

## **Linux installation made easy ...** by Matt O'Connor Nov 2002

I wrote this article to ease the fears of the Linux-curious admin or hobbyist. If you ever thought that Linux was too labor intensive or too intellectual for you, then I urge you to read this article! I will show you that even the most computer-illiterate person amongst us can set up a Linux system. I have a fair amount of experience with both the Slackware Linux distribution and the Red Hat Linux distribution. Since these represent the least and most user-friendly Linux distributions, I will discuss the installation process for each. This will give the reader a fairly good overview of the full Linux installation spectrum.

Linux is just a name for a UNIX knockoff operating system. The operating system is nothing more than a fancy collection of programs and other files that govern the system resources and provide services for user tasks. You are already probably familiar with several other operating systems, such as DOS and Windows.

The operating system normally manages the system processor, the memory, the disk resources, the system interfaces, and the peripherals. In the case of a multitasking operating system like Linux, the operating system also manages the scheduling of user tasks so that the user(s) can actually run many tasks at once. In a multiuser operating system like Linux, the operating system also protects each user from the actions of all the other users.

Since the common PC typically boots off a hard disk, that's where we will install the operating system. You could design a system to boot from other devices, such as the network interface, the tape drive, a floppy drive, or even from firmware. However, we will only concern ourselves with the installation of the operating system on a hard drive and with the installation of the boot loader on a hard drive.

Linux prefers its own custom filesystem, called ext2, to the FAT16 filesystem or FAT32 filesystem found in Windows-based computers. You could probably get Linux to run within these Windows filesystems, but this would seriously impact security and performance. The ext2 filesystem associates an owner, a group, and a privilege specification with each filename. This allows Linux to protect against unauthorized access to privileged files, which is not supported in the Windows filesystems.

Already, we can see the challenges involved in installing Linux on a computer. Somehow, we have to install a Linux ext2 filesystem on a hard drive. Then, we have to install the Linux operating system into that filesystem, and lastly, we need to install a boot loader onto the disk that will allow us to boot our Linux operating system. Since many people will want to run both Linux and Windows on the same computer, I shall discuss methods for providing this dual-boot capability.

### ***The install strategy***

Regardless of which Linux distribution you want to install, all of them use the same basic strategy. In this section, I will discuss the sequence of events that must happen in order to create a Linux system. Then, in the following sections, I will discuss the specifics of the Slackware installation and the Red Hat installation.

Installing an operating system reminds me of the old chicken and egg problem. What better way to build a Linux system than to start with an existing Linux system? Linux already has the ability to partition disks and to format disk partitions for the ext2 filesystem and for Linux swap partitions. So to this end, we will use a working Linux system, fully contained within two floppy disks.

The floppies contain a boot loader, the Linux kernel, and a minimal filesystem complete with the required utilities. When the Linux kernel is loaded, the user can then log in and run any of the command files contained within the floppy filesystem. Before doing anything else, we must use the fdisk command to partition a hard drive. Once we have at least one ext2 partition and one Linux swap partition, we can save the new partition table back to the disk. When we do this for real, we must be very careful not to destroy any existing Windows partitions! If you hose the partition table during this step, chances are that your Windows data will soon be history.

Once the partition table is correctly written, the ext2 partitions must be formatted using mkfs. Once they are formatted, they can then be "mounted," or incorporated into the filesystem. Next, files can be copied from a distribution media into the new filesystems.

Normally, these distributions are supplied on CD-ROM or they will have been previously copied into an existing DOS filesystem. People with fast Internet connections could actually copy the distribution files over the Internet, but I highly recommend installing from CD-ROM or from files that have already been downloaded. You don't want to get halfway through the install and then crash because your ISP went down!

The Linux distribution files for Slackware are stored in a compressed archive format, which can be uncompressed and retrieved using the tar command. The Red Hat distribution uses the Red Hat Package Manager (RPM) format, which requires a different application for unpacking the distribution. Either way, the distribution files contain all the executable files, library files, configuration files, etc. Everything you need to operate your system exists in these distributions.

Once the distributions are unpacked into your new Linux hard disk filesystems, you technically have a working Linux system! At this point, however, you have no method to gain access to your new Linux system. To complete your system, you must install the boot loader. Any boot loader that can comprehend the Linux filesystem will work. I will discuss the LILO boot loader, since it is the most popular.

## ***Important preinstallation suggestions and information***

Before beginning any Linux installation, I highly recommend backing up any important files. You can never tell what might go wrong, and the best way to be completely safe is to make a backup. Both the Slackware and the Red Hat installer are very well written and neither will destroy your data without your authorization. However, mistakes still happen, and I would hate for your first experience with Linux to be a bad one! So, if you take the time up front before you attempt an installation, your chances of making a mistake will be very low.

## ***Obtaining the Linux distribution***

If you have not obtained a CD-ROM version of either Slackware or Red Hat Linux, you will need to download the distribution from the Internet. You can download the CD-ROM image file from Red Hat Linux, which would allow you to burn your own CD-ROM if you own a CD-RW

drive. If you don't have a CD-RW drive, you can still download the independent distribution files into a DOS/Windows partition and install from there. For this approach, you need to download everything in this ftp directory.

Determine where to install Linux next. Decide whether you want to repartition your existing DOS disk or just add a separate Linux drive. Since hard drives are so cheap now, I recommend acquiring a new drive. If you do this, you stand a far smaller chance of getting confused and accidentally altering the partition table on the existing drive.

If you do decide to repartition your existing DOS drive, you must first defrag the drive so as to squish all the data up to the front of the existing partition. Next, you need to run a program like FIPS.EXE, which can split a partition into two pieces. You could also use PartitionMagic or your favorite partitioning tool. I have had good luck using FIPS, but again I would much rather see you just add a second drive.

If you bought a new disk specifically for your Linux system, consider making a DOS/Windows partition at the front of that drive. I normally put a 500-MB Windows 98 partition at the beginning of my new drives, because I always find that I need more Windows disk space. Linux can always access a DOS/Windows partition; however, the reverse is not true. Later, you can use this extra space to store either Windows or Linux files. You could even reformat it to be a Linux partition, if you wanted to. By the way, this would be an excellent place to copy your Linux distribution files in the event that you were doing an ftp-based install.

## ***Understand Linux device naming conventions and disk partitioning***

Before going forward, make sure you learn the Linux device names for your drives. Your first hard drive will be named /dev/hda and your second hard drive will be named /dev/hdb. The first partition on your first hard drive would be named /dev/hda1, while the second partition on that same drive would be named /dev/hda2, and so on. Assuming your Windows system is located in /dev/hda1, make sure you understand this and don't attempt to reformat that partition or change it in any way! Your Linux system should start either in /dev/hda2 or in /dev/hdb1, depending on whether you split the first disk or bought a second drive.

Prior to installation, you must decide how you will partition your hard drives. This is really no different than when you used the FDISK and FORMAT commands for DOS/Windows. At a minimum, you must create a root partition, named /, and a swap partition. UNIX, in general, has a fairly standard directory structure that almost always contains the following components:

- / —The root partition, the anchor of all path names
- /bin—System critical executable files, required for single user boot
- /etc—Control and configuration files such as the password file
- /usr—System software executable files, configuration files, library files, etc.
- /var—System log files and temporary files for editors and other applications
- /tmp—Temporary files, always can be read or written by any user

## ***Understand Linux swap partitions***

I always select the maximum swap partition size of 128 MB, and you should too! Swap space is needed whenever programs use more memory than physically exists in the system. Linux will kill off user processes if it runs out of memory, so having lots of swap space is a good idea. As you get more experienced with Linux, you might someday optimize system performance by moving your swap partition to a different hard drive than the one on which your data files and programs reside. Since the drive can only read or write from one physical location at a time, you don't want your system swapping functions slowing down your other disk accesses. But this is a suggestion for another day!

## ***Understand disk partitioning and mount points***

Before we go on, I think a review of partitions and mount points would be beneficial. Linux maps the physical disk partitions into the filesystem at a specified mount point. For example, your existing Windows C: drive will be accessed through the special file `/dev/hda1`, which represents the first partition of the first disk. When we want to access that disk partition within Linux, we will want to see it in our directory structure at a mount point of `/win98` or maybe `/mnt/win98`. If we do this, then a user can type a command such as `ls /mnt/win98` and the system will list the contents of the C: drive. The mount point is the position within the directory hierarchy in which the disk partition shall present itself.

The actual name of the mount point is nothing more than a directory file created with the `mkdir` command. You must always remember this fact because it can bite you when you least expect it! What happens if, for some reason (like an improper shutdown), a disk does not get mounted to its mount point? That directory name for the mount point still exists, and correctly privileged processes can still read and write that directory! Of course, this directory was intended to be nothing more than the placeholder for some other disk partition. So if you were to write a file to that directory, you might later discover your file missing when the system finally mounts that other partition. Your file will still be out there, but you won't have any way to access it, since every time you attempt to access that directory, you'll instead access the mounted partition! If you do inadvertently write a file into a mount point directory, all you have to do to recover the file is unmount the partition from that mount point, and you can then move the file.

When a directory is created within an existing directory, that new directory shall exist within the same partition as its parent. One way to overcome this is to use symbolic links. A symbolic link is nothing more than a directory entry that points to some other location in the directory hierarchy. For example, let's say you wanted to create a directory called `/home`, but you didn't want it to exist within the physical root partition. You could create a directory in the `/usr` partition called `/usr/home`, and then you would use the command `ln -s /usr/home /home`

This command creates a directory entry at `/home` that really points to the physical directory structure at `/usr/home`. Anytime you write a file to `/home`, it's really being written into `/usr/home`. Likewise, anytime you read a file from `/home`, you are actually reading from `/usr/home`. I will use this capability to adjust the directory structure to match the physical mapping that I desire.

## ***Planning your filesystem layout***

Remember: IDE disks only support four partitions in the partition table. Let's create an example system. The first partition on our example system will be a DOS/Windows partition. This will be true if you are installing Linux behind your existing Windows partition on your C: drive, and it

would still be true if you took my advice about creating a 500-MB DOS/Windows partition at the front of a new drive.

We already declared one DOS/Windows partition, and we know we need at least one Linux swap partition. Thus, we have only two partitions left, unless our system shall span multiple disk drives (which you might well want to do someday, but not right now). The kernel and the minimal set of files required for booting should be on an independent partition. Likewise, the user's home directories should be isolated from everything else on the system. In an ideal world, I would also want to have my /usr and /var mount points as separate partitions. The /usr contains the programs, libraries, and other files that implement system functions other than those most critical ones found in the root directory. The /var contains all the system log files, which might get large and thus could potentially crash the system if they fill the root partition.

A friend of mine sets up lots of Red Hat systems using a huge root partition, a fairly small /boot partition, and a swap partition. This is a very convenient strategy, and it does a good job of keeping the user partition isolated from the kernel and all the boot critical code. However, in this strategy, the user filesystem is really just a subdirectory of the root partition. In the event that a user somehow manages to corrupt this partition, you might not be able to access the critical stand-alone files, such as the cd or ls command, that would help you boot the system.

I tend to favor a different strategy for some very different objectives:

- I want the root partition to be a stand-alone bootable minimal system complete with all the files that one would need if he or she booted the system in single user mode.
- I want the rest of the system to be in a different physical partition so that I have the absolute minimum potential for damage to the root partition. Ideally, this would mean not having the /usr or /var located in the physical root partition.

Unfortunately, we don't always get what we want.

Thus, my recommendation would be to declare a root partition of size 128 MB or so. This will hold the kernel and all boot files, as well as the /bin directory and the /var directory. Next, I would declare the /usr partition to be the rest of the available space on the disk. Within the /usr partition, I would create /usr/home, and then I would create a symbolic link from /usr/home to /home. This points the /home directory to physical space in the /usr partition. Someday, you might wish to add another disk, and then you could redirect /home to point to that disk instead of /usr/home. Likewise, you could redirect /var to someplace else in the future, but that's probably not necessary.

So if you go with my friend's strategy, you would have a disk structure that looks like Table A.

**Table A**

Partition	Mount point	Description
/dev/hdb1	/mnt/win98	This is the Win98 filesystem we discussed.
/dev/hdb2	/boot	Kernel and related files go here; allocate 25 MB or so.
/dev/hdb3		Linux swap

This is your 128-MB swap space.  
/dev/hdb4  
  /

    This is where everything else goes.

    A very simple Linux partition table

Under /, we would find /bin, /etc, /usr, /var, and everything else occupying physical space within this partition on /dev/hdb4.

**Table B**

Partition	Mount point	Description
/dev/hdb1	/mnt/win98	This is the Win98 filesystem we discussed.
/dev/hdb2	/	This is where everything goes.
/dev/hdb3	Linux swap	This is your swap space.
/dev/hdb4	/usr	User data will be stored here.

## ***My Linux partition strategy***

Under the strategy listed in Table B, we would find /bin, /etc, /var, and everything else located in the /dev/hdb2 partition. This will install just fine, but after the install, we will go back and move /home to /usr/home and then create a symbolic link from /usr/home to /home so that /home points to physical space in /dev/hdb4 and not to space in /dev/hdb2. Using the following sequence of commands, this process is trivial:

```
cp -R /home /usr
rm -rf /home
ln -s /usr/home /home
```

This strategy is nice because it puts /bin and /etc together with the kernel in the root partition. If /usr ever gets hosed up, you can still boot your system and hopefully recover the data in /usr. You can use either of these strategies or even design one of your own choosing.

## ***Installing Slackware Linux***

You will need two formatted floppies to install Slackware Linux. Don't trust factory formatting, since any defect in the formatting might sabotage the installation. To save time later, format the floppies yourself.

Next, locate the RAWRITE.EXE utility in your distribution. Depending on how things are arranged, you might have to hunt around for this utility. I found it here. Using RAWRITE, you will want to create two disks; the first will be your boot floppy and the second will be your root floppy. To make the boot floppy, locate the boot image file bare.i (according to the Slackware Installation Help) and copy it to the boot floppy using the command:

```
RAWRITE BARE.I A:
```

I am assuming that you found the image file in the FLOPPIES directory. Adjust this path to reflect the correct directory in the event that your version of Slackware doesn't match. Similarly, create the root floppy using the command:

```
RAWRITE COLOR.GZ A:
```

I have never encountered anyone who had trouble with the graphics in color.gz, but in case you do, you can retry using text.gz instead. Once you have these two disks prepared, you can reboot the machine with the boot disk inserted into the floppy drive.

Once the two floppies have been created, you can insert the boot floppy into the floppy drive and then shut down Windows with the restart option. You might need to check your BIOS settings to make sure that your boot sequence starts with the A: drive. If this is not the case, then your system won't boot from the floppy! Assuming we leap that hurdle, the system will present you with a boot: prompt, at which you will most likely just want to press [Enter]. In the event that you have special needs, read over the text on the screen and see if anything applies. I hope that, if you copied all the needed files into a Windows partition, you won't need any special options. The system will soon prompt you to switch disks, and at that time, you can replace the boot disk with the root disk. After that, your system will come up with the slackware login: prompt. Just slightly above, it tells you that you may log in as root, so go ahead! Type root, press [Enter], and you will be sitting at the shell prompt, which is indicated by the # symbol.

It's show time now! From here, you can do many different things. One interesting thing to play with is the multiple shell availability. Press [Alt][F2] and you will discover a whole new virtual terminal. Press [Alt][F1] and you will be sent back to the original virtual terminal where you already logged in. This very nice feature allows you to display Slackware installation documentation in one virtual terminal while you perform the various tasks in the others. Log in on a second virtual terminal and then type the setup command. This starts the installation setup program. Next, select the HELP option to bring up the Slackware setup help. You can refer to that guide to get another opinion on how you might want to perform your install.

Now in another virtual terminal, choose either the fdisk command or the cfdisk command. Since I am most familiar with fdisk, I would invoke this command, as follows:

```
fdisk /dev/hfb
```

This allows us to edit the partition table of your second drive. If you type the p command, you will see any existing partition information. Assuming you already created a Win98 partition, you should see something like this.

You will then want to use the n command to add the new partitions. The sequence should go something like this. (Don't worry too much about the numbers it prints out when it asks you for a cylinder number. You don't really need to worry about the cylinder numbers.)

First cylinder

At the First cylinder prompt, always enter the lower of the two numbers.

Congratulations, you just entered your first partition entry! By default, it's automatically labeled as a Linux native partition. When we get done entering partition data, we will go back and change the types. Next, let's create the swap partition.

### **The t command**

The t command is used to change the partition type.

82

The 82 code is for the Linux swap type.

Finally, we enter the last partition.

Using the rest of the disk

In the case of our above partition, we will want to enter the last number for Last cylinder so we use the rest of the disk.

Next, inspect the partition table using the p command. If it looks right, you can use the w command to write it to the disk. When you inspect the partition table, look to make sure that you have one DOS/Windows partition, two Linux native partitions, and a Linux swap partition. Under the Start and End labels, verify that the start cylinder of each line directly follows the end cylinder of the previous line.

Once you have written out the partition table, you can go back to the virtual console where you have the setup program running. If you don't have setup running in another console, just start it now using the setup command.

The setup program has pretty good documentation built into it, and you can pretty much follow the menu in order. If you select the ADDSWAP option, this will automatically find your swap partitions and format them for you. After it completes, this step automatically asks you if you want to move into the next step: the selection of your TARGET. You should follow along with it. This option will format your root partition and any other Linux native partitions you might have. Accept the options as given, unless you have some strange reasons why you would prefer to have smaller block transfers. I recommend sticking with 4096 or larger, since a smaller size will require more block transfers to move the same amount of data.

Once the setup program has formatted a partition, it allows you to declare a mount point for it. The first Linux native partition you work on will become the root mount point. The second one, /dev/hdb4, should be called /usr (assuming you stuck with my strategy).

Next, the program will detect the various DOS/Windows partitions and ask you if you want to mount them. Certainly, you do want this, so select Yes! It will show you the various partitions out there, and you can select each one and give it a mount point. I tend to call them by names like /dosc and /dosd to remind me of what Windows would call them. You might call them /mnt/win98 or /mnt/dosc or whatever.

At this point, you have your filesystem pretty much in place. You have a root partition, a /usr partition, and an active swap partition. Not only that, but some of your configuration files are already set up in your /etc directory! All that's left is to start copying the distribution files into your filesystem. The setup program guides you by the nose to next select the SOURCE to copy from. In the event that you have the CD-ROM, try using it. If Linux recognizes your CD-ROM, then you are home free. In the old days, this used to be a problem, but I think Slackware (and most Linux distributions) recognizes most every CD-ROM drive out there now.

In the rare event that it doesn't work, go back to DOS/Windows, copy the CD-ROM to someplace in your DOS/Windows filesystem (that 500-MB partition I talked about earlier could come in handy now), and then repeat the previous steps. When you have to select your SOURCE, this time choose Install From A Hard Drive Partition. You will have to identify which partition contains the distribution and provide the path to the distribution. Aren't you glad you read the tutorial on disk partitions?

Next, the setup program asks you which files you want. I recommend installing absolutely everything! You can always remove it later, and disk space is cheap. After it finishes installing the distributions, it asks you which kernel you would like. Now you know why everybody prefers Red Hat! Choose your favorite kernel, or use the SKIP option that just keeps the kernel we have been using.

Believe it or not, we only have one step left. Once we put LILO onto the boot sector, we will have a way to boot up the system. Get a blank disk and create a LILO boot disk. In the event that something happens to your boot sector, this will get you back into the system. LILO lets you identify each partition that you want to be able to boot. It first asks you for your partition name and then asks you to come up with a unique identifier label. In our sample configuration, we might want to declare something like that in Table C.

Partition	LILO name
/dev/hda1	win98
/dev/hdb2	Linux

Make sure your identifier label is memorable!

After you configure LILO, the Slackware installation guides you through the configuration of your modem and your network. You can follow the menus and set all that up on your own. Fortunately, you can always go back and edit anything that you didn't do correctly the first time. Again, this is all stuff that Red Hat does for you, which is another good reason why you might want to check into the Red Hat distribution, which I will cover next!

## ***Installing Red Hat***

Most inexperienced Linux users will prefer the Red Hat installation procedure because of its simplicity. As an experienced Slackware user, I cannot say enough good things about the Red Hat install process! First, it only requires one floppy, and you make that in the same manner that you make the Slackware floppies, using RAWRITE.EXE. Red Hat keeps the floppy image in a file

under X:\IMAGES\boot.img, where X is the drive where your distribution CD-ROM or other media is located. Again, use a floppy that you personally formatted and you won't have any problems.

### ***One or two CD-ROMs***

I tried out the Red Hat CD-ROM with the boot floppy and it worked perfectly. However, it could only recognize the first CD-ROM drive in my system. (I have a CD-RW and a CD-R and it only saw the CD-RW.) This, of course, is not the definitive case.

Once you have your floppy created, reboot the system with the floppy in the drive. While the system is booting, put your Red Hat CD-ROM in the drive. The floppy will automatically come up and find the CD-ROM. It then automatically boots into an XWindows-based installer application. The application first asks you what language you want. Next, it asks for keyboard configuration information. With each question, it offers an extensive help text along the left side of the screen. I have never seen an installer application as good as this one, not even in Microsoft operating systems! After you declare your keyboard and mouse types, it asks you if you want to go with one of its canned configurations. At this screen, you can also declare the difference between an install vs. an upgrade. If you install, you typically format all the Linux partitions, so this is a nice safety feature. Also, you have the option of using fdisk as your partitioning tool. I really liked its Disk Druid partitioning tool, so I recommend you give it a try too.

I chose Custom on its install type, but you will have to play around and go with whatever you like. Personally, I like the OpenLook window manager, but it's not offered on this screen. Most people are using the KDE desktop these days, so that might be a good option. If you want to be in good graces with the GNU people, the GNOME desktop is the way to go. After you install, you should take a look around the Web to see what window manager best suits your needs. Linux lets your desktop take on any look that you would like. Most window managers have virtual desktops, and each has its own unique way of bebopping from one part of the desktop to another.

Once you make your desktop decision, go on to the next screen where you set up your partition information. First, go and assign mount points to the DOS/Windows partition. This will help you remember which one is the Windows partition. Next, use the Add button to create your new partitions (please review the section on preinstallation suggestions). When you leave that screen, you have the option to format the various Linux native partitions. You must format them, unless you are not installing for the first time.

After creating and formatting your new Linux partitions, the Red Hat installer asks you to set up your LILO configuration. It works the same as the Slackware installer and it has a great help text onscreen, so I won't dwell on it here.

For now, we will skip the network configuration screen, which comes next in the install process. You can always return to this using the upgrade option later. In the next screen, you set your time zone using a really nice graphical map. Next, you set your root password, and it even allows you to add additional users using a graphical tool. In the next screen, definitely leave the MD5 and shadow passwords options selected. Anything that improves security is a good idea! Next, you are asked to select which package groups you want. I highly recommend selecting everything. Again, disk space is cheap and the best way to learn everything that Linux offers is to have it available to you!

The Red Hat installer even sets up your XWindows server for you. All you have to do is identify your graphics card and monitor. This is really the last hurdle. After finishing this screen, all you have to do is let it go to work!

## ***Conclusion***

Linux installation has been made easy and painless. Now that hard drives are so inexpensive, one can hardly justify not giving Linux a try! Apart from the cost of the hard drive and the cost of your time, everything else is free. More importantly, the source is freely available. This means that if a bug exists, you can find someone who will fix it. If you need a new feature, you can find someone who will provide it!