

LPI exam 102 prep, Topic 106: Boot, initialization, shutdown, and runlevels

Junior Level Administration (LPIC-1) topic 106

Skill Level: Intermediate

Ian Shields (ishields@us.ibm.com)
Senior Programmer
IBM

04 Apr 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this second in a [series of nine tutorials](#), Ian introduces you to startup and shutdown on Linux®. By the end of this tutorial, you will know guide a system through booting, set kernel parameters, and shut down or reboot a system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The [Linux Professional Institute](#) (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics

LPI exam 102 topic	developerWorks tutorial	Tutorial summary
--------------------	-------------------------	------------------

Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	(This tutorial). Learn how to boot a system, set kernel parameters, and shut down or reboot a system. See detailed objectives below.
Topic 107	LPI exam 102 prep: Printing	Coming soon.
Topic 108	LPI exam 102 prep: Documentation	Coming soon.
Topic 109	LPI exam 102 prep: Shells, scripting, programming and compiling	Coming soon.
Topic 111	LPI exam 102 prep: Administrative tasks	Coming soon.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the [developerWorks tutorials for LPI exams 101 and 102](#), as well as the [entire set of developerWorks LPI tutorials](#).

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Boot, initialization, shutdown, and runlevels," the second of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to boot a system, set kernel parameters, and shut down or reboot a system.

is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Boot, initialization, shutdown, and runlevels: Exam objectives covered in this tutorial		
LPI exam objective	Objective weight	Objective summary
1.106.1 Boot the system	Weight 3	Guide the system through the booting process, including giving commands to the boot loader and setting kernel options at boot time. Learn how to check boot events in log files.
1.106.2 Change runlevels, and shut down or reboot system	Weight 3	Manage the runlevel of the system, and set the default runlevel, plus change to single-user mode, shut down and reboot the system. Learn how to alert users before switching runlevel, and how to properly terminate processes.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the [tutorials for exam 101](#). In particular, you should be very familiar with the material from the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. Boot the system

This section covers material for topic 1.106.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Guide the system through the booting process
- Give commands to the boot loader at boot time
- Give options to the kernel at boot time
- Check boot events in the log files

Boot overview

To summarize the boot process for PCs:

1. When a PC is turned on, the BIOS (*Basic Input Output Service*) performs a self test.
2. When the machine passes its self test, the BIOS loads the *Master Boot Record* (or *MBR*, usually from the first 512-byte sector of the boot drive). This is usually the first hard drive on the system, but may also be a diskette, CD, or USB key.
3. For a hard drive, the MBR loads a stage 1 boot loader, which is typically either the LILO or GRUB stage1 boot loader on a Linux system. This is another 512-byte, single-sector record.
4. The stage 1 boot loader usually loads a sequence of records called the stage 2 boot loader (or sometimes the stage 1.5 loader).
5. The stage 2 loader loads the operating system. For Linux, this is the kernel, and possibly an initial RAM disk (initrd).

At this point, your system should be able to install either of two popular boot loaders, LILO (the Linux LOader) or GRUB (the GRand Unified Boot loader). You should be able to use your chosen boot loader to boot normally as described above. Refer back to the "[LPI exam 101 prep \(topic 102\): Linux installation and package management](#)" tutorial if you need to review boot loader installation or basic booting.

To influence your system's boot process, you can:

1. Change the device from which you boot. You may normally boot from a hard drive, but you may sometimes need to boot from a floppy disk, a USB memory key, a CD or DVD, or a network. Setting up such alternate boot devices requires your BIOS to be appropriately configured. The method for doing this is specific to your system and its BIOS. This is beyond the scope of this tutorial or the requirements of this LPI objective, so consult your system documentation.
2. You can interact with the boot loader to select which of several possible configurations you may wish to boot. You will learn how to do this for both LILO and GRUB in this tutorial.
3. You can use GRUB or LILO to pass parameters to the kernel to control the way that your kernel starts the system once it has been loaded by the boot loader.

LILO

The LILO configuration file defaults to `/etc/lilo.conf`. Listing 1 shows an example from a system that is currently running Red Hat Enterprise Linux 3. The system has a Red Hat 9 partition whose root filesystem is mounted on `/mnt/hda7` and a Windows® XP system on `/dev/hda1`.

Listing 1. Sample LILO configuration

```
[root@lyrebird root]# cat /etc/lilo.conf
prompt
timeout=50
compact
default=latest-EL
boot=/dev/fd0
map=/boot/map
install=/boot/boot.b
message=/boot/message2
lba32
password=mypassword
restricted

image=/mnt/hda7/boot/vmlinuz-2.4.20-31.9
    label=redhat9
    alias=shrike
    initrd=/mnt/hda7/boot/initrd-2.4.20-31.9.img
    read-only
    append="hdd=ide-scsi root=LABEL=RH9"

image=/boot/vmlinuz-2.4.21-40.EL
    label=2.4.21-40.EL
    alias=latest-EL
    initrd=/boot/initrd-2.4.21-40.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.0.1.EL
    label=2.4.21-37a.EL
    initrd=/boot/initrd-2.4.21-37.0.1.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-37.EL
    label=2.4.21-37.EL
    initrd=/boot/initrd-2.4.21-37.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

image=/boot/vmlinuz-2.4.21-32.0.1.EL
    label=2.4.21-32.EL
    alias=early
    initrd=/boot/initrd-2.4.21-32.0.1.EL.img
    read-only
    append="hdd=ide-scsi root=LABEL=RHEL3"

other=/dev/hda1
    loader=/boot/chain.b
    label=WIN-XP
    alias=xp
```

Remember that whenever you make changes to `/etc/lilo.conf`, or whenever you install a new kernel, you **must** run `lilo`. The `lilo` program rewrites the MBR or the partition boot record to reflect your changes, including recording the absolute disk location of the kernel. If your configuration file includes Linux images from multiple partitions, you must mount the partitions because the `lilo` command

needs to access the partition to locate the image.

The `message` parameter in the LILO configuration file may refer to a text file or a specially created file in PCX format. The Red Hat Enterprise Linux distribution includes a graphical `/boot/message` file that shows a Red Hat logo. For the purposes of illustration, the configuration in Listing 1 uses a text file that is shown in Listing 2. Customizing graphical LILO messages is beyond the scope of this tutorial. Be warned that it is also not very well documented.

Listing 2. LILO text boot message

```
[root@lyrebird root]# cat /boot/message2
Booting lyrebird
```

If your configuration file does not include the `message` parameter, then you will see a very terse prompt, **LILO boot:**. Otherwise you will see either a text prompt or a graphical background and a menu. You may need to hold down the Shift key during boot to see the prompt, as a system may be configured so that it bypasses the prompt.

If you have the default prompt, or a custom text prompt, you may press the Tab key to display a list of available images to boot. You may either type the name of an image as shown in Listing 3, or press **Enter** to select the first entry. If you have a graphical menu, use the cursor movement keys to highlight the entry you wish to boot.

Listing 3. Sample LILO prompt

```
LILO

Booting lyrebird

boot:
latest-EL      shrike      redhat9      2.4.21-40.EL
2.4.21-37a.EL  2.4.21-37.EL  early       2.4.21-32.EL
xp             WIN-XP
boot: latest-EL
```

In addition to displaying the LILO configuration file, you may use the `-q` option of the `lilo` command to display information about the LILO boot choices. Add `-v` options for more verbose output. Two examples using the configuration file of Listing 1 are shown in Listing 4.

Listing 4. Displaying LILO configuration

```
[root@lyrebird root]# lilo -q
latest-EL      *
shrike
redhat9
2.4.21-40.EL
2.4.21-37a.EL
2.4.21-37.EL
early
2.4.21-32.EL
xp
WIN-XP
[root@lyrebird root]# lilo -q -v | tail +22 | head -n 9
shrike
Password is required for specifying options
Boot command-line won't be locked
```

```
No single-key activation
VGA mode is taken from boot image
Kernel is loaded "high", at 0x00100000
Initial RAM disk is 149789 bytes
No fallback
Options: "ro BOOT_FILE=/mnt/hda7/boot/vmlinuz-2.4.20-31.9 hdd=ide-scsi root=LABEL=RH9 "
```

GRUB

The GRUB configuration file defaults to `/boot/grub/grub.conf` or `/boot/grub/menu.lst`. If both are present, one will usually be a symbolic link to the other. Listing 5 shows an example from the same system that you saw above for LILO, although only a few of the entries are illustrated here.

Listing 5. Sample GRUB configuration

```
default=1
timeout=10
splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz
foreground=23334c
background=82a6bc
password --md5 $1$H8LlM1$cI0Lfs5.C06xFJYPQ8Ixz/
title Red Hat Linux (2.4.20-31.9)
    root (hd0,6)
    kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
    initrd /boot/initrd-2.4.20-31.9.img
    savedefault
    boot

title Red Hat Enterprise Linux WS A (2.4.21-40.EL)
    root (hd0,10)
    kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi
    initrd /boot/initrd-2.4.21-40.EL.img

title Win/XP
    rootnoverify (hd0,0)
    chainloader +1
```

GRUB provides a menu interface instead of LILO's prompt. It can also use a password encrypted with the MD5 algorithm as opposed to the plain text password of LILO. And, perhaps most importantly, changes made to the GRUB configuration file do not require GRUB to be reinstalled in the MBR. Note that many distributions will automatically update the GRUB (or LILO) configuration file when updating to a new kernel level, but if you install a new kernel yourself or create a new initial RAM disk, you may need to edit the configuration file.

GRUB also does not require a partition to be mounted in order to configure a boot entry for it. You will notice entries such as `root (hd0,6)` and `splashimage=(hd0,2)/boot/grub/fig1x.xpm.gz`. GRUB refers to your hard disks as `hdn`, where *n* is an integer starting from 0. Similarly, partitions on a disk are similarly numbered starting from 0.

So, on this system, `(hd0,2)` represents the primary partition `/dev/hda3`, while `(hd0,6)` represents the logical partition `/dev/hda7`. A floppy drive is usually `(fd0)`. Remember to quote these if you are invoking GRUB with parameters from a bash shell, for example, when installing GRUB onto a floppy or your MBR.

If you want a different background image for GRUB, you are limited to 14 colors. Your favorite JPEG image may look a little different when reduced to 14 colors. You

can see the effect in Figure 1, which shows the image from the above configuration file using a photo I took in Glacier Bay, Alaska. You may also want to pick suitable foreground and background colors for your textual prompts from the colors in the image; Figure 2 illustrates custom foreground and background colors.

Figure 1. Photo reduced to 14 colors for a GRUB splash image



Figure 2. Text and text background colors chosen from photo colors

Red Hat Enterprise Linux WS A (2.4.21-40.EL)

When the GRUB menu is displayed, you select a boot image using the cursor movement keys to move up and down the list.

Unlike LILO, GRUB behaves as a small shell, with several commands that allow you to do things such as edit the commands before they are executed or do things such as find and load a configuration file, or display files using a `cat` command. From the menu, you may press **e** on an entry to edit it, **c** to switch to a GRUB command line, **b** to boot the system, **p** to enter a password, and **Esc** to return to the menu or to the previous step. There is also a `grub` command, which creates a simulated shell in which you may test your GRUB configuration or your GRUB command skills. Within the GRUB shell, the `help` command provides a list of commands. Using `help commandname` provides help for the command named *commandname*. Listing 6 illustrates the help and the available commands.

Listing 6. Using the GRUB shell

```
[root@lyrebird root]# grub
Probing devices to guess BIOS drives. This may take a long time.
find FILENAME                geometry DRIVE [CYLINDER HEAD SECTOR [
halt [--no-apm]              help [--all] [PATTERN ...]
hide PARTITION               initrd FILE [ARG ...]
kernel [--no-mem-option] [--type=TYPE] makeactive
map TO_DRIVE FROM_DRIVE      md5crypt
module FILE [ARG ...]        modulenounzip FILE [ARG ...]
pager [FLAG]                 partnew PART TYPE START LEN
parttype PART TYPE           quit
reboot                       root [DEVICE [HDBIAS]]
rootnoverify [DEVICE [HDBIAS]] serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY]     setup [--prefix=DIR] [--stage2=STAGE2_
terminal [--dumb] [--no-echo] [--no-ed terminfo [--name=NAME --cursor-address
testvbe MODE                 unhide PARTITION
uppermem KBYTES              vbeprobe [MODE]

grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
```


Similar to `root`, but don't attempt to mount the partition. This is useful for when an OS is outside of the area of the disk that GRUB can read, but setting the correct root device is still desired. Note that the items mentioned in `root` which derived from attempting the mount will NOT work correctly.

```
grub>
```

As a practical example, you might continue with the previous example and use GRUB's `find` command to find configuration files. Next, you might load the configuration file from `(hd0,2)` which is `/dev/hda3`, as illustrated in Listing 7.

Listing 7. Using GRUB to find and load a GRUB configuration file

```
grub> find /boot/grub/menu.lst
(hd0,2)
(hd0,6)
(hd0,7)
(hd0,8)
(hd0,9)
(hd0,10)

grub> configfile (hd0,2)/boot/grub/menu.lst
```

When you load the configuration file, you might see a menu similar to that in Listing 8. Remember that this was done under the GRUB shell, which simulates the real GRUB environment and does not display the splash image. However, this is essentially the same as you would see superimposed on your splash image when you really boot the system using GRUB.

Listing 8. The GRUB menu

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| Red Hat Linux (2.4.20-31.9)                                     |
| Red Hat Linux (2.4.20-6)                                       |
| Red Hat Enterprise Linux WS A (2.4.21-40.EL)                  |
| Red Hat Enterprise Linux WS A (2.4.21-37.0.1.EL)              |
| Red Hat Enterprise Linux WS A (2.4.21-37.EL)                  |
| Red Hat Enterprise Linux WS A (2.4.21-32.0.1.EL)              |
| Red Hat Enterprise Linux WS A (2.4.21-27.0.4.EL)              |
| Red Hat Enterprise Linux WS A (2.4.21-27.0.2.EL)              |
| Red Hat Enterprise Linux WS A (2.4.21-27.0.1.EL)              |
| Red Hat Enterprise Linux WS A (2.4.21-20.EL)                  |
| Red Hat Enterprise Linux WS (2.4.21-27.0.1.EL)                |
| Red Hat Enterprise Linux WS (2.4.21-15.0.2.EL)                |
+-----+ v

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS or 'p' to enter a
password to unlock the next set of features.
```

The highlighted entry will be booted automatically in 5 seconds.

Suppose you highlighted the third entry for Red Hat Enterprise Linux WS A (2.4.21-40.EL) and the pressed `e` to edit it. You would see something similar to Listing 9.

Listing 9. Editing a GRUB configuration entry

```
GRUB version 0.93 (640K lower / 3072K upper memory)

+-----+
| root (hd0,10)                                                  |
| kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi |
| initrd /boot/initrd-2.4.21-40.EL.img                          |
+-----+
```

```
+-----+
|
| Use the ^ and v keys to select which entry is highlighted.
| Press 'b' to boot, 'e' to edit the selected command in the
| boot sequence, 'c' for a command-line, 'o' to open a new line
| after ('O' for before) the selected line, 'd' to remove the
| selected line, or escape to go back to the main menu.
|
+-----+
```

Again, you use the arrow keys to select the line to be edited, and then press **e** to edit it. For example, if you had deleted the partition `/dev/hda5`, then your root partition should now be `/dev/hda10` or `(hd0,9)` instead of `/dev/hda11` or `(hd0,10)`. Back up and edit the value. When done, press **Enter** to accept your change, or press **Esc** to cancel. Finally, press **b** to boot the system.

GRUB has enough capability to display files on your filesystem, and it is run as if it were the root user, so you really need to protect your system with GRUB passwords. Remember, however, that if a user can boot from removable media, that user can provide his or her own GRUB configuration. See the security section in the GRUB manual (see [Resources](#)) for more information on GRUB security and other aspects of GRUB. You may also view it on your system using the `info grub` command.

Kernel parameters

Kernel parameters (sometimes called boot parameters) supply the kernel with information about hardware parameters that it might not determine on its own, to override values that it might otherwise detect or to avoid detection of inappropriate values. For example, you might want to boot an SMP system in uniprocessor mode, or you may want to specify an alternate root filesystem. Some kernel levels require a parameter to enable large memory support on systems with more than a certain amount of RAM.

If you use LILO, you specify additional (or overriding) parameters after you type the name of the kernel to be booted. For example, if you had just built a new kernel called `/boot/vmlinuz-2.4.21-40.EL-prep`, you might enter a command to tell LILO to use it, as shown in Listing 10.

Listing 10. Specifying boot parameters with LILO

```
boot: latest-EL image=/boot/vmlinuz-2.4.21-40.EL-prep
```

With GRUB, you could type in another set of commands for the kernel and `initrd` statements, or, preferably, you could use the edit facility that you just learned about to edit an existing entry by adding `-prep` to the end of the existing kernel image name.

When the kernel finishes loading, it usually starts `/sbin/init`. This program remains running until the system is shut down. It is always assigned process ID 1, as you can see in Listing 11.

Listing 11. The init process

```
[root@lyrebird root]# ps --pid 1
  PID TTY          TIME CMD
    1  ?        0:00 s
```

```
1 ? 00:00:04 init
```

The `init` program boots the rest of your system by running a series of scripts. These scripts typically live in `/etc/rc.d/init.d` or `/etc/init.d`, and they perform services such as setting the system's hostname, checking the filesystem for errors, mounting additional filesystems, enabling networking, starting print services, and so on. When the scripts complete, `init` starts a program called `getty`, which displays the login prompt on consoles. Graphical login screens are handled differently as you learned in the tutorial "[LPI exam 102 prep, topic 105: Kernel](#)."

If your system will load a kernel, but cannot run `init` successfully, you may try to recover by specifying an alternate initialization program. For example, specifying `init=/bin/sh` will boot your system into a shell prompt with root authority, from which you might be able to repair the system.

You can find out more about the available boot parameters using the man pages for `bootparam`, or by browsing `/usr/src/linux/Documentation/ramdisk.txt`, which may be called `/usr/src/linux-$(uname -r)/Documentation/kernel-parameters.txt` on some systems.

Needless to say, if you have to apply the same set of additional parameters every time you boot, you should add them to the configuration file. Remember to rerun `lilo` if you are using LILO.

Boot events

During the Linux boot process, a large number of messages are emitted to the console, describing the kernel being booted, the hardware of your system, and other things related to the kernel. These messages usually flash by quickly and you probably won't be able to read them, unless there is a delay while the boot process waits for something, such as inability to reach a time server, or a filesystem that must be checked. With the advent of the Linux Bootsplash project (see [Resources](#)), these messages may be superimposed on a graphical background, or they may be hidden and replaced by a simple status bar. If your distribution supports the hidden mode, you will usually be able to switch back to displaying boot messages by pressing a key such as F2.

dmesg

It's nice to be able to go back and review the kernel messages. Since standard output is related to a process, and the kernel does not have a process identifier, it keeps kernel (and module) output messages in the *kernel ring buffer*. You may display the kernel ring buffer using the `dmesg` command, which displays these messages on standard output. Of course, you may redirect this output to a file for later analysis or forward it to a kernel developer for debugging purposes. Listing 12 illustrates some of the output that you might see.

Listing 12. Partial dmesg output

```
[root@lyrebird root]# dmesg | head -n 30
Linux version 2.4.21-40.EL (bhcompile@hs20-bc1-7.build.redhat.com) (gcc version 3.2.3
```

```

20030502 (Red Hat Linux 3.2.3-54)) #1 Thu Feb 2 22:32:00 EST 2006
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 0000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 000000000000e0000 - 00000000000100000 (reserved)
 BIOS-e820: 00000000000100000 - 000000000005f6f000 (usable)
 BIOS-e820: 000000000005f6f000 - 000000000005f6fb000 (ACPI data)
 BIOS-e820: 000000000005f6fb000 - 000000000005f700000 (ACPI NVS)
 BIOS-e820: 000000000005f700000 - 000000000005f780000 (usable)
 BIOS-e820: 000000000005f780000 - 0000000000060000000 (reserved)
 BIOS-e820: 00000000000fec00000 - 00000000000fec10000 (reserved)
 BIOS-e820: 00000000000fee00000 - 00000000000fee01000 (reserved)
 BIOS-e820: 00000000000ff800000 - 00000000000ffc00000 (reserved)
 BIOS-e820: 00000000000ffffc00 - 00000000000100000000 (reserved)
631MB HIGHMEM available.
896MB LOWMEM available.
NX protection not present; using segment protection
On node 0 totalpages: 391040
zone(0): 4096 pages.
zone(1): 225280 pages.
zone(2): 161664 pages.
IBM machine detected. Enabling interrupts during APM calls.
Kernel command line: ro root=LABEL=RHEL3 hdd=ide-scsi
ide_setup: hdd=ide-scsi
Initializing CPU#0
Detected 2392.059 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 4771.02 BogoMIPS
Page-cache hash table entries: 524288 (order: 9, 2048 KB)
Page-pin hash table entries: 131072 (order: 7, 512 KB)

```

The kernel ring buffer is also used for some events after the system is booted. These include certain program failures and hot-plug events. Listing 13 shows an entry for a program that failed with a segmentation fault and several entries related to plugging in a USB memory key.

Listing 13. Later events in kernel ring buffer

```

[root@attic4 ~]# dmesg |tail -n 19
main[15961]: segfault at 00000000000529000 rip 0000000000403b5d rsp 00007fffffd15d00
error 6
usb 1-4.3: new high speed USB device using ehci_hcd and address 4
scsi5 : SCSI emulation for USB Mass Storage devices
usb-storage: device found at 4
usb-storage: waiting for device to settle before scanning
 Vendor: Sony      Model: Storage Media    Rev: 0100
 Type:   Direct-Access          ANSI SCSI revision: 00
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
SCSI device sdb: 1014784 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 43 00 00 00
sdb: assuming drive cache: write through
sdb: sdb1
sd 5:0:0:0: Attached scsi removable disk sdb
usb-storage: device scan complete
SELinux: initialized (dev sdb1, type vfat), uses genfs_contexts

```

/var/log/messages

Once your system has started to the point of running `/sbin/init`, the kernel still logs events in the ring buffer as you just saw, but processes use the syslog daemon to log messages, usually in `/var/log/messages`. In contrast to the ring buffer, each syslog line has a timestamp, and the file persists between system restarts. This file is where you should first look for errors that occurred during the init scripts stage of booting.

Most daemons have names that end in 'd'. Listing 14 shows how to see the last few daemon status messages after a reboot.

Listing 14. Daemon messages form /var/log/messages

```
[root@lyrebird root]# grep "^Apr.*d\:" /var/log/messages|tail -n 14
Apr  2 15:36:50 lyrebird kernel: hdd: attached ide-scsi driver.
Apr  2 15:36:52 lyrebird apmd: apmd startup succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Setting hostname lyrebird:  succeeded
Apr  2 15:36:26 lyrebird rc.sysinit: Initializing USB keyboard:  succeeded
Apr  2 15:36:55 lyrebird sshd:  succeeded
Apr  2 15:36:55 lyrebird xinetd: xinetd startup succeeded
Apr  2 15:36:56 lyrebird ntpd:  succeeded
Apr  2 15:36:56 lyrebird ntpd:  succeeded
Apr  2 15:36:56 lyrebird ntpd:  succeeded
Apr  2 15:36:56 lyrebird ntpd: ntpd startup succeeded
Apr  2 15:36:57 lyrebird crond: crond startup succeeded
Apr  2 15:36:58 lyrebird atd: atd startup succeeded
Apr  2 15:36:58 lyrebird snastart: insmod: streams: no module by that name found
Apr  2 15:36:58 lyrebird rhnsd: rhnsd startup succeeded
```

You will also find logs for many other system programs in /var/log. For example, you can see the startup log for your X Window system that you learned about in the tutorial "[LPI exam 101 prep, Topic 110: The X Window System](#) ."

Section 3. Runlevels , shutdown, and reboot

This section covers material for topic 1.106.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Manage the runlevel of the system
- Change to single-user mode
- Set the default runlevel
- Shut down or reboot the system
- Alert users before switching runlevel
- Terminate processes properly

Runlevels

Runlevels define what tasks can be accomplished in the current state (or runlevel) of a Linux system. Every Linux system supports three basic runlevels, plus one or more runlevels for normal operation. The basic runlevels are shown in Table 3.

Table 3. Linux basic runlevels

Level	Purpose
-------	---------

0	Shut down (or halt) the system
1	Single-user mode; usually aliased as s or S
6	Reboot the system

Beyond the basics, runlevel usage differs among distributions. One common usage set is shown in Table 4.

Table 4. Other common Linux runlevels	
Level	Purpose
2	Multi-user mode without networking
3	Multi-user mode with networking
5	Multi-user mode with networking and the X Window System

The Slackware distribution uses runlevel 4 instead of 5 for a full system running the X Window system. Debian uses a single runlevel for any multi-user mode, typically runlevel 2. Be sure to consult the documentation for your distribution.

Default runlevel

When a Linux system starts, the default runlevel is determined from the `id:` entry in `/etc/inittab`. Listing 15 illustrates a typical entry for a system such as Red Hat Enterprise Linux, which uses runlevel 5 for the X Window System.

Listing 15. Default runlevel in `/etc/inittab`

```
[root@lyrebird root]# grep "^id:" /etc/inittab
id:5:initdefault:
```

Changing runlevels

There are several ways to change runlevels. To make a permanent change, you can edit `/etc/inittab` and change the default level that you just saw above.

If you just need to bring the system up in a different runlevel, you have a couple of ways to do this. For example, suppose you just installed a new kernel and need to build some kernel modules after the system booted with the new kernel, but before you start the X Window System. You might want to bring up the system in runlevel 3 to accomplish this. You do this at boot time by editing the kernel line (GRUB) or adding a parameter after the selected system name (LILO). Use a single digit to specify the desired runlevel (3, in this case). For example, you might edit a line from Listing 5 that you saw in the previous section to read as shown in Listing 16.

Listing 16. Setting default runlevel at boot time

```
kernel /boot/vmlinuz-2.4.21-40.EL ro root=LABEL=RHEL3 hdd=ide-scsi 3
```

Once you have finished your setup work in runlevel 3, you might want to switch to runlevel 5. Fortunately, you do not need to reboot the system. You can use the `telinit` command to tell the `init` process what runlevel it should switch to.

You can determine the current runlevel using the `runlevel` command, which shows the previous runlevel as well as the current one. If the first output character is 'N', the runlevel has not been changed since the system was booted. Listing 17 illustrates verifying and changing the runlevel.

Listing 17. Verifying and changing the runlevel

```
[root@lyrebird root]# runlevel
N 3
[root@lyrebird root]# telinit 5
[root@lyrebird root]# runlevel
3 5
```

If you use the `ls` command to display a long listing of the `telinit` command, you will see that it really is a symbolic link to the `init` command. The `init` executable knows which way it was called and behaves accordingly. Since `init` normally runs as PID 1, it is also smart enough to know if you invoke it using `init` rather than `telinit`. If you do, it will assume you want it to behave as if you had called `telinit` instead. For example, you may use `int 5` instead of `telinit 5` to switch to runlevel 5.

Single-user mode

In contrast to personal computer operating systems such as DOS or Windows, Linux is inherently a multiuser system. However, there are times when that can be a problem, such as when you need to recover a major filesystem or database, or install and test some new hardware. Runlevel 1, or *single-user mode*, is your answer for these situations. The actual implementation varies by distribution, but you will usually start in a shell with only a minimal system. Usually there will be no networking and no (or very few) daemons running. On some systems, you must authenticate by logging in, but on others you go straight into a shell prompt as root. Single-user mode can be a lifesaver, but you can also destroy your system, so always be very careful whenever you are running with root authority.

As with switching to regular multiuser runlevels, you can also switch to single-user mode using `telinit 1`. As noted in Table 3, 's' and 'S' are aliases for runlevel 1, so you could, for example, use `telinit s` instead.

Shutdown commands

While you can use `telinit` or `init` to stop multiuser activity and switch to single-user mode, you may also use the `shutdown` command. The `shutdown`

command sends a warning message to all logged on users and blocks further logins. It then signals `init` to switch runlevels. The `init` process then sends all running processes a `SIGTERM` signal, giving them a chance to save data or otherwise properly terminate. After 5 seconds, or another delay if specified, `init` sends a `SIGKILL` signal to forcibly end each remaining process.

By default, `shutdown` switches to runlevel 1 (single-user mode). You may specify the `-h` option to halt the system, or the `-r` option to reboot. A standard message is issued in addition to any message you specify. The time may be specified as an absolute time in `hh:mm` format, or as a relative time, `n`, where `n` is the number of minutes until shutdown. For immediate shutdown, use `now`, which is equivalent to `+0`.

If you have issued a delayed shutdown and the time has not yet expired, you may cancel the shutdown by pressing **Ctrl-c** if the command is running in the foreground, or by issuing `shutdown` with the `-c` option to cancel a pending shutdown. Listing 18 shows several examples of the use of `shutdown`, along with ways to cancel the command.

Listing 18. Shutdown examples

```
[root@lyrebird root]# shutdown 5 File system recovery needed
Broadcast message from root (pts/0) (Mon Apr  3 22:44:29 2006):

File system recovery needed
The system is going DOWN to maintenance mode in 5 minutes!

Shutdown cancelled.
[root@lyrebird root]# shutdown -r 10 Reloading updated kernel&
[1] 5388

Broadcast message from root (pts/0) (Mon Apr  3 22:45:15 2006):

Reloading updated kernel
The system is going DOWN for reboot in 10 minutes!
[root@lyrebird root]# fg
shutdown -r 10 Reloading updated kernel

Shutdown cancelled.
[root@lyrebird root]# shutdown -h 23:59&
[1] 5390
[root@lyrebird root]# shutdown -c

Shutdown cancelled.
[1]+  Done                  shutdown -h 23:59
```

You may have noticed that our last example did not cause a warning message to be sent. If the time till shutdown exceeds 15 minutes, then the message is not sent until 15 minutes before the event as shown in Listing 19. Listing 19 also shows the use of the `-t` option to increase the default delay between `SIGTERM` and `SIGKILL` signals from 5 seconds to 60 seconds.

Listing 19. Another shutdown example

```
[root@lyrebird root]# date;shutdown -t60 17 Time to do backups
Mon Apr  3 22:51:45 EDT 2006

Broadcast message from root (pts/0) (Mon Apr  3 22:53:45 2006):

Time to do backups
The system is going DOWN to maintenance mode in 15 minutes!
```

Listing 20. Rebooting the system

```
[root@lyrebird root]# reboot

Broadcast message from root (pts/0) (Mon Apr  3 22:58:27 2006):

The system is going down for reboot NOW!
```

And sure enough, the system did reboot back to runlevel 3, as is shown by the use of the `runlevel` and `uptime` commands in Listing 21.

Listing 21. Another example of rebooting the system

```
[ian@lyrebird ian]$ /sbin/runlevel
N 3
[ian@lyrebird ian]$ uptime
23:05:51 up 6 min,  1 user,  load average: 0.00, 0.06, 0.03
```

It is also possible to use `telinit` (or `init`) to shut down or reboot the system. As with other uses of `telinit`, no warning is sent to users, and the command takes effect immediately, although there is still a delay between `SIGTERM` and `SIGKILL` signals. For additional options of `telinit`, `init`, and `shutdown`, consult the appropriate man pages.

Halt, reboot, and poweroff

You should know about a few more commands related to shutdown and reboot.

- The `halt` command halts the system.
- The `poweroff` command is a symbolic link to the `halt` command, which halts the system and then attempts to power it off.
- The `reboot` command is a another symbolic link to the `halt` command, which halts the system and then reboots it.

If any of these are called when the system is not in runlevel 0 or 6, then the corresponding `shutdown` command will be invoked instead.

For additional options that you may use with these commands, as well as more detailed information on their operation, consult the man page.

Runlevel configuration

By now, you may be wondering why pressing **Ctrl-Alt-Delete** on some systems causes a reboot, or how all this runlevel stuff is configured. Remember the

field in `/etc/inittab`? Well, there are several other fields in `/etc/inittab`, along with a set of init scripts in directories such as `rc1.d` or `rc5.d`, where the digit identifies the runlevel to which the scripts in that directory apply. Listing 22 shows the entry for **Ctrl-Alt-Delete**, so you see why it causes the system to be rebooted.

Listing 22. Trapping ctrl-alt-delete

```
[root@lyrebird root]# grep -i ctrl /etc/inittab
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

The scripts used by `init` when starting the system, changing runlevels, or shutting down are typically stored in the `/etc/init.d` or `/etc/rc.d/init.d` directory. A series of symbolic links in the `rcn.d` directories, one directory for each runlevel *n*, control whether a script is started when entering a runlevel or stopped when leaving it. These links start with either a K or an S, followed by a two-digit number and then the name of the service, as shown in Listing 23.

Listing 23. Init scripts

```
[root@lyrebird root]# find /etc -path "*rc[0-9]*.d/???au*"
/etc/rc.d/rc0.d/K95audit
/etc/rc.d/rc0.d/K72autofs
/etc/rc.d/rc1.d/K95audit
/etc/rc.d/rc1.d/K72autofs
/etc/rc.d/rc2.d/S20audit
/etc/rc.d/rc2.d/K72autofs
/etc/rc.d/rc3.d/S20audit
/etc/rc.d/rc3.d/S28autofs
/etc/rc.d/rc4.d/K95audit
/etc/rc.d/rc4.d/S28autofs
/etc/rc.d/rc5.d/S20audit
/etc/rc.d/rc5.d/S28autofs
/etc/rc.d/rc6.d/K95audit
/etc/rc.d/rc6.d/K72autofs
[root@lyrebird root]# cd /etc/rc.d/rc5.d
[root@lyrebird rc5.d]# ls -l ???a*
lrwxr-xr-x 1 root root 15 Jan 11 2005 S20audit -> ../init.d/audit
lrwxr-xr-x 1 root root 14 Jan 11 2005 S26apmd -> ../init.d/apmd
lrwxr-xr-x 1 root root 16 Jan 11 2005 S28autofs -> ../init.d/autofs
lrwxr-xr-x 1 root root 13 Jan 11 2005 S95atd -> ../init.d/atd
```

Here you see that the `audit` and `autofs` services have *Knn* entries in all runlevels and *Snn* entries for both in runlevels 3 and 5. The S indicates that the service is started when that runlevel is entered, while the K entry indicates that it should be stopped. The *nn* component of the link name indicates the priority order in which the service should be started or stopped. In this example, `audit` is started before `autofs`, and it is stopped later.

Consult the man pages for `init` and `inittab` for more information.

Resources

Learn

- Review the entire [LPI exam prep tutorial series](#) on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the [LPIC Program](#), find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "[Basic tasks for new Linux developers](#)" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The [Linux Documentation Project](#) has a variety of useful documents, especially its HOWTOs.
- [The Linux System Administrator's Guide](#) introduces novices to Linux system administration.
- "[Boot loader showdown: Getting to know LILO and GRUB](#)" (developerWorks, August 2005) helps you contrast and compare these two contenders.
- The [GRUB Manual](#) has a wealth of information on GRUB.
- The [GRUB Splash Image Howto](#) will help you create your own GRUB splash image.
- Check out the [Bootsplash project](#) for a graphical boot process for the Linux kernel.
- The [LILO Mini-HOWTO](#) shows you how to use the Linux Loader.
- "[Automate OS switching on a dual-boot Linux system](#)" (developerWorks, March 2006) shows you how to switch between Linux and Windows on the same machine without manual intervention.
- [LPI Linux Certification in a Nutshell](#) (O'Reilly, 2001) and [LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 \(Exam Cram 2\)](#) (Que, 2004) are references for readers who prefer book format.
- Find more [tutorials for Linux developers](#) in the [developerWorks Linux zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Download [IBM trial software](#) directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Read [developerWorks blogs](#), and get involved in the developerWorks

community.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.

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

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

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