Chapter 2:

Learn basic Kernel Management

Learning objectives

Upon completing this chapter, the learner should be able to:

- Understand the concept of GRUB
- Boot systems into different runlevels manually
- Interrupt the boot process in order to gain access to a system
- Configure systems to boot into a specific target automatically
- Update the kernel package appropriately to ensure a bootable system
- Manage the kernel modules

Key terms

GRUB2

GRUB menu

grub2-mkconfig

grub.cfg

Linux kernel

Kernel parameters

Emergency target

Rescue target

Kernel version

Kernel image

lsmod modinfo modprobe

Kernel modules

dmesg

journalctl

uname

/proc

Table of content

1. Working with GRUB2

In RHEL7, a new, enhanced version of GRUB, called GRUB2, has been introduced, replacing the legacy GRUB, it supports both BIOS/MBR and UEFI/GPT combinations. In general, the GRUB2 boot loader is one of the first things that needs to be working well to boot a Linux server, this because after it has been loaded into memory and it takes control, it searches for the kernel, extracts its code into memory, decompresses it, and loads it based on the configuration defined in the /boot/grub2/grub.cfg file.

1.1. Understanding GRUB2

GRUB2 is installed in the boot sector of your server's hard drive and is configured to load a Linux kernel and the initramfs:

- The kernel is the heart of the operating system, allowing users to interact with the hardware that is installed in the server.
- The initramfs contains drivers that are needed to start your server. It contains a mini file system that is mounted during boot. In it are kernel modules that are needed during the rest of the boot process (for example, the LVM modules and SCSI modules for accessing disks that are not supported by default).

Normally, GRUB2 works just fine and does not need much maintenance. In some cases, though, you might have to change its configuration which mainly located in different files in the system, in the following we will learn about these files and how to deal with them.

1.2. The /etc/default/grub file

Values defined in this file are used to regenerate the /boot/grub2/grub.cfg file, which controls the behavior of GRUB at boot time. Any changes made to the grub file will only take effect after the grub2-mkconfig utility has been executed. Here are the default settings from the /etc/default/grub file, with an explanation in the table after.

Directive	Description
GRUB_TIMEOUT	Sets the wait time, in seconds, before booting off the default kernel. Default value is 5
GRUB_DISTRIBUTOR	Defines the name of the Linux distribution. "$(sed 's, release. *$, g' /etc/system-release)"
GRUB_DEFAULT	Boots the selected option from the previous system boot. saved
GRUB_DISABLE_SUBMENU	Enables/disables the appearance of GRUB submenu. true
GRUB_TERMINAL_OUTPUT	Sets the default terminal. "console"
GRUB_CMDLINE_LINUX	Specifies the command line options to pass to the kernel at boot time. "rd.lvm.lv=vg00/swap vconsole.font=latarcyrheb-sun16 crashker nel=auto vconsole.keymap=us rd.lvm.lv=vg00/root rhgb quiet"
GRUB_DISABLE_RECOVERY	Disables showing system recovery entries in the GRUB menu. true

Generally, you do not need to make any changes to this file, as the default settings are good enough for normal system operation. The /boot/grub2/grub.cfg file.

This is the main configuration file that controls the behavior of GRUB at boot time. This file is located in the /boot/grub2 directory on BIOS-based systems and in the /boot/efi/EFI/redhat directory on UEFI-based systems. This file can be regenerated manually with the grub2-mkconfig utility, or it is automatically regenerated when a new kernel is installed. During this process, any manual changes made to this file are lost.

Here is how you would run this utility to reproduce the grub.cfg file on BIOS and UEFI systems, respectively:

```
#grub2-mkconfig -o /boot/grub2/grub.cfg
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

When this utility runs, it uses the settings defined in the /etc/default/grub file and in the helper scripts located in the /etc/grub.d/ directory to regenerate this file for kernels located in the / boot directory.
If a new kernel is added to the system, existing kernel entries will remain in this file and can be chosen in the GRUB menu at startup to boot.

Exercise: Applying Modifications to GRUB2

1. Open the file /etc/default/grub with an editor and remove the rhgb and quiet options from the GRUB_CMDLINE_LINUX line.
2. From the same file, set the GRUB_TIMEOUT parameter to 10 seconds. Save changes to the file and close the editor.
3. From the command line, type grub2-mkconfig > /boot/grub2/grub.cfg to write the changes to GRUB2. (Note that instead of using the redirector > to write changes to the grub.cfg file, you could use the -o option. Both methods have the same result.)
4. Reboot and verify that while booting you see boot messages scrolling by.

1.3. The GRUB menu

The GRUB menu shows a list of bootable kernels at the top, where you can change the selection using the up or down arrow key, or also editing a selected kernel menu entry by:

- Pressing e or go to the grub> command prompt by pressing c. In the edit mode, GRUB loads the selected entry from the /boot/grub2/grub.cfg file in an editor, which you are allowed to modify before booting.
- You can press Ctrl+x to boot after making the change.
- Ctrl+c to switch into the grub> command prompt, Or press ESC to discard the changes made and go back to the main menu displays one of the entries and the action keys.
- The grub> prompt appears when you type c. While at the prompt, you can press the TAB key to view a list of all available commands that you can run to perform a desired action.

```
Setparams 'Centos Linuxs (3.10.0-1062. 4. 3. e17. x86_64) 7 (core)
        load_video
        set gfxpay load=keep
        insmod gzio
        insmod part_msdos
        insmod xfs
       set root='hd0, msdos1'
       if [x$ feature_platform_search_hint = xy]; then
           Search  --no  -floppy  --fs-uuid  -set=root  -hint-
bios=hd0, msdos1 -hin\
t-efi= hd0, msdos1 --hint-baremetal=ahci0, msdos1 --hint='hd0,
msdos1' a4818155-b\ a8b-4529-b9ae 2263fa202034
      else
          search  --no  -floppy  --fs  -uuid  --set=root  a4818155-
ba8b-4529-b9ae-2263\ fa202034
    press Ctrl-x to start, Ctrl-c for a command prompt or Escape
to
    discard edits and return to the menu. Pressing Tab lists
    possible completions
```

1.4. Modifying default GRUB2 options

If you enter the GRUB2 boot prompt to add kernel startup parameters, the contents of the /boot/grub2/grub.cfg file display. From here, you add one–time–only startup options. The following shows the relevant part of the grub.cfg file that takes care of loading the Linux kernel. Notice the line that starts with linux16, this line specifies all kernel boot parameters.

Partial Contents of the /boot/grub2/grub.cfg Configuration File

```
[root@LP2 ~]# cat /boot/grub2/grub.cfg
menuentry 'CentOS Linux (3.10.0-1062.4.3.el7.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-862.el7.x86_64-advanced-bf0d0b83-aeaf-4b5f-bb40-2ece311979c0' {
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' a4818155-ba8b-4529-b9ae-2263fa202034
        else
          search --no-floppy --fs-uuid --set=root a4818155-ba8b-4529-b9ae-2263fa202034
        fi
        linux16 /vmlinuz-3.10.0-1062.4.3.el7.x86_64 root=/dev/mapper/centos-root ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet LANG=en_US.UTF-8
        initrd16 /initramfs-3.10.0-1062.4.3.el7.x86_64.img
}
```

To apply modifications to the GRUB2 boot loader, the file /etc/default/grub is your entry point, do not change the contents of the /boot/grub2/grub.cfg configuration file directly.

- The most important line in this file is GRUB_CMDLINE_LINUX, which defines how the Linux kernel should be started. In this line, you can apply permanent fixes to the GRUB2 configuration.
- Some likely candidates for removal are the options rhgb and quiet. These options tell the kernel to hide all output while booting. That is nice to hide confusing messages for end users, but if you are a server administrator, you probably just want to remove these options. Without these you will not have to guess why your server takes a long time after a restart, you'll just be able to see.
- Another interesting parameter is GRUB_TIMEOUT. This defines the amount of time your server waits for you to access the GRUB2 boot menu before it continues booting automatically. If your server runs on physical hardware that takes a long time to get through the BIOS checks, it may be interesting to increase this time a bit.

1.5. Booting into specific targets

RHEL is booted into graphical target state by default. It can also be booted into other non-default, but less capable, operating targets from the GRUB menu. Additionally, in situations when it becomes mandatory to boot the system into an administrative state for carrying out a function that cannot be otherwise performed in other target states or for system recovery, RHEL offers emergency and rescue targets. These special targets can be entered by interacting with the GRUB interface, selecting a boot menu entry, pressing e to enter the edit mode, and supplying the desired target with the systemd.unit directive.

For instance, to boot into the emergency target, append systemd.unit=emergency.target (or simply 'emergency') to the default linux kernel line entry, as shown below:

```
       linux16 /vmlinux-3.10.0-123.e17.x86_64
root=UUID=964201bb-1e32-4794-a2\f2-7a33e2fb591a ro
rd.lvm.lv=vg00/swap
vconsole.font=latarcyrheb-sun16 crashke\rnel=auto
vconsole.keymap=us rd.lvm.lv=vg00/root rhgb quiet
systemd.unit=emer\gency.target_
```

Press Ctrl+x after making the modification to boot the system into the supplied target. You will be required to enter the root user password to log on. Run systemctl reboot after you are done to reboot the system.

Similarly, you can enter systemd.unit=rescue.target (or simply 1, s, or single) with the linux kernel line entry and press Ctrl+x to boot into the rescue target, which is also referred to as the single-user mode.

For this exercise, assume that the root user password has been lost or forgotten, and it needs to be reset. You will boot the system into a special shell in order to reset the root password.

Exercise: Resetting the root user password

1. Reboot or reset server1, and interact with GRUB by pressing a key before the autoboot timer runs out. Highlight the default kernel entry in the GRUB menu and press e to enter the edit mode. Scroll down and you will find a boot string similar to the following:

```
       linux16 /vmlinuz-3.10.0-1062.4.3.e17.x86_64
root=/dev/mapper/centos-ro\ot ro crashkernel=auto
rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet\
LANG=en_US.UTF-8
       initrd16 /initramfs-3.10.0-1062.4.3.e17.x86-64.img
```

2. Modify this kernel string and append "init = /sysroot/bin/sh" to the end of the line to look like:

```
       linux16 /vmlinuz-3.10.0-1062.4.3.e17.x86_64
root=/dev/mapper/centos-ro\ot ro crashkernel=auto
rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet\
LANG=en_US.UTF-8\
init=/sysroot/bin/sh
       initrd16 /initramfs-3.10.0-1062.4.3.e17.x86-64.img
```

3. Press Ctrl+x when done to boot to the special shell. The system mounts the root file system read-only on the /sysroot directory. Make /sysroot appear as mounted on / using the chroot command:

   # chroot /sysroot

4. Remount the root file system in read/write mode with the mount command:

   # mount –o remount,rw /

5. Enter a new password for root by invoking the passwd command:

passwd

6. Create an empty, hidden file called. autorelabel at the root of the directory tree to instruct the system to perform SELinux relabeling upon next reboot:

touch /. autorelabel

7. Exit out of the special shell:

exit

8. Reboot the system:

reboot

2. Linux Kernel

The Linux kernel is the heart of the operating system, it is the layer between the user who works with Linux from a shell environment and the hardware that is available in the computer on which the user is working. It is doing so by managing the I/O instructions it receives from the software and translating those to processing instructions that are to be executed by the central processing unit and other hardware in the computer. The kernel also takes care of handling essential operating system tasks like the scheduler, which makes sure that processes that are started on the operating system are handled by the CPU.

2.1. Installed kernel

The default kernel installed during the installation is usually adequate for most system needs; however, it requires a rebuild when a new functionality is added or removed. The new functionality may be introduced by installing a new kernel, upgrading the existing one, installing a new hardware device, or changing a critical system component.

RHEL allows us to generate and store several custom kernels with varied configuration and required modules, but only one of them is active at a time and the others may be loaded via GRUB. To list all installed kernel packages:

```
[root@LP2 ~]# yum installed kernel-*
Installed Packages
kernel.x86_64       3.10.0-862.el7      @anaconda
kernel.x86_64    3.10.0-1062.4.3.el7   @updates
kernel-tools.x86_64 3.10.0-862.el7     @anaconda
kernel-tools-libs.x86_64   3.10.0-862.el7 @anaconda
```

Likewise, an existing functionality that is no longer required may be removed to make the kernel smaller, resulting in improved performance and reduced memory utilization.

To control the behavior of the modules, and the kernel in general, several tunable parameters are set. It defines a baseline for kernel functionality. Some of these parameters must be tuned to allow certain applications and database software to be installed smoothly and operate properly.

Determining Kernel Version

To determine the version of the running kernel on the system, run the uname command:

```
[root@LP2 ~]# uname -r

3.10.0-1062.4.3.el7.x86_64


The output indicates the kernel version currently in use
is 3.10.0-1062.4.3.el7.x86_64
```

2.2. Kernel directory structure

Kernel and its support files are stored at different locations in the directory hierarchy, of which three locations: /boot, /proc, and /lib/modules —are of significance and are explained below.

2.2.1. The /boot File System

The /boot file system is created at system installation and its purpose is to store kernel and associated files. This file system also stores any updated or modified kernel data. Listing the content of the /boot produces the following information:

```
[root@lp2 ~]# ls /boot/
config-3.10.0-1127.19.1.el7.x86_64
config-3.10.0-1127.el7.x86_64
efi
grub
grub2
initramfs-0-rescue-cc7023446ea74f37ab8014f3e85845c5.img
initramfs-3.10.0-1127.19.1.el7.x86_64.img
initramfs-3.10.0-1127.19.1.el7.x86_64kdump.img
initramfs-3.10.0-1127.el7.x86_64.img
initramfs-3.10.0-1127.el7.x86_64kdump.img
symvers-3.10.0-1127.19.1.el7.x86_64.gz
symvers-3.10.0-1127.el7.x86_64.gz
System.map-3.10.0-1127.19.1.el7.x86_64
System.map-3.10.0-1127.el7.x86_64
vmlinuz-0-rescue-cc7023446ea74f37ab8014f3e85845c5
vmlinuz-3.10.0-1127.19.1.el7.x86_64
vmlinuz-3.10.0-1127.el7.x86_64
```

The output indicates that:

- the current kernel is vmlinuz-3.10.0-1062.4.3.el7.x86_64
- its boot image is stored in the initramfs-3.10.0-1062.4.3.el7.x86_64.img
- and configuration in the config-3.10.0-1062.4.3.el7.x86_64
- A sub-directory /boot/grub2 contains GRUB information as shown below:

```
[root@lp2 ~]# ll /boot/grub2/
total 32
-rw-r--r--. 1 root root   84 Oct 27 19:22 device.map
drwxr-xr-x. 2 root root   25 Oct 27 19:22 fonts
-rw-r--r--. 1 root root 5218 Nov 17 23:50 grub.cfg
-rw-r--r--. 1 root root 1024 Nov 17 23:50 grubenv
drwxr-xr-x. 2 root root 8192 Oct 27 19:22 i386-pc
drwxr-xr-x. 2 root root 4096 Oct 27 19:22 locale
```

The key file in /boot/grub2 is grub.cfg, which maintains a list of available kernels and defines the default kernel to boot, along with other information.

2.2.2. The /proc File System

/proc is a virtual file system and its contents are created in memory at system boot and destroyed when the system goes down. Underneath this file system lie current hardware configuration and status information.

Listing the content of the /proc produces several files and sub-directories:

- Some sub-directory names are numerical and contain information about a specific process, with process ID matching the sub-directory name. Within each sub-directory, there are files and further sub-directories that include information, such as memory segment specific to that particular process.

```
[root@lp2 ~]# ls /proc/
1      14     1812   1927   1982   2615   292    378   501   615   701   731
asound        filesystems   locks          self           vmstat
10     1484   1816   1928   1986   2620   294    379   51    616   702   732
buddyinfo     fs            mdstat         slabinfo       zoneinfo
103    1489   1840   1932   1989   27     295    38    510   618   703   733
bus           interrupts    meminfo        softirqs
11     1490   1848   1936   2      2732   296    391   52    624   704   734
cgroups       iomem         misc           stat
1118   1586   1865   1938   20     2779   299    392   53    625   708   735
cmdline       ioports       modules        swaps
1120   16     1886   1942   2004   279    300    4     6     629   709   741
consoles      irq           mounts         sys
1122   1729   1889   1944   2031   2796   3114   46    607   630   710   744
cpuinfo       kallsyms      mpt            sysrq-trigger
1130   1730   1892   1948   21     28     3262   476   608   66    712   790
crypto        kcore         mtrr           sysvipc
1136   1790   1894   1953   22     280    3316   477   609   673   714   8
devices       keys          net            timer_list
1139   1794   19     1959   23     281    3386   478   610   674   716   800
diskstats     key-users     pagetypeinfo   timer_stats
1140   18     1906   1962   24     282    3389   48    611   676   718   832
dma           kmsg          partitions     tty
1141   1800   1916   1965   25     286    35     49    612   678   720   865
driver        kpagecount    sched_debug    uptime
12     1801   1920   1972   26     287    36     5     613   680   725   9
execdomains   kpageflags    schedstat      version
13     1807   1926   1978   2603   29     37     50    614   7     729   acpi
fb            loadavg       scsi           vmallocinfo
```

- Other files and sub-directories contain configuration data for system components. If you wish to view configuration for a particular item, such as the CPU or memory, cat the contents of cpuinfo and meminfo files as shown below:

```
[root@lp2 ~]# cat /proc/cpuinfo
processor        : 0
vendor_id        : GenuineIntel
cpu family       : 6
model            : 142
model name       : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
stepping         : 9
microcode        : 0xb4
cpu MHz          : 2712.006
cache size       : 3072 KB
physical id      : 0
...
processor        : 1
vendor_id        : GenuineIntel
cpu family       : 6
model            : 142
model name       : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
stepping         : 9
microcode        : 0xb4
cpu MHz          : 2712.006
cache size       : 3072 KB
physical id      : 2
...
...
[root@lp2 ~]# cat /proc/meminfo
MemTotal:         3861308 kB
MemFree:          3023192 kB
MemAvailable:     3177028 kB
Buffers:            30124 kB
Cached:            324548 kB
SwapCached:             0 kB
Active:            340732 kB
Inactive:          281980 kB
...
```

The data stored under /proc is referenced by a number of system utilities, including top, ps, uname, and vmstat, to display information.

2.2.3. The /lib/modules Directory

This directory holds information about kernel modules. Underneath it are located sub-directories specific to the kernels installed on the system. For example, listing the content of /lib/modules below shows that there is only one kernel on this system:

```
[root@lp2 ~]# ll -h /lib/modules
total 8.0K
drwxr-xr-x. 7 root root 4.0K Nov 17 23:50 3.10.0-1127.19.1.el7.x86_64
drwxr-xr-x. 7 root root 4.0K Oct 27 19:22 3.10.0-1127.el7.x86_64
[root@lp2 ~]#
```

Now listing the content of the kernel sub-directory will show:

```
[root@lp2 ~]# ll -h /lib/modules/3.10.0-1127.19.1.el7.x86_64/
total 3.3M
lrwxrwxrwx.     1   root   root        44   Nov   17   23:48   build   ->
/usr/src/kernels/3.10.0-1127.19.1.el7.x86_64
drwxr-xr-x.  2 root root 4.0K Aug 25 20:27 extra
drwxr-xr-x. 12 root root 4.0K Nov 17 23:48 kernel
-rw-r--r--.  1 root root 840K Nov 17 23:50 modules.alias
-rw-r--r--.  1 root root 801K Nov 17 23:50 modules.alias.bin
-rw-r--r--.  1 root root 1.4K Aug 25 20:28 modules.block
-rw-r--r--.  1 root root 7.3K Aug 25 20:27 modules.builtin
-rw-r--r--.  1 root root 9.3K Nov 17 23:50 modules.builtin.bin
-rw-r--r--.  1 root root 267K Nov 17 23:50 modules.dep
-rw-r--r--.  1 root root 374K Nov 17 23:50 modules.dep.bin
-rw-r--r--.  1 root root  361 Nov 17 23:50 modules.devname
-rw-r--r--.  1 root root  140 Aug 25 20:28 modules.drm
-rw-r--r--.  1 root root   69 Aug 25 20:28 modules.modesetting
-rw-r--r--.  1 root root 1.8K Aug 25 20:28 modules.networking
-rw-r--r--.  1 root root  96K Aug 25 20:27 modules.order
-rw-r--r--.  1 root root  569 Nov 17 23:50 modules.softdep
-rw-r--r--.  1 root root 392K Nov 17 23:50 modules.symbols
-rw-r--r--.  1 root root 482K Nov 17 23:50 modules.symbols.bin
lrwxrwxrwx.  1 root root    5 Nov 17 23:48 source -> build
drwxr-xr-x.  2 root root 4.0K Aug 25 20:27 updates
drwxr-xr-x.  2 root root 4.0K Nov 17 23:48 vdso
drwxr-xr-x.  3 root root 4.0K Nov 17 23:49 weak-updates
[root@lp2 ~]#
```

There are several files and a few sub-directories here those hold module-specific information. One of the key sub-directories is lib/modules/3.10.0 –1062.4.3.el7.x86_64/kernel/drivers, which stores modules categorized in groups in various sub-directories as shown in the listing below:

```
[root@lp2          ~]#          ls          /lib/modules/3.10.0-
1127.19.1.el7.x86_64/kernel/drivers/
acpi         bluetooth  dax        gpio        i2c          isdn
mfd   nvdimm   platform  rtc       tty    video
ata          cdrom      dca        gpu         idle         leds
misc  nvme     power     scsi      uio    virtio
auxdisplay  char        dma        hid          iio          md
mmc   parport  powercap  ssb       usb    watchdog
base         cpufreq    edac       hv          infiniband  media
mtd   pci      pps       staging   uwb    xen
bcma        cpuidle    firewire  hwmon      input        memstick
net   pcmcia   ptp       target    vfio
block       crypto     firmware  hwtracing  iommu        message
ntb   pinctrl  pwm       thermal   vhost
[root@lp2 ~]#
```

Several module categories exist, such as ata, bluetooth, cdrom, firewire, input, net, pci, scsi, usb, and video. These categories contain modules to control the hardware components associated with them.

3. Managing the kernel

Managing the kernel involves performing several tasks, such as installing and updating the kernel, and listing, displaying, loading, unloading, and installing modules. It also includes the task of adding and removing modules to the initial ram disk; however, this is not usually done manually. This task is accomplished automatically when a new kernel is rebuilt.

- The tools that are used to install and update the kernel are the yum and rpm commands
- While those for managing modules are lsmod, modinfo, modprobe, and depmod. The module management tools are part of the kmod package, and they automatically take into account any dependent modules during their execution

3.1. Installing and updating

Unlike handling other package installs and upgrades, installing and updating kernel packages require extra care as you might end up leaving your system in an unbootable state. The new kernel addresses existing issues as well as adds bug fixes, security updates, new features, and support for additional hardware devices.

When you upgrade the Linux kernel, a new version is installed and used as the default one. The old version of the kernel file will still be available, though. This ensures that your computer can still boot if in the new kernel nonsupported functionality is included. To install a new version of the kernel, you can use the command yum upgrade kernel or yum install kernel command where both commands install the new kernel besides the old one.

The kernel files for the last four kernels that you have installed on your server will be kept in the /boot directory. The GRUB2 boot loader automatically picks up all kernels that it finds in this directory.

Exercise: Install a new Kernel

1. Run the uname command and check the version of the running kernel
2. Run the yum command to install the latest available kernel from the subscription management service using either the update or the install subcommand
3. Confirm that the kernel package has been updated
4. The /boot/grub2/grub.cfg file gets the newest kernel
5. Reboot the system and you will see the new kernel menu entry shows up in the GRUB boot list. The system will boot with this kernel as the install process has marked it as the default kernel
6. Run the uname command again after the reboot to confirm the loading of the new kernel

3.2. Working with kernel modules

In the old days of Linux, kernels had to be compiled to include all drivers that were required to support computer hardware. Other specific functionality needed to be compiled into the kernel as well. Since the release of Linux kernel 2.0 in the mid–1990s, kernels are no longer compiled but modular. A modular kernel consists of a relatively small core kernel and provides driver support through modules that are loaded when required. Modular kernels are very efficient, as only those modules that are really needed are included.

Linux kernel modules normally are loaded automatically for the devices that need them, but you will sometimes have to load the appropriate kernel modules manually. A few commands are used for manual management of kernel modules. The following table provides an overview.

An alternative method of loading kernel modules is by doing this through the /etc/modules-load.d directory. In this directory, you can create files to load modules automatically that are not loaded by the udev method already.

Command	Use
Lsmod	Lists currently loaded kernel modules.
Modinfo	Displays information about kernel modules.
Modprobe	Loads kernel modules, including all of their dependencies.
modprobe –r	Unloads kernel modules, considering kernel module dependencies.

The first command to use when working with kernel modules is lsmod, it lists all kernel modules that are currently used, including the modules by which this specific module is used. The command lsmod | head shows the output of the first 10 lines of the lsmod command.

```
[root@lp2 ~]# lsmod | head
Module                  Size  Used by
xt_CHECKSUM            12549  1
ipt_MASQUERADE         12678  3
nf_nat_masquerade_ipv4    13463  1 ipt_MASQUERADE
tun                    36164  1
devlink                60067  0
ip6t_rpfilter          12595  1
ip6t_REJECT            12625  2
nf_reject_ipv6         13717  1 ip6t_REJECT
ipt_REJECT             12541  4
[root@lp2 ~]#
```

If you want to have more information about a specific kernel module, you can use the modinfo command. This gives complete information about the specific kernel modules, including two interesting sections: the alias and the parms. A module alias is another name that can also be used to address the module. The following shows partial output of the modinfo e1000 command.

```
[root@lp2 ~]# modinfo xt_CHECKSUM
filename:                                    /lib/modules/3.10.0-1127.19.1.el7.x86_64/kernel/net/netfilter/xt_CHECKSUM.ko.xz
alias:          ip6t_CHECKSUM
alias:          ipt_CHECKSUM
description:    Xtables: checksum modification
author:         Michael S. Tsirkin <mst@redhat.com>
license:        GPL
retpoline:      Y
rhelversion:    7.8
srcversion:     188D3664B303A3171F4928C
depends:
intree:         Y
vermagic:            3.10.0-1127.19.1.el7.x86_64  SMP  mod_unload modversions
signer:         CentOS Linux kernel signing key
sig_key:
B1:6A:91:CA:C9:D6:51:46:4A:CB:7A:D9:B8:DE:D5:57:CF:1A:CA:27
sig_hashalgo:   sha256
[root@lp2 ~]#
```

To manually load and unload modules, you can use modprobe and modprobe -r commands. On earlier Linux versions, you may have used the insmod and rmmod commands. These should be used no longer because they do not consider kernel module dependencies.

Exercise: Managing kernel modules from the command line

1. Open a root shell and type lsmod | head. This shows all kernel modules currently loaded.
2. Type modprobe ext4 to load the ext4 kernel module. Verify that it is loaded, using the lsmod command again.
3. Type modinfo ext4 to get information about the ext4 kernel module. Notice that it does not have any parameters.
4. Type modprobe –r ext4 to unload the ext4 kernel module again.
5. Type modprobe –r xfs to try to unload the xfs kernel module. Notice that you get an error message as the kernel module is currently in use.

Managing kernel module parameters

You might sometimes want to load kernel modules with specific parameters. If this is the case, you first need to find out which parameter you want to use. If you have found the parameter you want to use, you can load it manually, specifying the name of the parameter followed by the value that you want to assign. To make this an automated procedure, you can create a file in the /etc/modprobe.d directory, where the module is loaded including the parameter you want to be loaded.

In the following Exercise, you see how to do this using the cdrom kernel module.

Exercise: Loading Kernel Modules with Parameters

1. Type lsmod | grep cdrom. If you have used the optical drive in your computer, this module should be loaded, and it should also indicate that it is used by the sr_mod module.

2. Type modprobe –r cdrom. This will not work because the module is in use by the sr_mod module.

3. Type modprobe –r sr_mod, modprobe –r cdrom. This will unload both modules.

4. Type modinfo cdrom. This will show information about the cdrom module including the parameters that it supports. One of these is the debug parameter, which supports a Boolean as its value.

5. Now use the command modprobe cdrom debug=1. This will load the cdrom module with the debug parameter set to on.

6. Type dmesg. For some kernel module, load information is written to the kernel ring buffer which can be displayed using the dmesg command. Unfortunately, this is not the case for the cdrom kernel module.

7. Create a file with the name /etc/modprobe. d/cdrom and give it the following contents: options cdrom debug=1, This will enable the parameter every time the cdrom kernel module will be loaded.

3.3. The use of kernel threads and drivers

The operating system tasks that are performed by the kernel are implemented by different kernel threads. Kernel threads are easily recognized with a command like ps aux. The kernel thread names are listed between square brackets.

Another important task of the Linux kernel is hardware initialization. To make sure that this hardware can be used, the Linux kernel uses drivers. Every piece of hardware contains specific features, and to use these features, a driver must be loaded. The Linux kernel is modular, and drivers are loaded as kernel modules.

3.4. Analyzing what the kernel is doing

To help analyze what the kernel is doing, some tools are provided by the Linux operating systems dmesg utility, /proc file system and the uname utility.

3.4.1. The dmesg utility

The first utility to consider whether detailed information about the kernel activity is required is dmesg. This utility shows the contents of the kernel ring buffer, an area of memory where the Linux kernel keeps its recent log messages.

In the dmesg output, all kernel-related messages are shown. Each message starts with a time indicator that shows at which specific second the event was logged. This time indicator is relative to the start of the kernel, which allows you to see exactly how many seconds have passed between the start of the kernel and a particular event. (Notice that the journalctl -k / --dmesg commands show clock time, instead of time that is relative to the start of the kernel.) This time indicator gives a clear indication of what has been happening and at which time it has happened.

3.4.2. The /proc file system

Another valuable source of information is the /proc file system. The /proc file system is an interface to the Linux kernel, and it contains files with detailed actual status information on what is happening on your server. Many of the performance related tools mine the /proc file system for more information.

As an administrator, you will find that some of the files in /proc are very readable and contain actual status information about CPU, memory, mounts, and more.

Take a look, for instance, at /proc/meminfo, which gives detailed information about each memory segment and what exactly is happening in these memory segments.

3.4.3. The uname utility

A last useful source of information that should be mentioned here is the uname command. This command gives different kinds of information about your operating system. Type, for instance, uname -a for an overview of all relevant parameters of uname -r to see which kernel version currently is used. This information also shows when using the hostnamectl status command.

Quiz

Chapter review questions

1. The systemd command may be used to rebuild a new kernel.

   a. True

   b. False

2. What is the location of the grub. efi file in the UEFI–based systems?

   a. /boot/efi/EFI/redhat

   b. /boot/grub2/grub. cfg

   c. /bin/efi/EFI/redhat

3. What is the location of the grub. cfg file in the BIOS–based systems?

   a. /boot/efi/EFI/redhat

   b. /boot/grub2/grub.cfg

   c. /bin/efi/EFI/redhat

4. Which file stores the location information of the boot partition on the BIOS systems?

   a. /boot/bios

   b. grub. conf

   c. /proc/partition

5. Which file would we want to view to obtain processor information?

   a. /proc/cpuinfo

   b. /proc/meminfo

   c. /proc/partition

6. Which file would we want to view to obtain memory information?

   a. /proc/cpuinfo

   b. /proc/meminfo

   c. /proc/partition

7. Which command can we use to determine the kernel release information?

   a. Kernel-release

   b. dmesg

   c. uname -r

8. The lsmod command is used to rebuild modules.

   a. True

   b. False

9. Which command can we use to unload a module?

   a. modprobe

   b. lsmod

   c. modinfo

10. We cannot use the yum command to upgrade a Linux kernel. True or False?

    a. True

    b. False

11. Which configuration file should you modify to apply common changes to GRUB2?

a. /boot/grub2/grub.cfg

b. /etc/grub cfg

c. /grub2/grub.config

12. After applying changes to GRUB 2 configuration, which command should you run?

a. modprobe

b. mkconfig

c. grub2-mkconfig

13. Which command shows a list of kernel modules that are currently loaded?

a. lsmod

b. modprobe

c. modinfo

14. How do you install a new version of the kernel?

a. yum update

b. yum update kernel

c. yum kernel install

15. Which command enables you to discover kernel module parameters?

a. lsmod

b. modinfo

c. modprob

16. How do you unload a kernel module?

a. modprobe –r

b. lsmod –r

c. modinfo

17. It contains drivers that are needed to start your server

a. initramfs

b. grub2

c. systemd

18. Setting the GRUB_TIMEOUT parameter to 10, will?

a. Restart the system after 10 seconds.

b. Boot the system into the default kernel after 10 seconds.

c. Shutdown the system after 10 seconds.

19. You can apply permanent fixes to the GRUB 2 configuration using the option:

a. GRUB_CMDLINE_LINUX

b. GRUB_TIMEOUT

c. GRUB_RHGB

20. To list all installed kernels use:

a. lsmod kernel–*

b. yum display kernels

c. yum installed kernel–*

Answers to chapter Review Questions

1. b
2. a
3. b
4. b
5. a
6. b
7. c
8. b
9. a
10. b
11. a
12. c
13. a
14. b
15. b
16. a
17. a
18. b
19. a
20. c