LILO
Generic boot loader for Linux
Version 0.9

Werner Almesberger
almesber@bernina.ethz.ch

February 28, 1993

Contents

LILO is a versatile boot loader for Linux. It does not depend on a specific file system, can boot Linux boot images and unstripped Linux kernels from floppy disks and from hard disks and can even boot other operating systems[1].

One of up to sixteen different boot images can be selected at boot time. Various parameters, such as the root device, can be set independently for each of them.

[1]PC/MS-DOS, DR DOS, OS/2, 386BSD, SCO UNIX, ...

LILO can even be used as the master boot record.

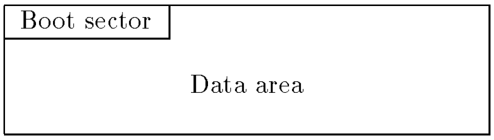
This document introduces the basics of disk organization and booting, continues with an overview of common boot techniques and finally describes installation and use of LILO in greater detail. The troubleshooting section at the end describes diagnostic messages and contains suggestions for most problems that have been observed in the past.

Please read the sections about installation and configuration if you're already using an older version of LILO. This distribution is accompanied by a file named `INCOMPAT` that describes further incompatibilities to older versions.

For the impatient: there is a quick-installation script that creates a simple but quite usable installation. See section 6.2 for details.

1 Disk organization

When designing a boot concept, it is important to understand all the subtleties of how MS-DOS organizes disks. The most simple case are floppy disks. They consist of a boot sector, some administrative data (FAT or super block, etc.) and the data area. Because that administrative data is irrelevant as far as booting is concerned, it is regarded as part of the data area for simplicity.



The entire disk appears as one device (i.e. `/dev/fd0`) on Linux.

The MS-DOS boot sector has the following structure:

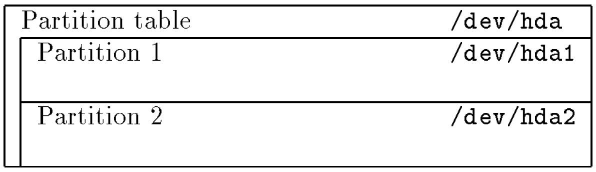
`0x000`	Jump to the program code
`0x003`	Disk parameters
`0x02C/0x03E`	Program code
`0x1FE`	Magic number (0xAA55)

LILO uses a similar boot sector, but it does not contain the disk parameters part. This is no problem for Minix or EXT file systems, because they don't look at the boot sector, but putting a LILO boot sector on an MS-DOS file system would make it inaccessible for MS-DOS.

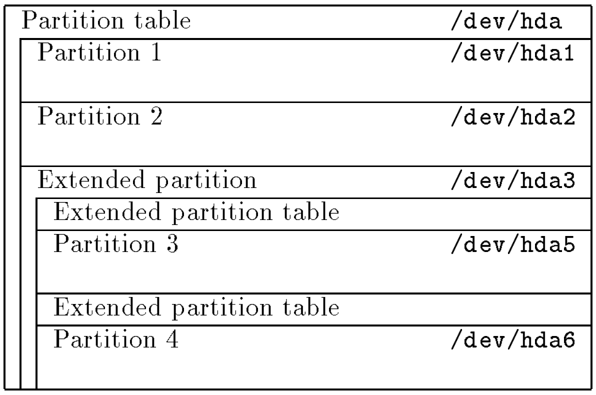
Hard disks are organized in a more complex way than floppy disks. They contain several data areas called partitions. Up to four so-called primary partitions can exist on an MS-DOS hard disk. If more partitions are needed, a primary partition is used as an extended partition that contains several logical partitions.

The first sector of each hard disk contains a partition table and an extended partition and **each** logical partition contains a partition table too.[2]

[2] Is it legal to have more than one extended partition ? Linux appears to be able to handle this, but is DOS ?



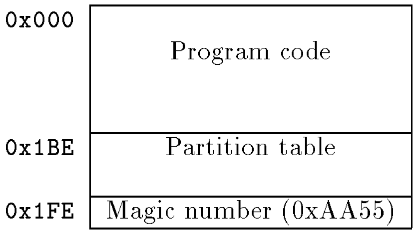
The entire disk can be accessed as `/dev/hda`, `/dev/hdb`, `/dev/sda`, etc. The primary partitions are `/dev/hda1` ... `/dev/hda4`.



This hard disk has two primary partitions and an extended partition that contains two logical partitions. They are accessed as `/dev/hda5` ...

Note that the partition tables of logical partitions are not accessible as the first blocks of some devices, while the main partition table, all boot sectors and the partition tables of extended partitions are.

Partition tables are stored in partition boot sectors. Only the partition boot sector of the entire disk is usually used as a boot sector. It is also frequently called the master boot record (MBR).



The LILO boot sector is designed to be usable as a partition boot sector. (I.e. there is room for the partiton table.) Therefore, the LILO boot sector can be stored at the following locations:

- boot sector of a Linux floppy disk. (`/dev/fd0`, ...)
- MBR of the first hard disk. (`/dev/hda`, ...)
- boot sector of a primary Linux file system partition on the first hard disk. (`/dev/hda1`, ...)
- partition boot sector of an extended partition on the first hard disk. (`/dev/hda1`, ...)[3]

It **can't** be stored at any of the following locations:

- boot sector of a non-Linux floppy disk or primary partition.
- a Linux swap partition.
- boot sector of a logical partition in an extended partition.[4]
- on the second hard disk. (Unless for backup installations, if the current first disk will be removed or disabled, or if some other boot loader that is capable of loading boot sectors from other drives is being used.)

[3]Most FDISK-type programs don't believe in booting from an extended partition and refuse to activate it. LILO is accompanied by a simple program (activate) that doesn't have this restriction.

[4]LILO can be forced to put the boot sector on such a partition by using the `-b` option or the `boot` variable. However, hardly any programs that operate as master boot records support booting from a logical partition.

2 Booting basics

When booting from a floppy disk, the first sector of the disk, the so-called boot sector, is loaded. That boot sector contains a small program that loads the respective operating system. MS-DOS boot sectors also contain a data area, where disk and file system parameters (cluster size, number of sectors, number of heads, etc.) are stored.

When booting from a hard disk, the very first sector of that disk, the so-called master boot record (MBR) is loaded. This sector contains a loader program and the partition table of the disk. The loader program usually loads the boot sector, as if the system was booting from a floppy.

Note that there is no functional difference between the MBR and the boot sector other than that the MBR contains the partition information but doesn't contain any (MS-DOS) disk parameter information.

The first 446 (0x1BE) bytes of the MBR are used by the loader program. They are followed by the partition table, with a length of 64 (0x40) bytes. The last two bytes contain a magic number that is sometimes used to verify that a given sector really is a boot sector.

There is a large number of possible boot configurations. The most common ones are described in the following sections.

2.1 MS-DOS alone

Master Boot Record	Boot sector	Operating system
DOS-MBR ⟶	**MS-DOS** ⟶	**COMMAND.COM**

This is what usually happens when MS-DOS boots from a hard disk: the DOS-MBR determines the active partition and loads the MS-DOS boot sector. This boot sector loads MS-DOS and finally passes control to `COMMAND.COM`. (This is greatly simplified.)

2.2 BOOTLIN

Master Boot Record	Boot sector	Operating systems
DOS-MBR ⟶	**MS-DOS** ⟶	**COMMAND.COM**
	⟶	**BOOTLIN** ⟶ **Linux**

A typical BOOTLIN setup: everything happens like when booting MS-DOS, but in `CONFIG.SYS` or `AUTOEXEC.BAT`, BOOTLIN is invoked. Typically, a program line `BOOT.SYS` is used to choose among configuration sections in `CONFIG.SYS` and `AUTOEXEC.BAT`. This approach has the pleasant property that no boot sectors have to be altered.

Installation:

- boot Linux.
- copy a bootable kernel image to your MS-DOS partition.[5]
- install `BOOT.SYS` and `BOOTLIN.SYS` on your MS-DOS partition and add them to your `CONFIG.SYS`. (The READMEs describe how this is done.)
- reboot.

Deinstallation:

- remove `BOOT.SYS` and `BOOTLIN.SYS` from your `CONFIG.SYS`.

2.3 LILO started by DOS-MBR

Master Boot Record	Boot sector	Operating system
DOS-MBR $\longrightarrow$	**LILO** $\longrightarrow$	**Linux**
$\longrightarrow$	other OS	

This is a "safe" LILO setup: LILO is booted by the DOS-MBR. No other boot sectors have to be touched. If the other OS (or one of them, if there are several other operating systems being used) should be booted without using LILO, the other partition has to be marked "active" with (e)fdisk or activate.

Installation:

- install LILO with its boot sector on the Linux partition.
- use (e)fdisk or activate to make that partition active.
- reboot.

Deinstallation:

- make a different partition active.
- install whatever should replace LILO or Linux.

[5] With Mtools or the MS-DOS FS.

2.4 Several alternate branches

Master Boot Record	Boot sector	Operating systems	
DOS-MBR ⟶	**MS-DOS** ⟶	**COMMAND.COM**	
	⟶	**BOOTLIN** ⟶	**Linux**
⟶	**LILO** ⟶	**Linux**	
	⟶	**MS-DOS** — ···	

An extended form of the above setup: the MBR is not changed and both branches can either boot Linux or MS-DOS. (LILO could also boot any other operating system.)

2.5 LILO started by BOOTACTV

Master Boot Record	Boot sector	Operating system
BOOTACTV ⟶	**LILO** ⟶	**Linux**
⟶	other OS	

Here, the MBR is replaced by BOOTACTV (or any other interactive boot partition selector) and the choice between Linux and the other operating system(s) can be made at boot time. This approach should be used if LILO fails to boot the other operating system(s).[6]

Installation:

- boot Linux.
- make a backup copy of your MBR on a floppy disk, e.g.
  `dd if=/dev/hda of=/fd/MBR bs=512 count=1`
- install LILO with the boot sector on the Linux partition.
- install BOOTACTV as the MBR, e.g.
  `dd if=bootactv.bin of=/dev/hda bs=446 count=1`
- reboot.

Deinstallation:

- boot Linux.
- restore the old MBR, e.g.
  `dd if=/MBR of=/dev/hda bs=446 count=1`

[6] And the author would like to be notified if booting the other operating system(s) doesn't work with LILO, but if it works with an other boot partition selector.

If replacing the MBR appears undesirable and if a second Linux partition exists (e.g. `/usr`, **not** a swap partition), BOOTACTV can be merged with the partition table and stored as the "boot sector" of that partition. Then, the partition can be marked active to be booted by the DOS-MBR.

Example:

```
# dd if=/dev/hda of=/dev/hda3 bs=512 count=1
# dd if=bootactv.bin of=/dev/hda3 bs=446 count=1
```

Warning: whenever the disk is re-partitioned, the merged boot sector on that "spare" Linux partition has to be updated too.

2.6 Shoelace started by BOOTACTV

Master Boot Record	Boot sector	Operating system
BOOTACTV ⟶	**Shoelace** ⟶	**Linux**
⟶	other OS	

Shoelace, LILO's predecessor, can be started by BOOTACTV as well, of course. The same indirection as outlined above is possible. There are probably many other ways to install Shoelace.[7]

2.7 LILO alone

Master Boot Record	Operating system
LILO ⟶	**Linux**
⟶	other OS

LILO can also take over the entire boot procedure. If installed as the MBR, LILO is responsible for either booting Linux or any other OS. This approach has the disadvantage, that the old MBR is overwritten and has to be restored (either from a backup copy, with `FDISK /MBR` on MS-DOS 5.0 or by overwriting it with BOOTACTV) if Linux should ever be removed from the system.

You should verify that LILO is able to boot your other operating system(s) before relying on this method.

Installation:

- boot Linux.
- make a backup copy of your MBR on a floppy disk, e.g.
  `dd if=/dev/hda of=/fd/MBR bs=512 count=1`

[7] Shoelace is considered obsolete and should not be used for new installations.

- install LILO with its boot sector as the MBR.
- reboot.

Deinstallation:

- boot Linux.
- restore the old MBR, e.g.
  `dd if=/fd/MBR of=/dev/hda bs=446 count=1`

If you've installed LILO to be the master boot record, you have to explicitly specify the boot sector (`-b` ...) when updating the map. Otherwise, it will try to use the boot sector of your current root partition, which may even work, but will leave your system unbootable.

2.8 Names

The following names have been used to describe boot sectors or parts of operating systems:

DOS-MBR is the original MS-DOS MBR. It scans the partition table for a partition that is marked "active" and loads the boot sector of that partition. Programs like MS-DOS' FDISK, Owen Le Blanc's fdisk for Linux (on MCC-interim or 0.97 rootimage, or named efdisk on the 0.96 rootimage) or activate (accompanies LILO) can change the active marker in the partition table.

MS-DOS denotes the MS-DOS boot sector that loads the other parts of the system (`IO.SYS`, etc.).

COMMAND.COM is the standard command interpreter of MS-DOS.

BOOTLIN is a program that loads a Linux boot image from an MS-DOS partition into memory and executes it. It is usually invoked from `CONFIG.SYS` and used in conjunction with a `CONFIG.SYS` configuration switcher, like BOOT.SYS.[8]

[8] BOOTLIN is available for anonymous FTP from
`tsx-11.mit.edu:/pub/linux/INSTALL/dos_utils/bootlin4.zip` or
`nic.funet.fi:/pub/OS/Linux/tools/bootlin.zip`,
BOOT.SYS is available for anonymous FTP from
`sunsite.unc.edu:/pub/Linux/system/Linux-boot/boot142.zip`,
`nic.funet.fi:/pub/OS/Linux/tools/boot142.zip` or
`wuarchive.wustl.edu:/mirrors/msdos/sysutl/boot142.zip`.

LILO can either load a Linux kernel or the boot sector of any other operating system. It has a first stage boot sector that loads the remaining parts of LILO from various locations.

BOOTACTV permits interactive selection of the partition from which the boot sector should be read. If no key is pressed within a given interval, the partition marked active is booted. BOOTACTV is included in the pfdisk package. There are also several similar programs, like PBOOT and OS-BS.[9]

Shoelace is a different boot loader for Linux. It is functionally similar to LILO, but it can only use the Minix file system. Because of that, Shoelace is obsolete and should not be used for new installations.

[9] pfdisk is available for anonymous FTP from
`sunsite.unc.edu:/pub/Linux/utils/disk-management/pfdisk.tar.Z` or
`nic.funet.fi:/pub/OS/Linux/tools/pfdisk.tar.Z`.
PBOOT can be found at the same sites in
`tsx-11.mit.edu:/pub/linux/INSTALL/dos_utils/pboot.zip` and
`nic.funet.fi:/pub/OS/Linux/tools/pboot.zip`.

3 Choosing the "right" boot concept

Although LILO can be installed in many different ways, the choice is usually limited by the present setup and therefore there is typically only a small number of configurations which fit naturally into an existing system.

For each concept, example `/etc/lilo/config` and `/etc/lilo/install` files are provided. `/etc/lilo/install` is a shell script that actually performs the installation (and updates) of the boot sector and the map file. It has to be created by the user when installing LILO and has to be run whenever LILO or the kernel is changed. (See section 6.3.) `/etc/lilo/config` is the new, preferred configuration method. It contains the same information as `/etc/lilo/install` in a more readable way. When using `/etc/lilo/config`, a standard `/etc/lilo/install` can be used, see section 5.1.2.

Each of the `/etc/lilo/install` files installs the following boot images:

- a traditional Linux boot image (`/linux`).
- an alternate traditional Linux boot image (`/linux.backup`).
- a chain loader to boot MS-DOS.

In all examples, the names of the AT-type hard disk devices (`/dev/hda`...) are used. Everything applies to SCSI disks (`/dev/sda`...) too.

3.1 One disk, Linux on a primary partition

If at least one primary partition of the first hard disk is used as a Linux file system (`/`, `/usr`, etc. but **not** as a swap partition), the LILO boot sector should be stored on that partition and it should be booted by the original master boot record or by a program like BOOTACTV.

	MBR	`/dev/hda`
	MS-DOS	`/dev/hda1`
→	Linux /	`/dev/hda2`

A typical `/etc/lilo/config` file would look like this:

```
compact
install = /etc/lilo/boot.b
image = /linux
image = /linux.backup
other = /dev/hda1
  table = /dev/hda
  label = msdos
```

or, old-style (`/etc/lilo/install`):

```
$ROOT/etc/lilo/lilo -c -i /etc/lilo/boot.b $* \
    /linux \
    /linux.backup \
    msdos=/etc/lilo/chain.b+/dev/hda1@/dev/hda
```

3.2 One disk, Linux on an extended partition

If no primary partition is available for Linux, but at least one logical partition of an extended partition on the first hard disk contains a Linux file system, the LILO boot sector should be stored in the partition sector of the extended partition and it should be booted by the original master boot record or by a program like BOOTACTV.

	MBR	`/dev/hda`
	MS-DOS	`/dev/hda1`
→	Extended	`/dev/hda2`
	Linux	`/dev/hda5`
	...	`/dev/hda6`

A typical `/etc/lilo/config` file for this configuration would look like this:

```
boot = /dev/hda2
compact
install = /etc/lilo/boot.b
image = /linux
image = /linux.backup
other = /dev/hda1
  table = /dev/hda
  label = msdos
```

or, old-style (`/etc/lilo/install`):

```
$ROOT/etc/lilo/lilo -b /dev/hda2 -c -i /etc/lilo/boot.b $* \
    /linux \
    /linux.backup \
    msdos=/etc/lilo/chain.b+/dev/hda1@/dev/hda
```

3.3 Two disks, Linux (at least partially) on the first disk

This case is equivalent to the configurations, where only one disk is in the system. The Linux boot sector resides on the first hard disk and the second disk is used later.

Only the location of the boot sector matters – everything else (`/etc/lilo/boot.b`, `/etc/lilo/map`, the root file system, a swap partition, other Linux file systems, etc.) can be located anywhere on the second disk.

3.4 Two disks, Linux on the second disk, the first disk has an extended partition

If there is no Linux partition on the first disk, but there is an extended partition, the LILO boot sector can be stored in the partition sector of the extended partition and it should be booted by the original master boot record or by a program like BOOTACTV.

	First disk		**Second disk**	
	MBR	/dev/hda	MBR	/dev/hdb
	MS-DOS	/dev/hda1	Linux	/dev/hdb1
→	Extended	/dev/hda2	...	/dev/hdb2
	...	/dev/hda5		
	...	/dev/hda6		

A typical `/etc/lilo/config` file for this configuration would look like this:

```
boot = /dev/hda2
compact
install = /etc/lilo/boot.b
image = /linux
image = /linux.backup
other = /dev/hda1
  table = /dev/hda
  label = msdos
```

or, old-style (`/etc/lilo/install`):

```
$ROOT/etc/lilo/lilo -b /dev/hda2 -c -i /etc/lilo/boot.b $* \
    /linux \
    /linux.backup \
    msdos=/etc/lilo/chain.b+/dev/hda1@/dev/hda
```

The program activate, that accompanies LILO, has to be used to set the active marker on an extended partition, because MS-DOS' FDISK and (e)fdisk refuse to do that. (Which is generally a good idea.)

3.5 Two disks, Linux on the second disk, the first disk has no extended partition

If there is neither a Linux partition nor an extended partition on the first disk, where a LILO boot sector could be stored, then there's only one place left: the master boot record.

In this configuration, LILO boots all other operating systems too.

	First disk		**Second disk**	
→	MBR	/dev/hda	MBR	/dev/hdb
	MS-DOS	/dev/hda1	Linux	/dev/hdb1
	...	/dev/hda2	...	/dev/hdb2

You should back up your old MBR before installing LILO and verify that LILO is able to boot your other operating system(s) before relying on this approach.

A typical `/etc/lilo/config` file for this configuration would look like this:

```
boot = /dev/hda
compact
install = /etc/lilo/boot.b
image = /linux
image = /linux.backup
other = /dev/hda1
  table = /dev/hda
  label = msdos
```

or, old-style (`/etc/lilo/install`):

```
$ROOT/etc/lilo/lilo -b /dev/hda -c -i /etc/lilo/boot.b $* \
    /linux \
    /linux.backup \
    msdos=/etc/lilo/chain.b+/dev/hda1@/dev/hda
```

4 The boot prompt

Immediately after it's loaded, LILO checks, whether one of the following is happening:

- any of the [Shift], [Control] or [Alt] keys is being pressed.
- [CapsLock] or [ScrollLock] is set.

If this is the case, LILO displays the `boot:` prompt and waits for the name of a boot image. Otherwise, it boots the default boot image[10] or – if a delay has been specified – waits for one of the listed activities until that amount of time has passed.

At the boot prompt, the name of the image to boot can be entered. Typing errors can be corrected with [BackSpace], [Delete], [Ctrl U] and [Ctrl X]. A list of known images can be obtained by pressing [?] (on the US keyboard) or [Tab].

If [Enter] is pressed and no file name has been entered, the default image is booted.

When using a modern kernel[11], LILO is also able to pass command-line options to it. Command-line options are words that follow the name of the boot image and that are separated by spaces. Currently, the kernel recognizes the option `root=`*device* and all current `init` programs also recognize the option `single`.

`single` boots the system in single-user mode. This bypasses all system initialization procedures and directly starts a root shell on the console. Multi-user mode can be entered by exiting the single-user shell or by rebooting.

`root=`*device* changes the root device. This overrides settings that may have been made in the boot image and on the LILO command line. *device* is the hexadecimal device number.[12]

`vga=`*mode* alters the VGA mode set at startup. The values `normal`, `extended`, `ask` or a decimal number are recognized. (See also page 25.)

Other options of the type *variable*`=`*value* than `root` and `vga` cause the respective variables to be set in the environment passed to `init`. The case of the variable name is preserved. (I.e. it's not automatically converted to upper case.)

[10] The default boot image is either the first boot image or the image that has been selected at the boot prompt.

[11] 0.98pl3 or newer.

[12] This is a list of device numbers of some frequently used devices:

/dev/fd0	200	/dev/hda1	301	/dev/sda1	801
/dev/fd1	201	/dev/hda2	302	/dev/sda2	802
...		...		...	
		/dev/hdb1	341	/dev/sdb1	811
		/dev/hdb2	342	/dev/sdb2	812
		...		...	

Example:

```
boot: linux single root=200
```

LILO always passes the string "`BOOT_IMAGE=`*name*" to the kernel, where *name* is the name by which the kernel is identified (e.g. the label). This variable can be used in `/etc/rc` to select different behaviour, depending on the kernel.

When booting automatically, the word "`auto`" is also passed on the command line. This can be used by `init` to suppress interactive prompts in the boot phase.

The details of selecting the boot image are somewhat complicated. The following tables illustrate them. First, if neither `prompt` is set nor a shift key is being pressed:

Externally provided cmd. line	Command line in map file	Automatic boot	Booted image
No	No	Yes	Default image
Yes	—	Yes	Specified by external command line
No	Yes	Yes	Specified by command line in map file

If `prompt` is not set and a shift key is being pressed:

Input timeout	Empty cmd.l.	Extern. cmd.l.	Cmd.l. in map file	Automatic boot	Booted image
No	No	—	—	No	Specified by the user
No	Yes	—	—	No	Default image
Yes	n/a	—	—	Yes	Default image

Finally, if the configuration variable `prompt` is set:

Input timeout	Empty cmd.l.	Extern. cmd.l.	Cmd.l. in map file	Automatic boot	Booted image
No	No	No	No	No	Specified by the user
No	Yes	No	No	No	Default image
Yes	n/a	No	No	Yes	Default image
n/a	n/a	Yes	—	Yes	Specified by external command line
n/a	n/a	No	Yes	Yes	Specified by command line in map file

The default image is typically the first image in the map file. However, after an unsuccessful boot attempt, the respective image becomes the default image.

5 Map installer

The map installer program `/etc/lilo/lilo` updates the boot sector and creates the map file. It is usually run from the shell script `/etc/lilo/install`. If the map installer detects an error, it terminates immediately and does not touch the boot sector and the map file.

Whenever the LILO map installer updates a boot sector, the original boot sector is copied to `/etc/lilo/boot.`*number*, where *number* is the hexadecimal device number. If such a file already exists, no backup copy is made.

5.1 Command-line arguments

The LILO map installer accepts the following command-line options:

`-b` *boot_device*
: Sets the name of the device that contains the boot sector. If `-b` is omitted, the boot sector is read from (and possibly written to) the device that is currently mounted as root.

`-c`
: Tries to merge read requests for adjacent sectors into a single read request. This drastically reduces load time and keeps the map smaller. Using `-c` is especially recommended when booting from a floppy disk.

`-C` *config_file*
: Specifies the new-style configuration file that is used by the map installer (see section 5.1.2). If `-C` is omitted, old-style configuration is used.

`-d` *tsecs*
: Specifies the number of tenths of a second LILO should wait before booting the first image. This is useful on systems that immediately boot from the hard disk after enabling the keyboard. LILO doesn't wait if `-d` is omitted.

`-D`
: Selects dump mode. Dumps are used during migration from earlier versions of LILO (see section 6.5). `-D` also implies `-t` and disables all other informational messages.

`-f` *disktab_file*
: Specifies the name of the disk parameter table (see section 5.3). The map installer looks for `/etc/lilo/disktab` if `-f` is omitted.

`-i` *boot_sector*
: Install the specified file as the new boot sector. If `-i` is omitted, the old boot sector is modified. A BIOS device code can be specified. `-i`

is usually a permanent part of the invocation of the map installer in `/etc/lilo/install`.

`-I` *name* [*options*]
: Translates the specified label name to the path of the corresponding kernel image and prints that path on standard output. This can be used to rebuild files that depend on the kernel (e.g. the `ps` database). The image name can be obtained from the environment variable `BOOT_IMAGE`. An error message is issued and a non-zero exit code is returned if no matching label name can be found. If an options argument is present, the following characters specify additional filtering:

  `i` only traditional boot images are found.

  `u` only (compound) unstripped kernels are found.

`-l`
: Generate linear sector addresses instead of sector/head/cylinder addresses. Linear addresses are translated at run time and do not depend on disk geometry. This is experimental and is not (yet) intended for common use.

`-m` *map_file*
: Specifies the location of the map file. If `-m` is omitted, a file `/etc/lilo/map` is used. A BIOS device code can be specified.

`-P`
: Perform certain corrections on corrupt partition tables, see `fix-table` on page 24.

`-q`
: List the currently mapped files.

`-r` *root_directory*
: Chroot to the specified directory before doing anything else. This is useful when running the map installer while the normal root file system is mounted somewhere else, e.g. when recovering from an installation failure with a bootimage/rootimage. The `-r` option is implied if the environment variable `ROOT` is set.[13] The current directory is changed to the new root directory, so using relative paths may not work.

`-R` *word* ...
: Stores the specified words in the map file. The boot loader uses those words as the default command line when booting the next time. That

[13] I.e. if your root partition is mounted on `/mnt`, you can update the map by simply running `ROOT=/mnt /mnt/etc/lilo/install`. If you're normally using the default boot sector, you have to specify it explicitly in this case: `-b /dev/`*device_name*. So the complete command may be something like this:
`ROOT=/mnt /mnt/etc/lilo/install -b /dev/hda1`

command line is removed from the map file by the boot loader by overwriting the sector immediately after reading it. The first word has to be the name of a boot image. If `-R` is not followed by any words, the current default command line in the map file is erased.[14] An error message is issued and a non-zero exit code is returned if the command line is not accepted.

`-s` *backup_file*
: Copy the original boot sector to *backup_file* (which may also be a device, e.g. `/dev/null`) instead of `/etc/lilo/boot.`*number*

`-S` *backup_file*
: Like `-S`, but overwrite an old backup copy if it exists.

`-t`
: Test only. This performs the entire installation procedure except replacing the map file and writing the modified boot sector. This can be used in conjunction with the `-v` option to verify that LILO will use sane values.

`-v`
: Turns on lots of progress reporting. Repeating `-v` will turn on more reporting. (`-v -v -v -v -v` is the highest verbosity level and displays all sector mappings before and after compaction.)

There are two ways how image files can be specified: the traditional approach is to specify them on the command line, typically in `/etc/lilo/install`. The new, preferred method uses a configuration file `/etc/lilo/config`. Section 6.5 describes the migration procedure.

The image files can reside on any media that is accessible at boot time. There's no need to put them on the root device, although this certainly doesn't hurt.

5.1.1 Old-style configuration

If no image files are specified on the command line, the currently mapped files are listed. Only the options `-m`, `-v` and `-r` can be used in this mode.

If at least one file name is specified, a new map is created for those files and they are registered in the boot sector. If the root device has been set in the images, it is copied into the descriptors in the boot sector. Changing the root device specification in a boot image after the map file has been built has no effect. If

[14] `-R` is typically used in reboot scripts, e.g.

```
#!/bin/sh
if /etc/lilo/install -R "$*"; then
    echo | shutdown -r now
fi
```

no root device has been set[15], the current root device is used. The root device can be overridden by appending it to the image specification, e.g.

$$\texttt{lilo}\ \underbrace{\texttt{foo}}_{\text{image}},\underbrace{\texttt{/dev/hda1}}_{\text{root}}$$

or

$$\texttt{lilo}\ \underbrace{\texttt{foo}}_{\text{image}},\underbrace{\texttt{0x302}}_{\text{root}}$$

It is perfectly valid to use different root settings for the same image, because LILO stores them in the image descriptors and not in the images themselves. Example:

```
lin-hd=/linux,/dev/hda2 \
lin-fd=/linux,/dev/fd0
```

LILO uses the main file name (without its path) of each image specification to identify that image. A different name can be used by prefixing the specification with *label*=, e.g.

```
msdos=/etc/lilo/chain.b+/dev/sda1@/dev/sda
```

5.1.2 New-style configuration

For new-style configuration, the file `/etc/lilo/install` always looks like this:

```
$ROOT/etc/lilo/lilo -C /etc/lilo/config $*
```

The configuration information is stored in the file `/etc/lilo/config` and consists of variable assignments. The following syntax rules apply:

- flag variables consist of a single word and are followed by whitespace or the end of the file.
- string variables consist of the variable name, optional whitespace, an equal sign, optional whitespace, the value and required whitespace or the end of the file.
- a non-empty sequence of blanks, tabs, newlines and comments counts as whitespace.

[15] Or if this is an unstripped kernel image.

- variable names are case-insensitive. Values are usually case-sensitive, but there are a few exceptions. (See below.)
- tabs and newlines are special characters and may not be part of a variable name or a value. The use of other control characters and non-ASCII characters is discouraged.
- blanks and equal signs may only be part of a variable name or a value if they are escaped by a backslash. An equal sign may not be the only character in a name or value.
- an escaped tab is converted to an escaped blank. An escaped newline is removed from the input stream. An escaped backslash (i.e. two backslashes) is converted to a backslash.
- comments begin with a number sign and end with the next newline. All characters (including backslashes) until the newline are ignored.

`/etc/lilo/config` begins with a possibly empty options section. Most options can also be set from the command line, but storing permanent options in the configuration file is more convenient. The following options are recognized:

`boot=`*boot_device* is equivalent to `-b` *boot_device*.

`compact` is equivalent to `-c`.

`delay=`*tsecs* is equivalent to `-d` *tsecs*.

`linear` is equivalent to `-l`.

`install=`*boot_sector* is equivalent to `-i` *boot_sector*.

`disktab=`*disktab_file* is equivalent to `-f` *disktab_file*.

`map=`*map_file* is equivalent to `-m` *map_file*.

`message=`*message_file* specifies a file containing a message that is displayed before the boot prompt. No message is displayed when waiting for a shifting key after printing "LILO ". The **FF** character ([Ctrl L]) clears the local screen. The size of the message file is limited to 65535 bytes. The map file has to be rebuild if the message file is changed or moved.

`verbose=`*level* is equivalent to *level* repetitions of *-v*. If `-v` is additionally specified on the command line, *level* is increased.

`backup=`*backup_file* is equivalent to `-s` *backup_file*.

`force-backup=`*backup_file* is equivalent to `-S` *backup_file*. `backup=`*backup_file* is ignored if `force-backup` appears in the same configuration file.

`prompt` forces entering the boot prompt without expecting any prior key-presses. Unattended reboots are impossible if `prompt` is set and `timeout` isn't.

`timeout=`*tsecs* sets a timeout (in tenths of a second) for keyboard input. If no key is pressed for the specified time, the first image is automatically booted. Similarly, password input is aborted if the user is idle for too long. The default timeout is infinite.

`serial=`*parameters* enables control from a serial line. The specified serial port is initialized and LILO is accepting input from it and from the PC's keyboard. Sending a break on the serial line corresponds to pressing a shift key on the console in order to get LILO's attention. All boot images should be password-protected if the serial access is less secure than access to the console, e.g. if the line is connected to a modem. The parameter string has the following syntax:
port`,`*bps parity bits*
The components *bps*, *parity* and *bits* can be omitted. If a component is omitted, all following components have to be omitted too. Additionally, the comma has to be omitted if only the port number is specified.

- *port* the number of the serial port, zero-based. 0 corresponds to `COM1` alias `/dev/ttys0`, etc. All four ports can be used (if present).
- *bps* the baud rate of the serial port. The following baud rates are supported: 110, 150, 300, 600, 1200, 2400, 4800 and 9600 bps. Default is 2400 bps.
- *parity* the parity used on the serial line. LILO ignores input parity and strips the 8th bit. The following (upper or lower case) characters are used to describe the parity: `n` for no parity, `e` for even parity and `o` for odd parity.
- *bits* the number of bits in a character. Only 7 and 8 bits are supported. Default is 8 if parity is "none", 7 if parity is "even" or "odd".

If `serial` is set, the value of `delay` is automatically raised to 20. Example: `serial=0,2400n8` sets `COM1` to the default parameters.

`fix-table` allows LILO to adjust 3D addresses in partition tables. Each partition entry contains a 3D (sector/head/cylinder) and a linear address of the first and the last sector of the partition. If a partition is not track-aligned and if certain other operating systems (e.g. PC/MS-DOS or OS/2) are using the same disk, they may change the 3D address. LILO can store its boot sector only on partitions where both address types correspond. LILO re-adjusts incorrect 3D start addresses if `fix-table` is set.
WARNING This does not guarantee that other operating systems may not attempt to reset the address later. It is also possible that this change

has other, unexpected side-effects. The correct fix is to re-partition the drive with a program that does align partitions to tracks.

`password` sets a password for all images. (See below.)

`restricted` relaxes the password protection. (See below.)

`optional` makes all images optional. (See below.)

Additionally, the kernel configuration parameters `root`, `vga` and `ramdisk` can be set in the options section. They are used as defaults if they aren't specified in the configuration sections of the respective kernel images. See below for a description.

If the option `-q` is specified on the command line, the currently mapped files are listed. Otherwise, a new map is created for the images described in the configuration file `/etc/lilo/config` and they are registered in the boot sector.

Each image description begins with a special variable and is followed by optional variables. The following variables can be used for all image descriptions that describe a Linux kernel:

`root=`*root_device* specifies the device that should be mounted as root. If the special name `current` is used, the root device is set to the device on which the root file system is currently mounted. If the root has been changed with `-r`, the respective device is used. If the variable `root` is omitted, the root device setting that is contained in the boot image is used. It can be changed with the `rdev` aka `setroot` program.

`vga=`*mode* specifies the VGA text mode that should be selected when booting. The following values are recognized (case is ignored):

- `normal` select normal 80x25 text mode.
- `extended` select 80x50 text mode. The word `extended` can be abbreviated to `ext`.
- `ask` stop and ask for user input (at boot time). A system where this option is being used is unable to reboot automatically.
- *number* use the corresponding text mode. A list of available modes can be obtained by booting with `vga=ask` and pressing [Enter].

If this variable is omitted, the VGA mode setting that is contained in the boot image is used. Some versions of `rdev` support manipulation of the VGA text mode setting in the boot image.

`ramdisk=`*size* specifies the size of the optional RAM disk. A value of zero indicates that no RAM disk should be created. If this variable is omitted, the RAM disk size that is configured into the boot image is used.

The variables **vga** and **ramdisk** are only available when running a kernel that supports them, e.g. 0.98pl6 and later. If one of *root*, *vga* or *ramdisk* is omitted in the configuration file and the corresponding value in the boot image is changed, LILO will use the new value.

It is perfectly valid to use different root settings for the same image, because LILO stores them in the image descriptors and not in the images themselves.

Example:

```
image = /linux
  label = lin-hd
  root = /dev/hda2
image = /linux
  label = lin-fd
  root = /dev/fd0
```

LILO uses the main file name (without its path) of each image specification to identify that image. A different name can be used by setting the variable `label` (see the example above). A different name for the same entry can be used by specifying the `alias=` *name* option.

Images are protected by a password if the variable **password** is set. If the variable **restricted** is set in addition to **password**, a password is only required to boot the respective image if parameters are specified on the command line (e.g. **single**). **password** and **restricted** can also be set in the options section to be the default password and password protection mode for all images.

If an image section contains the variable `optional` (or if that variable is set in the options section), the respective image is omitted if its main file is not available at map creation time. This is useful to specify test kernels that are not always present.

5.2 Boot image types

LILO can boot the following types of images:

- "traditional" boot images from a file.
- "traditional" boot images from a block device. (I.e. a floppy disk.)
- unstripped kernel executables.
- compound unstripped kernels.
- the boot sector of some other operating system.

The image type is determined either by the syntax that is used for the image specification (old-style configuration) or by the name of the initial variable of the configuration section (new-style configuration).

5.2.1 Booting "traditional" boot images from a file

If present, the root device definition is taken from the boot image. The image is specified as follows:

Old-style:

$$\mathit{file\_name}\,\big[\,:\mathit{BIOS\_code}\big]$$

Example: `/linux`

New-style:

$$\texttt{image=}\mathit{name}$$

The following additional variables are recognized: `root`, `vga`, `ramdisk`, `password`, `restricted`, `label`, `alias` and `optional`.

Example:

```
image = /linux
```

5.2.2 Booting "traditional" boot images from a device

The root device setting in the image is ignored.[16] The range of sectors that should be mapped, has to be specified. Either a range (*start*-*end*) or a start and a distance (*start*+*number*) have to be specified. *start* and *end* are zero based. If only the start if specified, only that sector is mapped.

Old-style:

$$\mathit{device\_name}\,\big[\,:\mathit{BIOS\_code}\big]\,\texttt{\#}\mathit{start}\,\big[\texttt{-}\mathit{end}\,\big|\,\texttt{+}\mathit{number}\big]$$

Example: `/dev/fd0#1+512`

New-style:

$$\texttt{image=}\mathit{name}$$

The following additional variables are recognized: `root`, `vga`, `ramdisk`, `password`, `restricted`, `label`, `alias`, `optional` and `range`.

Example:

```
image = /dev/fd0
  range = 1+512
```

[16] That device may not be available at the time LILO is installed and it is therefore impossible to read the settings from the image file.

5.2.3 Booting unstripped kernel executables

Unstripped kernel executables contain no root device information. The setup code of the kernel has also to be added to the kernel. First, it has to be copied to a suitable place and its header has to be removed, e.g.

```
dd if=/usr/src/linux/boot/setup of=/etc/lilo/setup.b \
  bs=32 skip=1
```

If this command is placed at the beginning of `/etc/lilo/install`, `setup.b` is automatically updated whenever that script is run because anything else changes.

Old-style:

The image specification looks like this:

$$setup\_name\,[:BIOS\_code]\,\texttt{+}kernel\_name\,[:BIOS\_code]$$

Example: `/etc/lilo/setup.b+/usr/src/linux/tools/system`

New-style:

`unstripped=`*name*

The following additional variables are recognized: `root`, `vga`, `ramdisk`, `password`, `restricted`, `label`, `alias`, `optional`, `boot` and `setup`.

`boot=`*boot_sector* specifies the floppy boot sector that should be loaded. It is used to provide defaults for settings like the root device, etc. The boot sector file can be created like the setup file. (See above.) The recommended name for the floppy boot sector file is `/etc/lilo/bsect.b`. If this variable is omitted, a zero-filled sector is loaded instead.

`setup=`*setup_file* specifies the setup file that should be loaded. This variable is required.

Example:

```
unstripped = /usr/src/linux/tools/system
  setup = /etc/lilo/setup.b
```

5.2.4 Booting compound unstripped kernels

Compound unstripped kernels are like ordinary unstripped kernels, but the kernel proper, the setup code and the floppy boot sector are in a single file instead of three files. This avoids the risk of version conflicts and generally makes the kernel easier to handle.

Compound kernels are generated with the `mkck` program. `mkck` is run in the top-level directory of the kernel source and writes the kernel to the file that has been specified on the command line. If no file has been specified, the file name `/linux` is used. `mkck` should be run by the kernel `Makefile`, e.g.

```
linux: $(CONFIGURE) Image
        if [ -f /linux ]; then mv /linux /linux.old; fi
        /etc/lilo/mkck
        /etc/lilo/install
```

`compound=`*name*

The following additional variables are recognized: `root`, `vga`, `ramdisk`, `password`, `restricted`, `label`, `alias` and `optional`.

Example:

```
compound = /linux
```

5.2.5 Booting a foreign operating system

LILO can even boot other operating systems, i.e. MS-DOS. To boot an other operating system, the name of a loader program, the device that contains the boot sector and the device that contains the partition table have to be specified.

The boot sector is merged with the partition table and stored in the map file.

Currently, only the loader `chain.b` exists. `chain.b` simply starts the specified boot sector.[17]

Old-style:

loader`+`*boot_dev* [`:` *BIOS_code*] `@` [*part_dev*]

Example: `/etc/lilo/chain.b+/dev/hda2@/dev/hda`

The name of the device that contains the partition table can be omitted if the respective operating system has other means to determine from which partition it has been booted. (E.g. MS-DOS usually stores the geometry of the boot disk or partition in its boot sector.)

New-style:

`other=`*name*

[17] The boot sector is loaded by LILO's secondary boot loader before control is passed to the code of `chain.b`.

The following additional variables are recognized: `password`, `restricted`, `label`, `alias`, `optional`, `loader`, `table` and `unsafe`.

`loader=`*chain_loader* specifies the chain loader that should be used. If this variable is omitted, `/etc/lilo/chain.b` is used.

`table=`*device* specifies the device that contains the partition table. LILO does not pass partition information to the booted operating system if this variable is omitted. (Some operating systems have other means to determine from which partition they have been booted.)

`unsafe` do not access the boot sector at map creation time. This disables some sanity checks. If the boot sector is on a floppy disk device, using `unsafe` avoids the need to put a readable disk into the drive when running the map installer. `unsafe` and `table` are mutually incompatible.

Example:

```
other = /dev/hda2
  table = /dev/hda
```

5.3 Disk parameter table

For floppies, IDE disks (or MFM, RLL, ESDI, etc.) and some SCSI disks, LILO can obtain the disk geometry information from the kernel. Unfortunately, this is not possible with all SCSI adapters[18] or with kernels older than 0.99pl0. In such cases, LILO reports either the error
`geo_query_dev HDIO_REQ`
or
`Device 0x`*number*`:  Got bad geometry` *sec*/*hd*/*cyl*

The file `/etc/lilo/disktab` is used to describe such disks. For each device (`/dev/sda` → 0x800, `/dev/sda1` → 0x801, etc.), the BIOS code, the disk geometry and the offset of the first sector of that partition (measured in sectors) have to be specified, e.g.

```
# /etc/lilo/disktab  -  LILO disk parameter table
#
# This table contains disk parameters for SCSI disks and non-
# standard parameters of IDE disks. Parameters in disktab
# _always_ override auto-detected disk parameters.
```

[18] The 0.99pl1 kernel provides geometry information for Adaptec AHA-1542 and AHA-1740, Future Domain TMC-1660 and TMC-1680 and WD-7000 adapters.

```
# Dev.  BIOS   Secs/   Heads/  Cylin-  Part.
# num.  code   track   cylin.  ders    offset

0x800   0x80   32      64      631     0       # /dev/sda
0x801   0x80   32      64      631     32      # /dev/sda1
0x802   0x80   32      64      631     204800  # /dev/sda2
```

(Those parameters are just a random example from my system. However, many SCSI controllers re-map the drives to 32 sectors and 64 heads. The number of cylinders does not have to be exact, but it shouldn't be lower than the number of effectively available cylinders.)

Note that the device number and the BIOS code have to be specified as hexadecimal numbers with the "0x" prefix.

The disk geometry parameters can be obtained by booting MS-DOS and running the program `DPARAM.COM` with the hexadecimal BIOS code of the drive as its argument, e.g. `dparam 0x80` for the first hard disk. It displays the number of sectors per track, the number of heads per cylinder and the number of cylinders.

The partition offset is printed by the Linux kernel when the SCSI disk is detected at boot time. Example:

```
sd0 :
 part 1 start 32 size 204768 end 204799
 part 2 start 204800 size 1087488 end 1292287
```

The first partition has an offset of 32 sectors, the second has an offset of 204800 sectors.

Because many SCSI controllers don't support more than 1 GB when using the BIOS interface, LILO can't access files that are located beyond the 1 GB limit of large SCSI disks on such controllers and reports errors in these cases.

6 Installation and updates

6.1 Compatibility

LILO 0.7 (and newer) is compatible with all kernels that don't support presetting of a swap device. This change has probably occurred at some 0.97 version. If the swap device preset code is disabled, you can also use 0.96c-pl1 (or newer) kernels. The kernel header files have to be in `/usr/include/linux` and if the kernel can be configured by running `make config`, this usually has to be done before LILO can be compiled.

`/bin/sh` has to be a real bourne shell. bash is sufficiently compatible, but some ksh clones may cause problems.

Using command-line options that are handled by init usually requires a comparably new version of init. All currently available init packages typically support new features of LILO in the respective next release.

A file named `INCOMPAT` is included in the distribution. It describes incompatibilities to older versions of LILO and may also contain further compatibility notes.

6.2 Quick installation

If you want to install LILO on your hard disk and if you don't want to use all its features, you can use the quick installation script. Read `QuickInst` for details.

QuickInst can only be be used for first-time installations, **not** to update or modify an existing installation of LILO. Be sure you've extracted LILO into a directory that doesn't contain any files of other LILO installations.

6.3 Normal first-time installation

First, you have to install the LILO files:

- extract all files from `lilo.`*version*`.tar.Z` in a new directory.[19]
- configure the `Makefile` (see section 6.7)
- run `make` to compile and assemble all parts.
- run `make install` to copy all LILO files to `/etc/lilo`. `/etc/lilo` should now contain the following files: `boot.b`, `chain.b`, `disktab` and `lilo`.

[19] Don't use `/etc/lilo` for the LILO source. Standard installation procedures fail if the source is in the same directory as the production binaries.

If you want to use LILO on a SCSI disk, you might have to determine the parameters of your SCSI disk(s) and put them into the file `/etc/lilo/disktab`. See section 5.3 for details.

The next step is to test LILO with the boot sector on a floppy disk:

- insert a blank (but low-level formatted) floppy disk into `/dev/fd0`.
- cd to `/etc/lilo`.
- run `./lilo -b /dev/fd0 -i boot.b -v -v -v` *kernel_image(s)*[20]
- reboot. LILO should now load its boot loaders from the floppy disk and then continue loading the kernel from the hard disk.

Now, your have to decide, which boot concept you want to use. Let's assume you have a Linux partition on `/dev/hda2` and you want to install your LILO boot sector there. The DOS-MBR loads the LILO boot sector.

- get a working bootimage and a rootimage, or a bootable rootimage. Verify that you can boot with this setup and that you can mount your Linux partition(s) with it.
- if the boot sector you want to overwrite with LILO is of any value (e.g. it's the MBR or it contains a boot loader you might want to use if you encounter problems with LILO), you should mount your rootimage and make a backup copy of your boot sector to a file on that floppy, e.g. `dd if=/dev/hda of=/fd/boot_sector bs=512 count=1`
- create a shell script `/etc/lilo/install` that installs and updates LILO on your hard disk, e.g.[21]
  `$ROOT/etc/lilo/lilo` *maybe_some_options* `-C /etc/lilo/config $* \`
  Then create the configuration file `/etc/lilo/config`, e.g.
  `install = /etc/lilo/boot.b`
  image specifications
  ...
  Be sure to use absolute paths for all files. Relative paths may cause unexpected behaviour when using the `-r` option.
- Now, you can check what LILO would do if you were about to install it on your hard disk:
  `/etc/lilo/install -v -v -v -t`

[20] If you've already installed LILO on your system, you might not want to overwrite your old map file. Use the `-m` option to specify an alternate map file name.

[21] A standard `/etc/lilo/install` named `install.new` is contained in the LILO distribution.

- If you need some additional boot utility (i.e. BOOTACTV), you should install that now.
- Run `/etc/lilo/install` to install LILO on your hard disk.
- If you have to change the active partition, use (e)fdisk or activate to do that.
- Reboot.

6.4 LILO update

When updating to a new version of LILO, the initial steps are the same as for a first time installation: extract all files, configure the `Makefile`, run `make` to build the executables and run `make install` to move the files to `/etc/lilo`.

The old versions of `boot.b` and `chain.b` are automatically renamed to `boot.old` and `chain.old`. This is done to ensure that you can boot even if the installation procedure is not finished. `boot.old` and `chain.old` can be deleted after the map file is rebuilt.

Because the locations of `boot.b` and `chain.b` have changed and because the map file format may be different too, you have to update the boot sector and the map file. Run `/etc/lilo/install` to do this.

6.5 Migration from old-style to new-style configuration

An existing pre-LILO 0.7 configuration can easily be replaced by a new-style configuration. The command-line option `-D` dumps the current settings on standard output. This is a typical migration procedure:

- upgrade to LILO 0.7 (or later)
- `cd` to `/etc/lilo`
- run `./install -D >config`
- add new variables `video` and `ramdisk` to `/etc/lilo/config` if you want to (and if your kernel supports it.)
- run `./lilo -C config -D | diff -c config -`
  to verify the correctness of the dump.[22]

[22] LILO may change the order of variables and the placement of whitespace in the second dump. It also removes comments.

- rename `/etc/lilo/install` to `/etc/lilo/install.old` and copy the standard install script `install.new` from your LILO source directory to `/etc/lilo/install`
- add commands to create boot and setup files (see section 5.2.3) to the new `/etc/lilo/install` if you're booting unstripped kernels.
- run `/etc/lilo/install`

6.6 Kernel update

Whenever any of the kernel files that are accessed by LILO is moved or overwritten, the map has to be re-built.[23] Run `/etc/lilo/install` to do this.

If the boot or setup code has changed and if you're booting unstripped kernels, you also have to update `bsect.b` and `setup.b`. This should be done in `/etc/lilo/install`, as described in section 5.2.3.

If you're frequently re-compiling the kernel, you should put an invocation of `/etc/lilo/install` into the kernel's top Makefile.

Example (.../`linux/Makefile`):

```
...
Image: boot/bootsect boot/setup tools/system tools/build
       cp tools/system system.tmp
       strip system.tmp
       tools/build boot/bootsect boot/setup system.tmp \
            $(ROOT_DEV) >/linux
       /etc/lilo/install
       rm system.tmp
       sync
...
```

Recent kernels already have a make target "lilo" that copies the kernel to `/vmlinux` and runs `/etc/lilo/install`

6.7 Build-time configuration

Starting with LILO 0.9, certain build-time parameters can be configured. They can either be edited in the top-level `Makefile` or they can be stored in a file `/etc/lilo/config.defines`. Settings in the `Makefile` are ignored if that file exists.

[23] It is advisable to keep a second, stable, boot image that can be booted if you forget to update the map after a change to your usual boot image.

The following items can be configured:

`IGNORECASE` Makes image name matching case-insensitive, i.e. "linux" and "Linux" are identical. This option is enabled by default. Note that password matching is always case-sensitive.

`NOAUTO` Disables putting the word `auto` on the command line when booting unattendedly. This is necessary to maintain the ability to enter single user mode when using an `init` that expects `single` to be the first word on the command line. `NOAUTO` is disabled by default.

`ONE_SHOT` Disables the command-line timeout (configuration variable `TIMEOUT`) if any key is pressed. This way, very short timeouts can be used if `PROMPT` is set. `ONE_SHOT` is disabled by default.

`READONLY` Disallows overwriting the default command line sector of the map file. This way, command lines set with `-R` stay in effect until they are explicitly removed. This option is disabled by default.

`/etc/lilo/config.defines` should be used if one wishes to make permanent configuration changes. The usual installation procedures don't touch that file. Example:

```
-DIGNORECASE -DONE_SHOT
```

7 Technical overview

This section contains a description of several internals of LILO. It is not necessary to understand them in order to install or use LILO.

7.1 Load sequence

The boot sector is loaded by the ROM-BIOS at address 0x07C00. It moves itself to address 0x9A000, sets up the stack (growing downwards from 0x9B000 to 0x9A200), loads the secondary boot loader at address 0x92000 and transfers control to it. It displays an "L" after moving itself and an "I" before starting the secondary boot loader.

The secondary boot loader loads the descriptor table at 0x9D200 and the sector containing the default command line at 0x9D600. If the default command line is enabled, its magic number is invalidated and the sector is written back to disk. This potentially dangerous operation can be disabled by defining `READONLY` when passing `second.S` through `cpp`. Next, the secondary boot loader checks for user input. If either the default is used or if the user has specified an alternate image, the floppy boot sector of that image is loaded at 0x90000[24], the setup part is loaded at 0x90200 and the kernel part is loaded at 0x10000. If the image being booted is not an unstripped kernel, the memory area 0x10000-0x8FFFF is zeroed before loading. During the load operation, the sectors of the map file are loaded at 0x9D000.

If the loaded image is a traditional boot image, control is transferred to its setup code. If it is an unstripped kernel, its BSS is zeroed first. If a different operating system is booted, things are a bit more difficult: the chain loader is loaded at 0x90200 and the boot sector of the other OS is loaded at 0x90400. The chain loader moves the partition table (loaded at 0x903BE as part of the chain loader) to 0x00600 and the boot sector to 0x07C00. After that, it passes control to the boot sector.

The secondary boot loader displays an "L" after being started and an "O" after loading the descriptor table.

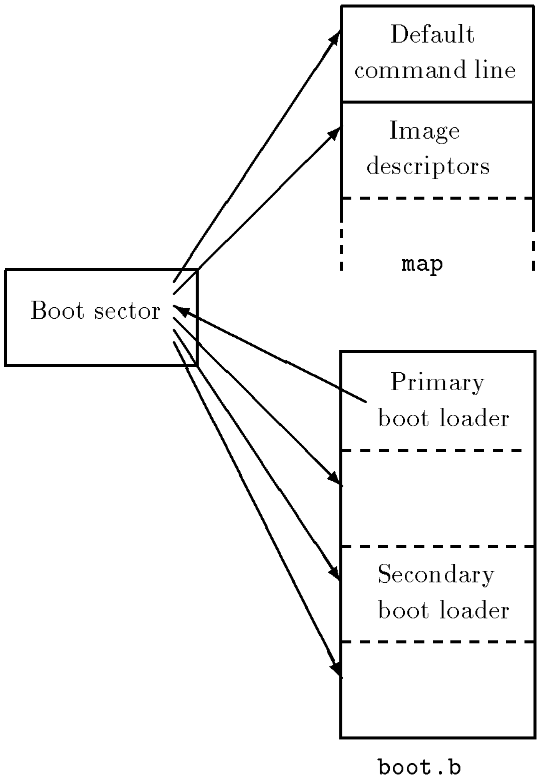
[24] The floppy boot sector is only used as a source of setup information.

Address	Area	Size
`0x00000`		1982 bytes
`0x007BE`	Partition table	64 bytes
`0x007FE`		29 kB
`0x07C00`	Boot load area	512 bytes
`0x07E00`		32.5 kB
`0x10000`	Kernel	448 kB
`0x80000`		192 kB
`0x90000`	Floppy boot sector	512 bytes
`0x90200`	Setup (kernel)	39.5 kB (2 kB used)
`0x9A000`	Primary boot loader	512 bytes
`0x9A200`	Stack	3.5 kB
`0x9B000`	Secondary boot loader	8 kB (3.5 kB used)
`0x9D000`	Map load area	512 bytes
`0x9D200`	Descriptor table	1 kB
`0x9D600`	Default command line	512 bytes
`0x9D800`		10 kB
`0xA0000`		

The area 0x90020-0x90023 is overlaid by command-line information while the secondary boot loader is running.

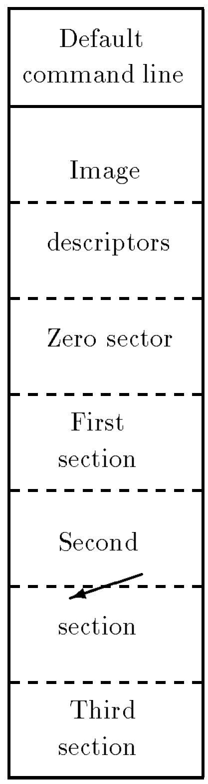
7.2 File references

This section describes the references among files involved in the boot procedures.



The boot sector contains the primary boot loader, the address of the default command line sector, the address of both descriptor table sectors and the addresses of the sectors of the secondary boot loader. The boot sector is copied from `boot.b`.

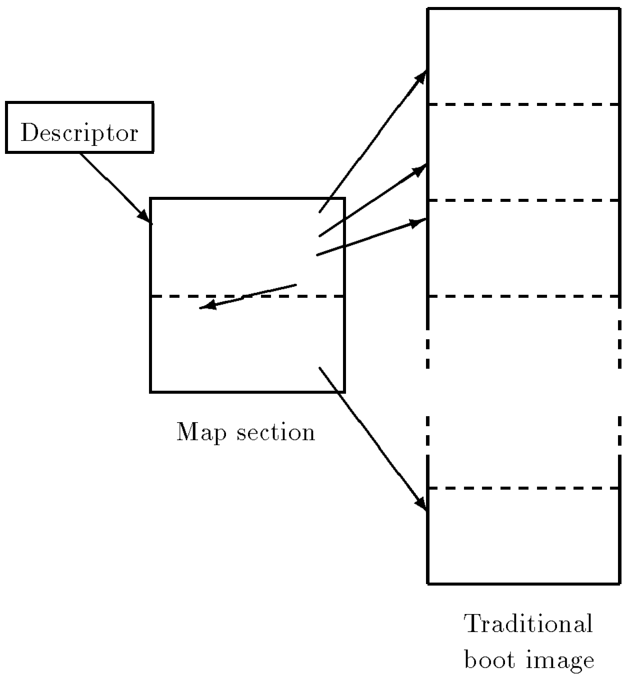
The primary boot loader can store up to eight sector addresses of the secondary boot loader.



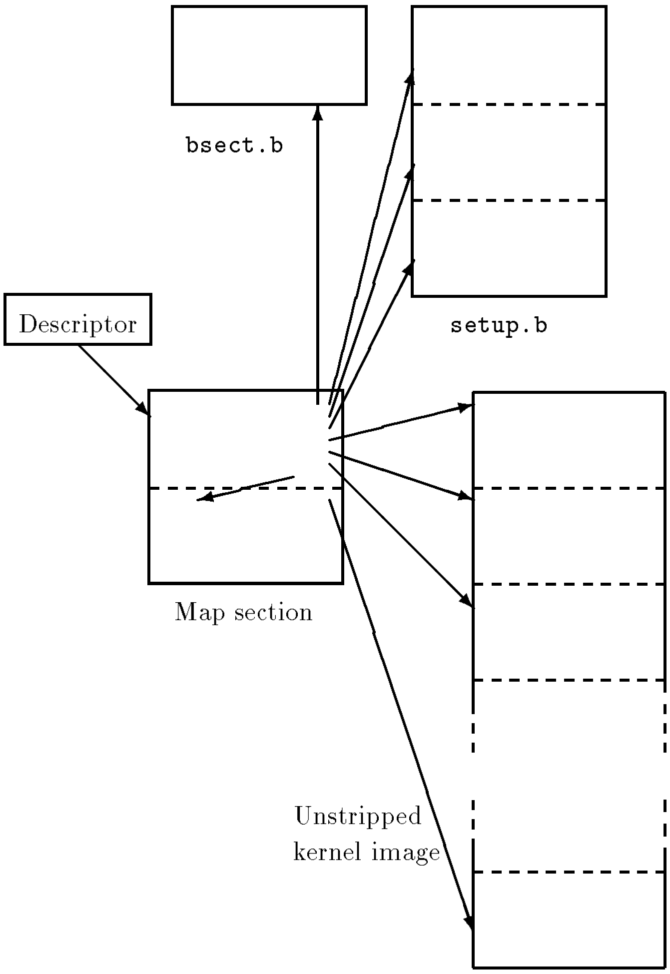
The map file consists of so-called sections and of special data sectors. Each section spans an integral number of disk sectors and contains addresses of sectors of other files.[25] The last address slot of each sector is either unused (if the map ends in this sector) or contains the address of the next sector in the section.

The four sectors at the beginning of the map file are special: the first sector contains the default command line, the next two sectors contain the boot image descriptor table and the fourth sector is filled with zero bytes. This sector is mapped whenever a file contains a "hole".

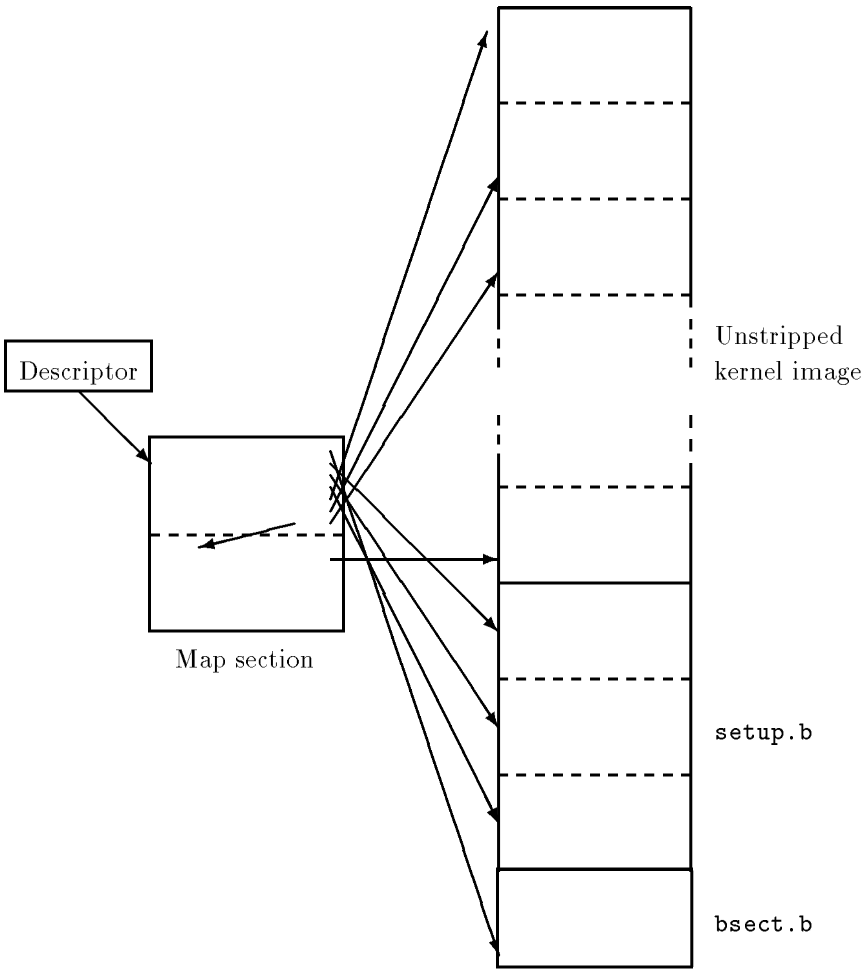
[25] There are two exceptions: 1. If a "hole" is being covered or if the floppy boot sector of an unstripped kernel has been omitted, the address of the zero sector is used. This sector is part of the map file. 2. When booting a different operating system, the first sector is the merged chain loader that has been written to the map file before that section.



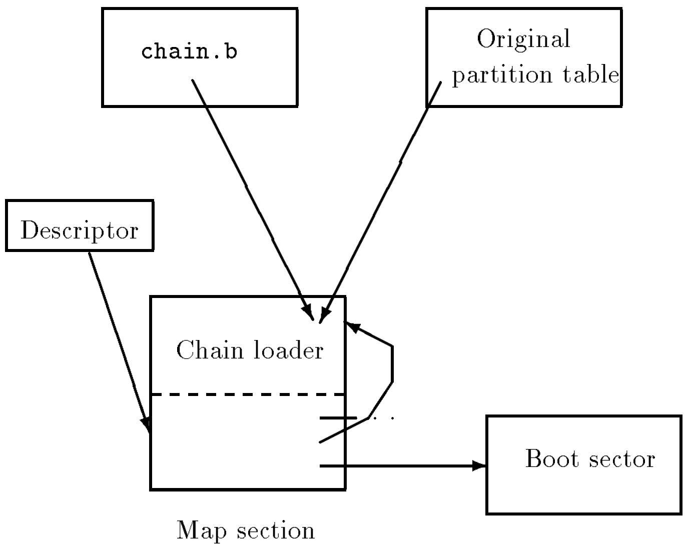
A traditional boot image consists simply of a sequence of sectors that are loaded. Images that are loaded from a device are treated exactly the same way as images that are loaded from a file.



Unstripped kernels consist of the floppy boot sector, the setup part and of the kernel file. The descriptor also contains information about the start and the size of the BSS segment. The boot loader clears BSS before starting the kernel.



Compound kernels are like unstripped kernels, but the floppy boot sector and the setup code are appended to the kernel file. The descriptor contains information about the start and the size of the BSS segment. The boot loader clears BSS before starting the kernel.



When booting another operating system, the chain loader (`chain.b`) is merged with the partition table[26] and written into the map file. The map section of this boot image starts after that sector and contains only the address of a dummy floppy boot sector (the zero sector, but its contents are irrelevant), the loader sector and of the boot sector of the other operating system.

7.3 Configuration parameters

The boot sector of each kernel contains a set of configuration parameters that have to be available at boot time before the kernel can access file systems. These parameters can be set when the kernel is compiled and later be changed with programs like `rdev`. LILO can overwrite the parameters at boot time.

The parameters are stored at the following (decimal) addresses:

502 this parameter is currently unused.

504 the size of the RAM disk in kilobytes. No RAM disk is created if this parameter is set to zero.

506 the text mode the VGA is set to.

0xFFFD the user is asked to specify the VGA mode at boot time.

0xFFFE uses 80x50 ("extended") mode.

[26] If the partition table is omitted, that area is filled with zero bytes.

0xFFFF uses 80x25 ("normal") mode.

Any other value selects the corresponding mode as displayed in the interactive VGA mode selection menu.

508 the minor number of the device that should be mounted as root.

509 the major number of the device that should be mounted as root.

7.4 Command-line interface

0.98pl3 and later kernels support processing of parameters that are provided by the boot loader. The parameter string is a NUL-terminated ASCII string that contains space-separated words or *variable=value* pairs. See section 4 for how they are interpreted.

The following descriptor has to be set up to pass a parameter string to the kernel:

0x90020 the magic number 0xA33F.

0x90022 the offset of the first byte of the command line relative to 0x90000.

7.5 External interface

LILO is able to receive its command line from a program that is booted before it. This externally provided command line is only used if the user does not use the normal mechanism to invoke the boot prompt.

The following register contents are expected:

DL contains the value 0xFE.

ES:SI points to the string "LILO". The string must be in upper case and no terminating character is needed. The string must not cross segment boundaries, i.e. **SI** must be below 0xFFFD.

ES:BX points to a NUL-terminated string that is used as the command line. This string has a maximum length of 78 characters (not including the terminating NUL) and must not cross segment boundaries.

There are two values of the externally provided command line that have a special meaning:

- an empty string (**ES:BX** points to a NUL byte) is interpreted as a request to enter the boot prompt and to accept keyboard input.

- a string that consists only of blanks is interpreted as a request to boot the default boot image.

LILO can also obtain the default command line from the map file. This command line is only used if no externally provided command line is available.

7.6 Default command line in map file

The first sector of the map file is reserved for a default command line. Unless the user invokes the boot prompt by pressing a shift key or unless an externally provided command line is present, the command line in the map file is interpreted as if it had been typed on the keyboard.

The first two bytes of the first sector of the map file have to contain the magic number `DC_MAGIC` (0xF4F2) in little-endian byte order. They are followed by a NUL-terminated string with a maximum length of 510 bytes, including the NUL. Note that the boot loader limits command lines to 78 characters after removing duplicate spaces.

The command line is disabled by either clobbering the magic number or by using an empty string (i.e. only a NUL byte) as the command line.

8 Troubleshooting

All parts of LILO display some messages that can be used to diagnose problems.

8.1 Map installer errors

Most messages of the map installer (`/etc/lilo/lilo`) should be self-explanatory. Some messages that indicate common errors are listed below:

`Can't put the boot sector on logical partition` *number*
: An attempt has been made to put LILO's boot sector on the current root file system partition which is on a logical partition. This usually doesn't have the desired effect, because common MBRs can only boot primary partitions. This check can be bypassed by explicitly specifying the boot partition with the `-b` option.

`Device 0x`*number*`: Got bad geometry` *sec/hd/cyl*
: The device driver for your SCSI controller does not support geometry detection. You have to use an `/etc/lilo/disktab` file.

`Device 0x`*number*`: Invalid partition table, entry` *number*
: The 3D and linear addresses of the first sector of the specified partition don't correspond. This is typically caused by partitioning a disk with a program that doesn't align partitions to tracks and using PC/MS-DOS or OS/2 later on that disk. LILO can attempt to correct the problem, see page 24.

`First sector of` *device* `doesn't have a valid boot signature`
: The first sector of the specified device does not appear to be a valid boot sector. You might have confused the device name.[27]

`geo_comp_addr: Cylinder number is too big (`*number* > `1023)`
: A file is located beyond the 1023th cylinder of a hard disk. LILO can't access such files, because the BIOS limits cylinder numbers to the range 0...1023. Try moving the file to a different place, preferably a partition that is entirely within the first 1023 cylinders of the disk.

`geo_query_dev HDIO_REQ`
: Your kernel does not support geometry detection for your SCSI disk(s). You have to use an `/etc/lilo/disktab` file.

[27] Because different partition programs may display the partitions in a different order, it is possible that what you think is your first partition isn't `/dev/hda1`, etc. A good method to verify the content of a partition is to try to mount it.

`Hole found in map file (`*location*`)`
: The map installer is confused about the disk organization. If you're running version 0.4 of LILO, upgrade to 0.5 or newer. Otherwise, please report this error.

item `doesn't have a valid LILO signature`
: The specified item has been located, but is not part of LILO. If *item* is the first boot sector, you've probably forgotten to specify the `-i` option or the `install` variable to install the LILO boot sector.

item `has an invalid stage code (`*number*`)`
: The specified item has probably been corrupted. Try re-building LILO.

item `is version` *number*`. Expecting version` *number*`.`
: The specified entity is either too old or too new. Make sure all parts of LILO (map installer, boot loaders and chain loaders) are from the same distribution. If *item* is the first boot sector, you have to run the map installer with the `-i` option to update it.[28]

`Kernel` *name* `is too big`
: The kernel image (without the setup code) is bigger than 512 kbytes. LILO would overwrite itself when trying to load such a kernel. Try removing some unused drivers and compiling the kernel again.

`Partition entry not found`
: The partition from which an other operating system should be booted isn't listed in the specified partition table. This either means that an incorrect partition table has been specified or that you're trying to boot from a logical partition. The latter usually doesn't work. You can bypass this check by omitting the partition table specification (e.g. `/dev/hda2@` or omitting the variable `table`).

`Sorry, don't know how to handle device` *number*
: LILO uses files that are located on a device for which there is no easy way to determine the disk geometry. Such devices have to be described in the file `/etc/lilo/disktab`.

8.2 LILO start message

When LILO loads itself, it displays the word "LILO". Each letter is printed before or after performing some specific action. If LILO fails at some point, the letters printed so far can be used to identify the problem. (See also section 7.1.)

[28] The expected version number may be different from the version number of the LILO package.

(*nothing*) No part of LILO has been loaded. LILO either isn't installed or the partition on which its boot sector is located isn't active.

`L` The first stage boot loader has been loaded and started, but it can't load the second stage boot loader. This usually indicates a media failure or a geometry mismatch (e.g. bad parameters in `/etc/lilo/disktab`).

`LI` The first stage boot loader was able to load the second stage boot loader, but has failed to execute it. This can either be caused by a geometry mismatch or by moving `/etc/lilo/boot.b` without running the map installer.

`LIL` The second stage boot loader has been started, but it can't load the descriptor table from the map file. This is typically caused by a media failure or by a geometry mismatch.

`LIL?` The second stage boot loader has been loaded at an incorrect address. This is typically caused by a subtle geometry mismatch or by moving `/etc/lilo/boot.b` without running the map installer.

`LIL-` The descriptor table is corrupt. This can either be caused by a geometry mismatch or by moving `/etc/lilo/map` without running the map installer.

`LILO` All parts of LILO have been successfully loaded.

8.3 Disk error codes

If the BIOS signals an error when LILO is trying to load a boot image, the respective error code is displayed. The following BIOS error codes are known:

`0x00` *Internal error.* This code is generated by the sector read routine of the LILO boot loader whenever an internal inconsistency is detected. This might be caused by corrupt files, try re-building the map file.

`0x01` *Illegal command.* This shouldn't happen.

`0x02` *Address mark not found.* This usually indicates a media problem. Try again several times.

`0x03` *Write-protected disk.* This shouldn't happen.

`0x04` *Sector not found.* This typically indicates a geometry mismatch. If you're booting a raw-written disk image, verify whether it was created for disks with the same geometry as the one you're using. If you're booting from a SCSI disk, you should check, whether LILO has obtained correct geometry data from the kernel or whether the contents of your `/etc/lilo/disktab` file corresponds to the real disk geometry. (See section 5.3.)

`0x06` *Change line active.* This should be a transient error. Try booting a second time.

`0x08` *DMA overrun.* This shouldn't happen. Try booting again.

`0x09` *DMA attempt across 64k boundary.* This shouldn't happen. Try omitting the `-c` option and report this problem to the author.

`0x0C` *Invalid media.* This shouldn't happen and might be caused by a media error. Try booting again.

`0x10` *CRC error.* A media error has been detected. Try booting several times, running the map installer a second time (to write "good data" over the bad spot), mapping out the bad sectors/tracks and, if all else fails, replacing the media.

`0x20` *Controller error.* This shouldn't happen.

`0x40` *Seek failure.* This might be a media problem. Try booting again.

`0x80` *Disk timeout.* The disk or the drive isn't ready. Either the media is bad or the disk isn't spinning. If you're booting from a floppy, you might not have closed the drive door. Otherwise, trying to boot again might help.