# Lilo mini-Howto


## Cameron Spitzer

cls@truffula.sj.ca.us

## Alessandro Rubini

rubini@linux.it



LILO is the most used Linux Loader for the x86 flavour of Linux; I'll call it Lilo rather than LILO here because I don't appreciate uppercase. This file describes some typical Lilo installations. It's intended as a supplement to the Lilo User's Guide. I think examples are informative even if your setup isn't much like mine. I hope this saves you trouble. Since Lilo's own documentation is very good, who's interested in the details is referred to /usr/doc/lilo*


# 1. Copyright and Licensing Information

according to the terms presented in the Open Publication License.

# 2. Preface

This document has not being maintained since Aug 1998. In June 1999 Cameron set up a web site to handle Lilo issues, aiming to help updating this document, so plese visit http://judi.greens.org/lilo/. Since I (Alessandro) haven't heard back from Cameron ever since, I took the chance to add this Preface when the document has been converted from LinuxDoc to DocBook (two sgml "variants") by Deb Richardson, in order to include it in the OSWG corpus.

While the document may still include interesting information, I don't think it serves the community's need any more. Current issues with Lilo are related to very big disks, modern hardware and BIOSes, multiple-booting with other Operating Systems and I've absolutely no experience with any of them. As the document has been sitting there unmodified for 15 months, I felt urged to add this note. If you are willing to take over maintainance or otherwise help, please get in touch with Cameron and either the LDP (http://www.linuxdoc.org) or the OSWG (http://www.oswg.org) – the latter is probably your best bet, as they demonstrated some interest in this specific document.

# 3. Introduction

Although the documentation found in Lilo's sources (the one installed in /usr/doc/lilo -version) is very comprehensive, most Linux users experience some trouble in building their own /etc/lilo.conf file. This document is meant to support them by giving the minimal information and by showing five sample installations:

- The first example is the classical "Linux and other" installation.
- The next one shows how to install Lilo on a hard drive connected as /dev/hdc that will boot as /dev/hda. This is usually needed when you install a new

Linux drive from your own running system. This also tells how to boot from SCSI disks when your BIOS is modern enough.

The third example shows how to boot a Linux system whose root partition can't be accessed by the BIOS.

The next sample file is used to access huge disks, that neither the BIOS nor DOS can access easily (this one is somehow outdated).

The last example shows how to restore a damaged disk, if the damage resulted from installing another operating system).

The last three examples are by Cameron, who wrote the original document. Alessandro, who took over maintainance for a while, doesn't run anything but Linux, so he can't check nor update them by himself. Feel free to send feedback through Cameron's web site (http://judi.greens.org/lilo/) or to send email to either of us.

# 4. Background Information and Standard Installation

When Lilo boots the system, it uses BIOS calls to load the Linux kernel off the disk (IDE drive, floppy or whatever). Therefore, the kernel must live in some place that can be accessed by the bios.

At boot time, Lilo is not able to read filesystem data, and any pathname you put in /etc/lilo.conf is resolved at installation time (when you invoke /sbin/lilo). Installation time is when the program builds the tables that list which sectors are used by the files used to load the operating system. As a consequence, all of these files must live in a partition that can be accessed by the BIOS (the files are usually located in the /boot directory, this means that only the root partition of your Linux system needs to be accessed via the BIOS).

Another consequence of being BIOS-based is that you must reinstall the loader (i.e., you must reinvoke /sbin/lilo) any time you modify the Lilo setup. Whenever you recompile your kernel and overwrite your old image you must reinstall Lilo.

## 4.1. Where Should I Install Lilo?

The boot= directive in /etc/lilo.conf tells Lilo where it should place its primary boot loader. In general, you can either specify the master boot record (/dev/hda) or the root partition of your Linux installation (is usually is /dev/hda1 or /dev/hda2).

If you have another operating system installed in your hard drive, you'd better install Lilo to the root partition instead of the MBR. In this case, you must mark the partition as “bootable” using the a command of fdisk or the b command of cfdisk. If you don't overwrite the master boot sector you'll find it easier to uninstall Linux and Lilo if needed.

## 4.2. How Should I Configure my IDE Hard Drives?

I personally don't use LBA or LARGE settings in the BIOS (but I only run Linux); they are horrible kludges forced on by design deficiencies in the PC world. This requires that the kernel lives in the first 1024 cylinders, but this is not a problem as long as you partition your hard drives and keep root small (as you should do anyways).

If your hard disk already carries another operating system, you won't be able to modify the BIOS settings, or the old system won't work any more. All recent Lilo distribution are able to deal with LBA and LARGE disk settings.

Note that the linear keyword in /etc/lilo.conf can help in dealing with geometry problems. The keyword instructs Lilo to use linear sector addresses instead of sector/head/cylinder tuples. Conversion to 3D addresses is delayed to run-time, therefore making the setup more immune to geometry problems.

If you have more than one hard disk and some of them are only used by Linux and are not involved in the boot process, you can tell your BIOS that they are not installed. Your system will boot more quickly and Linux will autodetect all the disks in no time. I often switch disks in my computers, but I never touch the BIOS configuration.

## 4.3. How Can I Interact at Boot Time?

When you see the Lilo prompt, you can hit the Tab key to show the list of possible choices. If Lilo is not configured to be interactive, press and hold the Alt or Shift key before the LILO message appears.

If you choose to boot a Linux kernel, you can add command-line arguments after the name of the system you choose. The kernel accepts many command-line arguments. All the arguments are listed in the BootPrompt-HOWTO by Paul Gortmaker, and I won't replicate it here. A few command line arguments, however, are particularly important and worth describing here:

root=: you can tell the Linux kernel to mount as root a different partition than the one appearing in lilo.conf. For example, my system has a tiny partition hosting a minimal Linux installation, and I've been able to boot the system after destroying my root partition by mistake.

init=: version 1.3.43 and newer of the Linux kernel can execute another command instead of /sbin/init, as specified on the command line. If you experience bad problems during the boot process, you can access the bare system by specifying init=/bin/sh (when you are at the shell prompt you most likely will need to mount your disks: try mount -w -n -o remount /; mount -a, and remember to umount -a before turning off the computer).

number: by specifying a number on the kernel command line, you instruct init to enter a specific run-level (the default is usually 3 or 2, according to the distribution you chose). Refer to the init documentation, to /etc/inittab and to /etc/rc*.d to probe further.

## 4.4. How Can I Uninstall Lilo?

When Lilo overwrites a boot sector, it saves a backup copy in /boot/boot.xxyy, where xxyy are the major and minor numbers of the device, in hex. You can see the major and minor numbers of your disk or partition by running ls -l /dev/device. For example, the first sector of /dev/hda (major 3, minor 0) will be saved in /boot/boot.0300, installing Lilo on /dev/fd0 creates /boot/boot.0200 and installing on /dev/sdb3> (major 8, minor 19) creates /boot/boot.0813. Note that Lilo won't create the file if there is already one so you don't need to care about the backup copy whenever you reinstall Lilo (for example, after recompiling your kernel). The backup copies found in /boot/ are always the snapshot of the situation before installing any Lilo.

If you ever need to uninstall Lilo (for example, in the unfortunate case you need to uninstall Linux), you just need to restore the original boot sector. If Lilo is installed in /dev/hda, just do dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1 (I personally just do cat /boot/boot.0300 > /dev/hda, but this is not safe, as this will restore the original partition table as well, which you might have modified in the meanwhile). This command is much easier to run than trying fdisk /mbr from a DOS shell: it allows you to cleanly remove Linux from a disk without ever booting anything but Linux. After removing Lilo remember to run Linux' fdisk to destroy any Linux partition (DOS' fdisk is unable to remove non-dos partitions).

If you installed Lilo on your root partition (e.g., /dev/hda2), nothing special needs to be done to uninstall Lilo. Just run Linux' fdisk to remove Linux partitions from the partition table. You must also mark the DOS partition as bootable.

# 5. The Simple Configuration

Most Lilo installations use a configuration file like the following one:

```
boot = /dev/hda   # or your root partition
delay = 10        # delay, in tenth of a second (so you can interact)
vga = 0           # optional. Use "vga=1" to get 80x50
#linear           # try "linear" in case of geometry problems.

image = /boot/vmlinux  # your zImage file
  root = /dev/hda1     # your root partition
  label = Linux        # or any fancy name
```

```
  read-only             # mount root read-only

other = /dev/hda4   # your dos partition, if any
  table = /dev/hda  # the current partition table
  label = dos       # or any non-fancy name
```

You can have multiple "image" and "other" sections if you want. It's not uncommon to have several kernel images configured in your lilo.conf, at least if you keep up to date with kernel development.

## 5.1. How to Deal with Big Kernels

If you compile a "zImage" kernel and it is too big to fit in half a megabyte (this is commong with new 2.1 kernels), you should build a "big zImage" instead: make bzImage. To boot a big kernel image nothing special is needed, but you need version 18 or newer of Lilo. If your installation is older, you should upgrade your Lilo package.

## 5.2. Other Sources of Information

In addition to the Lilo docs, there are a number of mini-howto's that can be useful for your needs. All of them are called Linux+foobarOS, for some foobarOS, they deal with coexistence of Linux and other operationg system(s). Also, Multiboot-with-LILO describes how the various Windows flavours can be made to coexist with Linux.

If you need some information about partitioning (how are partition tables stored on disk, what are the partition tipes, and other low-level information) you should check Andries Brouwer's site, at http://www.win.tue.nl/~aeb/.

# 6. Installing hdc to Boot as hda and Using bios=

Lilo allows to map the kernel image from one disk and instruct the BIOS to retrieve it from another disk. For example, it's common for me to install Linux on a disk I connect to hdc (master disk of secondary controller) and boot it as a standalong system on the primary IDE controller of another computer. I copied the installation floppy to a tiny partition, so I can run chroot in a virtual console to install hdc while I use the system to do something else.

The lilo.conf file I use to install Lilo looks like:

```
# This file must be used from a system running off /dev/hdc
boot = /dev/hdc   # overwrite MBR of hdc
disk = /dev/hdc   # tell how hdc will look like:
   bios = 0x80    #  the bios will see it as first drive
delay = 0
vga = 0

image = /boot/vmlinux  # this is on /dev/hdc1
  root = /dev/hda1     # but at boot it will be hda1
  label = Linux
  read-only
```

This configuration file must be read by a Lilo running off /dev/hdc1. The Lilo maps that get written the boot sector (/dev/hdc) must refer to the files in /boot (currently installed as hdc); such files will be accessed under hda when this disk will be booted as a standalone system.

I call this configuration file /mnt/etc/lilo.conf.hdc (/mnt is where hdc is mounted during the installation. I install Lilo by invoking cd /mnt; chroot . sbin/lilo -C /etc/lilo.conf.hdc. Refer to the manual page for chroot if this looks magic.

The bios= directive in lilo.conf is used to tell Lilo what the BIOS thinks of your devices. BIOS calls identify floppy disks and hard drives with a number: 0x00 and 0x01 select the floppy drives, 0x80 and the following numbers select hard disks (old BIOSes can only access two disks). The meaning of bios = 0x80 in the previous sample file is therefore “use 0x80 in your BIOS calls for /dev/hdc.

This Lilo directive can be handy in other situations, for example when your BIOS is able to boot from SCSI disks instead of IDE ones. When both IDE and SCSI devices are there, Lilo can't tell whether 0x80 will refer to one or the other because the user is able to choose it in the BIOS configuration menus, and the BIOS can't be accessed while Linux is running.

By default, Lilo assumes that IDE drives are mapped first by the BIOS, but this can be overridden by using instructions like these in /etc/lilo.conf:

```
disk = /dev/sda
  bios = 0x80
```

# 7. Using Lilo When the BIOS Can't See the Root Partition

I have two IDE drives, and a SCSI drive. The SCSI drive can't be seen from BIOS. The Linux Loader, Lilo, uses BIOS calls and can only see drives that BIOS can see. My stupid AMI BIOS will only boot from A: or C: My root file system is on a partition on the SCSI drive.

The solution consists in storing the kernel, map file, and chain loader in a Linux partition on the first IDE. Notice that it is not necessary to keep your kernel on your root partition.

The second partition on my first IDE (/dev/hda2, the Linux partition used to boot the system) is mounted on /u2. Here is the /etc/lilo.conf file I used.

```
#  Install Lilo on the Master Boot Record
#  on the first IDE.
#
boot = /dev/hda
#  /sbin/lilo (the installer) copies the Lilo boot record
#  from the following file to the MBR location.
install = /u2/etc/lilo/boot.b
```

```
#
#  I wrote a verbose boot menu.  Lilo finds it here.
message = /u2/etc/lilo/message
#  The installer will build the following file. It tells
#  the boot-loader where the blocks of the kernels are.
map = /u2/etc/lilo/map
compact
prompt
#  Wait 10 seconds, then boot the 1.2.1 kernel by default.
timeout = 100
#  The kernel is stored where BIOS can see it by doing this:
#      cp -p /usr/src/linux/arch/i386/boot/zImage /u2/z1.2.1
image = /u2/z1.2.1
label = 1.2.1
#  Lilo tells the kernel to mount the first SCSI partition
#  as root.  BIOS does not have to be able to see it.
root = /dev/sda1
#  This partition will be checked and remounted by /etc/rc.d/rc.S
read-only
#  I kept an old Slackware kernel lying around in case I built a
#  kernel that doesn't work.  I actually needed this once.
image = /u2/z1.0.9
label = 1.0.9
root = /dev/sda1
read-only
#  My DR-DOS 6 partition.
other = /dev/hda1
        loader=/u2/etc/lilo/chain.b
        label = dos
        alias = m
```

# 8. Accessing Huge Disks When the BIOS Can't

The system in my office has a 1GB IDE drive. The BIOS can only see the first 504 MB of the IDE. (Where MB means 2**10 bytes, not 10**6 bytes.) So I have

MS-DOS on a 350 MB partition /dev/hda1 and my Linux root on a 120 MB partition /dev/hda2.

MS-DOS was unable to install itself correctly when the drive was fresh. Novell DOS 7 had the same problem. Luckily for me, "Options by IBM" forgot to put the "OnTrack" diskette in the box with the drive. The drive was supposed to come with a product called "OnTrack Disk Manager." If you only have MSDOS, I guess you have to use it.

So I made a partition table with Linux' fdisk. MSDOS-6.2 refused to install itself in /dev/hda1. It said something like “this release of MS-DOS is for new installations. Your computer already has MS-DOS so you need to get an upgrade release from your dealer.” Actually, the disk was brand new.

What a crock! So I ran Linux' fdisk again and deleted partition 1 from the table. This satisfied MS-DOS 6.2 which proceeded to create the exact same partition 1 I had just deleted and installed itself. MS-DOS 6.2 wrote its Master Boot Record on the drive, but it couldn't boot.

Luckily I had a Slackware kernel on floppy (made by the Slackware installation program setup), so I booted Linux and wrote Lilo over MS-DOS' broken MBR. This works. Here is the /etc/lilo.conf file I used:

```
boot = /dev/hda
map = /lilo-map
delay = 100
ramdisk = 0             # Turns off ramdisk in Slackware kernel
timeout = 100
prompt
disk = /dev/hda # BIOS only sees first 500 MB.
   bios = 0x80          # specifies the first IDE.
   sectors = 63         # get the numbers from your drive's docs.
   heads = 16
   cylinders = 2100
image = /vmlinuz
  append = "hd=2100,16,63"
  root = /dev/hda2
  label = linux
```

```
  read-only
  vga = extended
other = /dev/hda1
  label = msdos
  table = /dev/hda
  loader = /boot/chain.b
```

After I installed these systems, I verified that the partition containing the zImage, boot.b, map, chain.b, and message files can use an MS-DOS file system, as long as it is not "stackered" or "doublespaced." So I could have made the DOS partition on /dev/hda1 500 MB.

I have also learned that "OnTrack" would have written a partition table starting a few dozen bytes into the drive, instead of at the beginning, and it is possible to hack the Linux IDE driver to work around this problem. But installing would have been impossible with the precompiled Slackware kernel. Eventually, IBM sent me an "OnTrack" diskette. I called OnTrack's technical support. They told me Linux is broken because Linux doesn't use BIOS. I gave their diskette away.

# 9. Booting from a Rescue Floppy

Next, I installed Windows-95 on my office system. It blew away my nice Lilo MBR, but it left my Linux partitions alone. Kernels take a long time to load from floppy, so I made a floppy with a working Lilo setup on it, which could boot my kernel from the IDE.

I made the lilo floppy like so:

```
  fdformat /dev/fd0H1440      #  lay tracks on virgin diskette
  mkfs -t minix /dev/fd0 1440 #  make file system of type minix
  mount /dev/fd0 /mnt         #  mount in the standard tmp mount point
  cp -p /boot/chain.b /mnt    #  copy the chain loader over
  lilo -C /etc/lilo.flop      #  install Lilo and the map on the diskette.
  umount /mnt
```

Notice that the diskette must be mounted when you run the installer so that Lilo can write its map file properly.

This file is /etc/lilo.flop. It's almost the same as the last one:

```
# Makes a floppy that can boot kernels from HD.
boot = /dev/fd0
map = /mnt/lilo-map
delay = 100
ramdisk = 0
timeout = 100
prompt
disk = /dev/hda     # 1 GB IDE, BIOS only sees first 500 MB.
   bios=0x80
   sectors = 63
   heads = 16
   cylinders = 2100
image = /vmlinuz
  append = "hd=2100,16,63"
  root = /dev/hda2
  label = linux
  read-only
  vga = extended
other = /dev/hda1
  label = msdos
  table = /dev/hda
  loader = /mnt/chain.b
```

Finally, I needed MS-DOS 6.2 on my office system, but I didn't want to touch the first drive. I added a SCSI controller and drive, made an msdos file system on it with Linux' mkdosfs, and Windows-95 sees it as D:. But of course MSDOS will not boot off of D:. This is not a problem when you have Lilo. I added the following to the lilo.conf in Example 2.

```
other = /dev/sda1
  label = d6.2
  table = /dev/sda
  loader = /boot/any_d.b
```

With this modification MSDOS-6.2 runs, and it thinks it is on C: and Windows-95 is on D:.