LILO
Generic boot loader for Linux
Version 0.6

Werner Almesberger
almesber@nessie.cs.id.ethz.ch

October 28, 1992

Contents

1 Disk organization **1**

2 Booting basics **4**

3 Choosing the "right" boot concept **9**

4 Technical overview **12**

5 The boot prompt **19**

6 Map installer **20**

7 Installation and updates **25**

8 Troubleshooting **28**

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. The root device can be set independently for each of them. LILO can even be used

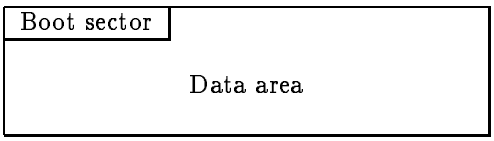
[1] MS-DOS, DR DOS, OS/2, 386BSD, ...

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.

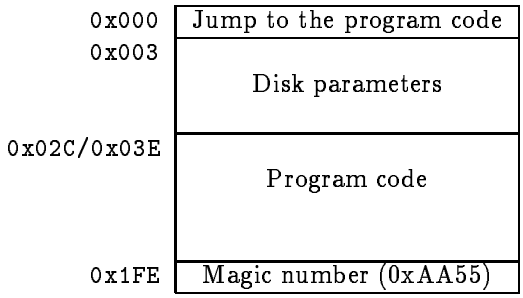
1 Disk organization

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.

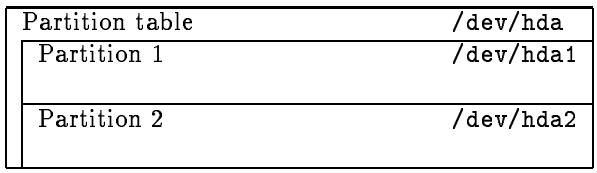
The MS-DOS boot sector has the following structure:



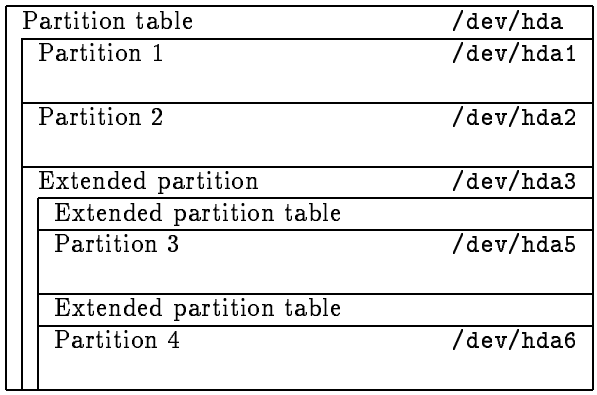
LILO uses a similar boot sector, but it does not contain the disk parameters part. This is no problem for Minix or EXT file systems, because they don't look at the boot sector, but putting a LILO boot sector on an MS-DOS file system would make it inaccessible for MS-DOS.

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]



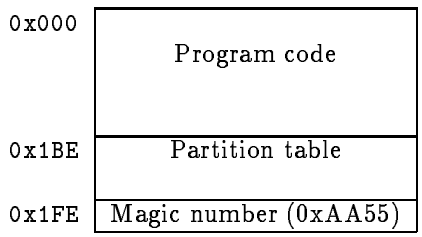
The entire disk can be accessed as `/dev/hda`, `/dev/hdb`, `/dev/sda`, etc. The primary partitions are `/dev/hda1` ... `/dev/hda4`.



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).



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

The LILO boot sector is designed to be usable as a partition boot sector. 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.
- on the second hard disk. (Unless for backup installations or if the current first disk will be removed or disabled.)

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 the disk parameters (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

[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.

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 $\longrightarrow$	**MS-DOS** $\longrightarrow$	**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 $\longrightarrow$	**MS-DOS** $\longrightarrow$	**COMMAND.COM**	
	$\longrightarrow$	**BOOTLIN** $\longrightarrow$	**Linux**

A typical BOOTLIN setup: everything happens like when booting MS-DOS, but in `CONFIG.SYS`, BOOTLIN is invoked. 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.[4]
- 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`.

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

2.3 LILO started by DOS-MBR

Master Boot Record	Boot sector	Operating system
DOS-MBR ⟶	**LILO** ⟶	**Linux**
⟶	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, 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.

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).[5]

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`

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

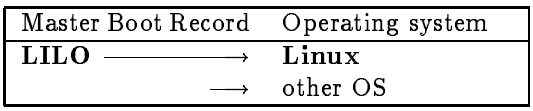
Master Boot Record	Boot sector	Operating system
BOOTACTV $\longrightarrow$	**Shoelace** $\longrightarrow$	**Linux**
$\longrightarrow$	other OS	

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.[6]

[5] 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.

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

2.7 LILO alone



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`
- 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.[7]

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.[8]

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.

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, an example `/etc/lilo/install` file is provided. This file is a shell script that actually performs the installation (and updates) of the boot

[7] 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 `nic.funet.fi:/pub/OS/Linux/tools/boot142.zip` or `wuarchive.wustl.edu:/mirrors/msdos/sysutl/boot142.zip`.

[8] pfdisk is available for anonymous FTP from `tsx-11.mit.edu:/pub/linux/attic/sources/sbin/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` `nic.funet.fi:/pub/OS/Linux/tools/pboot.zip`.

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 7.1.)

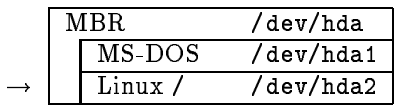
Each of the `/etc/lilo/install` files installs the following boot images:

- 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.

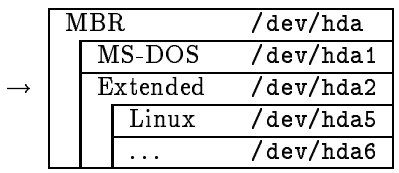


A typical `/etc/lilo/install` file would look like this:

```
$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.



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

```
$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
```

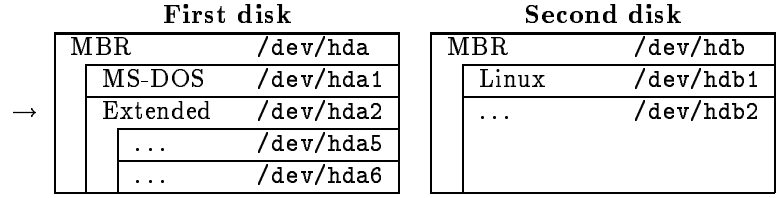
3.3 Two disks, Linux (at least partially) on the first disk

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.



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

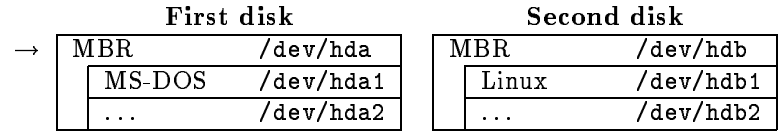
```
$ROOT/etc/lilo/lilo -b /dev/hda2 -c -i /etc/lilo/boot.b  $* \
    /linux \
    /linux.backup \
    msdos=/etc/lilo/chain.b+/dev/hda1@/dev/hda
```

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.



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/install` file for this configuration would look like this:

```
$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 Technical overview

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

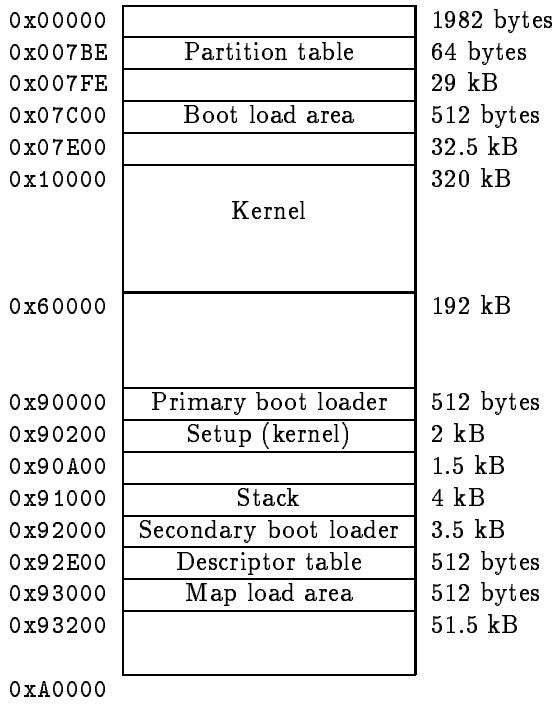
4.1 Load sequence

The boot sector is loaded by the ROM-BIOS at address 0x07C00. It moves itself to address 0x90000, sets up the stack (growing downwards from 0x92000 to 0x91000), 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 0x92E00 and checks for user input. If either the default is used or if the user has specified an alternate image, the setup part of that image 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 0x93000.

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.

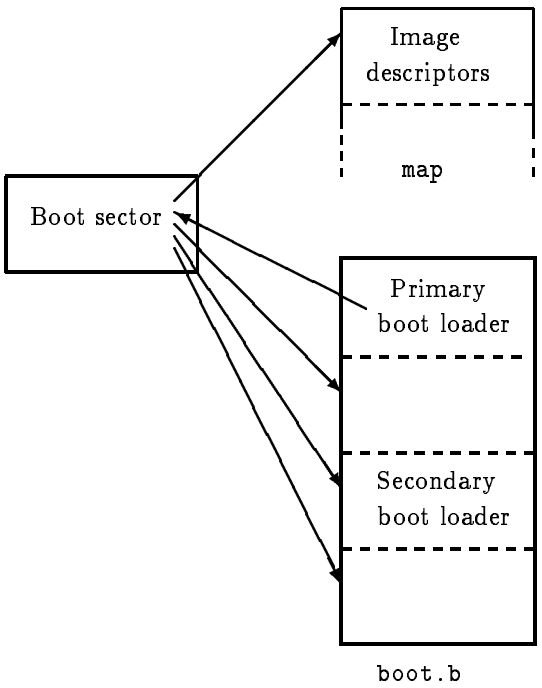
The secondary boot loader displays an "L" after being started and an "O" after loading the descriptor table.



The area 0x92020-0x92023 is overlaid by command-line information while the secondary boot loader is running.

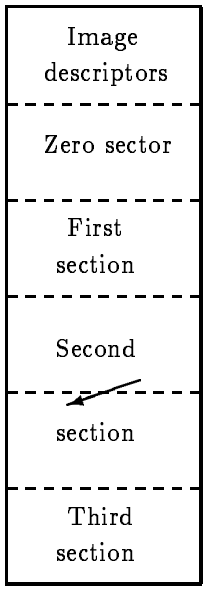
4.2 File references

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



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

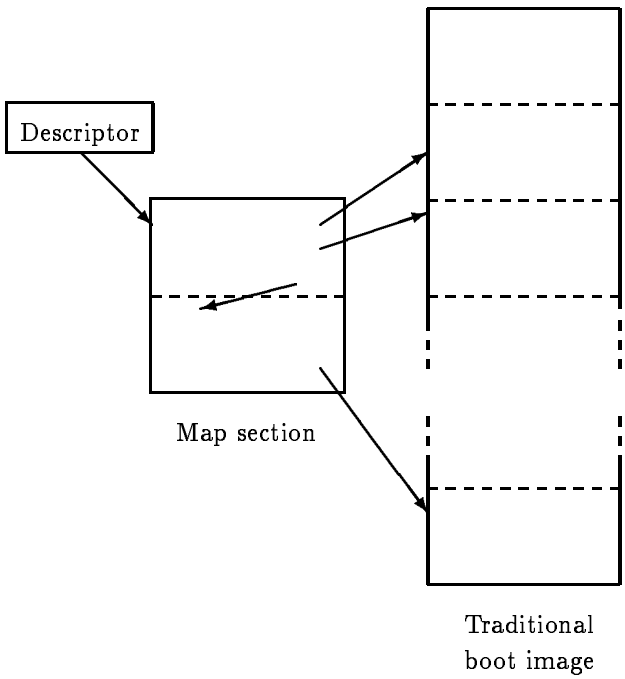
The primary boot loader can store up to four sector addresses of the secondary boot loader.



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.[9] 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.

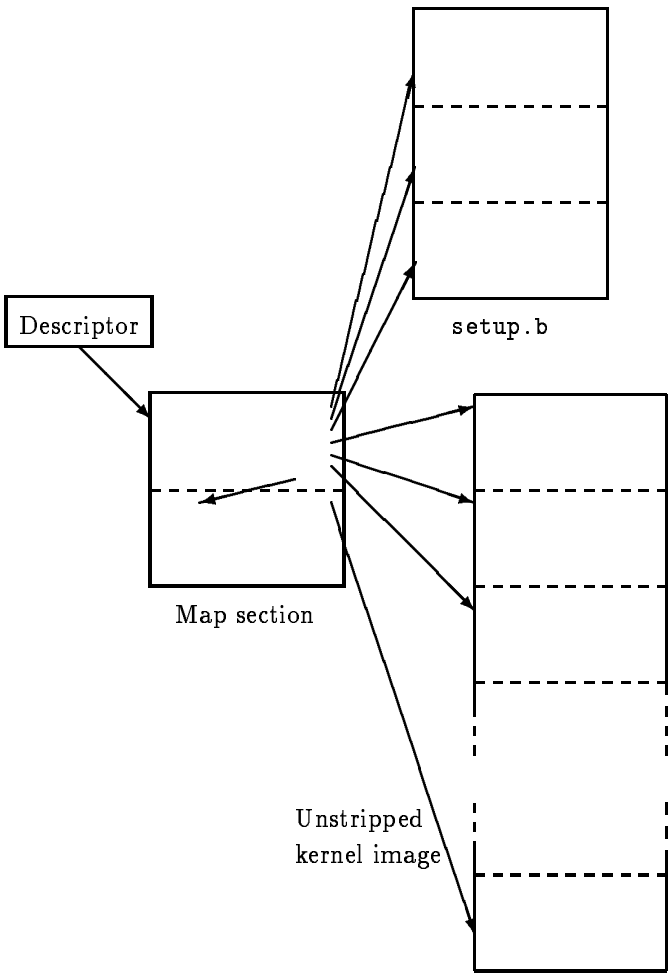
The two sectors at the beginning of the map file are special: the first sector contains the boot image descriptor table and the second sector is filled with zero bytes. This sector is mapped whenever a file contains a "hole".

[9] There are two exceptions: 1. If a "hole" is being covered, 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.

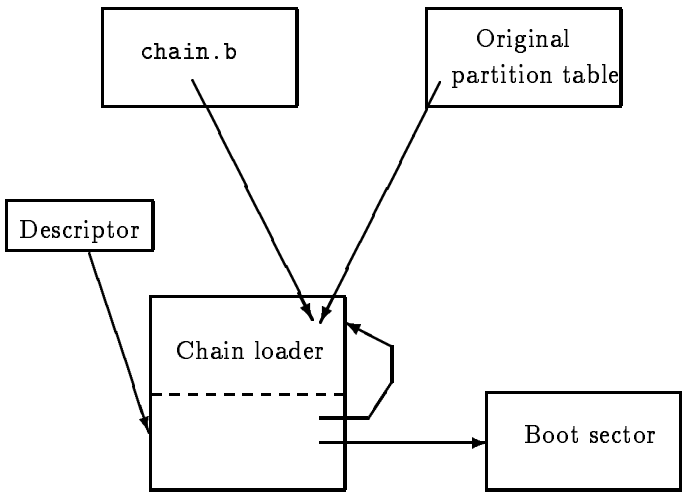


A traditional boot image consists simply of a sequence of sectors that are loaded. Images that are loaded from a device are treated exactly the same way as images that are loaded from a file.

The first sector of the boot image contains the floppy boot sector and is not mapped.



Unstripped kernels consist of 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.



When booting another operating system, the chain loader (`chain.b`) is merged with the partition table[10] and written into the map file. The map section of this boot image starts after that sector and contains only the address of the loader sector and of the boot sector of the other operating system.

4.3 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.

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:

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

- 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.

5 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[11] 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[12], 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 init from the next poeigl package will most likely recognize an option `single` (among others).

`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

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

[12] 0.98pl3 or newer.

hexadecimal device number.[13]

Example:

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

6 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.

The LILO map installer may create some device special files in your `/tmp` directory that are not removed if an error occurs. They are named `/tmp/dev.`*number*.

6.1 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.

`-d` *tsecs*
: Specifies the number of tenths of seconds 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.

[13] 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
		...		...	

`-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`.

`-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.

`-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.[14] The current directory is changed to the new root directory, so using relative paths may not work.

`-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.)

[14] I.e. if your root partition is mounted on `/mnt`, you can update the map by simply running `/mnt/etc/lilo/install` with the argument `-r /mnt`. 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:
`/mnt/etc/lilo/install -r /mnt -b /dev/hda1`
You also have to set the environment variable `ROOT` before running the update script, e.g.:
`export ROOT=/mnt`

If no image files are specified, 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. If 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.[16]

```
lilo  foo ,/dev/hda1
      image  root
```

or

```
lilo  foo ,0x302
      image  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
```

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.

If LILO doesn't guess the correct BIOS device code, it can be specified by appending a colon and the code to the file name, e.g. `/linux:0x80`[17]

LILO uses the first 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
```

6.2 Boot image types

LILO can boot the following types of images:

- "traditional" boot images from a file.

[15] Or if this is not a traditional boot image.

[16] For compatibility to older kernels, also a swap device can be set by appending its name or number to the root device specification (e.g. `foo,/dev/hda1,/dev/hda2`). It is discouraged to use this feature because recent kernels don't support this mechanism anymore.

[17] This is typically used to install LILO on a second disk that will be used as the first disk later.

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

The image type is determined by the syntax that is used for the image specification.

6.2.1 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:

$$\mathit{file_name}[:\mathit{BIOS_code}]$$

Example: `/linux`

6.2.2 Booting "traditional" boot images from a device

The root device setting in the image is ignored.[18] 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.

$$\mathit{device_name}[:\mathit{BIOS_code}]\#\mathit{start}[-\mathit{end}|+\mathit{number}]$$

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

6.2.3 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.

[18] 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.

The image specification looks like this:

$$setup_name[:BIOS_code]+kernel_name[:BIOS_code]$$

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

6.2.4 Booting a foreign operating system

LILO can even boot other operating systems, i.e. MS-DOS. This feature is new and may not yet work totally reliably. (Reported to work with PC-DOS 4.0, MS-DOS 5.0, DR-DOS 6.0, OS/2 2.0 and 386BSD.) 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:

$$loader+boot_dev[:BIOS_code]@[part_dev]$$

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.)

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.[19]

6.3 Disk parameter table

For floppies and IDE disks (or MFM, RLL, ESDI, etc.), LILO can obtain the disk geometry information from the kernel. Unfortunately, this is not possible with SCSI disks. 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.
```

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

```
# 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 and reports errors in these cases.

7 Installation and updates

7.1 First-time installation

You have to run the 0.96c-pl1 kernel or any newer release. First, you have to install the LILO files:

- extract all files from `lilo.`*version*`.tar.Z`
- 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`.

If you want to use LILO on a SCSI disk, you have to determine the parameters of your SCSI disk(s) and put them into the file `/etc/lilo/disktab`. See section 6.3 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`.
- chdir 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.
  `#!/bin/sh`
  `$ROOT/etc/lilo/lilo` *all_necessary_options* `-i /etc/lilo/boot.b $* \`
  kernel_images
  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.

- 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.

7.2 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, 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.

7.3 Kernel update

Whenever any of the kernel files that are accessed by LILO is moved or overwritten, the map has to be re-built.[21] Run `/etc/lilo/install` to do this.

If the setup code has changed and if you're booting unstripped kernels, you also have to update `setup.b`. This should be done in `/etc/lilo/install`, as described in section 6.2.3.

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
```

[21] 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.

```
        /etc/lilo/install
        rm system.tmp
        sync
...
```

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 FS 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.

`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.[22]

`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.

`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. 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 to install the LILO boot sector.

[22] 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.

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.[23]

`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@`).

`Sorry, don't know how to handle device` *number*
: LILO uses files that are located on a device (typically a SCSI disk) 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 4.1.)

(*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 activate it. This can either be caused by a geometry mismatch or by moving `/etc/lilo/boot.b` without running the map installer.

[23] The expected version number may be lower than the version number of the LILO package.

`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.

`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 the contents of your `/etc/lilo/disktab` file corresponds to the real disk geometry. (See section 6.3.)

`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.