration files.

GRUB is flexible. Not only can it be edited through its configuration file, but also it can be edited directly from the GRUB menu. If GRUB is password-protected, press P. You should now see the options shown in Figure 5-2. In that menu, you can press E to temporarily edit the file, press A to just add a directive to the end of the kernel command line, or press C to open a GRUB command prompt. This section is focused on booting into different runlevels.

Boot into Different Runlevels

To pass a parameter to the kernel through GRUB, press A at the first GRUB menu. This allows you to append the command sent to the kernel. You might then see a single line of commands similar to the following:

```
grub append> ro root=UUID=somelonghexadecimalnumber
rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us
crashkernel=auto rhgb quiet
```

Yeah, that's a lot of stuff that will be explained shortly. What matters for the RHCSA is that you can add more commands to the end of this line. For example, if you add the number **1** to the end of this line, Linux starts in single-user mode.

From single-user mode, type **exit** and the system will go into multiuser mode. If you have made changes or repairs to any partitions, the next step is to reboot the computer with the **reboot** command. At some point, changes made during a Red Hat exam should be tested with a reboot.

exam Watch

The reboot process takes valuable time. During an exam, you may want to wait until all relevant configuration changes are complete before testing the results with a reboot. Nevertheless, since changes must survive a reboot, you'll want to do so at least once.

The concept of the runlevel is detailed later in this chapter. For now, all you need to know is that when RHEL 6 is configured to boot into a GUI, it's configured to boot into runlevel 5 by default. That runlevel can be changed by appending a runlevel to the end of the kernel command line.

If you encounter a problem with a system booting into the GUI, the first thing to try is to add a **3** at the end of the kernel command line. If successful, it will boot RHEL 6 into text mode, with a command line console-based login.

If you need direct access into the root administrative account, add a **1** to the end of the kernel command line. That boots into runlevel 1, which automatically logs in to the account of the root administrative user. As that supports full root administrative privileges, including changes to the root administrative password, it's important to password-protect the GRUB menu, as discussed shortly.

In rare cases, some systems are so troubled, they don't boot into runlevel 1. In that case, two other runlevels are available:

- **single** Does everything but run the scripts listed in the /etc/rc1.d/ directory.
- **init=/bin/sh** Does not load init-related files; mounts only the top-level root directory (/) in read-only mode.

The runlevel option known as **emergency** is no longer recognized in RHEL.

Now you should understand how to boot into different runlevels during the boot process. As defined in the Red Hat Exam Prep guide, this is explicitly described as an RHCSA requirement:

Boot systems into different runlevels manually.

EXERCISE 5-1

Boot into a Different Runlevel

One key skill is knowing how to boot into a different runlevel. This exercise assumes you've configured RHEL 6 per Chapter 2, which sets the default runlevel as 5. Check your /etc/inittab file. If the current system reflects the defaults, it should read as follows:

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

Change this directive if needed and reboot your system. Now you can start the exercise.

1. When you see the following message, make sure to press any key to access the GRUB menu:

   ```
   Press any key to enter the menu
   ```

2. If the menu is password-protected, you'll have to press P to access the password prompt. Then you can enter the GRUB password. Next, you can press A to access the kernel command line.
3. At the end of the kernel command line, type a space followed by the runlevel of your choice. First, delete the **rhgb quiet**, enter **2**, and press B to boot this kernel.
4. Watch the boot messages. What kind of login screen do you see?
5. Log in to this system. You can use any existing user account.
6. Run the **reboot** command to restart this system.
7. Repeat Steps 1 through 3, but boot this system into runlevel **1**.
8. Watch the boot messages. What kind of login screen do you see? Do you have to log in at all?
9. Repeat Steps 1 through 3, but boot this system into runlevel **s**.
10. Watch the boot messages. What kind of login screen do you see? Do you have to log in at all?
11. Run the **reboot** command to restart this system.
12. Repeat Steps 1 through 3, but boot this system into runlevel **init=/bin/sh**.
13. Watch the boot messages. What kind of login screen do you see?
14. Run the **halt** command to stop this system.

Modify the System Bootloader

The RHCSA specifically requires that you need to know how to "modify the system bootloader." That means you need to know the GRUB configuration file in detail. It's available in the /boot/grub/grub.conf file. The following is a detailed analysis of a typical version of that configuration file. Some of the details, especially as related to the **kernel** command, is not necessary.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes
to this file
# NOTICE:  You have a /boot partition. This means that
#          all kernel and initrd paths are relative to /boot/,
eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/vda2
#          initrd /initrd-[generic-]version.img
# boot=/dev/vda
```

As you can see, even the comments are significant. The first line suggests that this file was created by Anaconda, the RHEL installation program. The next line notes that you don't have to rerun the **grub** command; in other words, configuration changes do not have to be written to the MBR. If you've previously configured GRUB (as is the default during the RHEL 6 installation process) there's already a pointer in the MBR that reads the current version of the GRUB configuration file.

The commented "NOTICE" in the configuration file appears when there's a separate partition for the /boot directory. The last commented line indicates the hard drive with the MBR—in this case, /dev/vda.

```
default=0
```

GRUB configuration options are organized in stanzas. The **default=0** may be slightly confusing, as it points to the first available stanza. A **default=1** would point to a second stanza; a **default=2** would point to a third stanza, and so on, if included in the configuration file.

```
timeout=5
```

In other words, if nothing is done in the five seconds specified by the **timeout=5** directive, GRUB automatically runs the commands in the first stanza.

```
splashimage=(hd0,0)/grub/splash.xpm.gz
```

The **splashimage** is the screen presented with the GRUB menu. Strangely enough, it's a black screen.

```
hiddenmenu
```

The **hiddenmenu** option does not display the menu automatically; instead, it displays the "Booting Red Hat Enterprise Linux" message described earlier. The directives that follow are the first stanza; additional stanzas are commonly shown for different versions of the kernel or even different operating systems such as Microsoft Windows.

```
title Red Hat Enterprise Linux (2.6.32-71.el6.x86_64)
```

The **title** directive displays the option shown in the GRUB menu.

```
        root (hd0,0)
```

Now this is really weird—there are two definitions for the word **root** in this file. First, the /boot directory in the GRUB configuration file is associated with root, in this case, **root(hd0,0)**.

```
root=UUID=somelonghexadecimalnumber
```

The UUID is an acronym for the universally unique identifier. It's a 128-bit number, expressed in hexadecimal (base 16) format. It's a unique number generated for each volume configured on RHEL 6. While you could use the **LABEL** directive from RHEL 5 or even the device file associated with the top-level root directory (**/**) volume, the UUID is the new default standard for RHEL 6 in the GRUB configuration file as well as the /etc/fstab file as discussed in Chapter 6.

The **root(hd0,0)** directive uses numbers starting with *0*. In other words, this directive points to the first partition on the first hard drive. The /boot directory is mounted on this partition. If it were the fifth partition on the first hard drive, for example, this directive would read *root(hd0,4)*.

The **timeout=5** directive specifies the time, in seconds, before GRUB automatically boots the default operating system. The **splashimage** directive locates the graphical GRUB screen. In this case, you can find it on the first partition of the first hard drive, in the /grub/splash.xpm.gz file. It happens to configure a black background. As **(hd0,0)** has been previously defined as the /boot directory, you can find the splash screen file in /boot/grub/splash.xpm.gz. The **hiddenmenu** directive means that the GRUB options are hidden, with the message shown here:

```
Booting Red Hat Enterprise Linux Server (2.6.32-71.el6.x86_64) in 5 seconds...
```

Each of the next two stanzas has a title normally associated with the operating system, such as:

```
title Red Hat Enterprise Linux Server (2.6.32-71.el6.x86_64)
```

This is a standard, based on the currently installed kernel. If multiple kernels are installed, you'll probably see more than one stanza. Both would boot RHEL 6, using a different kernel.

The next three lines specify the location of the /boot directory, the kernel, and the initial RAM disk, respectively. The **kernel** line may have a number of directives; it is actually one line in the actual GRUB configuration file.

```
root (hd0,0)
     kernel /vmlinuz-2.6.32-71.el6.x86_64 ro root=UUID=21754e41-bd5d-4faa-8c0d-
33c2e70950 rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=auto rhgb quiet
initrd /initrd-2.6.32-55.el6.i686.img
```

In this case, the /boot directory, as described earlier, is on the first partition of the first hard drive, as specified by **root (hd0,0)**. The kernel is specified by the vmlinuz-2.6.32-71.el6.x86_64 file, in the /boot directory. It's opened as read only (**ro**) to protect it from any accidental writes from the initial RAM disk; the actual top-level root directory is associated with the noted Universally Unique Identifier (UUID).

The **rd_NO_LUKS** disables detection of volumes encrypted to the Linux Unified Key Setup (LUKS) system. If a top-level root directory partition is encrypted, the system would be unbootable. The **rd_NO_LVM** disables detection of volumes configured with the Logical Volume Manager (LVM). The **rd_NO_MD** and **rd_NO_DM** options disables detection of the software version of Redundant Array of Independent Disks (RAID) devices. Except for RAID, which is not part of the Red Hat exams, these specialty systems are covered in Chapter 6.

The directives that follow specify language, font, and keyboard directives during the boot process, when the Initial RAM disk is loaded. These directives are **LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us**.

As suggested in the discussion on the Kickstart process in Chapter 2, in some cases you may need to change **crashkernel=auto** to **crashkernel=128M**. Finally, at the end of the line, the **rhgb quiet** directive hides the boot messages by default.

In most configurations, all that's absolutely needed is the following line:

```
kernel /vmlinuz-2.6.32-71.el6.x86_64 ro root=/dev/vda2
```

The initial RAM disk file (initrd-2.6.32-71.el6.x86_64.img) creates a temporary filesystem during the boot process. It includes kernel modules and user space programs needed to mount actual filesystems and run the first initialization programs.

The final three lines are trivial with respect to the Red Hat exams; however, many users see them when they configure their computers in a dual boot with Microsoft Windows. In most cases, you'll actually see the first line as:

```
title Other
```

On my laptop, I've modified it to specify the actual operating system on the other partition.

```
title Windows 7
```

With the following directive, GRUB points to the partition with Microsoft Windows 7, the second partition on the first hard drive:

```
rootnoverify (hd0,1)
```

This is different from the aforementioned **root (hd0,0)** directive, as **rootnoverify** does not attempt to mount the noted partition in a Linux fashion.

Finally, the **chainloader +1** directive points to the first sector of the noted partition, where Microsoft Windows 7 continues the boot process:

```
chainloader +1
```

More Options

Administrators can do more than just use GRUB to boot into a different runlevel. For example, if Linux does not recognize all of the RAM on the local system, try adding **mem=*xyz*M** (where *xyz*M represents the amount of memory on the local computer) at the end of the line. Alternatively, some problems related to graphics cards can be addressed by the **vga** directive; for example, the **vga=791** directive sets up a 16-bit 1024 × 768 screen.

A huge number of options are available, in a kernel-parameters.txt file. On RHEL 6, it's available from the kernel-doc package.

GRUB Security and Password Protection

If you've configured GRUB password protection during the RHEL installation process, you'll see a line similar to the following in the GRUB configuration file:

```
password --md5 $1$hfBhb8zA$sYrw4B1VzrrpPHpDtyhb.
```

This directive specifies a password hash, encrypted to the Message-Digest 5 algorithm. When it's included in the first GRUB commands, before the commands associated with individual kernels, it protects the GRUB menu. That can keep a cracker from breaking into this system and gaining access to the root administrative account.

If you want to create an MD5 password for this file, run the **grub-md5-crypt** command. You'll be prompted for a password that is converted to an MD5 hash. You can then copy this hash to the GRUB configuration file. If the **password** directive is included before a stanza, it password-protects the entire menu. Users won't be able to edit the GRUB menu unless they know the password. If the **password** directive is placed within a stanza, it prompts for the password before the option is selected.

On RHEL 6, a GRUB menu that's not password-protected may allow a user access to the root administrative account. To put it mildly, that's a serious security risk. Sure, access to the given system can be protected by other means. But password protection for the GRUB menu can at least ensure that anyone who gets access to that menu can't just boot into runlevel 1 (or any other runlevel) at will.

Anyone with access to a system CD/DVD drive can use the RHEL 6 installation DVD or network CD to start rescue mode. That also provides password-free access to the root administrative account. To prevent such security breaches, you may want to password-protect the BIOS/UEFI menu to prevent access or changes to the boot menu described earlier. Some systems may even include physical locks on the CD/DVD drive and even USB ports to prevent such unauthorized access to the root administrative account.

How to Update GRUB

If you've previously installed a different bootloader to the MBR, such as Microsoft's NTLDR or BOOTMGR, or TeraByte's BootIt Next Generation, just run the **grub-install** command. If it doesn't automatically write the GRUB pointer to the MBR, or multiple hard drives are available, you may need to include the hard drive device such as /dev/sdb. It's also possible to set up GRUB on a portable drive; just specify the device with the command.

When the GRUB configuration file is changed, no additional commands are required. The pointer from the MBR automatically reads the current version of the /boot/grub/grub.conf file.

Effects of GRUB Errors

An error in the GRUB configuration file can result in an unbootable system. For example, identifying the wrong volume as the root partition (/) can lead to a kernel panic. Other configuration errors in /boot/grub/grub.conf can also cause a kernel panic during the boot process.

Now that you've analyzed the GRUB configuration file, you can probably visualize some of the effects of errors in this file. If some of the filenames or partitions are wrong, GRUB won't be able to find critical files such as the Linux kernel.

First, if the **root (hd*x*,*y*)** directive does not point to the /boot directory, you'll see one of three possible error messages, described in Table 5-1.

If the GRUB configuration file is completely missing, you'll see a prompt similar to this:

```
grub>
```

on the job

It's possible to get to a* grub> *prompt from within Linux with the* grub *command. However, that option does not include command completion features; and some commands provide different information when Linux is running.

TABLE 5-1 GRUB Error Messages

Message	Description
Error 15: File not found	The partition was mounted; the kernel was not found on that partition. Cause: **root (hd*x*,*y*)** directive does not point to the partition with the /boot directory.
Error 17: Cannot mount selected partition	The partition is not formatted to a filesystem with files. Cause: the **root (hd*x*,*y*)** directive points to a partition formatted to a system such as Linux swap.
Error 22: No such partition	There is no partition specified by the **root (hd*x*,*y*)** directive.

The GRUB Command Line

If you see a GRUB command line, you may feel lost. To see a list of available commands, press the TAB key at the **grub>** prompt.

Some trial and error may be required. You should be able to find all detected hard drives on a standard PC from the BIOS/UEFI menus (SCSI drives can be a different story). You can use the **find** command to identify the partition with the GRUB configuration file.

By default, the /boot directory is mounted on a separate partition. For example, to find grub.conf on this particular system, start with the following command:

```
grub> find /grub/grub.conf
```

GRUB should return the partition with the /boot directory. In this case, it's the first partition on the first hard drive:

```
(hd0,0)
```

There's one more way to identify the partition with the /boot directory. Just run **root** at the **grub>** command line:

```
grub> root
(hd0,0): Filesystem type is ext2fs, partition type 0x83
```

Next, use that information to identify configured partitions:

```
grub> find (hd0,0)/grub/grub.conf
```

If the file is not on the noted partition, you'll see an "Error 15: File not found" error message. For the standard server1.example.com installation, the output includes the partitions with Linux-formatted drives:

```
(hd0,0)
(hd0,1)
(hd0,4)
```

We know that the /boot directory is on (hd0,0). While many administrators configure the /boot directory earlier in a drive sequence, it's not required. To confirm the location of grub.conf, run the **cat** command as follows to display the contents of the GRUB configuration file:

```
grub> cat (hd0,0)/grub/grub.conf
```

Now you can use these commands from the GRUB configuration file to boot Linux from the **grub>** command line. If the top-level root directory is normally

mounted on a partition, you may even confirm the contents of the /etc/fstab file with a command like the following:

```
grub> cat (hd0,1)/etc/fstab
```

Command completion works from the GRUB command line. For example, if you don't remember the name of the Kernel file, type **kernel /** and then press the TAB key to review the available files in the /boot directory.

EXERCISE 5-2

Using the GRUB Command Line

In this exercise, you'll use the printout of the GRUB configuration file from Exercise 5-1 to boot RHEL 6 manually. Look at the printout and identify the desired commands in the stanza. Now follow these steps:

1. Boot the system. When you see the following line at the top of the screen, press any key to access the GRUB menu:

   ```
   Press any key to enter the menu
   ```

2. If the GRUB configuration file is password-protected, you'll have to press P to access a password prompt.
3. Press C for a GRUB-based command line interface. You should see the **grub>** prompt.
4. Type in the commands listed in the selected stanza. Start by issuing the first **root** directive. If the command is successful, you should see output similar to:

   ```
   Filesystem type is ext2fs, partition type 0x83
   ```

   Other filesystems such as XFS lead to a slightly different result.
5. Enter the second command from your selected GRUB configuration file stanza, which specifies the kernel and root directory partition. Yes, this is a long line; however, you can use command completion (press the TAB key) to make it faster. In addition, the only important parts of the line are the kernel file, the "read-only" status, and the location of the top-level root directory, in other words:

   ```
   kernel /vmlinuz-2.6.32-71.el6.x86_64 ro root=/dev/vda2
   ```

If successful, you'll see the following message (the setup and size numbers may vary):

```
[Linux-bzImage, setup=0x1e00, size=0x16eb71]
```

6. Enter the third command from the stanza, which specifies the initial RAM disk command and file location. If successful, you'll see the following message (the numbers may vary):

```
[Linux-initrd @ 0x16544000, 0x19b6c7 bytes]
```

7. Now enter the **boot** command. If successful, Linux should now boot the selected kernel and initial RAM disk just as if you selected that option from the GRUB configuration menu.

Create Your Own GRUB Configuration File

If you need to rebuild the GRUB configuration file from scratch, a sample configuration file is available in the documentation associated with the GRUB package. As suggested in Chapter 3, that documentation is available in the /usr/share/doc directory. As the installed version of the grub package is 0.97, the directory with GRUB documentation is /usr/share/doc/grub-0.97.

In that directory, you'll find a menu.lst file. That file is used as the GRUB bootloader configuration file on some other Linux distributions. In fact, on RHEL 6, in the /boot/grub directory, the menu.lst file is soft-linked to the grub.conf file.

In that file, you can find the following commands, associated with the default RHEL 6 version of the configuration file, with comments:

```
# Boot automatically after 30 secs.
timeout = 30
# By default, boot the first entry
default = 0
```

In addition, there's a stanza associated with Linux:

```
# For booting GNU/Linux
title  GNU/Linux
root (hd1,0)
kernel /vmlinuz root=/dev/hdb1
#initrd /initrd.img
```

Of course, that stanza would need changes. The words after the **title** directive could be anything you want. The **root (hd1,0)** directive would have to be changed to the actual partition with the /boot directory, unless that directory is already configured on the first partition of the second hard drive. The *vmlinuz* would have to be changed to the full name of the file associated with the desired Linux kernel. You need to add a **ro** to make sure the kernel is initially loaded in read-only mode for the initial RAM disk. The */dev/hdb1* in that line would have to be changed to the device of the partition or volume where the top-level root directory is located.

There was a time when no Initial RAM disk file was required during the boot process. That is no longer true. Thus, the **initrd** directive line would have to be uncommented (by removing the # in front of the line), with the *initrd.img* filename changed to the name of the actual Initial RAM disk file in the /boot directory.

Of course, if the GRUB configuration file is missing and you weren't able to boot the system to even read this menu.lst file, you might need to resort to an option known as rescue mode.

An Option to Booting from GRUB: Rescue Mode

The troubleshooting objectives associated with the 2007 version of the RHCE exam prep guide suggest that you needed to be able to recover from a complete boot failure, such as if the GRUB configuration file were corrupt or missing. To that end, you might need to rebuild that GRUB configuration file from scratch. In other words, if you've tried to boot directly from the **grub>** prompt described earlier and failed, you might need to resort to the option known as rescue mode. That required access to the installation DVD, or the network boot disk.

exam Watch

The RHCSA and RHCE objectives no longer include a requirement associated with rescue mode. However, as the rescue of unbootable systems is part of the curriculum for Red Hat's RH254 course, it may be included in future versions of one of these exams.

To that end, boot from one of those media options. When you see the installation screen with the following options:

```
Install or upgrade an existing system
Install system with basic video driver
Rescue installed system
Boot from local drive
Memory test
```

Select the Rescue Installed System option and press ENTER. Rescue mode installs a stable minimal version of the RHEL 6 operating system on the local machine. It's in essence a console version of the "Live DVD" media available on other Linux distributions such as Knoppix, Ubuntu, and yes, even the Scientific Linux rebuild distribution. But none of those distributions will be available during a Red Hat exam. If you don't have access to a CD or DVD during an exam, that should mean that there's at least one method other than rescue mode for addressing a problem. However, practice from rescue mode can help you learn an intricate component such as GRUB.

on the job

For RHEL 6, it's best to use RHEL 6 rescue media. Such media uses a kernel compiled by Red Hat, customized for supported software. Nevertheless, options such as Knoppix are excellent.

To that end, even unbootable systems can be started in the rescue environment. It is functionally nearly the same prompt as was available from the installation DVD in previous versions of RHEL. To get that boot prompt from the boot screen with the options just listed, press the ESC key at that screen.

The first couple of steps should be easily recognizable for anyone who has installed RHEL 6. Specifically, you're prompted to enter a language and keyboard type standard for the installation process. New for RHEL 6 is the step shown in Figure 5-3. It illustrates where the rescue image can be located. If the Red Hat installation DVD or network boot disk is installed, you can select Local CD/DVD. Alternatively, you can select a local or network location for an installation server as described in Chapter 1.

If you select a network location, the next step prompts for network configuration, just as was done during the network installation process described in Chapter 1. If you select a local location, the next step asks if you want to set up networking on the local system. Unless access to a remote installation server such as that set up in Chapter 1 is needed, that's generally not required.

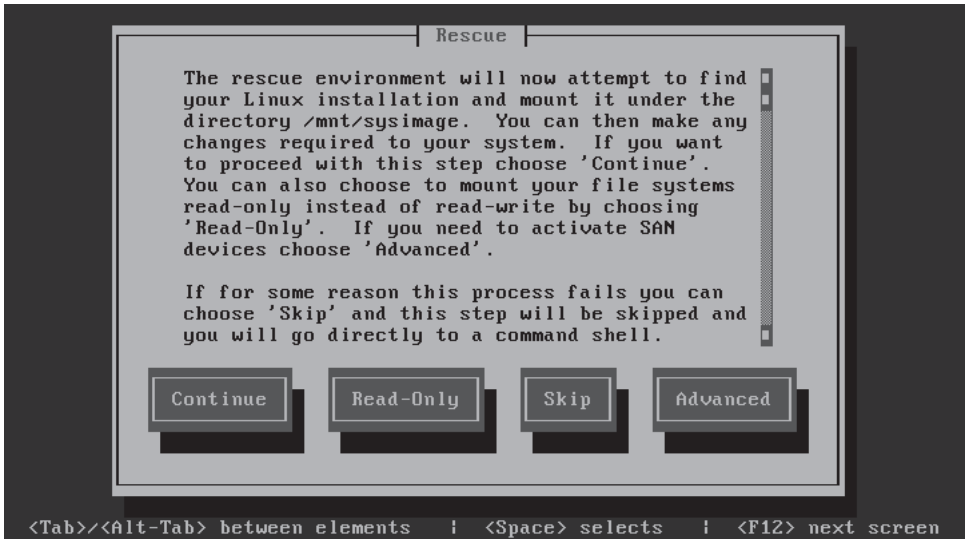
FIGURE 5-3

Options for the rescue image

If the rescue environment detects more than one local Linux system, you may be prompted for the volume device of the desired top-level root directory. Some trial and error may be required at that point. As it's highly unlikely that there will be more than one RHEL system on the same drive for an exam, no further explanation will be provided. In most cases, the next step will be the choice shown in Figure 5-4.

FIGURE 5-4

Options for the rescue environment



Generally, you'll choose Continue at this point. If you need to connect a network storage device, choose Advanced. Related devices such as iSCSI are described in Chapter 11. The Read-Only option mounts detected volumes in read-only mode. The Skip option moves straight to a command line interface.

The Continue option, as suggested in Figure 5-5, mounts all detected volumes as subdirectories of the /mnt/sysimage directory. After confirmation, you're given choices shown in Figure 5-6. The First Aid Kit quickstart menu includes diagnostics that may help recover some configuration files. But it is not included in the objectives for either exam. For now, just select Start Shell and press ENTER.

From the shell prompt, enter **chroot /mnt/sysimage** command. Since the regular top-level root directory for the system is mounted on the /mnt/sysimage directory, the **chroot** command mounts that filesystem as if the system were booted from a properly configured GRUB bootloader.

FIGURE 5-5

Rescue environment options for mounting

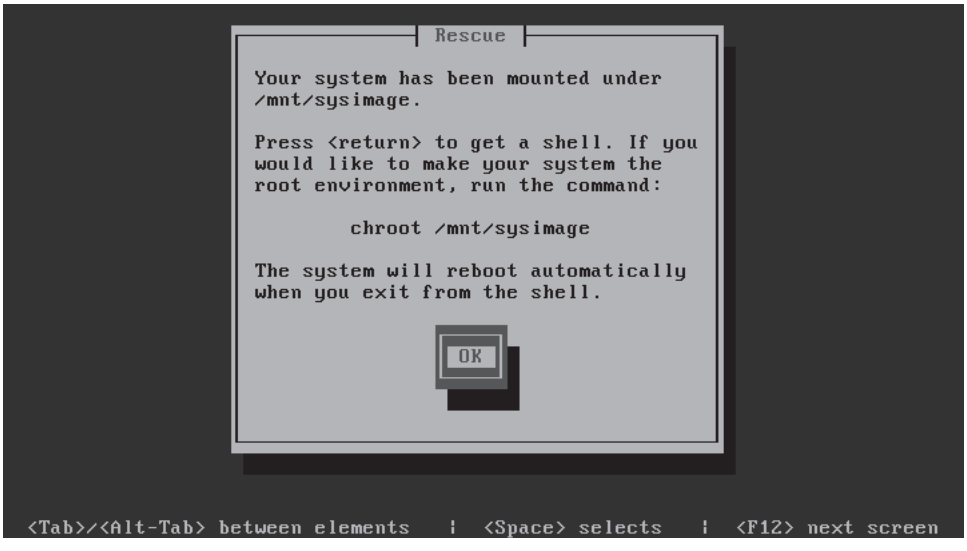


FIGURE 5-6

The First Aid Kit

CERTIFICATION OBJECTIVE 5.03

Between GRUB and Login

This section provides a basic overview of the boot process that occurs after the GRUB bootloader finds the kernel. Understanding what happens here can help you diagnose a wide variety of boot problems. The messages associated with the kernel provide a step-by-step view of the process.

The loading of Linux depends on a temporary filesystem, known as the initial RAM disk. Once the boot process is complete, control is given to init, known as the first process. Most Linux distributions, including RHEL 6, are converting from the old Unix init system, known as SysVinit to a system known as Upstart. This section will describe the contents of Upstart files in detail, through the configuration of terminals and login screens.

Also in the realm of the boot process are the commands that allow you to reboot and shut down a system normally.

Kernels and the Initial RAM Disk

Just a few messages after you select a kernel from the GRUB configuration menu, Linux hands over boot responsibilities to the kernel, with the help of a construct known as the initial RAM disk. It's actually a file system, as suggested by its filename in the /boot directory, initramfs.

That temporary filesystem is uncompressed within RAM. That filesystem is used to load information from key files such as the actual filesystems to be mounted from the /etc/fstab configuration file. It also loads init, as the first process.

To learn more, disable the **quiet** directive for the desired kernel in the GRUB configuration file. (Don't forget to restore it later.) Then you can watch as the messages pass quickly through the screen. Alternatively, you can reviews these messages in the /var/log/dmesg file or by running the **dmesg** command.

What you see depends on the hardware and configuration of the local system. Key messages include:

- The version of the kernel
- SELinux status, if active. By default, SELinux first starts in permissive mode, until the configured policy (enforcing) is loaded near the end of the boot process.
- Amount of recognized RAM (which does not necessarily match the actual amount of installed RAM)
- CPUs
- Kernel command line, specifying the logical volume or root filesystem label
- Freeing of memory associated with the initial RAM disk (initramfs)
- Hard drives and partitions (as defined by their device filenames, such as /dev /sda or /dev/vda1)
- Active filesystems
- Swap partitions

This file is filled with potential clues. If the system is loading the wrong kernel, you'll see evidence of that here. If Linux isn't using a partition that you've configured, you'll also see it here (indirectly). If SELinux isn't loading properly, you'll see it in messages toward the end of the file.

exam Watch

Remember that the Red Hat exams are not hardware exams. If you identify a problem with a key hardware component, such as a network card (which cannot be solved by some Linux command), inform your instructor/exam proctor. Don't be surprised, however, if she responds that it's not a hardware problem.

The First Process, Runlevels, and Services

The Linux kernel continues the boot process by calling the first process, **init**. The **init** process in turn runs /etc/rc.d/rc.sysinit, which performs a number of tasks, including network configuration, SELinux status, keyboard maps, system clock, partition mounts, and host names. It also loads the modules described in the previous section. It does even more: the default version of this file contains nearly 700 lines.

Through RHEL 5, **init** then used parameters configured in the /etc/inittab file. But that has changed. The /etc/inittab file now normally contains one line, which specifies the default runlevel.

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

As suggested by the comments at the start of /etc/inittab, there are seven available runlevels, 0 through 6. (There are actually a few other runlevels that may be used.) Linux services are organized by runlevel. Each runlevel is associated with a level of functionality. For example, in runlevel 1, only one user is allowed to connect to that Linux system. X11 mode, also known as runlevel 5, starts Linux into a GUI login screen, if appropriate packages are installed. As other Linux distributions define runlevels differently, the Red Hat definitions are shown in Table 5-2.

Runlevels are controlled by scripts, organized in runlevel-based directories. While the default runlevel is defined in /etc/inittab, you can override the default during the boot process from the GRUB menu.

One boot option described earlier in this chapter included the **single** command added to the end of the **kernel** directive during the boot process. It's different from the single-user mode associated with runlevel 1. As you'll see later in this chapter, there are scripts executed at each numbered runlevel.

TABLE 5-2 Red Hat Runlevels

Runlevel	Description
0	Halt
1	Single-user mode, for maintenance and repair
2	Multiuser, with some network services
3	Multiuser, with networking
4	Normally unused (but available)
5	X11, defaults to a GUI login screen; logins bring the user to a GUI desktop, with networking
6	Reboot (never set **initdefault** in /etc/inittab to this value)

In fact, each runlevel may be associated with a substantial number of scripts. Each script can start or stop Linux services such as printing (**cupsd**), scheduling (**crond**), the Apache web server (**httpd**), the Samba file server (**smbd**), and more. The starting and stopping of the right scripts becomes part of the boot process.

The default runlevel is specified in /etc/inittab by the **initdefault** directive. If that directive is set to 0, the system will shut down when Linux tries to boot. Likewise, if **initdefault** is set to **6**, Linux will enter a continuous reboot cycle.

Switch Between Runlevels

Now that you've examined the different runlevels available on RHEL 6, it's time to explore how to switch between runlevels. First, establish the current runlevel with the following command:

```
# runlevel
```

After a system is booted, the output that appears is normally something like:

```
N 5
```

That output refers to no previous runlevel (N), followed by the current runlevel (5). Now you can move to a different runlevel with the **init** or **telinit** command. While they're different commands, they're the same for our purposes. For example, the following command moves the system to runlevel 3:

```
# init 3
```

After that command is complete, rerun the runlevel command. The output confirms the result:

```
5 3
```

Now try something else. Since runlevel 0 is known as halt, what do you think happens when the following command is executed?

```
# telnint 0
```

Reboot and Shut Down a System Normally

RHEL 6 makes it easy to boot a system. Just power it up, and you're on your way. In contrast, you actually need to run a command to reboot or shut down a system. Both

are straightforward. As just suggested in the previous section, the following commands are one way to shut down and reboot a system, respectively:

```
# init 0
# init 6
```

As with many things in Linux, there is more than one way to shut down and reboot a system. In fact, the commands associated with those very actions serve the purpose:

```
# shutdown
# reboot
```

Upstart Replaces SysVInit

As Upstart is designed as a "drop-in" replacement for the older SysVInit system, in one way, not much has changed. But the way these first files are organized has changed greatly. As you might expect, the Upstart system is associated with the upstart package. To see the files installed with that package, run the following command:

```
# rpm -ql upstart
```

Review this list of files. It starts with a connection to the ConsoleKit, which (among other things) tracks user interaction with administrative tools. It continues with a message bus connection to the files in the /etc/init directory, as defined by the init-system-dbus.conf file. It then includes those runlevel control commands that you should be familiar with, as shown in Table 5-3.

TABLE 5-3 Runlevel Control Commands

Command	Description
halt	Moves to runlevel 0, which shuts down the system
init	Manages the current runlevel (different from the init daemon)
initctl	Controls the init daemon
poweroff	Moves to runlevel 0, which shuts down the system
runlevel	Lists current and previous runlevel
telinit	Manages the current runlevel

Those of you familiar with the latest Ubuntu releases should recognize the Upstart system. Be aware, the RHEL 6 implementation of Upstart is quite different.

Upstart Configuration Files

The way the first process starts others during the boot process depends on various Upstart configuration files. As with many other services, it starts with the init file in the /etc/sysconfig directory. It continues with various files in the /etc/init directory. Many of the filenames in that directory are descriptive, such as control-alt-delete.conf. Once Upstart configuration is complete, the boot process continues with runlevel controls, as discussed later in this chapter.

Basic /etc/sysconfig/init Configuration

The parameters in the /etc/sysconfig/init file specify how the system looks and feels during the boot process. The directives in the default version of the file are described in Table 5-4.

TABLE 5-4

Directives in /etc/sysconfig/init

Directive	Description
BOOTUP	Supports a color or verbose display
RES_COL	Specifies column for status labels, such as [OK]
MOVE_TO_COL	Sets the column to where the cursor moves
SETCOLOR_SUCCESS	Specifies the color for a successful command
SETCOLOR_FAILURE	Specifies the color for a failed command
SETCOLOR_WARNING	Specifies the color for a warning message
SETCOLOR_RESET	Specifies the reset color
LOGLEVEL	Notes the initial log level, modified later by rsyslog daemon
PROMPT	Supports interactive startup
AUTOSWAP	Works with automatic swap device detection; not required
ACTIVE_CONSOLES	Sets standard command line consoles
SINGLE	Defines the login shell for single-user mode

/etc/init/control-alt-delete.conf

The control-alt-delete.conf file is straightforward, with two commands. First the following command listens for the combination of the noted keys:

```
start on control-alt-delete
```

The line that follows starts with an **exec**, which executes the command that follows. The **shutdown -r now** command reboots the local system immediately. While there is no warning, it does send the message noted in quotes:

```
exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
```

If you prefer a delay before a reboot, substitute a number for the **now** switch. The number will be a time delay in the reboot process, in minutes.

/etc/init/init-system-dbus.conf

The comments at the start of the file describe its purpose: to connect the init process to the dbus server. In addition, it serves as the bridge between init and the configuration options in the other configuration files in the /etc/init directory. In other words, the following **kill** command listens for the dbus message bus and connects it to the process with a process identifier (PID) of 1.

```
exec /bin/kill -USR1 1
```

Plymouth files in /etc/init

Plymouth is the drop-in replacement for the Red Hat Graphical Boot (RHGB) system. While the **rhgb** directive is still used in the standard GRUB configuration file, RHGB-related packages are not used in RHEL 6. The **rhgb** directive now starts Plymouth splash screens. Three related files are normally stored in the /etc/init directory, which relate to what happens when the GUI stops:

- **plymouth-shutdown.conf** runs Plymouth splash screens as the system transitions from runlevel 5 to another runlevel.
- **splash-manager.conf** includes an appropriate message in the Plymouth splash screens when the system moves to runlevels 0 (halt) or 6 (reboot).
- **quit-plymouth.conf** quits the Plymouth service at the end of the boot process.

Readahead Files in /etc/init

The readahead system moves information from key files during the boot process into RAM, speeding access when needed. The files that are collected are in part defined in the /etc/sysconfig/readahead file. Three related files are normally stored in the /etc/init directory, which relate to what happens during the boot process:

- **readahead-collector.conf** uses the settings in /etc/sysconfig/readahead.
- **readahead-disable-services.conf** disables those services that can be trouble for readahead services; the only one disabled by default is the **audit** daemon associated with SELinux.
- **readahead.conf** actually starts the readahead process, assuming sufficient RAM is available.

Terminal Files in /etc/init

Linux systems can have serial and graphical-based terminals. They're configured, at least initially, by several files in the /etc/init directory. The prefdm.conf file is configured in runlevel 5 and is set up with the **respawn** command. In other words, if an X server crashes, say with the CTRL-ALT-BACKSPACE key combination, the **respawn** command restarts the GUI-based login screen, as defined in the /etc/X11/prefdm file. Four related files are normally stored in the /etc/init directory, which drive where and how terminals are started during the boot process.

- **prefdm.conf** starts and maintains the noted GUI login screen in runlevel 5.
- **start-ttys.conf** starts all terminals. Note how the X_TTY environment variable starts the GUI in the first virtual terminal. Pay attention to the **ACTIVE_CONSOLES** directive in the /etc/sysconfig/init file. Which one determines what virtual terminals get started?
- **tty.conf** refers to a "getty," which is another name for the standard command line terminal. Note how it is disabled in runlevels 0, 1, and 6, which corresponds to the halt, single-user, and reboot runlevels.
- **serial.conf** is used when there's a remote connection over a serial port.

Resource Control

In Linux, resource control relates to runlevels. Different scripts may be run in different runlevels; for example, if the default runlevel is 3, one of these files makes sure that the scripts in the /etc/rc3.d directory are executed during the boot process.

Four related files are normally stored in the /etc/init directory, based on the default runlevel defined in /etc/inittab:

- **rcS.conf** runs the /etc/rc.d/rc.sysinit script only during the boot process.
- **rcS-sulogin.conf** starts the use of a terminal in single-user mode, and then moves to the standard runlevel.
- **rc.conf** executes the scripts in the runlevel; it can be defined in /etc/inittab or the **init/telinit** commands. The executed scripts are in the /etc/rc*n*.d/ directory, where *n* is the runlevel.

Terminals and Login Screens

The login terminals in Linux are virtual consoles. Most Linux systems, including RHEL 6, are configured with six standard command line virtual consoles. These consoles are numbered from 1 to 6. When configured with a GUI and a login manager, other Linux distributions include a seventh virtual console, with a graphical login screen. That's one place where RHEL 6 is a bit different, as in most cases, it substitutes the graphical login screen for the first virtual console. That applies just for the graphical login screen. If you start the GUI with the **startx** command, the GUI is run in the seventh virtual console.

What does that all mean? In Linux, you can switch between virtual consoles with an ALT-function key combination. For example, ALT-F2 brings you to the second virtual console. You can switch between adjacent virtual consoles by pressing ALT-RIGHT ARROW or ALT-LEFT ARROW. For example, to move from virtual console 2 to virtual console 3, press ALT-RIGHT ARROW. If you're in a GUI virtual console, add the CTRL key. So in RHEL 6, if the GUI is installed and you're in the first virtual console, you'd have to press CTRL-ALT-F2 to get to the second virtual console.

When you log in to a regular virtual console, Linux returns a command line shell. The default shell is defined in the /etc/passwd file described in Chapter 6. When you log in to a GUI virtual console, Linux returns the configured GUI desktop. For more information on the Linux GUI, see Chapter 8.

Through RHEL 5, virtual consoles were configured in /etc/inittab. Now that Upstart has replaced SysVinit, virtual consoles are configured in files in /etc/sysconfig /init and the /etc/init directory, as described earlier in this chapter.

Virtual consoles really bring the multiuser capabilities of Linux to life. At work (or during a Red Hat exam), you might review a man page on one console, compile a program in another, and edit a configuration file in a third virtual console. Other users who are connected through a network can do the same thing at the same time.

CERTIFICATION OBJECTIVE 5.04

Control by Runlevel

Despite the aforementioned move from SysVinit to Upstart, Linux is still focused on the runlevel. The management of services is customized by runlevel. As Upstart includes the same functional commands as SysVinit, you can still control the current runlevel with commands like **init** and **telinit**.

Linux is highly customizable. So it makes sense that the services that start in each runlevel can be customized. Of course, even the scripts themselves can be customized, if you wish to go so far. While GUI tools are available to help customize scripts, it is generally a lot faster to customize them from the command line interface.

Functionality by Runlevel

As described earlier, the basic functionality of each runlevel is listed in the /etc/inittab file, as listed in Table 5-2. But those are just general descriptions. Actual functionality is determined by the scripts run at each runlevel. And that's highly customizable. For example, while runlevel 4 is normally not used in RHEL 6, there's nothing preventing you from setting up a custom group of services in that runlevel. Per the /etc/init/rc .conf file, the following command is executed during the boot process:

```
exec /etc/rc.d/rc $RUNLEVEL
```

Take a look at that file, rc in the /etc/rc.d directory. With the following command, it looks for a directory associated with the runlevel:

```
[ -d /etc/rc$runlevel.d ] || exit 0
```

In other words, if the default runlevel is 3, it looks to see if the /etc/rc3.d directory exists. It should exist on any standard RHEL 6 system. Take a quick look at the files in that directory. One sample is shown in Figure 5-7.

Take a look at the scripts in other runlevels. Run the following commands:

```
# ls /etc/rc0.d
# ls /etc/rc1.d
# ls /etc/rc2.d
# ls /etc/rc3.d
# ls /etc/rc4.d
# ls /etc/rc5.d
# ls /etc/rc6.d
```

FIGURE 5-7

Scripts in runlevel 3

```
[root@server1 init]# \ls /etc/rc3.d/
K01certmonger  K80lldpad          S11auditd          S26haldaemon
K01smartd      K80sssd            S12rsyslog         S26udev-post
K02oddjobd     K84wpa_supplicant  S13cpuspeed        S28autofs
K10psacct      K86cgred           S13irqbalance      S50bluetooth
K10saslauthd   K87multipathd      S13iscsi           S55sshd
K15httpd       K87restorecond     S13rpcbind         S80postfix
K35vncserver   K88nslcd           S15mdmonitor       S82abrtd
K50dnsmasq     K89rdisc           S20kdump           S90crond
K50netconsole  S00microcode_ctl   S22messagebus      S95atd
K60nfs         S01sysstat         S23NetworkManager  S97libvirtd
K69rpcsvcgssd  S02lvm2-monitor    S24avahi-daemon    S97rhnsd
K73ypbind      S05cgconfig        S24nfslock         S98libvirt-guests
K74nscd        S07iscsid          S24rpcgssd         S99firstboot
K74ntpd        S08ip6tables       S24rpcidmapd       S99local
K75ntpdate     S08iptables        S25netfs
K80fcoe        S10network         S26acpid
[root@server1 init]#
```

Do you notice any patterns? What are the filenames of the scripts that are started in runlevels 0 and 6? You should discover script filenames that match the documented functionality of that runlevel, such as S01halt and S01reboot.

Now return to the /etc/rc.d/rc file. Slightly later in the file, you can find a loop that starts with the following command:

```
for i in /etc/rc$runlevel.d/K* ; do
```

That command takes the scripts that start with a *K*, in numeric order. In other words, based on Figure 5-7, the K01certmonger script is run first, and the K89rdisc script is run last. Now run the **ls -l** command on the /etc/rc3.d directory. You'll see links to scripts of the same name (without the *K* and the number) in the /etc/init.d directory.

Now examine the next stanza in the /etc/rc.d/rc file. You'll see a loop that starts with the following command:

```
for i in /etc/rc$runlevel.d/S* ; do
```

Based on the files shown in Figure 5-7, the S01sysstat script is run first, and the S99local script is run last. As suggested by the other commands in the stanza, those scripts are started, using their links to the actual scripts in the /etc/init.d directory.

Now it's time to take a look at the innards of runlevel scripts, as many of them can do more than just start and stop a service.

The Innards of Runlevel Scripts

Runlevel scripts are executed whenever a system moves to a different runlevel. So the scripts associated with the default runlevel are executed during the boot process. Appropriate scripts are also executed when you change runlevels; for example, when you run the **init 3** command, Linux executes the scripts in the /etc/rc3.d/ directory.

But administrators like yourself can control Linux scripts directly. As noted earlier, scripts in directories such as /etc/rc3.d are linked to the actual scripts in the /etc/init.d directory. Take some time to look at these scripts. As an example, scan the /etc/init.d/sshd script. The second active line pulls in configuration options from the /etc/sysconfig/sshd file. Much of the rest of the script describes actions associated with command options, as summarized by the **echo** directive. Exit the script and run the following command:

```
# /etc/init.d/sshd
```

It should return the following output, straight from the **echo** directive:

```
Usage: /etc/init.d/sshd {start|stop|restart|reload|force-reload|condrestart|try-
restart|status}
```

In other words, the following command stops the Secure Shell (SSH) service:

```
# /etc/init.d/sshd stop
```

Alternatively, the service command can be used with the noted options; for example, the following command reloads the SSH configuration file without stopping or starting the service:

```
# service sshd reload
```

The options shown from the SSH service are described in Table 5-5.

Service Configuration from the Command Line

It's generally fastest to control services at the command line. The **chkconfig** command gives you a simple way to maintain different runlevels within the /etc/rc.d directory structure. First, try the **chkconfig --list** command. As shown in Figure 5-8, you'll see the whole list of services in the /etc/init.d directory, along with their normal status in all seven runlevels:

But the **chkconfig** command can do more. With that command, you can add, remove, and change services; list startup information; and check the state of a

TABLE 5-5

Service Control Commands

Command	Description
start	Starts the service if it's currently not running
stop	Stops the service if it is currently running
restart	Stops and then restarts the service
reload	If the service is currently running, it loads the current version of the configuration file(s) with any changes. The service is not stopped, users who have previously connected are not kicked off
force-reload	Restarts a service if it's already running; otherwise, makes sure the new service is started with the latest version of a configuration file
condrestart	Stops and then restarts the service, only if it is already running
try-restart	Same as condrestart
status	Lists the current operational status of the service

particular service. For example, the following command checks the runlevels where the Postfix service is set to start:

```
# chkconfig --list postfix
postfix 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

FIGURE 5-8

Service status by runlevel

```
nslcd           0:off   1:off   2:off   3:off   4:off   5:off   6:off
ntpd            0:off   1:off   2:off   3:off   4:off   5:off   6:off
ntpdate         0:off   1:off   2:off   3:off   4:off   5:off   6:off
oddjobd         0:off   1:off   2:off   3:off   4:off   5:off   6:off
postfix         0:off   1:off   2:on    3:on    4:on    5:on    6:off
psacct          0:off   1:off   2:off   3:off   4:off   5:off   6:off
rdisc           0:off   1:off   2:off   3:off   4:off   5:off   6:off
restorecond     0:off   1:off   2:off   3:off   4:off   5:off   6:off
rhnsd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
rpcbind         0:off   1:off   2:on    3:on    4:on    5:on    6:off
rpcgssd         0:off   1:off   2:off   3:on    4:on    5:on    6:off
rpcidmapd       0:off   1:off   2:off   3:on    4:on    5:on    6:off
rpcsvcgssd      0:off   1:off   2:off   3:off   4:off   5:off   6:off
rsyslog         0:off   1:off   2:on    3:on    4:on    5:on    6:off
saslauthd       0:off   1:off   2:off   3:off   4:off   5:off   6:off
smartd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
sssd            0:off   1:off   2:off   3:off   4:off   5:off   6:off
sysstat         0:off   1:on    2:on    3:on    4:on    5:on    6:off
udev-post       0:off   1:on    2:on    3:on    4:on    5:on    6:off
vncserver       0:off   1:off   2:off   3:off   4:off   5:off   6:off
wpa_supplicant  0:off   1:off   2:off   3:off   4:off   5:off   6:off
ypbind          0:off   1:off   2:off   3:off   4:off   5:off   6:off
[michael@server1 ~]$
```

This indicates that the Postfix e-mail server is configured to start in runlevels 2, 3, 4, and 5. If you want to make sure the Postfix service does not start in runlevel 4, execute the following command:

```
# chkconfig --level 4 postfix off
```

Run the **chkconfig --list postfix** command again to confirm the change. Now Postfix is configured to run only on runlevels 2, 3, and 5. To turn it back on for runlevel 4, run the same command, substituting **on** for **off**. With **chkconfig**, you can also add or delete services with the **--add** and **--del** switches. Installing a service sets up the appropriate links within the /etc/rc.d directory hierarchy. Uninstalling that service removes the associated links from the same hierarchy.

The commands need not even be that complex. If you leave out the runlevel, the following commands automatically deactivate the Postfix service in all runlevels and then activate it in runlevels 2, 3, 4, and 5:

```
# chkconfig postfix off
# chkconfig postfix on
```

The Text Console Service Configuration Tool

If you're managing a substantial number of services, the command line might not be quite so efficient. You don't need a GUI, just the **ntsysv** tool, which can be started with the command of the same name. However, it's a bit tricky; by default, it only changes the status of selected services in the current runlevel.

But you can do more. For example, if to activate several services in runlevels 2, 3, 4, and 5, start **ntsysv** with the following command (don't forget the double-dash):

```
# ntsysv --level 2345
```

```
For this section, I've started ntsysv with the noted command and
disabled the bluetooth service, as shown in Figure 5-9.
```

Once changes are complete, you can check the result with the following command:

```
# chkconfig --list bluetooth
bluetooth 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

You should also be able to confirm the changes in appropriate runlevel directories. For example, before the bluetooth service was deactivated, the following file was available for runlevel 3:

```
S50bluetooth
```

FIGURE 5-9

Service management with ntsysv

After the noted changes were made, the S50bluetooth file was replaced with the following:

```
K83bluetooth
```

The GUI Service Configuration Tool

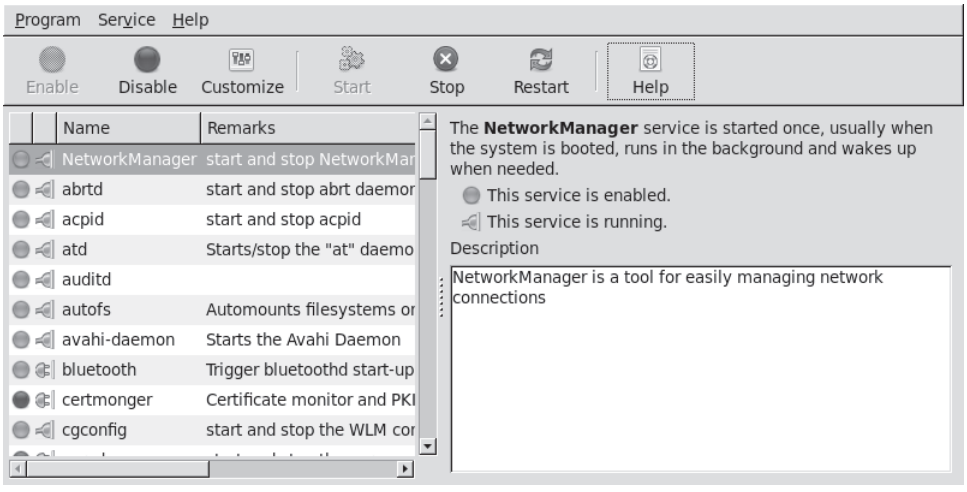
The Service Configuration tool shown in Figure 5-10 allows you to select the services that are to be activated in runlevels 2 through 5. One way to start it is from a command line in the GUI with the **system-config-services** command. The options are fairly self-explanatory. Click Customize for a list of runlevels affected by any changes made through this tool<=>.

exam Watch

When you configure or repair a service, use* chkconfig *(or a related utility such as* ntsysv *or* system-config-services*) to make sure that the service is activated at the appropriate runlevels. Otherwise, you may not get full credit for your work.

FIGURE 5-10

The Service Configuration tool



CERTIFICATION OBJECTIVE 5.05

Network Configuration

Network configuration is a key element in the administration of RHEL 6, and the Red Hat exams. Red Hat sets up a number of key configuration files in the /etc/sysconfig directory. Other configuration files such as /etc/resolv.conf and even /etc/hosts are also important. You can configure them with a text editor or different configuration tools.

But before making any permanent changes to configuration files, such changes should be tested with configuration commands. And Linux includes a number of commands that can help configure temporary changes to the network configuration of a system.

It's important to learn the configuration files in the /etc/sysconfig/network-scripts, /etc/sysconfig, and /etc directories. These are crucial to the network configuration of RHEL 6. If you have a network problem, a permanent solution probably involves changing files in one of these directories.

Network Configuration Commands

From the command line, you can reconfigure the network in a number of ways. The **ifup** and **ifdown** commands can activate and deactivate network devices. The **ifconfig** command supports detailed changes to individual network devices. The **route** command supports review of and changes to routing tables.

ifup/ifdown

You can activate or deactivate the adapter of your choice with the **ifup** and **ifdown** commands. For example, either of the following commands will activate the eth0 network adapter. The eth0 device is the standard for the first Ethernet adapter on a system.

```
ifup ifcfg-eth0
ifup eth0
```

As might be expected, you can reverse the process by substituting the **ifdown** command. Each installed network adapter has a corresponding ifcfg-* file in the /etc/sysconfig/network-scripts directory. When the **ifup** command is used, it starts the given network adapter based on its configuration file in that directory.

ifconfig

The **ifconfig** command is powerful in the area of network configuration. By itself, it lists all currently active network adapters. It can also be used to configure and display network devices. One sample output is shown in Figure 5-11.

The output shown reflects a typical configuration on most systems, with a loopback and a real network adapter, as labeled by their device filenames, lo and eth0 respectively. The loopback adapter supports network tests when a real network connection is not available.

Some systems have a number of network adapters. For example, my home server has two Ethernet cards, a loopback adapter, a virtual bridge to KVM virtual machines, and two virtual bridges to VMware virtual machines. That's six network adapters. But that may not be everything; the **ifconfig** command lists only active adapters by default. To see the full list, run the **ifconfig -a** command.

It may help to isolate a specific adapter; for example, the following command isolates settings associated with the first Ethernet adapter:

```
# ifconfig eth0
```

FIGURE 5-11

The ifconfig command

```
[root@server1 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:55:D0:A2
          inet addr:192.168.122.50  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe55:d0a2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16895 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15364 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5725595 (5.4 MiB)  TX bytes:6162800 (5.8 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:15889 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15889 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9089760 (8.6 MiB)  TX bytes:9089760 (8.6 MiB)

[root@server1 ~]#
```

The **ifconfig** command can also be used to configure network interfaces. For example, you can assign a new IP address for eth0 with the following command:

```
# ifconfig eth0 192.168.122.250
```

The first parameter, **eth0**, tells you which interface is being configured. The next argument, **192.168.122.250**, indicates the new IP address being assigned to this interface. To make sure the change worked, issue the **ifconfig** command again (with the name of the adapter device) to view its current settings.

With the right switch, the **ifconfig** command can modify a number of other settings for your network adapter. Some of these switches are shown in Table 5-6.

Of course, you'll want to make sure the changes survive a reboot, whether it be for the exam, or for a server that you want to administer remotely. That depends on appropriate changes to configuration files in the /etc/sysconfig/network-scripts directory, described shortly. And any changes made with the ifconfig command are, by definition, temporary.

TABLE 5-6 Command Switches for ifconfig

Switch	Description
up	Activates the specified adapter.
down	Deactivates the specified adapter.
netmask ***address***	Assigns the *address* subnet mask.
broadcast ***address***	Assigns the *address* as the broadcast address. Rarely required, since the default broadcast address is standard for most current networks.
metric **N**	Allows you to set a metric value of *N* for the routing table associated with the network adapter.
mtu **N**	Sets the maximum transmission unit as *N*, in bytes.
-arp	Deactivates the Address Resolution Protocol, which collects network adapter hardware addresses.
promisc	Activates promiscuous mode. This allows the network adapter to read all packets to all hosts on the LAN. Can be used to analyze the network for problems or to try to decipher messages between other users.
-promisc	Deactivates promiscuous mode.

Routing Tables with route or netstat -r

The **netstat** command is used to display a plethora of network connectivity information. The most commonly used option, **netstat -r**, is used to display local routing tables. Here's a sample **netstat -nr** output:

```
# netstat -nr
Kernel routing table
Destination    Gateway        Genmask          Flags MSS Window  irtt Iface
192.168.122.0  0.0.0.0        255.255.255.0    U     40  0          0 eth0
0.0.0.0        192.168.122.1  0.0.0.0          UG    40  0          0 eth0
```

Did you notice the use of the **-n** flag? **-n** tells **netstat** to display addresses as IP addresses, instead of as host names. This makes it a little easier to see what's going on. One equivalent option is the **route -n** command.

The Destination column lists networks by their IP addresses. The *default* destination is associated with all other IP addresses. The Gateway column indicates gateway addresses. If the destination is on the LAN, no gateway is required, so 0.0.0.0 is shown in this column. The Genmask column lists the network mask. Networks look for a route appropriate to the destination IP address. The IP address is compared against the destination networks, in order. When the IP address is found to be part of one of these networks, it's sent in that direction. If there is a gateway address, it's sent to the computer with that gateway. If you know IPv4 addresses, you should remember that 0.0.0.0 is the default IP address. In other words, all data that isn't going to the 192.168.122.0 network is sent through that network's gateway IPv4 address, 192.168.122.1.

The **netstat** and **route** commands do have different purposes. For example, the **route** command can also add routes. For example, for systems with only one network adapter, the following command adds a default route over the aforementioned network gateway IP address. If desired, you can substitute **0.0.0.0** for **default**.

```
# route add default gw 192.168.122.1
```

Of course, you may want to set up a route to a specific network through a specific device. The following command sets up a route to the noted network/subnet mask combination through the second Ethernet device:

```
# route add -net 192.168.100.0 netmask 255.255.255.0 dev eth1
```

The **netstat** command can do more. With the right combination of switches, it can help identify open services on the local system. One command I like to use to assess potential vulnerabilities of a Linux system is:

```
# netstat -atunp
```

Where the **netstat** command listens for all (**-a**) network connections, using both the TCP (**-t**) and UDP (**-u**) protocols, in numeric (**-n**) format, with the PID and program associated with each line. Figure 5-12 illustrates the output on the baseline server.

In the middle of the output, note the Foreign Address of 192.168.122.1:48092. The 48092 port number is just the return communications port. The corresponding Local Address of 192.168.122.50:22 specifies a port number 22 for a connection from the system at 192.168.122.1. Near the top of the file, there's a second entry with the same PID that identifies the associated SSH daemon (sshd) connection. Other lines in this output identify other open services, which may or may not be blocked by firewalls and more, as discussed in Chapters 4 and 10.

FIGURE 5-12 Output from the netstat -atunp command

```
[root@server1 ~]# netstat -atunp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name
tcp        0      0 0.0.0.0:44742               0.0.0.0:*                   LISTEN      1320/rpc.statd
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN      1212/rpcbind
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN      15993/sshd
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN      1574/master
tcp        0      0 127.0.0.1:6010              0.0.0.0:*                   LISTEN      16304/0
tcp        0      0 192.168.122.50:22           192.168.122.1:48092         ESTABLISHED 16304/0
tcp        0      0 :::44907                    :::*                        LISTEN      1320/rpc.statd
tcp        0      0 :::111                      :::*                        LISTEN      1212/rpcbind
tcp        0      0 :::22                       :::*                        LISTEN      15993/sshd
tcp        0      0 ::1:6010                    :::*                        LISTEN      16304/0
udp        0      0 0.0.0.0:648                 0.0.0.0:*                               1320/rpc.statd
udp        0      0 0.0.0.0:42020               0.0.0.0:*                               1320/rpc.statd
udp        0      0 0.0.0.0:963                 0.0.0.0:*                               1212/rpcbind
udp        0      0 0.0.0.0:5353                0.0.0.0:*                               1291/avahi-daemon:
udp        0      0 0.0.0.0:111                 0.0.0.0:*                               1212/rpcbind
udp        0      0 0.0.0.0:33915               0.0.0.0:*                               1291/avahi-daemon:
udp        0      0 :::963                      :::*                                    1212/rpcbind
udp        0      0 :::60636                    :::*                                    1320/rpc.statd
udp        0      0 :::111                      :::*                                    1212/rpcbind
[root@server1 ~]#
```

arpas a Diagnostic Tool

The Address Resolution Protocol associates the hardware address of a network adapter with an IP address. The **arp** command displays a table of hardware and IP addresses on the local computer. With **arp**, you can detect problems such as duplicate addresses on the network, or you can manually add **arp** entries as required. Here's a sample **arp** command, showing all **arp** entries in the local database:

```
# arp
Address              HWtype  HWaddress           Flags Mask            Iface
192.168.0.121        ether   52:A5:CB:54:52:A2   C                     eth0
192.168.0.113        ether   00:A0:C5:E2:49:02   C                     eth0
```

If the **arp** table is empty, there are no recent connections to other computers on the local network. The Address column lists known IP addresses, usually on the LAN. The HWtype column shows the hardware type of the adapter, while the HWaddress column shows the hardware address of the adapter.

The **arp** command can help you with duplicate IP addresses, which can stop a network completely. To remove the offending machine's **arp** entry from your **arp** table, use the **-d** option:

```
# arp -d bugsy
```

This removes all **arp** information for the host bugsy. To add an **arp** entry, use the **-s** option:

```
# arp -s bugsy 00:00:c0:cf:a1:33
```

This entry will add the host bugsy with the given hardware address to the **arp** table. Hardware addresses are required; IP addresses won't work in this case.

Dynamic Host Configuration Protocol (DHCP) Clients

If the local system is set up as a Dynamic Host Configuration Protocol (DHCP) client, it should be noted as such in the associated configuration file for the network card, described shortly. Even if a system is configured with static IP address information, it can be reconfigured with the appropriate DHCP client, the **dhclient** command.

Of course, if other systems depend on a definitive static IP address, the use of the **dhclient** command can cause trouble. For example, other systems may not be able to find key servers on that system.

Network Configuration Files

The network configuration file that provides the foundation for RHEL 6 networking is /etc/sysconfig/network. It can contain up to six directives, as described in Table 5-7. If a directive from that table does not appear in the /etc/sysconfig/network file, the directive does not apply. For example, if you don't see the **GATEWAYDEV** directive, there's probably only one network card on the local system.

TABLE 5-7

/etc/sysconfig /network Directives

Directive	Description
NETWORKING	Can be yes or no, to configure or not configure networking.
NETWORKING_IPV6	Can be yes or no, to configure networking under IPv6.
NISDOMAIN	Should be set to the name of the NIS domain, if the local system is connected to an NIS network.
HOSTNAME	Sets the host name of the local computer. If you don't see this directive, it may be set by a DHCP server.
GATEWAY	Sets the IP address for the gateway for your network. If you don't see this directive, it may be set by a DHCP server.
GATEWAYDEV	Sets the network device, such as eth0, that this computer uses to reach a gateway. There's no need for this directive if there's only one network card.

In most cases, /etc/sysconfig/network contains at least two directives:

```
NETWORKING=yes
HOSTNAME=yourhostname
```

There is no requirement for the other directives shown in Table 5-7, including an NIS domain, for IPv6 networking, and a gateway address. Some of these directives, including the **HOSTNAME**, may be set by the DHCP server.

The /etc/sysconfig/network-scripts Files

The /etc/sysconfig/network-scripts directory is where Red Hat Enterprise Linux stores and retrieves networking information. With available Red Hat configuration tools, you don't have to touch these files, but it's good to know they're there. A few representative files are shown in Table 5-8.

Some of the commands in /etc/sysconfig/network-scripts may be hard-linked to files in the /etc/sysconfig/networking/devices and /etc/sysconfig/networking/profiles/default directories.

TABLE 5-8

/etc/sysconfig /network-scripts Files

File in /etc/sysconfig/network-scripts	Description
ifcfg-lo	Configures the loopback device, a virtual device that confirms proper installation of TCP/IP.
ifcfg-*	Each installed network adapter, such as eth0, gets its own ifcfg-* script. For example, eth0 is given file ifcfg-eth0. If the NetworkManager is used, it may be ifcfg-System_eth0. This file includes the IP address information required to identify this network adapter on a network.
network-functions	This script contains functions used by other network scripts to bring network interfaces up and down.
ifup-* and ifdown-*	These scripts activate and deactivate their assigned protocols. For example, ifup-ppp brings up a PPP device, usually a telephone modem.

Now take a look at an ifcfg-eth0 file. Based on the static network configuration defined in Chapter 2, the associated directives are:

```
DEVICE="eth0"
BOOTPROTO="static"
BROADCAST="192.168.122.255"
DNS1="192.168.122.1"
GATEWAY="192.168.122.1"
HWADDR="52:54:00:55:D0:A2"
IPADDR="192.168.122.50"
NETMASK="255.255.255.0"
NM_CONTROLLED="yes"
ONBOOT="yes"
```

Most of these directives are straightforward, as they define the device as the first Ethernet network card (eth0), with static networking, using a defined IP address for broadcast, a DNS server, the network gateway, the network card, and the network mask. Note that the hardware address of the network card shown here matches the hardware address defined in Figure 5-11. Yes, this eth0 network card is enabled during the Linux boot process.

One directive shown is extra cryptic: **NM_CONTROLLED**. That directive specifies whether the card can be configured and controlled with the Red Hat Network Manager. It's a terrific convenience for desktop users with multiple network connections, but it's less important for servers with single stable physical network connections.

Red Hat Configuration Tools

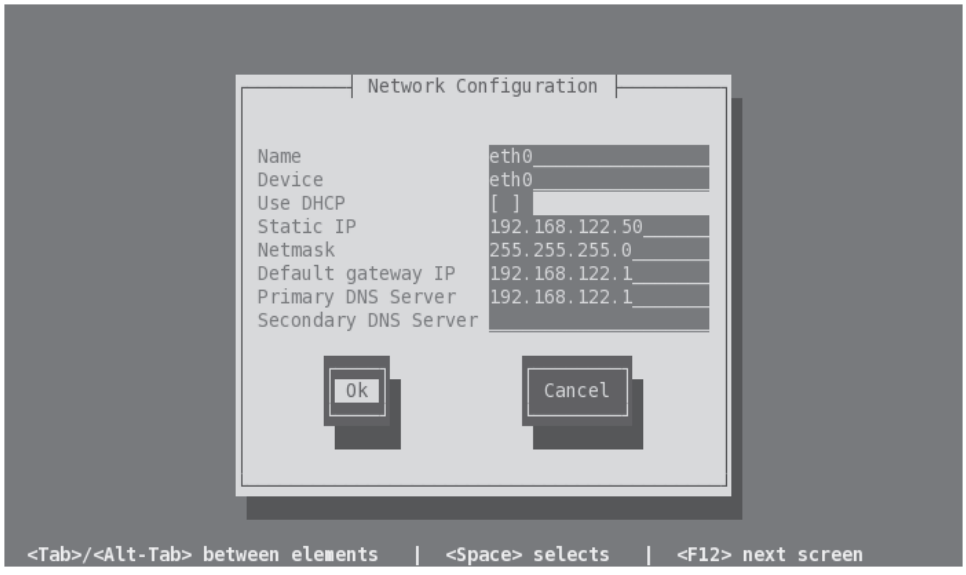
As with many other Red Hat configuration tools, there's a console and a graphical tool available. The console tool, naturally, can be run from text consoles. To open it, run the **system-config-network-tui** command. It's the same tool accessible from the **setup** command at the console; in either case, it starts an easily configurable Network Configuration tool shown in Figure 5-13.

With RHEL 6, the NetworkManager service replaces the Network Administration Tool. Not only does that mean the GUI configuration tool has changed, it also means the GUI tool is not installed by default. To use the GUI NetworkManager configuration tool, you may need to first install the associated package with the following command:

```
# yum install NetworkManager-gnome
```

FIGURE 5-13

The Console Network Configuration tool

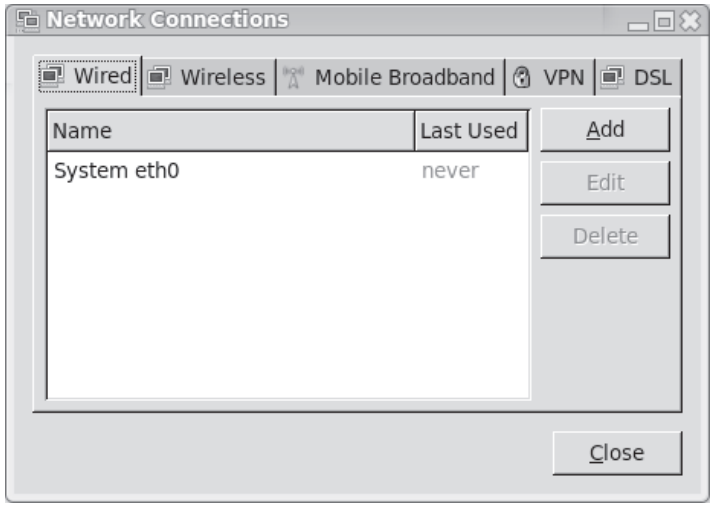


You can then start the GUI Network Connections tool shown in Figure 5-14 with the following command:

```
# nm-connection-editor
```

FIGURE 5-14

Network Connections tool.

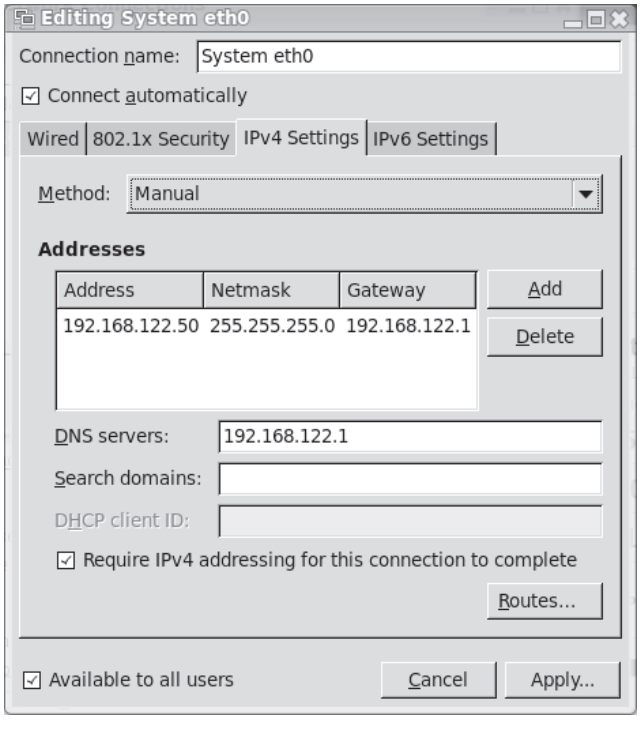


EXERCISE 5-3

Modify Network Interfaces with the Network Connections Tool

This exercise depends on the active use of the NetworkManager service, and the accompanying GUI configuration tool. You'll see what's added to the key configuration file, courtesy of the noted tool.

1. Back up the configuration for the network card to be changed. If it's the first Ethernet card, back up the ifcfg-eth0 file from the /etc/sysconfig/network-scripts directory. Make sure a copy of the file exists in both directories.
2. Start the Network Connections tool. From a GUI terminal, press ALT-F2, and enter **nm-connection-editor** in the text box that appears. This opens the Network Connections tool.
3. Select the Wired tab if it isn't already active.
4. Select the adapter that you want to modify, and then click Edit. If it is an Ethernet adapter, you'll see an Editing System window similar to the one shown in the next illustration.



5. Select the IPv4 Settings tab. Highlight the current IP address, and then click Delete to remove the current IP address information. Click Add. Set an IP address to **192.168.1.11** and the network mask (Subnet Mask) to **255.255.255.0**. (If needed, substitute a different IP address to isolate this system from the local network.)
6. Click Apply. If prompted, enter the root administrative password. Back in the Network Connections tool window, click Close.
7. At the command prompt, run **ifconfig** from a command line interface to check the IP address settings of the target network card. Did anything happen?
8. Review the contents of the revised network configuration file. If it's the first Ethernet card, the filename includes the terms ifcfg and eth0 in the /etc/sysconfig/network-scripts directory. Are the changes reflected there? If the filename had changed, then it's likely that you forgot to save a copy of the ifcfg-eth0 file in the noted directory.
9. Run the **ifdown eth0** and **ifup eth0** commands to apply the changes from the new version of the ifcfg-eth0 file.
10. Rerun the **ifconfig** command to make sure it worked.
11. Compare the original and new versions of the configuration file. You could open them up in matching terminals, or apply the **diff** command.
12. What are the differences? Besides the different IP address information, what else was added by the Network Connections tool? The additional directives should be straightforward. Note how the Network Connections tool added a UUID to uniquely identify the configuration.
13. Restore the original version of the configuration file for the associated network card; if it's the first Ethernet card, it's ifcfg-eth0. If you forgot to save the ifcfg-eth0 file in the noted directory, it might now be named ifcfg-System_eth0.

exam Watch

If you want to use the Network Connections tool during the exam, make sure you know the tool inside-out. Make sure you know how to install it. Experiment with the available options. Make some changes, and check the effect on the files I've described.

Configure Name Resolution

The final piece in network configuration is typically name resolution. In other words, does the local system have the information required to translate domain names such as mcgraw-hill.com to IP addresses such as 198.45.24.143?

Name resolution was easy when Unix was first being developed. When the predecessor to the Internet was first put into use, the worldwide computer network had four hosts, one computer at each of four different universities. It was easy to set up a static file with a list of each of their names and corresponding addresses. That file has evolved into what is known in Linux as /etc/hosts.

But now the Internet is more complex. While you could try to set up a database of every domain name and IP address on the Internet in the /etc/hosts file, that would take almost forever. That's why most users set up connections to DNS, Domain Name Service, servers. On RHEL 6, that's still documented in the /etc/resolv.conf configuration file. On the baseline system created in Chapter 2, that file contains two lines:

```
search example.com
nameserver 192.168.122.1
```

The first line appends the example.com domain to given hostnames; for example, if you were to run the **ping** command on the server1 system, it would actually search for the server1.example.com system. The second line specifies the IP address associated with the local DNS server. As an RHCE, you need to know how to configure a caching-only DNS server; that subject is covered in Chapter 17. DNS configuration is not an RHCSA requirement.

On smaller networks, some administrators set up an /etc/hosts file as a database for the name of each system and IP address on the local network. If desired, administrators could even set up a few IP addresses of domains on the Internet. For the systems described in earlier chapters, an appropriate /etc/hosts file might contain the following directives:

```
192.168.122.50 server1.example.com
192.168.122.150 tester1.example.com
192.168.100.100 outsider1.example.org
```

But if you've configured a connection to a DNS server and systems in /etc/hosts, what's searched first? The search order is specified by one line in the /etc/nsswitch.conf configuration file, which searches for host names first in local files (/etc/hosts) followed by any accessible DNS servers.

```
hosts:  files dns
```

EXERCISE 5-4

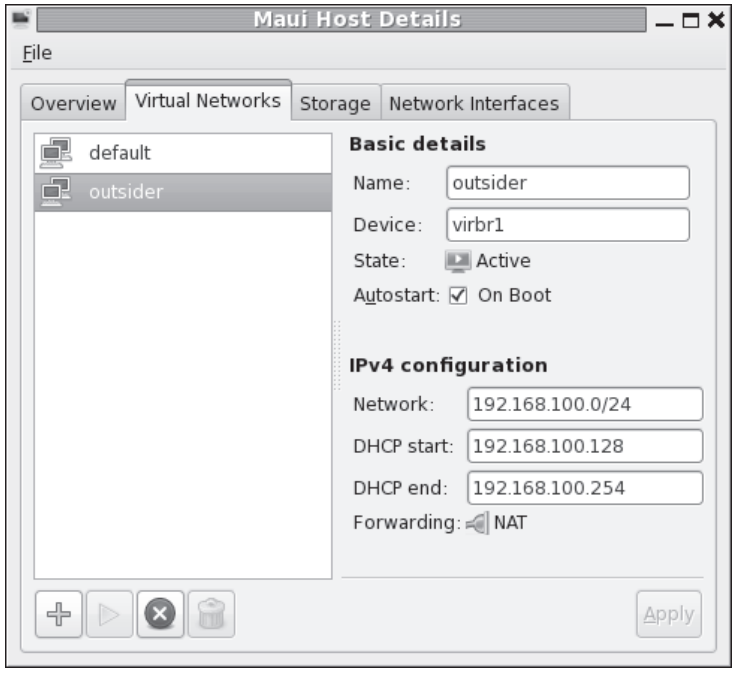
Revise Network Interfaces on a Cloned System

Now that you know something of networking on a RHEL system, you're ready to modify the network interfaces on a cloned system. Since cloned systems originally have the same network interfaces as the original, they should be changed before they're connected to a local network. Otherwise, there will be two systems on the same network, declaring their ownership of the same IP addresses. And that would be trouble for both systems.

These instructions assume that you want to set up the cloned system as the outsider1.example.org system described in Chapter 1, with an IP address of 192.168.100.100, on the 192.168.100.0/24 network. If an outsider1.example.org system already exists, you can substitute a different system name on a different IP address. (Some home network devices such as cable modems are configured on the 192.168.100.0 network. In that case, you may need to use a different network address.)

If you've set up a cloned system on a virtual machine such as KVM, the first steps involve deleting the current network card and creating a new network card. As this book is modeled on KVM-based virtual machines, the steps are based on that system. While not tested, the same principles should help you reconfigure networking on a cloned system on a different type of virtual machine such as KVM.

1. Open the Virtual Machine Manager with the **virt-manager** command.
2. Connect to the system with the target virtual machines, normally **localhost (QEMU)**.
3. Click Edit | Host Details, and click the Virtual Networks tab.
4. Select Add (which may be a plus sign). Follow the prompts to create the new virtual network, including the IP address range given. Call this the **outside** network (or another name of your choice). Do not delete the current default network; both networks are required. Traffic between the two networks can be routed through the physical host system.
5. Configure the system to forward information to the physical network, using Network Address Translation (NAT). The result should look like the illustration.



6. Return to the main Virtual Machine Manager window. Select the cloned system. Click View | Details.
7. Select the current network device and click Remove.
8. Click Add Hardware, select the Network hardware type, select the outside network, and create the new network card.
9. Click View | Console. Boot the cloned system into runlevel 1, which should not start networking.
10. Edit the /etc/sysconfig/network file. Be sure to make a change to the **HOSTNAME** directive, in this case to outsider1.example.org.
11. Examine the /etc/sysconfig/network-scripts/ifcfg-eth0 file. Don't make any changes yet. Since you've changed network cards, the system will detect it and use the next available network device.
12. Move to runlevel 3 with a command like **telinit 3**. Unless you've made some unusual changes, that should start networking.

13. Run the **ifconfig -a** command. It'll display the detected Ethernet device, along with its hardware address. The detected Ethernet device will be different form eth0, perhaps eth1. You'll need both pieces of information, along with the desired IP address information described earlier.
14. Open the /etc/sysconfig/network-scripts/ifcfg-eth0 file. Make appropriate changes to the **DEVICE**, the various IP addresses, and the **HWADDR** directives.
15. Apply the **cp** or **mv** commands to the ifcfg-eth0 file. The new name should reflect the new Ethernet device, such as ifcfg-eth1 for the eth1 device.
16. Copy the resulting file to the /etc/sysconfig/networking/devices and /etc /sysconfig/networking/profiles/default directories.
17. Restart networking with the **/etc/sysconfig/network restart** command.
18. Verify the network device and routing table have the right settings with the **ifconfig** and **route -n** commands.

Assuming IP Forwarding has been enabled on the physical host system, and there are no blocks such as firewall rules, the virtual machines should now all be able to communicate with outside networks. In Chapter 10, you'll explore how to set appropriate IP forwarding rules based on **iptables** commands documented in the /etc/sysconfig/iptables file. For more information on firewalls and IP Forwarding, see Chapters 4 and 10, respectively. If you're studying only for the RHCSA exam, open the /etc/sysctl.conf file, make sure to set **net.ipv4.ip_forward=1** and run the **sysctl -p** command.

CERTIFICATION OBJECTIVE 5.06

Time Synchronization

The configuration of a Network Time Protocol (NTP) client is straightforward. While it's no longer part of the RHCSA objectives, it's still part of one of the prep courses for the RHCSA, RH124. It would be easy for Red Hat to include this topic as part of future RHCSA objectives. Therefore, this section provides a minimal overview of the client files and the associated GUI configuration tool.

There are good reasons to keep different systems running on the same clock. Otherwise, one system that may take web orders may miss the lack of inventory from a second system that manages a warehouse database, or even see out-of-date production figures from a third system associated with a manufacturing assembly line.

An NTP Client

Every system, real or virtual, starts with a hardware clock. The time on that clock may depend on the power in a battery; over time, batteries lose power and many hardware clocks end up losing time. The installation process on RHEL 6 normally sets the hardware clock to UTC, which is essentially identical to Greenwich Mean Time (GMT). Linux bases time changes such as Daylight Saving Time on the use of UTC.

Every RHEL 6 system includes a time zone configured in the /etc/sysconfig/clock file. The contents are simple; a typical system may include the following line:

```
ZONE="America/Los Angeles"
```

The default NTP configuration file, /etc/ntp.conf, is set up to connect to standard Red Hat servers that are part of the NTP pool project. Collectively, any errors from these servers relative to actual time is minimized.

```
server 0.rhel.pool.ntp.org
server 1.rhel.pool.ntp.org
server 2.rhel.pool.ntp.org
```

Users of rebuild distributions such as CentOS will see different Universal Resource Identifiers (URIs), such as 0.centos.pool.ntp.org.

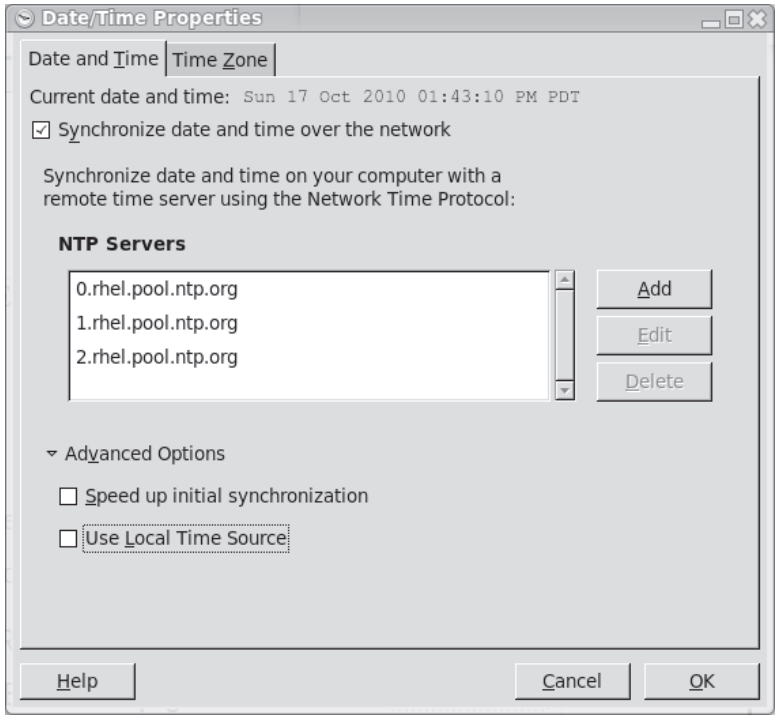
Date/Time Properties

With the Date/Time Properties configuration tool, you can set the date, time, time zone, and NTP server for the local system. To start it in the GUI, run the **system-config-date** command. This opens the Date/Time Properties window shown in Figure 5-15.

You should recognize the URIs for the NTP servers as the pool servers previously discussed. In other words, the Date/Time Properties tool is a front end for editing the /etc/ntp.conf file. The advanced option to Speed Up Initial Synchronization minimizes the time lag effect of polling multiple NTP servers. The Use Local Time

FIGURE 5-15

Date/Time Properties tool



Source directive uses the local system as its own server. The Time Zone tab supports changes to the local time zone, handy for users who travel.

CERTIFICATION SUMMARY

This chapter covered the basic boot process of an RHEL system. It starts with the hardware POST and continues with the BIOS or UEFI systems. Once it finds boot media, it moves to the first stage of the GRUB bootloader. The GRUB menu allows you to select and customize the kernel to be booted. GRUB can be secured, as it supports booting into different runlevels, including a runlevel which supports root-level access without a user password.

Once you've selected an option from GRUB, it hands control to the kernel. It starts with a temporary filesystem known as the Initial RAM disk. Once essential drivers and filesystems are loaded, you can find out more about what happens through /var/log/dmesg and the drivers it loads. It hands control to the First Process,

also known as **init**, as configured through the newer Upstart system with related files in the /etc/init directory.

Linux services are controlled in part by runlevel. While RHEL 6 uses the Upstart system, the default runlevel is still configured in /etc/inittab, and the service scripts in the /etc/rcn.d directories, where n represents the runlevel, determine which services are started and stopped. The status of those scripts in each runlevel can be configured with the **chkconfig** command (along with other tools). Runlevel scripts are linked to real scripts in the /etc/init.d directory, which can be used to start, stop, restart, reload, even status a service, and more.

For most users, network configuration is also an important part of the boot process. Most related configuration files can be found in the /etc/sysconfig/network-scripts directory. While you can modify the network configuration with the **ifup**, **ifdown**, **ifconfig**, and **route**, and **dhclient** commands, you can detail the current status of the network with the **ifconfig**, **route**, **netstat**, and **arp** commands. If you prefer to configure with administrative tools, RHEL 6 includes Network Manager and system-config-network packages.

You may need to set up local systems as NTP clients. The Date/Time Properties tool can help configure key configuration files such as /etc/sysconfig/ntp and /etc/ntp.conf.

TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 5.

The BIOS and the UEFI

- ❑ While not strictly a part of the exam, it's important to know the basics of the BIOS and the UEFI.
- ❑ You can change the boot sequence from the BIOS/UEFI menu.
- ❑ Once the BIOS/UEFI detects a designated boot drive(s), it hands control to GRUB via the master boot record (MBR) of the appropriate drive.

Bootloaders and GRUB

- ❑ RHEL 6 uses the traditional version of GRUB, version 0.97.
- ❑ The GRUB configuration file is organized in stanzas.
- ❑ From the GRUB menu, you can boot into a runlevel other than the default. You can even boot into a runlevel that provides root administrative access without an account password.
- ❑ The GRUB menu, along with individual options, can be password-protected.
- ❑ The GRUB configuration file specifies a kernel, a root directory volume, and an initial RAM disk for each operating system.
- ❑ If the GRUB configuration file is missing, you may be able to boot from the grub> prompt with information on the /boot directory partition, the Linux kernel file, the top-level root directory, and the Initial RAM disk file.

Between GRUB and Login

- ❑ You can analyze boot messages through /var/log/dmesg or the **dmesg** command.
- ❑ There are six different runlevels available; the default is configured in /etc/inittab.
- ❑ Upstart has replaced SysVInit, with configuration files in the /etc/init directory.

- Once the kernel boots, it hands control to init, also known as the First Process.
- Different services are started based on the default or chosen runlevel.

Control by Runlevel

- The default runlevel configured in /etc/inittab runs scripts in the associated /etc/rc*n*.d directory, where *n* is the runlevel.
- Runlevel scripts in the /etc/rc*n*.d directories are links to actual scripts in the /etc/init.d directory.
- You can use runlevel scripts in the /etc/init.d directory to control a service with the start, stop, restart, reload, and other commands.
- The services that start in each runlevel can be controlled with the **chkconfig** command, as well as the tools associated with the **ntsysv** and **system-config-services** commands.

Network Configuration

- Important network configuration commands include **ifup**, **ifdown**, **ifconfig**, **route**, **netstat**, and **arp**.
- Key network configuration files are located primarily in the /etc, /etc/sysconfig, and /etc/sysconfig/network-scripts directories.
- You can edit the network configuration files directly, or configure network connections with the tools available through the **system-config-network-tui** or **nm-connection-editor** commands
- Name resolution depends on appropriate settings in the /etc/resolv.conf, /etc/hosts, and /etc/nsswitch.conf files.

Time Synchronization

- The NTP service can help keep systems in sync with servers configured in the /etc/ntp.conf file.
- NTP clients can be configured with the Date/Time Configuration tool.

SELF TEST

The following questions will help measure your understanding of the material presented in this chapter. As no multiple-choice questions appear on the Red Hat exams, no multiple-choice questions appear in this book. These questions exclusively test your understanding of the chapter. It is okay if you have another way of performing a task. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer for many of these questions.

The BIOS and the UEFI

1. On what part of the boot hard drive is the first stage of the GRUB bootloader typically located?

_____

Bootloaders and GRUB

2. When you see the GRUB configuration menu, what command would you use to modify the kernel arguments? Assume that GRUB is not password-protected.

_____

3. What command would you enter at the **kernel** command line to boot into single-user mode?

_____

4. If you see the **root(hd0,1)** directive in the GRUB configuration file, on what partition is the /boot directory? Assume the GRUB configuration file is properly configured.

_____

Between GRUB and Login

5. What temporary file system is loaded directly from the GRUB menu?

_____

6. What one-word command can you use to read the kernel initialization messages?

_____

7. In what directory can you find the configuration files associated with the first process?

_____

8. What is the name of the system that controls splash screens, configured in the /etc/init directory?

_____

Control by Runlevel

9. In what directory can you find the scripts that control services? Hint: scripts in specific runlevels are linked to the actual scripts in this directory.

_____

10. Name three actions that can be typically run from scripts in the directory associated with question 9.

_____

_____

_____

Network Configuration

11. What command lists all network devices currently available on the local system, including those that are not active?

_____

12. What command lists the current routing table of the local system?

_____

Time Synchronization

13. What command starts the GUI-based configuration tool for time synchronization?

_____

LAB QUESTIONS

Several of these labs involve installation exercises. You should do these exercises on test machines only. The second Lab of Chapter 1 sets up KVM for this purpose. However, some readers may not have hardware that supports KVM. Options to KVM include virtual machine solutions such as VMware, available from www.vmware.com, or Virtualbox, open-source edition, available from www.virtualbox.org. Red Hat presents its exams electronically. For that reason, most of the labs in this and future chapters are available from the CD that accompanies the book, in the Chapter5/ subdirectory. It's available in .doc, .html, and .txt formats, in the filename starting with 56505-labs. In case you haven't yet set up RHEL 6 on a system, refer to the first lab of Chapter 2 for installation instructions. The answers for each lab follows the Self Test answers for the fill-in-the-blank questions.

SELF TEST ANSWERS

The BIOS and the UEFI

1. For the BIOS/UEFI to hand control over to Linux, it needs to identify the Master Boot Record (MBR) of the boot hard drive.

Bootloaders and GRUB

2. From the GRUB menu, the command that modifies kernel arguments is **a**.
3. To boot into single-user mode from the GRUB **kernel** command line, you'd enter the **1** command; while not identical, **single** is also acceptable.
4. The **root (hd0,1)** directive documents the /boot directory on the second partition on the first hard drive.

Between GRUB and Login

5. The temporary filesystem loaded from the GRUB menu is the initial RAM disk filesystem, also known by its filename, initramfs.
6. The one-word command that you can use to read the kernel initialization messages is **dmesg**.
7. The configuration files associated with the first process are located in the /etc/init directory.
8. The name of the system that controls splash screens on RHEL 6 is Plymouth.

Control by Runlevel

9. Scripts that control services are located in the /etc/init.d directory. The /etc/rc.d/init.d directory is also an acceptable answer. While there are scripts that qualify in other directories, /etc/init.d is the location for the great majority of such scripts.
10. Typical actions that can be run from scripts in the /etc/init.d directory include **start**, **stop**, **restart**, **reload**, and more. Any of the actions listed near the end of a file in that /etc/init.d directory is an acceptable answer.

Network Configuration

11. The **ifconfig -a** command lists all detected network devices, active and inactive.

12. The route command lists the current routing table of the local system. Other acceptable answers include **route -n**, **netstat -r**, and **netstat -rn**.

Time Synchronization

13. The **system-config-date** command starts the Date/Time Configuration tool.

LAB ANSWERS

Yes, there are many Linux systems that run for years at a time without a reboot. But reboots are sometimes required, such as when newer kernels are installed. So when configuring a Linux system, make sure any changes survive a reboot. Otherwise, your supervisor may not believe that you ever did the work. On a Red Hat exam, you won't get credit unless changes survive a reboot.

Lab 1

If successful, this lab will show you how to change the default runlevel, along with the relative importance of the options in the GRUB bootloader. Normally, it's best to back up a file before making changes. However, the RHEL 6 version of the /etc/inittab file is pretty simple, as it contains only one line of significance:

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

This assumes the default runlevel is 5; if there's a different default runlevel for your system, substitute accordingly.

Lab 2

This lab should make a point: it's far too easy in the default configuration of RHEL 6 to get access to the root administrative account.

If you had to back up the /etc/shadow file and were successful with this lab, you should not have to restore that file from backup. The **passwd** commands executed during this lab should have restored the intended password for the root administrative user. One way to meet the requirements of this lab is with the following steps:

1. Power up the local system. During the boot process, when you see the following message (the operating system name and version number may vary), press a key.

```
Booting Red Hat Enterprise Linux Server (2.6.32-71.el6.x86_64) in 5 seconds....
```

2. Edit the default option. If the GRUB bootloader is already password-protected, press P and enter the password at the given prompt.
3. Press A to edit the **kernel** command line. You should see a line with a cursor similar to this:

```
< rhgb quiet
```

4. Add a **1** to the end of the kernel command line, as follows, and press ENTER.

```
< rhgb quiet 1
```

When the system boots, you should see entries similar to the following, including a command line prompt:

```
Telling INIT to go to single user mode.
[root@tester1 /]#
```

5. Try the **passwd** command. It should immediately prompt you for a new password. Go ahead and enter a password, confirming the new password when prompted.

In some configurations, the **passwd** command may not work in single-user mode. In that case, you'll have to edit the /etc/shadow file. Back it up first, perhaps to the /root directory. The first line in that file should look similar to this:

```
root:$6a24fdsaj..a432:14972:0:99999:7
```

In this case, delete the contents of the second column (between the first and second colons). If you're editing /etc/shadow in the vi editor, you'll have to save and exit with the **:wq!** command.

6. Test the result. Reboot the system, or run the **init 3** command.
7. Open a text login console. Log in as the root administrative user with the new root password. If you had to delete the password column in /etc/shadow, the system will log you in as the root administrative user without a password. In that case, you'll immediately want to set a root administrative password with the following command:

```
# passwd
```

Lab 3

In Lab 2, you should have seen how easy it is for a user with access to the GRUB menu to access the root administrative account, without the password. Now you should understand the importance of password protection of the GRUB menu. Successful completion of this lab should be easy to confirm. Reboot the system, and press a key in the five seconds before the default option is booted to get to

the GRUB menu. In that menu, you should see the **'p' to enter a password to unlock the next set of features** message. One way to meet the requirements of this lab is with the following steps:

1. Power up the local system. During the boot process, when you see the following message (the operating system name and version number may vary), press a key.

```
Booting Red Hat Enterprise Linux Server (2.6.32-71.el6.x86_64) in 5 seconds....
```

2. If the GRUB bootloader is already password-protected, you'll see the following message:

```
Press enter to boot the selected OS or 'p' to enter a password to unlock
the next set of features.
```

   In the default RHEL 6 installation, the GRUB bootloader is not password-protected. The following steps assume that you're working with such a system.

3. Select a kernel and press ENTER to boot RHEL 6.
4. While it's possible on some systems to copy a hashed password from a text console, it is more convenient to do so from the GUI. To that end, log in to the GUI as the root administrative user.
5. Open two command line consoles in the GUI. One method is to click Applications | System Tools | Terminal, and repeat.
6. Open the GRUB configuration file in a text editor. While there are links from other files, the actual location is /boot/grub/grub.conf.
7. At the second command line interface, run the **grub-md5-crypt** command. Enter a password when prompted and repeat when prompted to confirm. The typed-in password won't be shown. The encrypted password starts with the $1. (In this case, the password is an encrypted hash of the word *redhat*.)

```
# grub-md5-crypt
Password:
Retype password:
$1$KdqKv/$T1L0luzEOU2zFJFm8UpCv0
```

8. In the GRUB configuration file, open a line before the first **title** directive. Type in the following:

```
password --md5
```

9. Copy the password from the first command line console, and append it to the line in the GRUB configuration file; it should read as follows:

```
password --md5 $1$KdqKv/$T1L0luzEOU2zFJFm8UpCv0
```

10. Save the changes to the GRUB configuration file. Reboot the system, and repeat Steps 1 and 2. You should now see the password prompt suggested in Step 2. The GRUB menu is now password protected.

While it's easy to delete the **password** directive from the GRUB configuration file, as configured in this lab, I suggest that you keep that directive. In fact, you should add it to all systems, at least those which may be accessed by other users.

Lab 4

After completing this lab, two stanzas should exist in the /boot/grub/grub.conf configuration file. They might appear similar to Figure 5-16. The only difference between the two stanzas is the **title** directive, the **password** directive, and the number **1** at the end of the **kernel** command.

To really test the result, reboot the system, select the Single-User Mode option, and then enter an incorrect password when prompted. What happens?

If desired, you can now remove the second stanza from the GRUB configuration file, /boot/grub /grub.conf. Alternatively, you can restore grub.conf from the backup location.

FIGURE 5-16

Sample GRUB configuration file with Single-User Mode stanza

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/vda2
#          initrd /initrd-[generic-]version.img
#boot=/dev/vda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
password --md5 $1$A.CLv/$.jw8JKlyQkknUM3siBf981
title Red Hat Enterprise Linux (2.6.32-71.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-71.el6.x86_64 ro root=UUID=ffe37c21-53ab-46ab-b9d
f-4b856046e18f rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=l
atarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=auto rhgb quiet
        initrd /initramfs-2.6.32-71.el6.x86_64.img
title Single User Mode
        password --md5 $1$A.CLv/$.jw8JKlyQkknUM3siBf981
        root (hd0,0)
        kernel /vmlinuz-2.6.32-71.el6.x86_64 ro root=UUID=ffe37c21-53ab-46ab-b9d
f-4b856046e18f rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 SYSFONT=l
atarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us crashkernel=auto rhgb quiet 1
        initrd /initramfs-2.6.32-71.el6.x86_64.img
~
```

Lab 5

The script executed in this lab moved the grub.conf configuration file to the /root/backup directory. If you understand GRUB well, you should have been able to boot the system from the grub> prompt.

Otherwise, you can recover the grub.conf file by booting into the rescue mode described in this Chapter. From the rescue mode command line prompt, you should be able to restore the original configuration with the following commands:

```
# chroot /mnt/sysimage
# cp /root/backup/grub.conf /boot/grub/
```

If that doesn't work, review Figure 5-16 for clues on what to put in the file. The UUID shown in the figure is almost definitely not the UUID for partition where your top-level root directory is mounted.

Alternatively, review the menu.lst file in the /usr/share/doc/grub-0.97 directory. If you absolutely need to include the UUID number for the volume, it's available in the default version of the /etc/fstab file. If you can't find it there, and the top-level root directory is mounted on partition /dev/vda2, you can find the appropriate UUID number with the following command:

```
# dumpe2fs /dev/vda2 | grep UUID
```

Lab 6

When you've completed this lab, you'll understand the relative importance of files in the /etc/sysconfig directory, even relative to the files in the /etc/init directory. The results may be a bit surprising, as init is the first process.

It's possible to configure up to 12 virtual terminals, which matches the number of function keys available on most keyboards. If you want to set up 12 virtual terminals (and that would be an interesting problem for the RHCSA exam), look at the /etc/securetty file and related man pages with the **man -k securetty** command. One way to accomplish the tasks in this lab is with the following steps:

1. Open the /etc/init/start-ttys.conf file. Change the following directive to limit the active consoles to terminals 1 and 2:

   ```
   env ACTIVE_CONSOLES=/dev/tty[1-6]
   ```

2. To test the result, move to runlevel 1 and then move to runlevel 3. Hint: you can use the **init** or **telint** commands for that purpose.
3. What happens? Can you still log into terminals 3, 4, 5, and 6?
4. Now edit the /etc/sysconfig/init file. Note the ACTIVE_CONSOLES directive. Use it to limit the active consoles to terminals 1 and 2.

5. Rerun Step 3. Did the changes to the /etc/sysconfig/init file do the trick?
6. Next, comment out the applicable directive in the /etc/sysconfig/init file, and then rerun Step 3. What consoles are still active?
7. What can you conclude about the importance of the /etc/sysconfig/init and the files in the /etc/init directory?
8. When complete, just remember to restore the original versions of the /etc/sysconfig/init and /etc/init/start-ttys.conf files.

Lab 7

This lab is straightforward; it substituted a /etc/sysconfig/network file with the **NETWORKING=no** directive. After the **ifdown eth0** command is run, when the /etc/init.d/network script is restarted, it reads this file and does not activate networking.