Boot the System

Information

These notes were originally written in the year 2000 as part of a set of LPI Exam 101 training materials. The LPI training course at Bromley College was subsequently discontinued and some of the sections of the notes modified and incorporated into our one-day System Administration Courses. The remainder of the notes have now been made publicly available on the linuxtraining.org.uk website.

If you are a beginner please do not be put off of training courses by these notes, as they are rather technical. On the other hand if you are a more experienced Linux user we hope you find the coverage of this topic refreshingly clear.

For full details of our current Linux training please visit the site:
http://ce.bromley.ac.uk/linux

If you have reached this page from a search engine and wish to see the full contents list for the published notes please visit the site:
http://www.linuxtraining.org.uk

We hope you find these notes useful, but please remember that they apply to the 2.2 kernel. I will update them when I have the time.

Clive Gould - 21st December 2004

Boot the System

Objective 1

Boot the system: Guide the system through the booting process, including giving options to the kernel at boot time, and check the events in the log files. Involves using the commands: dmesg (lilo). Involves reviewing the files: /var/log/messages, /etc/lilo.conf, /etc/conf.modules, /etc/modules.conf

The Boot Process

You can either boot Linux using a boot loader located on either a hard disk or a floppy disk. The process followed when Linux boots from hard disk is as follows:

1. Power on self test (POST) ROM bios routines
2. System boot
3. The kernel is uncompressed and loaded into memory
4. The root filesystem is mounted
5. The init daemon is started.

Install Boot Loader - lilo

System boot is achieved using a boot loader. The **lilo** command installs a boot loader that will be activated next time you boot. LILO provides an efficient method of booting Linux as well as other operating systems, such as DOS, UNIX, OS/2, Win 9x and NT. LILO can handle up to sixteen different boot images on different partitions, each of which can be password protected.

The recommended place to install LILO is the Master Boot Record (MBR), unless the MBR already starts another boot loader such as System Commander or OS/2's boot manager, in which case you will need to install LILO on the first sector of the root partition, and configure the existing boot loader on the MBR to start LILO as required.
LILO allows the boot sector file, map file and boot images to reside on different partitions. LILO can be installed using the **/sbin/lilo** command without any options.

Boot the System

As LILO is loaded it displays the word LILO, one character at a time. Only if the complete word is displayed is LILO fully loaded. The LILO prompt error table below explains this further:

Prompt	*Description*
L<nn>	Where nn represents one of 16 disk error codes. This is usually a media failure
LI	The second stage boot loader loaded but couldn't run. This is likely to be a geometry mismatch or that /boot/boot.b could not be found
LIL	The descriptor table (map file) could not be read. This is usually a media failure. The boot.map file contains the location of the kernel.
LIL?	The second stage boot loader loaded at an incorrect address. This is likely to be a geometry mismatch or that /boot/boot.b could not be found
LIL-	LILO found a corrupt descriptor table. This is likely to be a geometry mismatch or an inability to find the map file
LILO	LILO ran successfully

Assuming the prompt option is enabled in /etc/lilo.conf, LILO will stop and issue a boot prompt. It will then wait for a few seconds for any optional input from the user, and failing that it will then boot the default system.

Typical system labels that people use in the LILO configuration files are linux and backup and msdos. If you press the [TAB] key LILO will display a list of available labels. If you want to type in a boot argument, you type it in here, after typing in the system label that you want LILO to boot from, as shown in the examples below.

`LILO:` **`linux root=/dev/hda1`** - mount /dev/hda1 as root partition
`LILO:` **`linux single`** - boot linux in single user mode

For detailed information on passing options to the kernel see the **BootPrompt-HOWTO** in **/usr/doc/HOWTO**.
Options can be also be passed to LILO on the command line (see the appropriate info page), but LILO is more usually configured by editing the file **/etc/lilo.conf**. A sample lilo.conf file from a dual boot Linux/Win98 machine is illustrated below:

```
default=dos
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50

image=/boot/vmlinuz-2.2.12-20
          label=linux
          initrd=/boot/initrd-2.2.12-20.img
          read-only
          root=/dev/hda5

other=/dev/hda1
          label=dos
```

The parameters used in the above example are explained in the table below:

Parameter	***Explanation***
default	Tells LILO which operating system to boot by default. If this is not supplied LILO will default to the first o/s image listed in the lilo.conf file
boot	Tells LILO which drive to boot from
map	Tells LILO which mapping file to use
install	Tells LILO where to find the boot sector file
prompt	Tells LILO to issue a prompt. If this parameter is omitted LILO will boot directly into the default operating system
timeout	Tells LILO how long to wait for user response after the prompt is issued before booting the default operating system. Timeout is expressed in tenths of a second e.g. 50 = 5 seconds wait.
image	Tells LILO the name of the kernel to load
label	Here you can define the text string which LILO will expect you to enter at the prompt so that the chosen operating system boots.
read-only	Tells LILO to mount the root as read only
root	Tells LILO which partition to mount as root
other	Used to tell LILO about other operating system boot choices

Boot the System

Before you try editing /etc/lilo.conf make a backup copy of the original and a boot floppy disc, as it is very easy to make a mistake and end up with an unbootable machine. After making changes to /etc/lilo.conf make sure to run the **/sbin/lilo** command so that the new information is written to the boot record.

The Kernel

The kernel of the operating system is the software which interacts with the computer hardware and applications connect on demand to the kernel. The kernel is kept as small as possible and supports dynamically loaded device drivers (modules) which allow portions of the kernel to be written as separate objects which can be loaded or unloaded on a running system.

With open source software, even numbered versions are stable releases, whereas odd numbered versions are developmental releases of code, which may not be stable. Kernel version numbers follow the convention, **`major.minor.patchlevel`** Major is the version number, which rarely changes, the minor version number indicates the current "strain" of the kernel release and patchlevel is the the patch number of the current kernel.

For instance as of August 2000, the current stable kernel is 2.2.16, whereas the kernel under development is 2.3.X. The next stable kernel will be 2.4, which is currently at patchlevel test6. This will only be released once the kernel project maintainer (Linus Torvalds) is satisfied that it is fully ready for release. Information on the latest kernel developments can be found at the following web site: **www.linuxhq.com/kernel**

You can find out what kernel you are running by using the **uname -a** command as illustrated below:

```
[clive@redhat clive]$ uname -a
Linux redhat.bromley.ac.uk 2.2.5-15 #1 Mon Apr 19 22:21:09 EDT
1999 i586 unknown
```

Here you can see we are running kernel version 2.2.5-15 which was last compiled on April 19th 1999.

Boot the System

Print or Control the Kernel Ring Buffer- dmesg

While loading, the kernel prints messages to the display and also saves them in the file **/var/log/messages**. You can either look at this log file directly or use the **dmesg** command.

The syntax for the dmesg command is shown below:

dmesg *option(s)*

Without any options dmesg displays the last messages (8196 bytes) in the kernel ring buffer.

Using the -n option you can determine the level at which logging messages are displayed on the screen. This does not affect the contents of the log file, just the level of detail of which it is displayed on the screen. For instance the command **dmesg -n 1** displays just panic messages on the console.

For faultfinding the dmesg command can help you print out your boot messages. Instead of copying the messages by hand, you need only type:

```
[clive@redhat clive]$ dmesg > boot.messages
```

You can then email the boot.messages file to someone who can help you debug the problem.

Messages saved by the kernel at boot include:

- The console type and font
- Detection of any PCI bus and any PCI cards present
- An estimate of the processor speed
- The amount of system memory available
- The type of CPU
- The version number of the kernel
- Any device drivers which are loaded
- The amount of swap space
- Network adapters and settings

Boot the System

The init Process

After the kernel has finished loading and has mounted the root filesystem it starts the init daemon. This is responsible for starting and restarting processes that the rest of the system will use, and it stays active until you shut the system down.

The configuration file for the init process is **/etc/inittab**. The exact contents of this file will vary from one distribution to another. You can edit /etc/inittab while the system is running, but take great care editing this file. Make a backup copy of the original and a boot floppy disc, as it is very easy to make a mistake and end up with an unbootable machine.

An example of an /etc/inittab file is illustrated below:

```
# inittab       This file describes how the INIT process
should set up the system in a certain run-level.

# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not
have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)

id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

```
# Things to run in every runlevel.
ud::once:/sbin/update


# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few
minutes of power left. Schedule a shutdown for 2 minutes from
now. This does, of course, assume you have powerd installed
and your UPS connected and working correctly.

pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System
Shutting Down"

# If power was restored before the shutdown kicked in, cancel
it.

pr:12345:powerokwait:/sbin/shutdown -c "Power Restored;
Shutdown Cancelled"

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Each command line in the /etc/inittab file contains four fields separated by a colon. These fields are:

```
ID:runlevel:action:process
```

The purpose of these fields is explained in the table below:

Field	*Description*
`ID`	One or characters identifying the entry, usually the device name
`runlevel`	Indicates the run levels to which this line will apply. If this is left blank then this line will apply to all runlevels
`action`	How to handle the entry
`process`	The command to execute

The table below shows some valid entries from the action field:

Action	*Description*
`ctrlaltdel`	Traps the CTRL-ALT-DEL key combination and performs an orderly shutdown.
`initdefault`	Sets the default runlevel. This will normally be set to either full multiuser mode or X11.
`powerfail`	This is executed in the event of a power failure, assuming you have UPS, of course.
`respawn`	Restarts the process if it stops.
`sysinit`	Executes the system initialisation script.
`wait`	Runs the script once

Any changes you make to /etc/inittab file will not be implemented until init re-reads its configuration file. This occurs when the runlevel is changed, you reboot the system, a power failure occurs, or you use **init q.**

Boot the System

Modules

The kernel loads and unloads certain modules on demand according to the appropriate level rc script called by the init process. Details of the modules and the parameters passed to them are contained in either the file **/etc/conf.modules** or the file **/etc/modules.conf**.

You can view the device drivers the kernel is loading and change their configuration settings by editing these files. In reality only one file is needed for this purpose. This was originally named /etc/conf.modules, but for the sake of consistency it was later decided to use the filename /etc/modules.conf.

This is confusing and it is currently recommended that you use /etc/modules.conf, but retain /etc/conf.modules as a soft link to it for compatibility. (Otherwise you might find yourself trying to configure a network adapter or a sound card and find your settings have been ignored because you put them in the wrong file!)
An example of an /etc/modules.conf file is illustrated below:

```
[clive@redhat clive]$ cat /etc/conf.modules
alias scsi_hostadapter aic7xxx     # the scsi adapter
alias eth0 3c509                   # the network adapter
alias parport_lowlevel parport_pc  # the parallel port
pre-install pcmcia_core /etc/rc.d/init.d/pcmcia start

# You might need an options line for some net adapters:
options 3c509 io=0x300 irq=10
```

The alias parameter is used to form a look up table to link the module names used by the kernel to the names of the actual device drivers.

The options parameter can be used to specify the configuration options to be used with each of the device drivers.