The Linux Installation HOWTO

Eric Steven Raymond

Thyrsus Enterprises

esr@thyrsus.com

Revision History

Revision 5.2	22 February 2001	Revised by: esr
LDP Styleguide markup fixes.		
Revision 5.1	29 January 2001	Revised by: esr
Minor corrections for the post−2.1 world.		
Revision 5.0	21 August 2000	Revised by: esr
First DocBook version.		

This document describes how to obtain and install Linux software. It is the first document which a new Linux user should read to get started.

Table of Contents

Table of Contents

1. Introduction

1.1. Purpose of this document

Linux is a freely−distributable implementation of Unix for inexpensive personal machines (it was developed on 386s, and now supports 486, 586, Pentium, PowerPC, Sun Sparc, ARM and DEC Alpha hardware, and even the IBM System 390 mainframe!). It supports a wide range of software, including X Windows, Emacs, TCP/IP networking (including SLIP), and many applications.

This document assumes that you have heard of and know about Linux, and now want to get it running. It focuses on the Intel base version, which is the most popular, but much of the advice applies on Power PCs, Sparcs and Alphas as well.

1.2. Other sources of information

If you are new to Linux, there are several sources of basic information about the system. The best place to find these is at the at Linux Documentation Project home page. You can find the latest version of this document there.

You should probably start by browsing the resources under General Linux Information; the Linux INFO−SHEET and the Linux META−FAQ. The `Linux Frequently Asked Questions' document contains many common questions (and answers!) about Linux −− it is a ``must read'' for new users.

You can find help for common problems on the USENET newsgroups comp.os.linux.help and comp.os.linux.announce.

The Linux Documentation Project is writing a set of manuals and books about Linux, all of which are freely distributable on the net and available from the LDP home page.

The book ``*Linux Installation and Getting Started''* is a complete guide to getting and installing Linux, as well as how to use the system once you've installed it. It contains a complete tutorial to using and running the system, and much more information than is contained here. You can browse it, or download a copy, from the LDP home page.

Finally, there is a rather technical Guide to x86 Bootstrapping. This document is NetBSD− rather than Linux−oriented, but contains useful material on disk configuration and boot managers for multi−OS setups.

Please do *not* email me asking for installation help. Even if I had the time to handle such requests, troubleshooting by mail is much less efficient than asking help from your local Linux user's group. You can find worldwide contact information for Linux user groups on the LDP site.

1.3. New versions of this document

New versions of the Linux Installation HOWTO will be periodically posted to comp.os.linux.help and comp.os.linux.announce and news.answers. They will also be uploaded to various Linux WWW and FTP sites, including the LDP home page.

You can also view the latest version of this on the World Wide Web via the URL http://www.linuxdoc.org/HOWTO/Installation−HOWTO.html.

1.4. Feedback and Corrections

If you have questions or comments about this document, please feel free to mail Eric S. Raymond, at `<esr@thyrsus.com>`. I welcome any suggestions or criticisms. If you find a mistake with this document, please let me know so I can correct it in the next version. Thanks.

Please do *not* mail me questions about how to solve hardware problems encountered during installation. Consult *Linux Installation and Getting Started*, bug your vendor, or consult the Linux newsgroup comp.os.linux.setup. This HOWTO is intended to be rapid, painless guide to *normal* installation −− a separate HOWTO on hardware problems and diagnosis is in preparation.

2. Recent Changes

- Added the `Buy, Don't Build' section.
- Added the material on booting from CD−ROM.

3. The Easiest Option: Buy, Don't Build

Linux has now matured enough that there are now system integrators who will assemble a workstation for you, install and configure a Linux, and do an intensive burn–in to test it before it's shipped to you. If you have more money than time, or you have stringent reliability or performance requirements, these integrators provide a valuable service by making sure you won't get hardware that's flaky or dies two days out of the box.

For those of us without a champagne budget, the rest of this HOWTO is about how to install Linux yourself.

4. Before You Begin

Before you can install Linux, you'll need to be sure your machine is Linux−capable, and choose a Linux to install. The Linux Pre−installation checklist may help you organize configuration data before you begin.

4.1. Hardware requirements

What kind of system is needed to run Linux? This is a good question; the actual hardware requirements for the system change periodically. The Linux Hardware−HOWTO, gives a (more or less) complete listing of hardware supported by Linux. The Linux INFO−SHEET, provides another list.

For the Intel versions, a hardware configuration that looks like the following is required:

Any 80386, 80486, Pentium or Pentium II processor will do. Non−Intel clones of the 80386 and up will generally work. You do not need a math coprocessor, although it is nice to have one.

The ISA, EISA, VESA Local Bus and PCI bus architectures are supported. The MCA bus architecture (found on IBM PS/2 machines) has been minimally supported since the 2.1.x kernels, but may not be ready for prime time yet.

You need at least 4 megabytes of memory in your machine. Technically, Linux will run with only 2 megs, but most installations and software require 4. The more memory you have, the happier you'll be. I suggest an absolute minimum of 16 megabytes if you're planning to use X−Windows; 64 is better.

Of course, you'll need a hard drive and an AT−standard drive controller. All MFM, RLL, and IDE drives and controllers should work. Many SCSI drives and adaptors are supported as well; the Linux SCSI−HOWTO contains more information on SCSI. If you are assembling a system from scratch to run Linux, the small additional cost of SCSI is well worth it for the extra performance and reliability it brings.

You will need a 3.5" floppy drive. While 5.25" floppies are supported under Linux, they are little−enough used that you should not count on disk images necessarily fitting on them. (A stripped−down Linux can actually run on a single floppy, but that's only useful for installation and certain troubleshooting tasks.)

You also need an MDA, Hercules, CGA, EGA, VGA, or Super VGA video card and monitor. In general, if your video card and monitor work under MS−DOS then it should work under Linux. However, if you wish to run the X window system, there are other restrictions on the supported video hardware. The Linux XFree86−HOWTO, contains more information about running X and its requirements.

You'll want a CD−ROM drive. If it's ATAPI, SCSI, or true IDE you should have no problem making it work (but watch for cheap drives advertising "IDE" interfaces that aren't true IDE). If your CD−ROM uses a proprietary interface card, it's possible the installation kernel you're going to boot from floppy won't be able to see it −− and an inaccessible CD−ROM is a installation show−stopper. Also, CD−ROMs that attach to your parallel port won't work at all. If you're in doubt, consult the Linux CD−ROM HOWTO for a list and details of supported hardware.

So−called "Plug'n'Play" jumperless cards can be a problem. Support for these is under active development, but not there yet in the 2.0.25 kernel. Fortunately this is only likely to be a problem with sound or Ethernet cards.

If you're running on a box that uses one of the Motorola 68K processors (including Amiga, Atari, or VMEbus machines), see the Linux/m68k FAQ for information on minimum requirements and the state of the port. The FAQ now says m68k Linux is as stable and usable as the Intel versions.

4.2. Space requirements and coexistence

You'll need free space for Linux on your hard drive. The amount of space needed depends on how much software you plan to install. Today most installations require somewhere in the ballpark of a gigabyte of space. This includes space for the software, swap space (used as virtual RAM on your machine), and free space for users, and so on.

It's conceivable that you could run a minimal Linux system in 80 megs or less (this used to be common when Linux distributions were smaller), and it's conceivable that you could use two gigabytes or more for all of your Linux software. The amount varies greatly depending on the amount of software you install and how much space you require. More about this later.

Linux will co−exist with other operating systems, such as MS−DOS, Microsoft Windows, or OS/2, on your hard drive. (In fact you can even access MS−DOS files and run some MS−DOS programs from Linux.) In other words, when partitioning your drive for Linux, MS−DOS or OS/2 live on their own partitions, and Linux exists on its own. We'll go into more detail about such ``dual−boot'' systems later.

You do *not* need to be running MS−DOS, OS/2, or any other operating system to use Linux. Linux is a completely stand−alone operating system and does not rely on other OSs for installation and use.

In all, the minimal setup for Linux is not much more than is required for most MS−DOS or Windows 3.1 systems sold today (and it's a good deal less than the minimum for Windows 95!). If you have a 386 or 486 with at least 4 megs of RAM, then you'll be happy running Linux. Linux does not require huge amounts of diskspace, memory, or processor speed. Matt Welsh, the originator of this HOWTO, used to run Linux on a 386/16 MHz (the slowest machine you can get) with 4 megs of RAM, and was quite happy. The more you want to do, the more memory (and faster processor) you'll need. In our experience a 486 with 16 megabytes of RAM running Linux outdoes several models of expensive workstations.

5. Time requirements

Start to finish, a modern Linux installation from CD−ROM can be expected to take from ninety minutes to three hours.

5.1. Choosing a Linux distribution

Before you can install Linux, you need to decide on one of the ``distributions'' of Linux which are available. There is no single, standard release of the Linux software−−−there are many such releases. Each release has its own documentation and installation instructions.

Linux distributions are available both via anonymous FTP and via mail order on diskette, tape, and CD−ROM. There are many checklists and comparative reviews of Linux distributions out there. The Linux Weekly News site, in addition to being an excellent general source of news and information, carries a weekly report on distributions with pointers to many of them.

In the dim and ancient past when this HOWTO was first written (1992−93), most people got Linux by tortuous means involving long downloads off the Internet or a BBS onto their DOS machines, followed by an elaborate procedure which transferred the downloads onto multiple floppy disks. One of these disks would then be booted and used to install the other dozen. With luck (and no media failures) you'd finish your installation many hours later with a working Linux. Or maybe not.

While this path is still possible (and you can download any one of several distributions from Metalab), there are now much less strenuous ways. The easiest is to buy one of the high−quality commercial Linux distributions distributed on CD−ROM, such as Red Hat, Debian, Linux Pro, or WGS. These are typically available for less than $50 at your local bookstore or computer shop, and will save you many hours of aggravation.

You can also buy anthology CD−ROMs such as the InfoMagic Linux Developer's Resource set. These typically include several Linux distributions and a recent dump of major Linux archive sites, such as metalab or tsx−11.

In the remainder of this HOWTO we will focus on the steps needed to install from an anthology CD−ROM, or one of the lower−end commercial Linuxes that doesn't include a printed installation manual. If your Linux includes a paper manual some of this HOWTO may provide useful background, but you should consult the manual for detailed installation instructions.

6. Installation Overview

It's wise to collect configuration information on your hardware before installing. Know the vendor and model number of each card in your machine; collect the IRQs and DMA channel numbers. You probably won't need this information −− but if it turns out you do, you'll need it very badly.

If you want to run a "dual−boot" system (Linux and DOS or Windows or both), rearrange (repartition) your disk to make room for Linux. If you're wise, you'll *back up everything first!*.

6.1. First Installation Steps: The Easy Way

If you have an EIDE/ATAPI CDROM (normal these days), check your machine's BIOS settings to see if it has the capability to boot from CD−ROM. Most machines made after mid−1997 can do this.

If yours is among them, change the settings so that the CD−ROM is checked first. This is often in a 'BIOS FEATURES' submenu of the BIOS configuration menus.

Then insert the installation CD−ROM. Reboot. You're started.

If you have a SCSI CDROM you can often still boot from it, but it gets a little more motherboard/BIOS dependent. Those who know enough to spend the extra dollars on a SCSI CDROM drive probably know enough to figure it out.

6.2. First Installation Steps: The Hard Way

- Make installation floppies.
- Boot an installation mini−Linux from the floppies in order to get access to the CD−ROM.

6.3. Continuing the Installation

- Prepare the Linux filesystems. (If you didn't edit the disk partition table earlier, you will at this stage.)
- Install a basic production Linux from the CD−ROM.
- Boot Linux from the hard drive.
- (Optional) Install more packages from CD−ROM.

6.4. Basic Parts of an Installation Kit

Here are the basic parts of an installable distribution:

- The README and FAQ files. These will usually be located in the top−level directory of your CD−ROM and be readable once the CD−ROM has been mounted under Linux. (Depending on how the CD−ROM was generated, they may even be visible under DOS/Windows.) It is a good idea to read these files as soon as you have access to them, to become aware of important updates or changes.

- A number of bootdisk images (often in a subdirectory). If your CD−ROM is not bootable, one of these is the file that you will write to a floppy to create the boot disk. You'll select *one* of the above bootdisk images, depending on the type of hardware that you have in your system.

The issue here is that some hardware drivers conflict with each other in strange ways, and instead of attempting to debug hardware problems on your system it's easier to use a boot floppy image with only the drivers you need enabled. (This will have the nice side effect of making your kernel smaller.)

- A rescue disk image. This is a disk containing a basic kernel and tools for disaster recovery in case something trashes the kernel or boot block of your hard disk.
- RAWRITE.EXE. This is an MS−DOS program that will write the contents of a file (such as a bootdisk image) directly to a floppy, without regard to format.

You only need RAWRITE.EXE if you plan to create your boot and root floppies from an MS−DOS system. If you have access to a UNIX workstation with a floppy drive instead, you can create the floppies from there, using the `dd' command, or possibly a vendor−provided build script. See the man page for dd(1) and ask your local UNIX gurus for assistance. There's a dd example later in this document.

- The CD−ROM itself. The purpose of the boot disk is to get your machine ready to load the root or installation disks, which in turn are just devices for preparing your hard disk and copying portions of the CD−ROM to it. If your CD−ROM is bootable, you can boot it and skip right to preparing your disk.

7. Installation In Detail

7.1. Getting prepared for installation

Linux makes more effective use of PC hardware than MS−DOS, Windows or NT, and is accordingly less tolerant of misconfigured hardware. There are a few things you can do before you start that will lessen your chances of being stopped by this kind of problem.

First, collect any manuals you have on your hardware −− motherboard, video card, monitor, modem, etc. −− and put them within easy reach.

Second, gather detailed information on your hardware configuration. One easy way to do this, if you're running MS−DOS 5.0, or up, is to print a report from the Microsoft diagnostic utility msd.exe (you can leave out the TSR, driver, memory−map, environment−strings and OS−version parts). Among other things, this will guarantee you full and correct information on your video card and mouse type, which will be helpful in configuring X later on.

Third, check your machine for configuration problems with supported hardware that could cause an un−recoverable lockup during Linux installation.

- It is possible for a DOS/Windows system using IDE hard drive(s) and CD ROM to be functional even with the master/slave jumpers on the drives incorrectly set. Linux won't fly this way. If in doubt, check your master−slave jumpers!
- Is any of your peripheral hardware designed with neither configuration jumpers nor non−volatile configuration memory? If so, it may require boot−time initialization via an MS−DOS utility to start up, and may not be easily accessible from Linux. CD−ROMs, sound cards, Ethernet cards and low−end tape drives can have this problem. If so, you may be able to work around this with an argument to the boot prompt; see theLinux Boot Prompt HOWTO for details).
- Some other operating systems will allow a bus mouse to share an IRQ with other devices. Linux doesn't support this; in fact, trying it may lock up your machine. If you are using a bus mouse, see the Linux Bus Mouse HOWTO, for details.

If possible, get the telephone number of an experienced Linux user you can call in case of emergency. Nine times out of ten you won't need it, but it's comforting to have.

Budget time for installation. That will be about one hour on a bare system or one being converted to all−Linux operation. Or up to three hours for a dual−boot system (they have a much higher incidence of false starts and hangups).

7.2. Creating the boot and root floppies

(This step is only needed if you can't boot from a CD−ROM.)

Your Linux CD−ROM may come with installation aids that will take you through the process of building boot, root, and rescue disks with interactive prompts. These may be an MS−DOS installation program (such as the Red Hat **redhat.exe** program) or a Unix script, or both.

If you have such a program and can use it, you should read the rest of this subsection for information only. Run the program to do actual installation −− its authors certainly knew more about the specific distribution than I, and you'll avoid many error−prone hand−entry steps.

More detailed information on making bootdisks, see the Linux Bootdisk HOWTO.

Your first step will be to select a boot−disk image to fit your hardware. If you must do this by hand, you'll generally find that either (a) the bootdisk images on your CD−ROM are named in a way that will help you pick a correct one, or (b) there's an index file nearby describing each image.

Next, you must create floppies from the bootdisk image you selected, and optionally from the rescue disk images. This is where the MS−DOS program RAWRITE.EXE comes into play.

Next, you must have two or three *high−density* MS−DOS formatted floppies. (They must be of the same type; that is, if your boot floppy drive is a 3.5" drive, both floppies must be high−density 3.5" disks.) You will use RAWRITE.EXE to write the bootdisk images to the floppies.

Invoke it with no arguments, like this:

```
C:\> RAWRITE
```

Answer the prompts for the name of the file to write and the floppy to write it to (such as A:). RAWRITE will copy the file, block−by−block, directly to the floppy. Also use RAWRITE for the root disk image (such as COLOR144). When you're done, you'll have two floppies: one containing the boot disk, the other containing the root disk. Note that these two floppies will no longer be readable by MS−DOS (they are ``Linux format'' floppies, in some sense).

You can use the dd(1) commands on a UNIX system to do the same job. (For this, you will need a UNIX workstation with a floppy drive, of course.) For example, on a Sun workstation with the floppy drive on device `/dev/rfd0`, you can use the command:

```
$ dd if=bare of=/dev/rfd0 obs=18k
```

You must provide the appropriate output block size argument (the `obs' argument) on some workstations (e.g., Suns) or this will fail. If you have problems the man page for dd(1) may be be instructive.

Be sure that you're using brand−new, error−free floppies. The floppies must have no bad blocks on them.

Note that you do not need to be running Linux or MS−DOS in order to install Linux. However, running Linux or MS−DOS makes it easier to create the boot and root floppies from your CD−ROM. If you don't have an operating system on your machine, you can use someone else's Linux or MS−DOS just to create the floppies, and install from there.

7.3. Repartitioning your DOS/Windows drives

On most used systems, the hard drive is already dedicated to partitions for MS−DOS, OS/2, and so on. You'll need to resize these partitions in order to make space for Linux. If you're going to run a dual−boot system, it's strongly recommended that you read one or more of the following mini−HOWTOS, which describe different dual−boot configurations.

* The DOS−Win95−OS2−Linux mini−HOWTO.
* The Linux+Win95 mini−HOWTO.
* The Linux+NT−Loader mini−HOWTO

Even if they are not directly applicable to your system, they will help you understand the issues involved.

> **Note:** Some Linuxes will install to a directory on your MS−DOS partition. (This is different than installing *from* an MS−DOS partition.) Instead, you use the ``UMSDOS filesystem'', which allows you to treat a directory of your MS−DOS partition as a Linux filesystem. In this way, you don't have to repartition your drive.

I only suggest using this method if your drive already has four partitions (the maximum supported by DOS) and repartitioning would be more trouble than it's worth (it slows down your Linux due to filename translation overhead). Or, if you want to try out Linux before repartitioning, this is a good way to do so. But in most cases you should re−partition, as described here. If you do plan to use UMSDOS, you are on your own −− it is not documented in detail here. From now on, we assume that you are NOT using UMSDOS, and that you will be repartitioning.

A *partition* is just a section of the hard drive set aside for a particular operating system to use. If you only have MS−DOS installed, your hard drive probably has just one partition, entirely for MS−DOS. To use Linux, however, you'll need to repartition the drive, so that you have one partition for MS−DOS, and one (or more) for Linux.

Partitions come in three flavors: *primary*, *extended*, and *logical*. Briefly, primary partitions are one of the four main partitions on your drive. However, if you wish to have more than four partitions per drive, you need to replace the last primary partition with an extended partition, which can contain many logical partitions. You don't store data directly on an extended partition−−−it is used only as a container for logical partitions. Data is stored only on either primary or logical partitions.

To put this another way, most people use only primary partitions. However, if you need more than four partitions on a drive, you create an extended partition. Logical partitions are then created on top of the extended partition, and there you have it−−−more than four partitions per drive.

Note that you can easily install Linux on the second drive on your system (known as D: to MS−DOS). You simply specify the appropriate device name when creating Linux partitions. This is described in detail below.

Back to repartitioning your drive. It used to be that there was no way to resize partitions without destroying the data on them. Nowadays there are partitioning utilities that can resize non−destructively; they know about the structure of file systems, can find the free space on a file system, and can move file data around on the partition to move free space where it needs to be in order for a resize to work properly. It's still suggested that you make a full backup before using one of these, in case of program or human error.

Under Linux GNU parted allows you to create, destroy, resize and copy partitions. It supports ext2, FAT16, and FAT32 filesystems, Linux swap devices; it also knows about MS−DOS disk labels. Parted is useful for creating space for new operating systems, reorganising disk usage, copying data between hard disks, and disk imaging. It is relatively new code, but is reported to work well and not trash data.

There is a non−destructive disk repartitioner available for MS−DOS, called FIPS. With FIPS, a disk optimizer (such as Norton Speed Disk), and a little bit of luck, you should be able to resize MS−DOS partitions without destroying the data on them.

The older method of resizing a partition, if you don't have one of these resizing partition editors available, is to delete the partition(s), and re−create them with smaller sizes. If you use this method, you absolutely must make a backup in order to save any of your data.

The classic way to modify partitions is with the program **FDISK**. For example, let's say that you have an 80 meg hard drive, dedicated to MS−DOS. You'd like to split it in half−−−40 megs for MS−DOS and 40 megs for Linux. In order to do this, you run **FDISK** under MS−DOS, delete the 80 meg MS−DOS partition, and re−create a 40 meg MS−DOS partition in its place. You can then format the new partition and reinstall your MS−DOS software from backups. 40 megabytes of the drive is left empty. Later, you create Linux partitions on the unused portion of the drive.

In short, you should do the following to resize MS−DOS partitions with **FDISK**:

1. Make a full backup of your system.
2. Create an MS−DOS bootable floppy, using a command such as
   ```
   FORMAT /S A:
   ```
   Copy the files **FDISK.EXE** and **FORMAT.COM** to this floppy, as well as any other utilities that you need. (For example, utilities to recover your system from backup.)
3. Boot the MS−DOS system floppy.
4. Run **FDISK**, possibly specifying the drive to modify (such as C: or D:).
5. Use the FDISK menu options to delete the partitions which you wish to resize. *This will destroy all data on the affected partitions.*
6. Use the FDISK menu options to re−create those partitions, with smaller sizes.
7. Exit FDISK and re−format the new partitions with the **FORMAT** command.
8. Restore the original files from backup.

Note that MS−DOS FDISK will give you an option to create a ``logical DOS drive''. A logical DOS drive is just a logical partition on your hard drive. You can install Linux on a logical partition, but you don't want to create that logical partition with MS−DOS fdisk. So, if you're currently using a logical DOS drive, and want to install Linux in its place, you should delete the logical drive with MS−DOS FDISK, and (later) create a logical partition for Linux in its place.

The mechanism used to repartition for OS/2 and other operating systems is similar. See the documentation for those operating systems for details.

7.4. Creating partitions for Linux

After repartitioning your drive, you need to create partitions for Linux. Before describing how to do that, we'll talk about partitions and filesystems under Linux.

7.4.1. Partition basics

Linux requires at least one partition, for the *root filesystem*, which will hold the Linux kernel itself.

You can think of a *filesystem* as a partition formatted for Linux. Filesystems are used to hold files. Every system must have a root filesystem, at least. However, many users prefer to use multiple filesystems−−−one for each major part of the directory tree. For example, you may wish to create a separate filesystem to hold all files under the `/usr` directory. (Note that on UNIX systems, forward slashes are used to delimit directories, not backslashes as with MS−DOS.) In this case you have both a root filesystem, and a `/usr` filesystem.

Each filesystem requires its own partition. Therefore, if you're using both root and `/usr` filesystems, you'll need to create two Linux partitions.

In addition, most users create a *swap partition*, which is used for virtual RAM. If you have, say, 4 megabytes of memory on your machine, and a 10−megabyte swap partition, as far as Linux is concerned you have 14 megabytes of virtual memory.

When using swap space, Linux moves unused pages of memory out to disk, allowing you to run more applications at once on your system. However, because swapping is often slow, it's no replacement for real physical RAM. But applications that require a great deal of memory (such as the X window system) often rely on swap space if you don't have enough physical RAM.

Nearly all Linux users employ a swap partition. If you have 4 megabytes of RAM or less, a swap partition is required to install the software. It is strongly recommended that you have a swap partition anyway, unless you have a great amount of physical RAM.

The size of your swap partition depends on how much virtual memory you need. It's often suggested that you have at least 16 megabytes of virtual memory total. Therefore, if you have 8 megs of physical RAM, you might want to create an 8−megabyte swap partition. Note that there are platform−dependent limits on the size of swap partitions; see the Partition−HOWTO if you want to create a swap partition larger than 1GB.

You can find more on the theory of swap space layout and disk partitioning in the Linux Partition mini−HOWTO (http://www.linuxdoc.org/HOWTO/mini/Partition.html).

Note: it is possible, though a bit tricky, to share swap partitions between Linux and Windows 95 in a dual−boot system. For details, see the Linux Swap Space Mini−HOWTO.

Gotcha #1: If you have an EIDE drive with a partition that goes above 504MB, your BIOS may not allow you to boot to a Linux installed there. So keep your root partition below 504MB. This shouldn't be a problem for SCSI drive controllers, which normally have their own drive BIOS firmware. For technical details, see the Large Disk Mini−HOWTO.

Gotcha #2: Are you mixing IDE and SCSI drives? Then watch out. Your BIOS may not allow you to boot directly to a SCSI drive.

7.4.2. Sizing partitions

Besides your root and swap partitions, you'll want to set up one or more partitions to hold your software and home directories.

While, in theory, you could run everything off a single huge root partition, almost nobody does this. Having multiple partitions has several advantages:

- It often cuts down the time required for boot−time file−system checks.
- Files can't grow across partition boundaries. Therefore you can use partition boundaries as firebreaks against programs (like Usenet news) that want to eat huge amounts of disk, to prevent them from crowding out file space needed by your kernel and the rest of your applications.

* If you ever develop a bad spot on your disk, formatting and restoring a single partition is less painful than having to redo everything from scratch.

On today's large disks, a good basic setup is to have a small root partition (less than 80 meg), a medium−sized /usr partition (up to 300 meg or so) to hold system software, and a /home partition occupying the rest of your available space for home directories.

You can get more elaborate. If you know you're going to run Usenet news, for example, you may want to give it a partition of its own to control its maximum possible disk usage. Or create a /var partition for mail, news, and temporary files all together. But in today's regime of very cheap, very large hard disks these complications seem less and less necessary for your first Linux installation. For your first time, especially, keep it simple.

7.5. Booting the installation disk

The first step is to boot the bootdisk you generated. Normally you'll be able to boot hands−off; the boot kernel prompt will fill itself in after 10 seconds. This is how you'll normally boot from an IDE disk.

What's actually happening here is this: the boot disk provides a miniature operating system which (because the hard drive isn't prepared) uses a portion of your RAM as a virtual disk (called, logically enough, a `ramdisk').

The boot disk loads onto the ramdisk a small set of files and installation tools which you'll use to prepare your hard drive and install a production Linux on it from your CD−ROM.

(In times past this was a two−stage−process, involving a second disk called a `root disk'; this changed when kernel modules were introduced.)

By giving arguments after the kernel name, you can specify various hardware parameters, such as your SCSI controller IRQ and address, or drive geometry, before booting the Linux kernel. This may be necessary if Linux does not detect your SCSI controller or hard drive geometry, for example.

In particular, many BIOS−less SCSI controllers require you to specify the port address and IRQ at boot time. Likewise, IBM PS/1, ThinkPad, and ValuePoint machines do not store drive geometry in the CMOS, and you must specify it at boot time. (Later on, you'll be able to configure your production system to supply such parameters itself.)

Watch the messages as the system boots. They will list and describe the hardware your installation Linux detects. In particular, if you have a SCSI controller, you should see a listing of the SCSI hosts detected. If you see the message

```
SCSI: 0 hosts
```

Then your SCSI controller was not detected, and you will have to figure out how to tell the kernel where it is.

Also, the system will display information on the drive partitions and devices detected. If any of this information is incorrect or missing, you will have to force hardware detection.

On the other hand, if all goes well and your hardware seems to be detected, you can skip to the following section, ``Loading the root disk.''

To force hardware detection, you must enter the appropriate parameters at the boot prompt, using the following syntax:

```
linux <parameters...>
```

There are a number of such parameters available; we list some of the most common below. Modern Linux boot disks will often give you the option to look at help screen describing kernel parameters before you boot.

* *hd=cylinders,heads,sectors* Specify the drive geometry. Required for systems such as the IBM PS/1, ValuePoint, and ThinkPad. For example, if your drive has 683 cylinders, 16 heads, and 32 sectors per track, enter
  ```
  linux hd=683,16,32
  ```

  tmc8xx=memaddr,irq Specify address and IRQ for BIOS−less Future Domain TMC−8xx SCSI controller. For example,

  ```
  linux tmc8xx=0xca000,5
  ```

  Note that the *0x* prefix must be used for all values given in hex. This is true for all of the following options.

* *st0x=memaddr,irq* Specify address and IRQ for BIOS−less Seagate ST02 controller.
* *t128=memaddr,irq* Specify address and IRQ for BIOS−less Trantor T128B controller.
* *ncr5380=port,irq,dma* Specify port, IRQ, and DMA channel for generic NCR5380 controller.
* *aha152x=port,irq,scsi_id,1* Specify port, IRQ, and SCSI ID for BIOS−less AIC−6260 controllers. This includes Adaptec 1510, 152x, and Soundblaster−SCSI controllers.

If you have questions about these boot−time options, please read the Linux *SCSI HOWTO*, which should be available on any Linux FTP archive site (or from wherever you obtained this document). The *SCSI HOWTO* explains Linux SCSI compatibility in much more detail.

7.5.1. Choosing Console or X installation

After boot, all current Linuxes run a screen−oriented installation program which tries to interactively walk you through these steps, giving lots of help.

You will probably get the option to try to configure X right away so the installation program can go graphical. If you choose this route, the installation program will quiz you about your mouse and monitor type before getting to the installation proper. Once you get your production Linux installed, these settings will be saved for you. You will be able to tune your monitor's performance later, so at this stage it makes sense to settle for a basic 640x480 SVGA mode.

X isn't necessary for installation, but (assuming you can get past the mouse and monitor configuration) many people find the graphical interface easier to use. And you're going to want to bring up X anyway, so trying it early makes some sense.

Just follow the prompts in the program. It will take you through the steps necessary to prepare your disk, create initial user accounts, and install software packages off the CD−ROM.

In the following subsections we'll describe some of the tricky areas in the installation sequence as if you were doing them by hand. This should help you understand what the installation program is doing, and why.

7.5.2. Using fdisk and cfdisk

Your first installation step once the root−disk Linux is booted will be to create or edit the partition tables on your disks. Even if you used FDISK to set up partitions earlier, you'll need to go back to the partition table now and insert some Linux−specific information now.

To create or edit Linux partitions, we'll use the Linux version of the **fdisk** program, or its screen−oriented sibling **cfdisk**.

Generally the installation program will look for a preexisting partition table and offer to run **fdisk** or **cfdisk** on it for you. Of the two, **cfdisk** is definitely easier to use, but current versions of it are also less tolerant of a nonexistent or garbled partition table.

Therefore you may find (especially if you're installing on virgin hardware) that you need to start with **fdisk** to get to a state that **cfdisk** can deal with. Try running **cfdisk**; if it complains, run **fdisk**. (A good way to proceed if you're building an all−Linux system and **cfdisk** complains is to use **fdisk** to delete all the existing partions and then fire up **cfdisk** to edit the empty table.)

A few notes apply to both **fdisk** and **cfdisk**. Both take an argument which is the name of the drive that you wish to create Linux partitions on. Hard drive device names are:

- `/dev/hda` First IDE drive
- `/dev/hdb` Second IDE drive
- `/dev/sda` First SCSI drive
- `/dev/sdb` Second SCSI drive

For example, to create Linux partitions on the first SCSI drive in your system, you will use (or your installation program might generate from a menu choice) the command:

```
cfdisk /dev/sda
```

If you use **fdisk** or **cfdisk** without an argument, it will assume `/dev/hda`.

To create Linux partitions on the second drive on your system, simply specify either `/dev/hdb` (for IDE drives) or `/dev/sdb` (for SCSI drives) when running **fdisk**.

Your Linux partitions don't all have to be on the same drive. You might want to create your root filesystem partition on `/dev/hda` and your swap partition on `/dev/hdb`, for example. In order to do so just run **fdisk** or **cfdisk** once for each drive.

In Linux, partitions are given a name based on the drive which they belong to. For example, the first partition on the drive `/dev/hda` is `/dev/hda1`, the second is `/dev/hda2`, and so on. If you have any logical partitions, they are numbered starting with `/dev/hda5`, `/dev/hda6` and so on up.

> **Note:** You should not create or delete partitions for operating systems other than Linux with Linux **fdisk** or **cfdisk**. That is, don't create or delete MS−DOS partitions with this version of **fdisk**; use MS−DOS's version of **FDISK** instead. If you try to create MS−DOS partitions with Linux **fdisk**, chances are MS−DOS will not recognize the partition and not boot correctly.

Here's an example of using **fdisk**. Here, we have a single MS−DOS partition using 61693 blocks on the drive, and the rest of the drive is free for Linux. (Under Linux, one block is 1024 bytes. Therefore, 61693 blocks is about 61 megabytes.) We will create just two partitions in this tutorial example, swap and root. You should probably extend this to four Linux partitions in line with the recommendations above: one for swap, one for the root filesystem, one for system software, and a home directory area.

First, we use the ``**p**'' command to display the current partition table. As you can see, `/dev/hda1` (the first partition on `/dev/hda`) is a DOS partition of 61693 blocks.

```
Command (m for help):   p
Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes

     Device Boot  Begin   Start     End  Blocks   Id  System
  /dev/hda1   *       1       1     203   61693    6  DOS 16-bit >=32M

Command (m for help):
```

Next, we use the ``**n**'' command to create a new partition. The Linux root partition will be 80 megs in size.

```
Command (m for help):  n
Command action
    e   extended
    p   primary partition (1-4)
p
```

Here we're being asked if we want to create an extended or primary partition. In most cases you want to use primary partitions, unless you need more than four partitions on a drive. See the section ``Repartitioning'', above, for more information.

```
Partition number (1-4): 2
First cylinder (204-683):  204
Last cylinder or +size or +sizeM or +sizeK (204-683): +80M
```

The first cylinder should be the cylinder AFTER where the last partition left off. In this case, `/dev/hda1` ended on cylinder 203, so we start the new partition at cylinder 204.

As you can see, if we use the notation ``+80M'', it specifies a partition of 80 megs in size. Likewise, the notation ``+80K'' would specify an 80 kilobyte partition, and ``+80'' would specify just an 80 byte partition.

```
Warning: Linux cannot currently use 33090 sectors of this partition
```

If you see this warning, you can ignore it. It is left over from an old restriction that Linux filesystems could only be 64 megs in size. However, with newer filesystem types, that is no longer the case... partitions can now be up to 4 terabytes in size.

Next, we create our 10 megabyte swap partition, `/dev/hda3`.

```
Command (m for help): n
Command action
    e   extended
    p   primary partition (1-4)
p

Partition number (1-4): 3
First cylinder (474-683):  474
Last cylinder or +size or +sizeM or +sizeK (474-683):  +10M
```

Again, we display the contents of the partition table. Be sure to write down the information here, especially the size of each partition in blocks. You need this information later.

```
Command (m for help): p
Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes

     Device Boot  Begin   Start     End  Blocks   Id  System
  /dev/hda1   *       1       1     203   61693    6  DOS 16-bit >=32M
  /dev/hda2         204     204     473   82080   83  Linux native
  /dev/hda3         474     474     507   10336   83  Linux native
```

Note that the Linux swap partition (here, `/dev/hda3`) has type ``Linux native''. We need to change the type of the swap partition to ``Linux swap'' so that the installation program will recognize it as such. In order to do this, use the **fdisk** ``t'' command:

```
Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 82
```

If you use ``**L**'' to list the type codes, you'll find that 82 is the type corresponding to Linux swap.

To quit **fdisk** and save the changes to the partition table, use the ``w'' command. To quit **fdisk** WITHOUT saving changes, use the ``q'' command.

After quitting **fdisk**, the system may tell you to reboot to make sure that the changes took effect. In general there is no reason to reboot after using **fdisk**−−−modern versions of **fdisk** and **cfdisk** are smart enough to update the partitions without rebooting.

7.5.3. Post−partition steps

After you've edited the partition tables, your installation program should look at them and offer to enable your swap partition for you. Tell it yes.

(This is made a question, rather than done automatically, on the off chance that you're running a dual−boot system and one of your non−Linux partitions might happen to look like a swap volume.)

Next the program will ask you to associate Linux filesystem names (such as /, /usr, /var, /tmp, /home, /home2, etc.) with each of the non−swap partitions you're going to use.

There is only one hard and fast rule for this. There must be a root filesystem, named /, and it must be bootable. You can name your other Linux partitions anything you like. But there are some conventions about how to name them which will probably simplify your life later on.

Earlier on I recommended a basic three−partition setup including a small root, a medium−sized system−software partition, and a large home−directory partition. Traditionally, these would be called /, /usr, and /home. The counterintuitive `/usr' name is a historical carryover from the days when (much smaller) Unix systems carried system software and user home directories on a single non−root partition. Some software depends on it.

If you have more than one home−directory area, it's conventional to name them /home, /home2, /home3, etc. This may come up if you have two physical disks. On my personal system, for example, the layout currently looks like this:

```
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sda1              30719   22337      6796      77%   /
/dev/sda3             595663  327608    237284      58%   /usr
/dev/sda4            1371370    1174   1299336       0%   /home
/dev/sdb1            1000949  643108    306130      68%   /home2
```

The second disk (sdb1) isn't really all /home2; the swap partitions on sda and sdb aren't shown in this display. But you can see that /home is the large free area on sda and /home2 is the user area of sdb.

If you want to create an partition for scratch, spool, temporary, mail, and news files, call it /var. Otherwise you'll probably want to create a /usr/var and create a symbolic link named /var that points back to it (the installation program may offer to do this for you).

7.6. Installing software packages

Once you've gotten past preparing your partitions, the remainder of the installation should be almost automatic. Your installation program (whether EGA or X−based) will guide you through a series of menus which allow you to specify the CD−ROM to install from, the partitions to use, and so forth.

Here we're not going to document many of the specifics of this stage of installation. It's one of the parts that varies most between Linux distributions (vendors traditionally compete to add value here), but also the simplest part. And the installation programs are pretty much self−explanatory, with good on−screen help.

7.7. After package installations

After installation is complete, and if all goes well, the installation program will walk you through a few options for configuring your system before its first boot from hard drive.

7.7.1. LILO, the LInux LOader

LILO (which stands for LInux LOader) is a program that will allow you to boot Linux (as well as other operating systems, such as MS−DOS) from your hard drive.

You may be given the option of installing LILO on your hard drive. Unless you're running OS/2, answer `yes'. OS/2 has special requirements; see Custom LILO Configuration below.

Installing LILO as your primary loader makes a separate boot diskette unnecessary; instead, you can tell LILO at each boot time which OS to boot.

7.7.2. Making a production boot disk (optional)

You may also be given the chance to create a ``standard boot disk'', which you can use to boot your newly−installed Linux system. (This is an older and slightly less convenient method which assumes that you will normally boot DOS, but use the boot disk to start Linux.)

For this you will need a blank, high−density MS−DOS formatted diskette of the type that you boot with on your system. Simply insert the disk when prompted and a boot diskette will be created. (This is not the same as an installation bootdisk, and you can't substitute one for the other!)

7.7.3. Miscellaneous system configuration

The post−installation procedure may also take you through several menu items allowing you to configure your system. This includes specifying your modem and mouse device, as well as your time zone. Follow the menu options.

It may also prompt you to create user accounts or put a password on the root (administration) account. This is not complicated and you can usually just walk through the screen instructions.

8. Booting Your New System

If everything went as planned, you should now be able to boot Linux from the hard drive using LILO. Alternatively, you should be able to boot your Linux boot floppy (not the original bootdisk floppy, but the floppy created after installing the software). After booting, login as *root*. Congratulations! You have your very own Linux system.

If you are booting using LILO, try holding down *shift* or *control* during boot. This will present you with a boot prompt; press *tab* to see a list of options. In this way you can boot Linux, MS−DOS, or whatever directly from LILO.

9. After Your First Boot

You should now be looking at the login prompt of a new Linux, just booted from your hard drive. Congratulations!

9.1. Beginning System Administratration

Depending on how the installation phase went, you may need to create accounts, change your hostname, or (re)configure X at this stage. There are many more things you could set up and configure, including backup devices, SLIP/PPP links to an Internet Service Provider, etc.

A good book on UNIX systems administration should help. (I suggest *Essential Systems Administration* from O'Reilly and Associates.) You will pick these things up as time goes by. You should read various other Linux HOWTOs, such as the *NET−3−HOWTO* and *Printing−HOWTO*, for information on other configuration tasks.

9.2. Custom LILO Configuration

LILO is a boot loader, which can be used to select either Linux, MS−DOS, or some other operating system at boot time. Chances are your distribution automatically configured LILO for you during the installation phase (unless you're using OS/2, this is what you should have done). If so, you can skip the rest of this section.

If you installed LILO as the *primary* boot loader, it will handle the first−stage booting process for all operating systems on your drive. This works well if MS−DOS is the only other operating system that you have installed. However, you might be running OS/2, which has its own Boot Manager. In this case, you want OS/2's Boot Manager to be the primary boot loader, and use LILO just to boot Linux (as the *secondary* boot loader).

An important gotcha for people using EIDE systems: due to a BIOS limitation, your boot sectors for any OS have to live on one of the first two physical disks. Otherwise LILO will hang after writing "LI", no matter where you run it from.

If you have to configure LILO manually, this will involve editing the file `/etc/lilo.conf`. Below we present an example of a LILO configuration file, where the Linux root partition is on `/dev/hda2`, and MS−DOS is installed on `/dev/hdb1` (on the second hard drive).

```
# Tell LILO to install itself as the primary boot loader on /dev/hda.
boot = /dev/hda
# The boot image to install; you probably shouldn't change this
install = /boot/boot.b

# The stanza for booting Linux.
image = /vmlinuz       # The kernel is in /vmlinuz
  label = linux        # Give it the name "linux"
  root = /dev/hda2     # Use /dev/hda2 as the root filesystem
  vga = ask            # Prompt for VGA mode
  append = "aha152x=0x340,11,7,1"  # Add this to the boot options,
   # for detecting the SCSI controller
```

```
# The stanza for booting MS-DOS
other = /dev/hdb1      # This is the MS-DOS partition
  label = msdos        # Give it the name "msdos"
  table = /dev/hdb     # The partition table for the second drive
```

Once you have edited the `/etc/lilo.conf` file, run **/sbin/lilo** as *root*. This will install LILO on your drive. Note that you must rerun **/sbin/lilo** anytime that you recompile your kernel in order to point the boot loader at it properly (something that you don't need to worry about just now, but keep it in mind).

Note how we use the *append* option in `/etc/lilo.conf` to specify boot parameters as we did when booting the bootdisk.

You can now reboot your system from the hard drive. By default LILO will boot the operating system listed first in the configuration file, which in this case is Linux. In order to bring up a boot menu, in order to select another operating system, hold down *shift* or *ctrl* while the system boots; you should see a prompt such as

```
Boot:
```

Here, enter either the name of the operating system to boot (given by the *label* line in the configuration file; in this case, either *linux* or *msdos*), or press *tab* to get a list.

Now let's say that you want to use LILO as the secondary boot loader; if you want to boot Linux from OS/2's Boot Manager, for example. In order to boot a Linux partition from OS/2 Boot Manager, unfortunately, you must create the partition using OS/2's **FDISK** (not Linux's), and format the partition as FAT or HPFS, so that OS/2 knows about it. (That's IBM for you.)

In order to have LILO boot Linux from OS/2 Boot Manager, you only want to install LILO on your Linux root filesystem (in the above example, `/dev/hda2`). In this case, your LILO config file should look something like:

```
boot = /dev/hda2
install = /boot/boot.b
compact

image = /vmlinuz
  label = linux
  root = /dev/hda2
  vga = ask
```

Note the change in the *boot* line. After running **/sbin/lilo** you should be able to add the Linux partition to Boot Manager. This mechanism should work for boot loaders used by other operating systems as well.

10. Administrivia

10.1. Terms of Use

10.2. Acknowledgements

My grateful acknowledgement to Matt D. Welsh, who originated this HOWTO. I removed much of the Slackware−specific content and refocused the remainder of the document on CD−ROM installation, but a substantial part of the content is still his.

The 4.1 version was substantially improved by some suggestions from David Shao <dshao@best.com>.