Informational

Instructional

Installation

System

Server Setup

Link button:

LINUX GUIDE
by Joshua Go

Comments, requests, and submissions are also accepted through the guestbook.

Introduction

Ever since I started using GNU/Linux around January of 1997, I've noticed continual change in the state of the GNU/Linux operating system. Large corporations have pledged support for the Linux community and ported their applications to run on Linux. Before my eyes, small startups have grown into monstrous vendors, drawing respected investors. At the same time, I've seen Linux websites come and go; some of them slowly died off, while others continue to prosper immensely. One of the major key factors behind this dynamic change is the exponential growth of the user base.

It seems rather conceited to think that I could make even a small difference in the acceptance of the GNU/Linux operating system. Thanks to e-mails sent in over the lifetime of this guide, however, I am under the impression that I have directly (and sometimes indirectly) impacted the acceptance of GNU/Linux. Seeing my site linked to from several prominent spots on the Internet has further inflated my ego although I'm not responsible for all of the work here.

I must warn you that some people in the Linux community are reluctant about the widespread acceptance of the GNU/Linux operating system. They're scared about the "dumbing down" of users. I am sincerely concerned about that as well, so will push myself to ensure that users will be properly informed and educated. However, what I am driving at is that when you ask for help, some people will be a little more intolerant than you might have expected. Even in #LinuxHelp on the Undernet, which I consider a friendly and helpful place, it is not uncommon for those who are helping out to be irked by users bombarding them with questions. I hope to do my part by providing a place to read up on the most common problems.

If you find something that doesn't seem to be covered here, by all means, do not hesitate to drop by #LinuxHelp on the Undernet (IRC) and ask. I will be there sometimes as **JoshTech** or **jtg**.

Lastly, I urge you to be patient with GNU/Linux. The first couple of weeks will *probably* be rough on you, but after that, it will be a breeze if you persevere in learning. (I must note that with proper documentation, there is a high probability that you will be able to pick it up in less than a week.)

Comments, questions, suggestions, corrections? Send them all to jtg@computers.iwz.com.

Linux Commands

Created: October 28, 1998
Last updated: June 9, 1999
Development stage: Beta

These commands will work with most (if not all) distributions of Linux as well as most (?) implementations of Unix. They're the commands that everybody knows. To be able to survive in Linux, you should know these. There aren't always handy-dandy tools for X that shield you, especially if you're managing your own system, stuff often goes wrong and you're forced to work with the bare minimum.

I hope I didn't scare you.

Index

TOC

Navigation

Navigating around the files and directories of your hard drive could be a dreaded task for you, but it is necessary knowledge. If you were a user of command prompt interfaces such as MS-DOS, you'll have little trouble adjusting. You'll only need to learn a few new commands. If you're used to navigating using a graphical file manager, I don't know how it'll be like, but some concepts might require a little more clarification. Or maybe it'll be easier for you. Who knows? Everyone is different.

cd

As you might already have guessed, the **cd** command changes directories. It's a very common navigation command that you'll end up using, just like you might have done in MS-DOS.

You must put a space between **cd** and the ".." or else it won't work; Linux doesn't see the two dots as an extension to the cd command, but rather a different command altogether. It'll come to make sense if it doesn't already.

ls

The **ls** letters stand for **lis**t. It basically works the same way as the **dir** command in DOS. Only being a Unix command, you can do more with it. :-)

Typing `ls` will give you a listing of all the files in the current directory. If you're new to Linux, chances are that the directories you are commonly in will be empty, and after the **ls** command is run, you aren't given any information and will just be returned to the command prompt (the shell).

There are "hidden" files in Linux, too. Their file names start with a dot, and doing a normal **ls** won't show them in a directory. Many configuration files start with a dot on their file names because they would only get in the way of users who would like to see more commonly used items. To view hidden files, use the **-a** flag with the **ls** command, i.e. `ls -a`.

To view more information about the files in a directory, use the **-l** flag with **ls**. It will show the file permissions as well as the file size, which are probably what are the most useful things to know about files.

You might occasionally want to have a listing of all the subdirectories, also. A simple **-R** flag will do, so you could look upon `ls -R` as a *rough* equivalent of the `dir /s` command in MS-DOS.

You can put flags together, so to view all the files in a directory, show their permissions/size, and view all the files that way through the subdirectories, you could type `ls -laR`.

pwd

This command simply shows what directory you're in at the moment. It stands for "Print Working Directory". It's useful for scripting in case you might ever want to refer to your current directory.

File Management

TOC

A lot of people, surprisingly for me, prefer to use graphical file managers. Fortunately for me, I wasn't spoiled like that and used commands in DOS. That made it a bit easier for me to make the transition to Linux. Most of the file management Linux gurus do is through the command line, so if you learn to use the commands, you can brag that you're a guru. Well, almost.

cp

Copying works very much the same. The **cp** command can be used just like the MS-DOS **copy** command, only remember that directories are separated with slashes (/) instead of backslashes (\). So a basic command line is just **`cp filename1 filename2`**.

There are other extensions to the **cp** command. You can use the **-f** command to force it. You can use the **-p** command to preserve the permissions (and also who owns the file, but I'm not sure).

You can move an entire directory to its new destination. Let's say you want to copy a directory (and all of its contents) from where you are to be `/home/jack/newdirectory/`. You would type **`cp -rpf olddirectory /home/jack/newdirectory`**. To issue this command you would have to be in the directory where the subdirectory "olddirectory" is actually located.

ln

A feature of **lin**king files is available in Linux. It works by "redirecting" a file to the actual file. It's referred to as a **symbolic link**. Don't confuse this term with the linking of programs, which is when binary programs are connected with libraries that they need to load in order to run.

The most simple way that I've ever used **ln** to create symbolic links is **`ln -s existing_file link`**. Evidently there's a hard link and a symbolic link; I've been using a symbolic link all along. You can also use the **-f** flag to force the command line to overwrite anything that might have the symbolic link's file name already.

To remove a symbolic link, simply type **`rm symbolic_link`**. It won't remove the file that it's linked to.

mv

The **mv** command can be used both to move files and to rename them. The syntax is **`mv fileone filetwo`**, where "fileone" is the original file name and "filetwo" will be the new file name.

You can't move a directory that is located in one partition to another, unfortunately. You can copy it, though, using **`cp -rpf`**, and then remove it with **`rm -rf`** later on. If you have only a single partition that makes up your filesystem then you have very little to worry about in this area.

rm

The **rm** command is used for **rem**oving files. You use it just like the **del** or **delete** command in MS-DOS. Let's say you want to remove a file called `foobar` in your current directory. To do that, simply type **`rm foobar`**. Note that there is no "Recycle Bin" like in Windows 95. So when you de file, it's gone for good.

To delete something in some other directory, use the full path as the file name. For example, if you want to delete a file called "windows" that's in the directory /usr/local/src/, you would type **`rm /usr/local/src/windows`**.

To remove an entire directory and its contents, type **`rm -rf /directory`** where "/directory" is the path to the directory that you want to delete. If you're wondering, the "rf" stands for "recursive" and "force". Be very careful with this command, as it can wreak havoc easily if misused.

Editing

If you haven't figured out how important a text editor is, you soon will. Graphical interfaces can't shield you forever, and those utilities have their limits. Besides, if you're reading this page, I'm inclined to think that you want to be able to customize beyond the capabilities of graphical utilities. You want to work at the command prompt. I know you do.

The basic syntax to invoke these text editors is the same. Type the name of the editor followed by the file you want to edit, separated by a space in between. Non-existent files will be blank. Blank files will be blank as well.

emacs

To use GNU Emacs (or its counterpart, XEmacs), there are really only two commands you need to know. Heck, they're the only ones *I* know.

While you're editing a certain file with **emacs** or **xemacs**, you can save it with the **`[Ctrl]-x [Ctrl]-s`** keystrokes. Then to exit, type **`[Ctrl]-x [Ctrl]-c`**.

pico

The instructions for using **pico** are located on the screen. You save the file by using the **`[Ctrl]-o`** keystroke (for write-*o*ut) and exit with **`[Ctrl]-x`**.

As a permanent solution, you probably don't want to use **pico**. It lacks real power. Since I am such a wuss, however, I still have the bad habit of using **pico** once in a while. Why? By pressing **`[Ctrl]-j`** I can get entire paragraphs wrapped into a nice justified block. I don't know how to do that with the other text editors.

vim

Most modern distributions include **vim**, derived from the infamously arcane Unix editor, **vi**. (It stands for **vi** **Im**proved, as a matter of fact.)

Using **vim** is different in that there are several modes in which you use it. To do actual editing of the files, press **`[ESC] i`** (both separately). Then to save it, press **`[ESC] : w`**. Escape, the colon, and "w" should be keyed in one after the other. Finally, to quit, type **`[ESC] : q`**. The same rules apply as in previous **vim** commands.

You can use "w" and "q" at the same time to enable yourself to write to the file and then quit right afterwards. Just press **`[ESC] : w q`**.

If you don't have **vim** installed, try **vi** instead.

Monitoring Your System

An important part of system administration (especially with your own system) is being able to know what's going on.

tail

The program **tail** allows you to follow a file as it is growing. Most often, I use it to follow `/var/log/messages`. I do that by typing **`tail -f /var/log/messages`**. Of course, you can use anything else, including the other logs in `/var/log/`. Another file you may want to keep an eye out for is `/var/log/secure`.

If you want to leave that running all the time, I recommend having some sort of terminal program in X, logged in as root through **su**.

Another program you may want to look at is **head**. It monitors the top of the file specified, instead of the bottom.

top

This program shows a lot of stuff that goes on with your system. In the program, you can type:

1. **M** for memory usage information
2. **P** for CPU information
3. **q** to quit

Once you try it, you can see that **top** shows you the memory usage, uptime, load average, CPU states, and processes.

w

Typing **`w`** will tell you who is logged in. This can be helpful if you're the only one who uses your computer and you see someone logged in that's not supposed to be.

Another alternative is **who**.

Shutting Down and Rebooting

To shut down your system, type **`shutdown -h now`**, which tells the **shutdown** program to begin system halt immediately. You can also tell it to halt the system at a later time, I think, but you'll have to consult the **shutdown** manual page for that (**`man shutdown`**).

To do a reboot, you can either type **`reboot`** or **`shutdown -r`**. You can also use the famous Ctrl-Alt-Delete combination to reboot, which you might already be familiar with.

Shutting down and restarting properly (as described above) will prevent your filesystem from being damaged. Filesystem damage is the most obvious of the consequences, but there are probably other things out there that I don't know about. The point is, shut down your system properly.

There are (rare!) cases in which the machine might lock up entirely, and prevent you from being able to access a command prompt. Only then will your last resort be to do a forced reboot (just pressing the restart button on the case).

Related Books

1. Learning the VI Editor - a book to teach you how to use the standard text editor for all Unix systems.
2. Learning GNU Emacs - another book on another powerful text editor. It's good especially if you program or simply prefer Emacs.
3. Linux System Administration Handbook - for those who are using their Linux box as a server

---

TOC

Using chmod

Author: jbm
Created on: August 27, 1998
Last modified: September 16, 1998
Status: Alpha

`chmod` has a bit of a bad reputation for being confusing. I'll be honest with you: at first, it makes very little sense. You just have to be patient and try to understand the basics. This document aims to help you in gaining that understanding.

Basic Unix Filesystem Security (UID, GID, and permissions)

Knowing the basic Unix filesystem security issues will help you dramatically. Explained here are concepts to make your understanding of these issues skyrocket. Well, that's what we'll try to do, anyway.

UID/GID

In Unix, every user is a member of a group. For example, when you add a user, that user is normally a member of the group **`users`**. Almost all users are members of the group **`users`**. Notable exceptions are normally members of **`wheel`** or **`admin`**, the administrative groups. Note that the computer doesn't see the username when it does anything, but it sees a user ID. A user ID is (technically) a number, while a username is a string associated with that number (in pseudo-C, it's char *usernames(userid)). The same holds true for groups and group IDs. User ID is normally abbreviated "uid" and group ID is normally abbreviated "GID". I'll use these abbreviations from here on out.

Ownership

Each file in the directory structure is owned by a user. Each user is owned by a group. To keep things flexible, each file is *also* owned by a group - by default it's owned by the group that owns the user.

Why This Stuff is Important: Permissions

Permissions define what users of various groups can do with the file. There are three basic things you can do with any file: read from it, write to it, and execute it. Permissions is based on this concept, combined with UID/GID, and the hierarchy of users/groups.

You set permissions using the **`chmod`** command, and you can do it one of two ways: with letters or with numbers. Numbers is the preferred method (fairly easy to use, once you get the hang of it). In fact, I feel that letters are about useless, unless you are doing advanced tasks (and by the time you know enough to need such functionality, you should be able to read man pages). Understand that I'm simplifying things a bit - for a better reference see **`man chmod`** - but this information should be enough to get you through most normal maintenance tasks.

chmod by the Numbers

The basic format for chmod is **`chmod xyz file.foo`**. x, y, and z are each a number between 0 and 7. Each number represents the permissions of a group - x is for the user that owns the file, y is for the group that owns the file (normally the user's group), and z is for everybody else. To determine the actual values for each number, you use the following method: start with x = 0. If you want to be able to read from the file, add four. x can be 0 or 4 at this point in time. If you want to be able to write to the file, add two. x can now be 0, 2 (a write-only file??), 4 (read-only file), or 6 (read/write file). If you want to be able to execute the program, add one. You now have a full range of possible numbers:

```
     Number   | Permissions
   ----------+------------------------------------------------------------------------
       0      |   None - cannot read or write or execute
       1      |   Can execute, but cannot read or write
       2      |   Write-only, cannot read or execute (??)
       3      |   Write-able/executable
       4      |   Read-only, cannot write to or execute
       5      |   Read-only executable, cannot write to
       6      |   Readable Writeable file, but not executable (ie: text file)
       7      |   Readable Writeable Executable file - most programs are this
```

You use the same process to determine the number representing the permissions for the group that owns the file (y) and for the rest of the world (z). It's typically a bad idea to **`chmod 777`** any file, as it allows the world to replace the program with whatever they'd like.

Note that root can mess with files however they darn well please. When root uses **`chmod`** on a file, the ownerships do not change, but the permissions are changed. If you want to keep a user from accessing a file that they own, you must change the ownership to root (or anyone else, for that matter). You can change ownership using **`chown`**.

Setting execute for a directory allows that directory to be read. That is, you can see what's in it.

chown

`chown` is a lot simpler than **`chmod`**. You simply do **`chown new-owner. filename.foo`** as root and "new-owner" now owns the file and the file's group is set to new-owner's group. If you'd rather keep the group ownership the same, drop off the "." after "new-owner". You can change only group ownership with this, too: **`chown new-owner.new-group filename.foo`**. To just change the group ownership do **`chown .new-group filename.foo`** and it will leave the user ownership alone, while changing the group ownership.

Security Considerations

Be careful. Don't give the world write access to anything executable and just generally think before you do. Neither I, the Guide, or anybody else takes responsibility if for some reason this document messes things up, either by my stupidity, *your stupidity*, or freak accident. Those *italics* are to emphasize that you should be careful to avoid stupid mistakes with chmod - it can and probably will create some small security holes, especially if misused or used carelessly. So be cautious. End of security ramble.

Linux Basics and Tips

On this page, I (and many others) just wrote some stuff that did not deserve (or need) their own section because of their simple nature. I suggest anyone new to Linux and Unix read this section so that you will jump in with enough knowledge. You probably have a lot more to add to this, so make sure you e-mail it to me to help more people.

- **Free Space:**
  To find out how much disk space you have left on your hard drive or mounted floppy drive, type **df**.
- **Program Info:**
  For details about using a program you can use the "man" command to access the manual page for that particular program. For a description of the command "ls" you would type **man ls**. Some programs do not have manual pages ("manpages"), and there are manpages that aren't really much help. However, as you get more advanced with Linux you will find manual pages to be an invaluable resource. Mark Schreiber wants to remind us of the **info** command as well. Use **info** when looking for Texinfo documentation.
- **Deleting Directories:**
  To remove an entire directory without being prompted if you want to delete each file, type **rm -rf directory-name**, where "directory-name" is the name of the directory. Be careful with this! (Also works with the **cp** (copy) command, when you overwrite old files.) In case you're wondering, the **r** is for *recursive* (moves through subdirectories) and the **f** stands for *force*.
- **Shutting Down:**
  Shut down your system properly. Use the command **shutdown -h now** as root. If you want to reboot, type **reboot** or **shutdown -r now**. Mark Schreiber reminds us that the Ctrl-Alt-Delete combination works as well, and does not require a root login.
- **Finding Stuff:**
  Find any file on your system by typing **find / -name <name>**. The "/" is telling it to start from "/" and search everything under that, which is everything on your filesystem. In place of "name" put the name of the file you want to look for. You can also use "*name*" to find anything with the string "name" in its filename, for example.
- **MOTD:**
  When a user logs in, you can edit **/etc/motd** to print out a message on their screen. That file must be edited when you are root. Just type in the plain text you want to display when someone logs in.
- **Do not IRC as root:**
  Do not log on to IRC as root. Even if Nobody decides to try to penetrate your system, you are going to be banned from a lot of the good technical help channels and servers.
- **Bailing Out:**
  If you're stuck in a program that won't respond, try the "Ctrl-c" combination. In many cases even that won't work, so use "Ctrl-z" to force it to stop. It's best to exit the program the proper way, though.
- **Managing RPMs:**
  In any distribution that uses the Red Hat package manager (Red Hat, Caldera, and TurboLinux are

TOC

a few) you can just use the **rpm** command to install it from your CD. Use **rpm -i <filename>** install a package; **rpm -U <filename>** to upgrade a package; and **rpm -i --force <filename>** you installed a package, but didn't use **rpm** to remove it and want to reinstall it. The file name should end in "*.rpm". If **rpm** doesn't work on your system then you don't have it installed.

- **Upper-case to Lower-case:**
  This little script renames all the stuff in a directory to lowercase... **`for x in *; do mv $x `echo $x | tr [A-Z] [a-z]`; done`**. (code by Warren Blumenow)
- **More Managing RPMs:**
  To find out if you have an RPM package installed, the syntax is **rpm -qa | grep "part_of_package_name"**.
- **Boot Scripts:**
  The startup scripts are usually located in `/etc/rc.d/`. Inside that directory you may find files such as `rc.local`, `rc.sysinit`, `rc.serial`, and even some subdirectories. The files contained in there may vary with each distribution. If you want to start a program when Linux boots, you probably want to put it in the `rc.local` file.
- **Continuing FTP:**
  To continue downloading an incomplete/interrupted file in FTP, use **reget <filename>** after logging in to the FTP server.
- **Finding Text in Files:**
  To search for a text string, the basic syntax is **grep "text string" /directory/you/want/to/start/searching/from**. If you don't want it to be case sensitive then use **grep -i "text string" /directory/to/start"**.
- **Exiting `vi`:**
  To exit from **vi**, the standard text editor, type [ESC], **:** and then **q**. Typing **:** will put you in command mode; **wq** will quit and write any changes to the file; [ESC] **i** will put you back in insert mode so that you can type text in.
- **Red Hat Oopsies:**
  Regularly check the Red Hat Errata pages for updates to your Red Hat distribution. The URL is http://www.redhat.com/support/docs/errata.html.
- **Fresh Meat:**
  Check for Linux software updates at FreshMeat. Or, join and subscribe to the mailing list to have the day's announcements mailed to you every night.
- **Filename Completion:**
  Tab filename completion allows you to type in portions of a filename or program, and then press [TAB], and it will complete the filename for you. If there's more than one file or program that starts with what you already typed in, it will beep, and then when you press [TAB] again it lists all the files that start with what you initially type. Play around with it a little bit to see what its limits are, how it can work for what you do, etc.
- **`ifconfig`:**
  You can check to see what your current IP addresses are using the **ifconfig** command.
- **HOWTO-reading HOWTO:**
  When looking through a Linux HOWTO, try to read only what you think you'll need. If you're still

confused, then try additional portions of the HOWTO that you think will be helpful. This is u
if you've figured out part of the problem and want to know how to do the rest.

- **How big is it, anyway?:**
  To find out how much space a directory and its contents take up, use the command **du -b --total** to give you the output in bytes. There's a lot more you can do with this command, just read the manpage using **man du**. Another good trick is to do **du -b | sort +0n -r > foo-du.txt**, then use **less** to look for really big files.
- **Development packages:**
  In case you decided to skip installing development packages during the installation of your distribution, and now can't compile source into binaries that you can use ("command not found" when you're following the guides on this page), install the following packages: autoconf, automake, make, gcc, egcs, glibc, glibc-devel, kernel-headers, libstdc++, XFree86-devel, and binutils. This will probably vary depending on your distribution, but they're basically what you'll need. (Based on experiments performed on Parker Mead)
- **Long Filenames:**
  Linux can handle long filenames--with spaces, too! However, it's not a good idea to use spaces in filenames, as it creates problems for many programs. To refer to them you'll have to use quotation marks. So if I have a file called `That report I was working on` then I'll have to edit it using **`pico "That report I was working on"`**.
- **^D:**
  Pressing ^D (Ctrl-D) on an empty line will close that shell, either with the **exit** or **logout** command, whichever is appropriate. When you've already typed something, and then you press ^D, you'll see a list of filenames that start with the thing you typed! (Sybren Stuvel)
- **How Do I...?:**
  When you know what you want to do but not what command you need, try typing **apropos *subject***. This will print out a list of all the man pages having to do with *subject*. (Paul Winkler)
- **Instant Replay:**
  Another nice bash feature: typing **!*string*** will execute the most recent command that began with *string*. (Paul Winkler)
- **^Z:**
  ^Z (Control-z) stops a job. After stopping a job, you can do two things with it - start it again in the foreground or start it again in background. To start the job in foreground again, use **fg n**; to start the job in background use **bg n**, where n is the job number. **jobs** will list all the jobs (along with numbers) you currently have running. Note that "all the jobs you currently have running" is only on the current tty - if you're logged in as **joe** on tty1 and tty2, and you stop a job on tty2, you can't do anything to it from tty1 (play with the **jobs**, **bg** and **fg** commands - you'll see what I mean). Also note that job numbers are not the same as process id (PID) numbers, so DON'T TRY "kill 2" WHEN 2 IS THE JOB NUMBER FOR ANYTHING. (Instead, try **kill %2**. Thanks to Mark Schreiber for this one.) If you have many jobs running background and make the simple mistake of killing a job number instead of a PID, you risk having some large nasty problems. Trust me on that one. Why? Because a lot of very important processes are loaded with very low PIDs, and it's a bad idea to be signalling them without a good reason. (jbm)
- **No more root Logins:**

TOC

To never have to log on as root again (having to log on as root is generally looked down upon bad practice), try two very useful utilities: **su** and **sudo**. "su" allows users to become the super if they know the superuser password. Simply type **su** at the prompt, and the program will ask fo the password. Enter it in, and poof - your $ is now a #. Type **exit** to get back to being a normal user. (Note: you can also use **su username**; "su" will prompt you for "username"'s password. If you are a superuser, you can just do **su username** and it won't prompt for a password. Use "exit" to get back to being your usual self). This is very useful for installing system-wide programs, or moving things around. (jbm)

- **Less is More:**
  Most Linux distributions come with a version of **less** that will automagically handle gzip'd and zip'd files. This means you can just do **`less my-big-gzipped-text-file.gz`** and it will act just like normal **less**! Also, **less** can read the directory structure of tar files - gzip'd ones, too. So to make sure that new package you got actually creates its own directory, do **`less foobar-1.0.tar.gz`** to see where all it puts things; it'll spit out a listing somewhat like **`ls -l`**. (jbm)
- **Screenshots:**
  To grab a screenshot in X, use **xv**. Open up **xv** and click "grab". Set the delay to 5 seconds and check "Hide XV windows". Click Grab and then click on the root window (your desktop) and leave the mouse cursor over the desktop (not over a window). The edge of your screen will blink and your computer will beep to tell you it's grabbed the screenshot. Move the **xv** window a little bit, and suddenly your whole screen will be filled with the screenshot image. Right-click on that to bring up the control window, and save the image. (jbm)

Got any tips of your own to add? Let me know and your name and a link to your e-mail address or website will be included.

Modifying Your Partitions and Help on LILO

TOC

Last updated: September 19, 1998

If you were really careful during installation (or can predict how much hard drive space you'll need later on), you might not need to read this, but if you suddenly decided or found out that you want to extend your partitions, shrink or extend your swap space, added a new hard drive, or something of that sort, I'll explain how to do those things for you right here.

First, type **`fdisk`** when you're **root** at the Linux prompt. You might remember this from installation, when you were setting up your partitions for Linux to use.

You probably also want to be able to boot any new partitions using LILO (if you're *using* LILO). First you have to edit `/etc/lilo.conf` so that LILO will know which partitions are there to boot. As of now I still haven't figured out everything about LILO, but it would help to take a look at your `lilo.conf` and figure it out to an extent; you can also read the lilo.conf manual page by typing `man 5 lilo.conf`. The first entry put down is the one LILO will boot if you just press 'enter' or leave it alone. On my system, I have LILO boot Linux on default and boot my DOS FAT partition when I type 'dos'. Here's what my `lilo.conf` file looks like:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz
        label=linux
        root=/dev/hda2
        read-only
other=/dev/hda1
        label=dos
        table=/dev/hda
```

After you edit `/etc/lilo.conf`, **don't immediately reboot**. Type **`lilo`** at the command prompt and see if it displays something like this (with no error message; if there is, then edit `lilo.conf` until you get it right):

```
Added linux *
Added dos
```

The asterisk (*) after `linux` shows that it boots Linux if I don't enter anything or just press Enter. If you have more than two bootable partitions, then it should show whatever partitions you enabled in `/etc/lilo.conf`. According to what I figured out, using the `image` label in `lilo.conf` is only for your Linux or Unix-based partitions with boot images, and the label `other` is used for other types of partitions. Just a guess, really.

Another common problem people have with LILO is that the computer reads the hard drive, starts to display "LILO", but then stops when it only displays "LI". A possible solution might be to add the line `linear` somewhere in /etc/lilo.conf; I heard it's a disk geometry problem. If you have another solution to this, I'd sure appreciate it if you shared it. So if that problem appears, you should boot a kernel using LOADLIN.EXE, which should be on your Linux distribution's CD-ROM, and a copy of the kernel (a file called vmlinuz).

I can't predict all the possible uses you might have, but I'll just list some basic pieces of information you might want to use to format, repartition, and make swap space.

- To format a new partition I've just made, I'd use **`mkfs -t ext2 /dev/hda4`**.
- The syntax for formatting a new ext2 partition is **`mkfs -t ext2 <full path to device>`**.
- To make new swap space, I'd use **`mkswap /dev/hda5`**.
- The syntax for creating new swap space is **`mkswap <full path to device> [number of blocks]`**.

To make Linux mount a filesystem that you want automatically, you should edit `/etc/fstab`. It is read each time you boot up Linux. Here's what mine looks like:

```
/dev/hda2                /                        ext2    defaults        1 1
/dev/hda4                /usr/local/src           ext2    defaults        1 1
/dev/hda5                swap                     swap    defaults        0 0
/dev/fd0                 /mnt/floppy              ext2    noauto          0 0
none                     /proc                    proc    defaults        0 0
```

The configuration I have is for Linux to mount my main partition (/dev/hda2) on `/`, mount my second ext2 Linux partition (/dev/hda4) to `/usr/local/src`, enable my swap space (/dev/hda5), and to mount the /proc filesystem to `/proc` (/proc contains information about your system and the hardware; it's not part of the actual harddisk). When it says `defaults` I think it automatically mounts it to where you specified. The entries are separated after each line.

Suggestions

Stefan had a suggestion about the problem with LILO's tendency to sometimes only display the "LI": "*It's a little bit annoying if you've spent 2h or more on installing your Linux and then discovers that the LILO doesn't work. I can't even count the hours I've spent to solve this problem. Then I finally discovered another solution. To be sure that this problem never occurs, I do a `fdisk /mbr` from a windows/dos boot diskette before installing my linux. In this way you rewrite the boot-table on your boot harddisk and the geometry-problem, or whatever it is, never occurs.*".

Using RPM

Created on June 27, 1998
Last updated on September 19, 1998
Development stage: Alpha

RPM is the Red Hat Package Manager, developed by Red Hat Software. It's a packaging format used in several Linux distributions, such as Red Hat Linux, Caldera OpenLinux, and TurboLinux. The RPM homepage is at http://www.rpm.org.

Since RPM is a package manager, you can install, remove, build, rebuild, and view your packages. In this document I'll cover everything except building RPM software packages, which you *probably* won't need to do right now.

Installing RPM Packages

Your Linux distribution, if it uses RPM, already installed RPM packages. You can install additional RPMs from your CD-ROM or download them from an FTP server. RPM packages end in ".rpm".

The most basic syntax for installing an RPM package is **rpm -i package-1.2.3-4.i386.rpm**. RPM file names usually consist of the package name ("package"), the version ("1.2.3"), the build number or patch level by Red Hat ("4"), the architecture it was built to run on, ("i386"), and the "rpm" extension.

Instead of using just "-i" in the RPM command line to install RPMs, you probably want to use "-ivh" so you can see what's going on.

Forcing

Sometimes you have to force a package to install, because maybe you had it installed, but some of its files got deleted and you want to have the complete package again. Simply add "--force" to your RPM command line while installing a package. If you don't, you'll probably get complaints that the package is already installed.

If you're forcing a package to install, your command line will then look something like **rpm -i --force package-1.2.3-4.i386.rpm**. Forcing a package to install *usually* won't hurt anything, but don't use the "--force" option all the time.

Upgrading

Often, new versions of software comes out, and new RPM packages are built for them. That's when the "-U" flag of RPM becomes useful. That's a capital "U", not a lowercase "u".

If there's an update to the previous package we had, it would be noted in the filename. So an upgrade to "package-1.2.3-4.i386.rpm" would be something like "package-2.0.3-1.i386.rpm". To install it, the command line looks like **rpm -U package-2.0.3-1.i386.rpm**.

Like installing, you can also see more of what's going on. Instead of just "-U" you can use "-Uvh".

Removing RPM Packages

If you don't want to have a package around anymore, you can remove it. Use the "-e" flag for **rpm**. Let's say we want to remove the package installed by package-1.2.3-4.i386.rpm. Don't type the entire filename. Just type something like **rpm -e package-1.2.3-4**. Even **rpm -e package-1.2.3** will work, if there aren't more RPM packages of **package-1.2.3** lying around, which may have been installed with the "--force" flag.

Ignoring Dependencies

Many RPM packages depend on other RPM packages to run. A lot of cases are programs that need libraries to run, or programs that need other programs to run. It's generally a bad idea to remove packages without paying attention to what depends on those packages, which RPM warns you about on default, unless you use the "--nodeps" flag.

RPM will also complain about dependencies if you try to upgrade a bunch of packages. Upgrading also involves removing the old package, so you can use the "--nodeps" flag with upgrading as well.

Viewing All Packages

To get a simple printout of all the packages you have installed through RPM, use the command **rpm -qa**. It's as simple as that.

Checking for a Package

To see what packages you have installed, use their names to find them. For example, if we want to find all RPM packages installed that have "pack" somewhere in their names, we type **rpm -qa | grep pack**.

What we're doing there is printing out all the packages but sending it to the **grep** program so that grep can search for the text string we specified ("pack").

Viewing Package Information

To find out information about our package, type **rpm -qi package-1.2.3**. The "-q" we've been using so far stands for Query, and the "i" stands for Information.

Here's a sample output from a package called "perl":

```
Name        : perl                        Distribution: Manhattan
Version     : 5.004                             Vendor: Red Hat Software
Release     : 6                             Build Date: Fri May  8
12:08:09 1998Install date: Tue Jun 16 10:36:49 1998   Build Host:
porky.redhat.com
Group       : Utilities/Text                Source RPM:
perl-5.004-6.src.rpm
Size        : 11696746
Packager    : Red Hat Software
Summary     : Practical Extraction and Report Language
Description :
Perl is an interpreted language optimized for scanning arbitrary text
```

```
files, extracting information from those text files, and printing rep
based on that information.  It's also a good language for many system
management tasks.  The language is intended to be practical (easy to use,
efficient, complete) rather than beautiful (tiny, elegant, minimal).
```

As you can see, there's a lot of good, detailed information you can find by using **rpm -qi**.

Rebuilding an RPM from a Source RPM

There are few situations that the typical user will actually *need* to rebuild RPM packages, but I had to do it when I upgraded my JPEG library. The programs that used the JPEG library wouldn't load JPEGs just because I had a new version and they were expecting an older version.

I had to take the source RPM packages of those programs and use them to rebuild into an RPM package. That way they would expect the correct version of the JPEG library I had installed. One of those packages was the GIMP. I took the source RPM (gimp-1.0.0-1.src.rpm) and typed **rpm --rebuild gimp-1.0.0-1.src.rpm**. That recompiled it, re-linked it with the new JPEG library so it would know what version to expect, and created three RPM packages. That's what the .src.rpm files are for: they're source RPMs for people who need to rebuild the package for some reason.

Related Pages

1. Installing Software Packages

Questions, comments, and suggestions should be sent to jtg@computers.iwz.com through e-mail, the guestbook, or help.html.

Installing Software Packages

Created on September 30, 1997
Last updated on September 19, 1998
Development stage: Beta

If you've ever wondered how everyone else knows how to install software packages and you don't, read on. Here, I'll try to explain what you should look for when you're downloading all sorts of programs from all over, as well as some examples.

This covers installing tarred and gzipped archives, not any distribution-specific software packages. This document should work will all Linux distributions, not just the ones that use RPM. If you're using Red Hat Linux or any other distribution that uses RPM, you can read up on using RPM.

Summary

1. Download the software, which is usually in the form *.tar.gz.
2. Extract the archive. Using **tar -zxvf archive.tar.gz** will usually work.
3. Do the standard **./configure ; make ; make install** to configure, compile, and install the software.
4. If you can't find instructions, try looking for a file called INSTALL or README in the directory.

Downloading

The first step in installing a new program is download the archive. These archives usually come in *.tar.gz files (or *.tgz; same type of compression, different name) which are stored on FTP servers such as **sunsite.unc.edu**. On servers that aren't fully dedicated to serving out Linux software, the software you're looking for is usually stored in the `/pub/linux` directory. A lot of servers also have Unix software in `/pub/unix`, which means they're applications that can run on all (or most) flavors of Unix, of which Linux is also included. Basically, just use your common sense when browsing or searching for programs on these FTP servers.

There are a lot of Linux software archives out there, but you probably already have a single program in mind that you want to install. Anyway, here are some FTP sites that contain tons of software you can use with Linux:

- http://sunsite.unc.edu/pub/Linux
- ftp://tsx-11.mit.edu/pub/linux
- ftp://ftp.cc.gatech.edu/pub/linux

I know I missed at least a couple of good, popular ones. If you're bored, you can try e-mailing it to me.

Basic Stuff for the FTP Program

Linux comes with a default **ftp** program, which probably came with your distribution. With Red Hat it's NetKit, with Slackware it's somewhere in disk set N, and for other distributions I have no idea. Let me know if you can help me out here so I won't sound like an idiot. (Too late, huh?).

Anyway, the **ftp** program will get you into an FTP server. The way you use it is **ftp ftp.host.somewhere.com**, and you'll be connected to that server. For example, if I were to connect to sunsite.unc.edu, I would type **`ftp sunsite.unc.edu`** at the main Linux prompt. If there's an error, like the server being down, just type **open host.elsewhere.com** when you're at the "`ftp>`" command line.

Now, on to more important stuff. Like how to get your files. Assuming you've changed to the directory you want using the **cd** command (it works just like MS-DOS), key in **ls**. If it's a large directory, scroll up using the Shift-PageUp keystroke, use your mouse to highlight the filename, type in "**get**, hit the spacebar, and then click the right mouse button. Finally, press [Enter] and you should see some FTP language being spit out on your screen.

Sometimes you may have to get multiple files. Say you want to get the GIMP, and you've found the directory at `ftp.gimp.org`. You want to get everything that starts with "gimp". First type in, at the FTP command prompt, **prompt**. That's right, just type **prompt**. That will get the FTP client to stop asking you whether you want to get each file when you do the next step. (You can also turn it back on by typing it in again.) Now, use **mget** followed by any files that you want to get. I suggest using the "*" character to save you some typing. If I wanted to get all the GIMP file, for example, I'd type **mget gimp*tar.gz** when in the directory for the GIMP archives.

After you've downloaded the file, it's in the directory that you started the **ftp** program from. Most software packages can be extracted anywhere, but just move it to `/usr/src/` as a good habit. Use **`mv filename.tar.gz /usr/src/`** so that it will be in the directory `/usr/src` now.

Extracting the Package

Once you've got your package downloaded, it's time to extract it. The method of compression that was used to store it might be different; sometimes people might only use .tar to put all the contents of a directory in one file, and sometimes they may use GNU zip (gzip), with a file extension of *.gz, which I think is convenient for compressing single files. Most of the time, though, software packages will come in a *.tar.gz or *.tgz format and will usually extract to one or more directories with the files inside them.

Anyway, now that you've got the file, you should know how to extract it (duh!). If it's a *.tar.gz file or a *.tgz file, you would use **tar -zxvf [filename]**. For example, if you have a file you downloaded called `gimp-0.99.17.tar.gz` and you downloaded it into `/usr/local/src/`, you'd do a `cd /usr/local/src/` first and then extract the archive using `tar -zxvf gimp-0.99.15.tar.gz`.

The **tar -zxvf** command line will extract a file, showing you the files and directories which are being extracted. Most Linux/Unix software packages in the *.tar.gz format create a directory after extraction, but watch out for the ones that don't.

Compiling and Installing

Most of the time, you'll have to have **`make`** installed. This is what helps compile large programs and most Unix software packages. However, before running this, you should help the software get to know your system. There's usually a file called **configure** in the directory that was extracted. Run it by typing

./configure when you're in the directory.

That will print out a bunch of stuff that mentions stuff about checking whether gcc works, if automake is installed, if some .h file can be used, that sort of stuff.

After running **./configure** you should have everything ready to compile. You shouldn't need to worry about optimizing anything right now; you just want that package to run well, which the authors of the software as well as the configure script should have already helped set up for. Usually you can just type **make**, which will compile the source code for the program. Compiling it is basically taking those text files that contain instructions, the source, and converting it into binary files that will create your program.

Sometimes you'll have to type something else after **make** so that it knows what to do. I had to do this when installing a mail server. Instead of just `make`, I had to type `make lnx` so that it would know that I was running Linux. There's `make depend`, which, I guess, configures what depends on what.

Usually, after the `make`, you'll only need to do a `make install`. Since `make` compiles the binary executables (the program), so `make install` installs them to the proper place.

Binaries

Most software packages under Linux are available as binary packages, as well. That means you don't have to go to the trouble (or fun) of compiling the package yourself.

You should always read the README or INSTALL file, but as a general rule, you can safely use the `./configure ; make ; make install` procedure. It probably won't compile anything since binaries are already present, though.

Keeping Logs

If you are the root user on the system, you should keep a log of where the files installed to, in case you ever want to remove them. Create a directory in your home directory (/root/) called `install_logs` using `mkdir /root/install_logs`. That will create the directory `/root/install_logs` so you can have text files telling you exactly where programs installed their files to.

Where are these text files going to come from, though? Well, assuming that you use a standard `make install`, just type in `make install > /root/install_logs/program-1.0`, where program-1.0 is the name and version of your software. That basically installs the files where they're supposed to be, while at the same time writing to a file exactly what would have been echoed out to the screen.

Even if you already did `make install`, you can still do a `make install > /root/install_logs/program-1.0`. That will reinstall the binaries, but that usually won't matter, since they're the same files, just being written to the same place again. I guess there are some instances where you wouldn't want to do this, but generally it's safe to do many `make install` commands.

This isn't absolutely necessary, but it will help you keep track of where your software files are, in case you want to get rid of them.

TOC

Cleaning Up

Usually when you're done and you know that the program is working, you can type **`rm -rf directory`**, where "directory" is the directory created when you extracted a file. You won't need that anymore, because it's just source code, but if you want to look at the source code and modify the program, you should keep it around.

Tips

When you can't seem to figure out what to do, just try reading a file called INSTALL in the directory that the tarred and gzipped file extracted. If there is no INSTALL file, check for a file called README.

Related Pages

1. Using RPM

Questions, comments, suggestions, contributions, any feedback at all? I sure would like you hear what you have to say about my work. My e-mail address is jtg@computers.iwz.com.

Configuring and Troubleshooting X

TOC

Last modified: July 16, 1999
Development stage: Beta

This section is guaranteed to be long, so if you want to read the whole thing, it's your choice. This includes just some basic things for Red Hat 5.2, based on XFree86 version 3.3.3.1. This information should also work for other Linux distributions.

I'm assuming you already have XFree86 installed, along with its libraries. If not, install them. In Red Hat, TurboLinux, Mandrake, SuSE, and Stampede, they come in RPM files that start with "XFree86". If you want to install the files from the tarred and gzipped form, read Upgrading and Installing X.

What's an X Server?

An X server is sort of like a video driver (that's the best I can explain it...) that the system uses to run the graphical user interface. There are too many changing X servers to list here, but there are SVGA, VGA, and Monochrome, which are the most basic. There are also commercial X servers which I know virtually nothing about.

The X server you choose to use depends on the type of chip your video card uses to process graphics. It also determines how many colors will be displayed in X (the amount of video memory you have also determines that). When you key in **startx** and the error under "A Common Error Message" appears, you should make sure that you installed the server that you configured for.

The X server I used was the S3V. That stands for S3 ViRGE, since my Diamond Stealth 3D 2000 used that chip. When I got a Creative Labs Graphics Blaster Riva TNT (PCI), I switched to the SVGA server because the SVGA server was the one that had support for the Riva TNT chipset.

Configuring X

TOC

If there is any kind of error given when you execute the **startx** script, you probably should run **Xconfigurator** or **xf86config**. These will configure the keyboard, the mouse, the video card, the monitor, and some other stuff so that X will run properly. The settings are stored in a file called XF86Config. On my system, it's in `/etc/X11/XF86Config`. You can edit that with a text editor such as pico, vi, joe, jed, or emacs.

A lot of people have trouble configuring X, especially beginners and long-time text mode users. The best thing to do when there are no manuals available is to guess. It might take a few hundred tries, but using defaults will usually get you by. For example, I didn't know which RAMDAC and Clockchip setting to use when eventually I read the little comments/tips on the screen and decided to try *not* using a RAMDAC or Clockchip setting for my card. It turns out that I wasn't supposed to use those settings in the first place.

It will help you *greatly* if you know your hardware settings or have your hardware manuals handy. If not, experiment.

A Common Error Message

The following error message is very common among new users, and often very confusing as well:

```
_X11TransSocketUNIXConnect: Can't connect: errno = 111
giving up.
xinit:  Connection refused (errno 111):  unable to connect to X server
xinit:  No such process (errno 3):  Server error.
```

Solution One

One solution to this is to install the server you chose for your video card's chipset (the type and brand of chip it uses to process video information). In Red Hat, you install the RPM package that describes which server it is by the first few characters in its filename. The common X servers are as following, with their RPM package filenames in parentheses:

- AGX (XFree86-AGX-3.3.2-8.i386.rpm)
- I128 (XFree86-I128-3.3.2-8.i386.rpm)
- Mach32 (XFree86-Mach32-3.3.2-8.i386.rpm)
- Mach64 (XFree86-Mach64-3.3.2-8.i386.rpm)
- Mach8 (XFree86-Mach8-3.3.2-8.i386.rpm)
- Mono (XFree86-Mono-3.3.2-8.i386.rpm) - Black and white display
- P9000 (XFree86-P9000-3.3.2-8.i386.rpm)
- S3 (XFree86-S3-3.3.2-8.i386.rpm)
- S3V/S3 ViRGE (XFree86-S3V-3.3.2-8.i386.rpm)
- SVGA (XFree86-SVGA-3.3.2-8.i386.rpm) - Works with most cards, 256 colors

- VGA (XFree86-VGA16-3.3.2-8.i386.rpm) - works with most cards, 16 colors
- W32 (XFree86-W32-3.3.2-8.i386.rpm)

To avoid as much trouble as possible, install these RPM packages if you're using Red Hat.

1. XFree86-3.3.2-8.i386.rpm - XFree86
2. XFree86-devel-3.3.2-8.i386.rpm - Lets you compile stuff
3. XFree86-libs-3.3.2-8.i386.rpm - X libraries
4. XFree86-SVGA-3.3.2-8.i386.rpm - 256 color X server
5. XFree86-100dpi-fonts-3.3.2-8.i386.rpm - Fonts
6. XFree86-75dpi-fonts-3.3.2-8.i386.rpm - Fonts
7. Xconfigurator-3.57-2.i386.rpm - Helps with configuring X

For more information on installing and upgrading X, you might want to read Installing and Upgrading X.

If you want to run at more than 256 colors use another X server that is made especially for the type of chip that's on your video card. The special X servers are made for the newer PCI video cards, but ISA graphics accelerator cards could possibly also use the special X servers.

Solution Two

Another possible cause for the "_X11TransSocketUNIXConnect" error is that the mode that is chosen (or the mode that you set it to) is not valid. This can be because of monitor settings that are too high or too low for your video card. I discovered this when I set my monitor's hsync (horizontal sync) and vsync (vertical sync) ranges to settings that were too high, although I had the proper X server for my card. Then I tried them really low, and then I was allowed into X--but the resolution was very low. I finally decided on something in the middle and that seemed to work.

Those monitor settings (hsync and vsync ranges I think) will also have an effect on what resolution you can run at. Again, setting those too low will get X to make you run at a very low resolution, so that everything looks huge and blocky. Try entering your own ranges if your **monitor manual** has the settings. You can also select what type of monitor you have from XFree86's list of monitors. My MAG Innovision DX15T was not listed but something close was, the DX1595, and it had the exact hsync and vsync ranges that my monitor had (when I finally found it in the manual).

Otherwise, you can try the lowest setting and work your way up, try the highest setting and work your way down, or choose something in the middle and see if your monitor supports it. Be careful: if the setting you choose is too high, you could seriously damage your monitor.

The Infamous Gray Screen

Many people encounter the problem of starting X (startx), seeing an X-shaped mouse-pointer-thingy, a gray screen, and being taken back to text mode with no window manager loading and no error messages. If it doesn't take you back to text mode, try the Ctrl-Alt-[Backspace] combination.

The way I found I could solve this was to create a **.xinitrc** in my home directory and putting the line `exec afterstep` in it. That seemed to work fine and it loaded the AfterStep window manager. As you can probably figure out, you'll have to replace **afterstep** with whatever window manager you're using. If I wanted to use fvwm, I'd put in **exec fvwm** instead.

Now suppose you want to run multiple window managers; of course, you can run only one at a time in each X session, but you don't want to type everything in again. The solution is to type in the startup lines of each window manager then comment out all the ones that you don't want to use, leaving one line uncommented so that it'll run. Here's how my .xinitrc file looks like:

```
# exec afterstep
# exec fvwm95-2
if [ -f $HOME/GNUstep/Library/WindowMaker/.workspace_state ]; then
        sh $HOME/GNUstep/Library/WindowMaker/.workspace_state &
fi
exec wmaker
```

That is set to run WindowMaker when I start up, and the commented lines, the ones with the pound signs (#) at the beginning of the lines, are ignored. If you leave everything commented out, it'll give you the gray screen with the X-shaped mouse, so if that happens, you should know what to look for. :)

Alias to Start X

On my system, I start X by typing **x** at the prompt, and it starts up in 24 bits per pixel mode when I do that. I can type **startx -- -bpp 24**, but that takes time to type up. I can do this because I added a line to `/etc/bashrc` that goes like this:

```
alias x='startx -- -bpp 24'
```

Of course, if you prefer to run at some other video mode other than 24 bits per pixel (16.7 million colors, I think) then you would replace "24" with 15, 16, 32, or whatever you may wish to run as the default.

With Slackware, you might have to edit `/etc/profile` instead of `/etc/bashrc` if it doesn't look in `/etc/bashrc` when a user logs in. I don't think aliases are *supposed* to go in `/etc/profile`, but it works.

Turning Off Virtual Desktop

XFree86 usually annoys new users with something called virtual desktop. This can be described as a screen bigger than the one displayed, so that when a user moves the mouse to the edge of the screen, it scrolls down instead of staying there.

Only after using Linux for a year did I discover how to turn that off. Yeah, it gives me more desktop space, but as new users frequently complain, it's annoying. :)

Anyway, the way I turned it off was by finally finding the right place to edit in the `/etc/X11/XF86Config` file. The following is the area that you should edit:

```
Section "Screen"
```

```
    Driver      "accel"
    Device      "Diamond Stealth 3D 2000"
    Monitor     "My Monitor"
    Subsection "Display"
        Depth       8
        Modes       "800x600"
        ViewPort    0 0
        Virtual     800 600
    EndSubsection
    Subsection "Display"
        Depth       16
        Modes       "800x600"
        ViewPort    0 0
        Virtual     800 600
    EndSubsection
    Subsection "Display"
        Depth       24
        Modes       "800x600" "640x480"
        ViewPort    0 0
        Virtual     800 600
    EndSubsection
    Subsection "Display"
        Depth       32
        Modes       "800x600" "640x480"
        ViewPort    0 0
        Virtual     800 600
    EndSubsection
EndSection
```

This can be found towards the end of the file. At first I heard of people editing this file and successfully getting the results that they wanted; I tried it, but I edited the wrong part of the file and that didn't make a difference at all.

Anyway, the way I got rid of my virtual desktop was to set it to be the same resolution as the resolution I was running at. In my case that was 800 by 600 (all modes from 8 bpp to 32 bpp). You can also try commenting out the lines starting with "Virtual" by putting a pound sign (#) at the beginning of the line.

Additional Fonts

Most people might not be satisfied with the fonts included with the standard distribution of X, so if you're one of them, read Zach Beane's tutorial on adding fonts to X.

The fonts themselves are available from ftp.cdrom.com/pub/os2/fonts. They're scalable Adobe Type1 fonts, which means that they won't get jagged when enlarged to large font sizes.

You can also get a couple of packages containing fonts from ftp.gimp.org/pub/gimp/fonts. They're the .tar.gz files.

With Zach Beane's tutorial, it's also possible to add other types of fonts, such as 75dpi and 100dpi.

Mouse Trouble

There are a lot of possible problems people might encounter with mice. Just make sure you have it pointing to the correct device (usually `/dev/ttyS0`, which is COM1 under DOS and Windows) when the X configuration utility that you use (xf86config, Xconfigurator, etc.) asks you where your mouse is. If you have a PS/2 mouse, use `/dev/psaux`.

If you want to edit `/etc/X11/XF86Config`, look under the section "Pointer". Here's what it looks like:

```
Section "Pointer"
    Protocol    "Microsoft"
    Device      "/dev/ttyS0"
```

If it's on another device, such as COM2 under DOS, it would be `/dev/ttyS1`. COM3 under DOS/Windows would then be `/dev/ttyS2`, and so on.

andres salomon wrote in to say that if your PS/2 mouse doesn't work in X but works in normal text mode, you should try killing the GPM mouse services (**`killall gpm`** as root should do the trick). He also mentioned a workaround, which he didn't successfully use. It's adding `/dev/gpmdata` to /etc/X11/XF86Config, supposedly, and starting GPM with the "-R" flag.

Fotis Karpenissiotis wrote, "*I had several problems using a Microsoft Mouse (Serial Mouse 2.0A) under XFree86 3.3.2. (Under 3.3.1 it worked fine). The solution I found was to start gpm using:*

```
gpm -t ms -R
```

and the following lines in XF86Config:

```
Section "Pointer"
       Protocol    "MouseSystems"
       Device      "/dev/gpmdata"
       Emulate3Buttons
```

That's about all the information I can give about mouse trouble and their solutions right now, but if you've got a different problem and a different solution then please let me know.

Newer Cards

Dragoon and Larry Elmore wrote in with tips reminding those of us with newer cards to head over to http://www.suse.de/XSuSE/XSuSE_E.html. SuSE creates a Linux distribution, but they also create X servers for video cards/chipsets that XFree86 has not included yet. The XFree86 Project only releases new X servers every six months, so in that time span in between, SuSE releases stuff. It's worth checking out if you're not having much luck with that new video card and X.

TOC

If you have anything from NVidia, find their Open Source XFree86 patches/modifications on their website at http://www.nvidia.com/. This includes support for cards with Riva TNT and Riva TNT2

If you can't find an X server for your card using any of the above methods, just about the last thing you can try is going to your card or chipset maker's website. Unfortunately, some video card companies do not like to cooperate with the XFree86 project.

Send any and all feeback to jtg@computers.iwz.com. Feedback is welcome and highly encouraged. You can also make use of the help form or guestbook, which is really more like a feedback form.

Installing and Upgrading X

TOC

Last updated: June 8, 1999
Development stage: Beta

When Linux users refer to X, they mean the graphical interface for Unix systems. More information about XFree86 (a free implementation of X) is at http://www.xfree86.org. You need to be in X in order to run many graphical applications. It is what you would use to create the equivalent of a Windows desktop under Linux.

The subject of writing here is XFree86 3.3.3.1, the "new" version at the time of the last major update. There are various reasons one would want to upgrade their version of X. The latest release might have better hardware support. Alternatively, one might want the updated X applications and libraries. Lastly, it could be that some people do not feel comfortable when they are not using the latest version of a software package. If you do not have X installed, this is also the page to read.

The first step to upgrade your version of X is to back up your old one, just in case the procedure on this page does not work. (If you do not have enough disk space nor want to go through the trouble, there is really no need to worry about it. It is just a good habit to develop). You can do this by changing your working directory to `/usr/X11R6` which should work in most systems. Your system might be different, but that is the way it is laid out on my machine.

If you do not have X installed yet, just create an `X11R6` directory as a subdirectory in `/usr/`.

This is how your X directory (`/usr/X11R6/` in most systems) should look like if you already have X installed:

```
bin
doc
include
lib
man
```

To back it up, I suggest you **tar** and **gzip** them into one file and keep it in there. On my system, I named it `x11r6.old.tar.gz` to make it easy to tell what it contains later on. I did that by typing this in `/usr/X11R6`:

```
# tar -czvf x11r6.old.tar.gz bin doc include lib man
```

Now that you have backed up the old X, go to one of the following FTP sites to download the various XFree86 .tgz files. I will specify the files you need to download later in this document.

North America

ftp://ftp.XFree86.org/pub/XFree86
ftp://ftp2.XFree86.org/pub/XFree86

TOC

ftp://ftp.infomagic.com/pub/mirrors/XFree86-current
ftp://ftp.rge.com/pub/X/XFree86
ftp://ftp.varesearch.com/pub/mirrors/xfree86
ftp://ftp.cs.umn.edu/pub/XFree86

Europe

ftp://fvkma.tu-graz.ac.at/pub/XFree86
ftp://gd.tuwien.ac.at/hci/X11/XFree86
ftp://ftp.gwdg.de/pub/xfree86/XFree86
ftp://ftp.cs.tu-berlin.de/pub/X/XFree86
ftp://ftp.uni-erlangen.de/pub/Linux/MIRROR.xfree86
ftp://ftp.uni-stuttgart.de/pub/X11/Xfree86
ftp://ftp.funet.fi/pub/X11/XFree86
ftp://ftp.ibp.fr/pub/X11/XFree86
ftp://ftp.unina.it/pub/XFree86
ftp://ftp.pvv.unit.no/pub/XFree86
ftp://sunsite.doc.ic.ac.uk/packages/XFree86

Asia and Australia

ftp://x.physics.usyd.edu.au/pub/XFree86
ftp://ftp.netlab.is.tsukuba.ac.jp/pub/XFree86
ftp://ftp.iij.ad.jp/pub/X/XFree86/XFree86

You should read the file RELNOTES in the XFree86 distribution to see which files you should get from the distribution, but just to spare you some scrolling and searching, here is what you should download:

- X332VG16.tgz
- X332prog.tgz
- X332bin.tgz
- X332doc.tgz
- X332fnts.tgz
- X332lib.tgz
- X332man.tgz
- X332set.tgz
- preinst.sh
- postinst.sh

You should also download the specific X server for your card. For example, I use an S3 ViRGE on my Diamond Stealth 3D 2000, so I downloaded `X33S3V.tgz`. So if you are using an Imagine 128 card, then you should download the X server for that (I128). Some cards are not supported totally, but most will work with the SVGA server (256-colors). My Creative Labs Graphics Blaster Riva TNT uses th NVidia Riva TNT chipset, and what worked with that was the SVGA server. If it seems that your ca not supported, try going over to SuSE's website. They have X servers that are not integrated into the official XFree86 distribution yet, and that often means that they have X servers for the new cards. The XFree86 Project only releases a new version about every six months.

TOC

All that should be put into `/usr/X11R6/` or whatever your X directory is. You can run the .sh files using `sh file.sh` or make sure the .sh files are set to be executable. If not, do `chmod +x preinst.sh postinst.sh`. After that, run preinst.sh by doing `./preinst.sh`; answering "yes" to the defaults should be okay. Now, all you do is extract all the .tgz files and then run `postinst.sh`. That should put all the files where they are supposed to be. If already you are an avid X user, you can just resume your normal routine with your system with X; you will not need to read any further.

If you are installing X for the first time, you should set the X binary directory to be part of your path. You can do this by editing `/etc/profile` with your preferred text editor and adding "/usr/X11R6/bin" to the line with PATH. That will set the environment variable, $PATH, to all the directories specified for it to look for the name of the program that you typed in.

If you are a first-time X user, the way to start X is by using the `startx` command. On my system, that starts up X in 8 bpp (eight bits per pixel, which is 256 colors) mode on default. The way to start up X in higher color modes if that happens to you is **startx -- -bpp 16**; with that command line, X will start up in 16 bits per pixel mode, which is approximately 65,536 colors. Other valid bpp modes you can use are usually 1, 4, 15, 24, and 32. The number of colors in relation to bits per pixel is 2 to the power of the number of bits per pixel.

If you have trouble with X or some error message appears, try Configuring and Troubleshooting X.

You will probably also need a window manager if your system has never had X on it before. There are a lot of window managers out there, so just look for one and choose at http://www.plig.org/xwinman. The one I recommend is WindowMaker, which I have an installation guide for as well.

Questions, comments, corrections, suggestions? Send them to me at jtg@computers.iwz.com.

Red Hat Linux Installation from the CD-ROM

Last updated: July 20, 1999
Development stage: Beta

This installation guide will help you get through the Red Hat Linux CD-ROM installation; the CD-ROMs I order for Linux are from CheapBytes, for about 2 dollars each (plus about 5 dollars for shipping and handling); what's on the CD-ROM is basically what you can download from Red Hat FTP mirrors. You can also get those kinds of CD-ROMs from Linux Central. It seemed a little faster when Linux Central sent me CD-ROMs. If you want to go with the official Red Hat set, this guide still works.

This installation guide is based on my experience in installing Red Hat Linux from the CD-ROM. I'm assuming that you have a PC with a 386 or higher processor, and that you are installing this from the CD-ROM. Use common sense and adjust whatever settings you feel may be necessary; I just tell the reader to do stuff so the installation will go faster and be less confusing.

Getting Started

Take a few deep breaths, and brace yourself. The first time around, this could be very daunting. But the end result--your reward--will be worth it.

Create a Bootdisk

Go to the DOS prompt and insert a 1.44 megabyte floppy into your a: drive. From the CD-ROM prompt (drive D I'll assume... you know what drive it is), go to the directory where the DOS Utilities are (\DOSUTILS). Type `rawrite`. Make sure you formatted the disk for DOS before this. The source file should be a file called `boot.img` in the `\images` directory (so you would type in "\images\boot.img" as the target) and the target should be your a: drive (type "a:"). After you do this, set the disk aside to use l

If you have the official Red Hat Linux distribution, the bootdisk should already have been made for you. This is how to create a bootdisk in case those disks are damaged. If you have the CheapBytes version, the top would have helped you.

You don't have to use a bootdisk to get into the installation program if you're using DOS without Windows running. You can just type `autoboot` from the \DOSUTILS\ directory.

Backup

If you want to retain your old DOS/Windows system data, create a DOS system disk. Take another floppy that is new or that you don't need, and at the DOS prompt, type `format a: /s`. After you're done formatting, look for the files `FDISK.EXE` and `FORMAT.COM` in a directory with all the DOS programs. Copy those files to the DOS system disk you just created.

Make sure you backed up all the programs that you want to keep. **Defragment your hard drive in Windows or DOS so you can split it up (aka resize).**

Installing

Take out the DOS disk and then put in the Linux disk that you prepared before. Reboot again, and make sure the CD is in the drive. It will display a message: `LILO boot:` . Leave it alone (or type 'linux'), and it will start up the installation program.

Partitioning

When you get to the point where it asks you about the partitions and swap space for Linux, select *Edit*. You might be asked if you want to use Disk Druid or fdisk. I prefer **fdisk** so that is what I will explain. You can use the information about **fdisk** to use Disk Druid as well.

Once you're in the fdisk program, print out your partition information (p). Looking at the index of commands also helps greatly (m). If you want to resize your Windows/DOS partition to make room for Linux, delete (d) it first and then re-create it with a smaller size (taking up less cylinders). If you want to wipe it out altogether, you can just delete the whole thing and give Linux more space. Remember to leave room for the swap space. Remember to tag ("t") your DOS partition (number 6).

For the swap space, create an extended partition out of the room left over on the hard drive after deciding how much to give to Linux and how much to give to Windows. After that, create a logical partition (partition number 5) that takes up the same cylinders as the extended partition. Then tag (t) it as Linux swap (number 82).

After creating all the partitions you should print out your partition table. See if it looks okay. If not, go back and resize the partitions or tag the partitions as what they're supposed to be.

When I say "resizing" in reference to using **fdisk**, I mean deleting your old DOS/Windows partition and re-creating it in a smaller size. This is necessary, of course, if you have no room left on your hard drive to make a Linux partition. **It is also important to defragment your Windows partition before you resize it.** That is necessary in order to move data to the beginning of a partition so it doesn't get lost when the partition is resized.

TOC

Partition Size

You can set aside a Linux partition as big as you want; if you have the room, give it a gigabyte (a comfortable installation will work with 200 megs but extra space is recommended in case you want to download a bunch of stuff). Leave the remaining partitions for DOS (which you've already created) and the Linux swap space (which should be at least twice the amount of RAM that you have). If Linux is going to be your main operating system, consider giving it more generous portions (4-5 gigs perhaps, although I don't know what I would do with that much space).

Next make an extended partition, and then make another partition *within* the extended partition, called a **logical partition**. The logical partition should be partition number 5 or above, no matter whether you already have 4 partitions made before it or not. The number of cylinders is proportional to how much space you have on your hard drive, so if you want to give half of the drive to Linux, then use half the number of total cylinders when making partitions. You'll probably want to use all the cylinders left on the hard drive when making these last two partitions.

Tag Partitions

After this, tag the partitions; the primary partition should be tagged as "Linux native". The other partitions should be tagged as Linux swap space. Try to stay organized by tagging the partitions as soon as you make them (except for the extended; you **have** to create a logical partition within the extended partition or else the partitioning program will give you an error about having to delete it); otherwise, just print the partition table by typing 'p' from the main Linux partitioning program (fdisk), which will print out the information about the partitions and their type on your hard drive. If all this is done, then type `w` to write to the partition table and exit.

You will find yourself at the screen you were last in, so press enter on "Done" or "Okay" since you already edited the partition tables for Linux. If you have an older, slower computer, it might take Linux a while to "initialize the swap space" (which it is really just "formatting" it, really). Just sit back and wait, and/or do something else while it does this.

Splitting Your Hard Drive

In case you still want to keep your old DOS/Windows partition, you must resize it. The tool I prefer to use is the fdisk utility in Red Hat installation (though DiskDruid is available for you). I go about it by "deleting" the DOS/Windows partition in Linux fdisk (during Red Hat installation). (Note that it does not really get deleted right at the moment when you choose to delete it.) Next, I re-create it, only smaller, so it takes up only however much of the drive that I want. When doing this, be sure to make it big enough so that it will at least hold your existing data, so none of it gets damaged or lost.

After re-creating the DOS/Windows partition in a smaller size, be sure that you tag it properly. Then, create your Linux partition(s) (tagged as Linux native) and your swap space (tagged as Linux swap).

Software Installation

When Linux installation is finished setting up swap space, it will ask you what programs to install. Choose what you're interested in, modify what is about to be installed, and proceed. It might take a long time if you have an older computer. You can easily manually select what packages you want to install through the menu that Red Hat installation provides, or choose to install everything (I think by checking off the box at the bottom).

If you want to be able to build programs on your system from their source code (compiling), or get into programming sometime, be sure to install the development packages. I would also suggest installing the networking packages. It's safe to install all the packages, but if you're short on hard drive space just choose the packages that you don't think you'll need, and if you need them later, you can install them separately.

Installing LILO, the Bootloader

When the programs are done installing, write the Master Boot Record to `/dev/hda` (`/dev/sda` if you have a SCSI system.. if you don't know what I'm talking about then don't worry about it, just use /dev/hda). You want to do this at sector 0 (the beginning of the HD) because there's where the BIOS on your computer looks on your hard drive to boot up (if you want to use some other bootloader, that's fine, but LILO is really the only one I know how to use).

Starting Linux

When Linux prompts you that installation is complete, just reboot with no floppies in your floppy drive(s). When the screen says `LILO boot:`, type `Linux` (doesn't matter what is capitalized), and Linux will start up. If you don't do anything, LILO will just load what is on the first partition on your hard drive. You can later configure it so that it will load something else on default. To do that, refer to the fdisk section of the guide.

Now, when it's time for you to login, your username will be `root`, and you use the password that you typed in during install. Your Linux adventure/struggle begins.

Jumping Points

If you don't know what to do now, there's plenty! You can get your graphical interface set up, you can get connected to the Internet, and learn some Linux commands. And that's just the beginning. There are more pages in this guide to keep you occupied for a while. :-)

1. Accessing Your Floppy Drive, CD-ROM Drive, and Other Partitions
2. Configuring X
3. Installing Software Packages
4. Linux Commands
5. Setting Up a Dialup PPP Connection

You should also check for updates to Red Hat at http://www.redhat.com/support/docs/errata.html. These should plug up security holes and fix bugs that might have been in the software packages. To install these, read the quick guide on using RPM.

Author's Notes

When you use the Linux fdisk (`FIPS.EXE` in `\DOSUTILS\` on the Red Hat CD-ROM, I think) to partition your hard drive, it will retain all the data that is already on your MS-DOS FAT partition, provided that your hard drive is not overly fragmented. When you do a `dir` command in DOS, however, it will show its size before it was repartitioned, so DOS/Windows will think that your partition was never modified in the fi place.

If you already know quite a bit about Linux and want to create separate partitions that are to be mounted for each root filesystem, I would suggest that you make `/usr/` the largest partition. More information on modifying your partitions is available in the fdisk section of the guide. That guide is useful especially for changing your Linux filesystem after you get Linux up and running.

Comments

```
Fips.exe _definitely_ needs a defragged drive to work. fips won't
chop off space that contains files. This could be an issue when one is
running Norton Utilities which puts rescue information at the very last
cluster of the DOS partition, even after defragging (which would normally
gather all files at the beginning.). Also, one needs PartitionMagic to do
this kind of work on a FAT32 disk, as they come with win95b.

Mans Axel Nilsson              Sound Engineer
http://hem.passagen.se/mansaxel
```

Shawn Ormond wrote about his MetroX problem during installation. MetroX, as far as I know, is only in the offical Red Hat distribution, not the 2 dollar CheapBytes version. But here it is anyway:

```
Well, I fixed the problem....Maybe you might wanna make a note of this
in your manual or something....I don't know, but the solution was kinda
weird....On Red Hat Linux 5, the installation, gives you an option of
installing MetroX.  Well, I usually installed it, whenever I reinstalled
it. (I installed it for the 15th time today :c) )
Well, this time I didn't.  Let everything run its course.  And there we
go, I got full color.  I don't know exactly why it did that, maybe a
conflict between MetroX and the regular X-Windows thingy....I dunno, but
from what I understand MetroX has to do with X-Windows.  Anyway, thanks
for responding to my email, I appreciated it.

                                                        Sincerely,
                                                        Shawn Ormond
```

That's a lesson to be learned... if XFree86 supports your card, there's no need for MetroX. More information on configuring X or whatnot is through another document I wrote, Configuring and Troubleshooting X. However, that doesn't list what video cards XFree86 supports; you might want to try http://www.xfree86.org or one of its mirror sites.

Summary

- Partition your hard drive with DOS's `FDISK` and make sure you leave room for other partitions.
- Format your hard drive in DOS.

TOC

- Insert a boot disk rawritten with `boot.img`.
- Run the installation.
- Make a primary Linux partition, then extended, and then logical. (The last two are for the swap space.)
- Tag your partitions.
- Reboot and at the " LILO boot: " prompt, type Linux.
- Login as "`root`" and enter the password that you set.

Send an e-mail if you encounter problems. You can also make use of the help form.

Red Hat Hard Drive Installation

TOC

Last updated: July 27, 1999
Development stage: Alpha

To prevent any confusion, this is when you download the required files through FTP to your hard drive. This will also work if you copied the contents of the CD-ROM to your hard drive for reasons such as unsupported CD-ROM type. Just remember, if you're copying from CD-ROM, don't download what you already have.

If it's not too inconvenient, I would suggest buying a CD-ROM from CheapBytes or Linux System Labs to save you much trouble in downloading.

Download It

The first thing you do is make sure you have at least 500 megs free on your hard drive so that you can download the tree (the files you need in order to install Red Hat Linux) and so you can split it up. Before you play with partitioning, be sure to defragment the DOS partition that you're going to split so that all your data is at the beginning of the hard drive and you don`t lose anything in the process of resizing. After you've done that, find a site with the Red Hat distribution. Below is a list of fast ones that I've used myself.

The username is 'anonymous' or 'ftp', and the password is usually whatever you want it to be. Some FTP sites will only let you in if you use your e-mail address as the password, like you should be doing.

- sunsite.unc.edu/pub/Linux/distributions/redhat/current
- ftp.cc.gatech.edu/pub/linux/distributions/redhat/current

When downloading, the first thing you have to do is create a tree structure like this example. (unless your FTP program creates all the subdirectories for you; I heard WS_FTP does it)

```
C:\REDHAT\
  |----> RedHat - a subdirectory of C:\RedHat\, so it'll be C:\RedHat\RedHat
           |----> RPMS          -- binary packages - important
           |----> base          -- small filesystem setup archives - important
           |----> instimage     -- image used for graphical installs
  |----> images                 -- boot and ramdisk images - download to follow this
guide
  |----> dosutils               -- installation utilities for DOS - download to follow
this guide
  |----> doc                    -- various FAQs and HOWTOs
  |----> misc                   -- source files, install trees
  |----> COPYING                -- copyright information
  |----> RPM-PGP-KEY            -- PGP signature for packages from Red Hat
  |----> SRPMS                  -- Source RPMS for Red Hat distribution (not really
needed unless you want to build RPMs)
```

Make sure that you have all the files in "RPMS" and "base". If you miss one critical package, you could potentially mess up your installation. However, some large packages you probably won't need immediately are TeX, LaTeX, and Emacs. Those are pretty large packages that I haven't found much use for. You can download them later, though, if you need them.

You don't need to get the SRPMS directory. In fact, according to Matt Alcala all you need for a successful installation is the "RPMS" directory and the "base" directory. Also get the stuff in the "images" directory.

Backup?

I suggest you do not take chances and backup your system before you attempt to install it (if you have the ability and hardware to do so).

TOC

If you can't do a backup, at least defragment your DOS/Windows partition so you can repartition your hard drive later.

Of course, if you want to rid your system of Microsoft altogether, you can just delete your DOS/Windows partition.

Getting Into Red Hat Installation

After downloading it all, either get the boot image (it should be `C:\RedHat\images\BOOT.IMG`) or Autoboot (`AUTOBOOT.BAT`). Some prefer Autoboot because it boots from MS-DOS mode and runs Red Hat setup in a snap. If you get the bootdisk, get RAWRITE.EXE (in the DOSUTILS directory on the FTP server) also and "rawrite" it to a formatted floppy disk. Make sure you have the supplementary image (`C:\RedHat\images\SUPP.IMG` (so you can put in the supplementary disk when Red Hat Installation asks for it) and "rawrite" that to another formatted floppy. Set those aside for later.

When you're using RAWRITE, it will ask you to specify the source and target. The source should be whatever .IMG file you're trying to write to the disk, and the target should usually be A:, or whatever your floppy drive is.

Reboot with the bootdisk that has the image `BOOT.IMG` on it. This will take you to Red Hat's installation program. Once you're in there, it will ask at some point to insert the supplementary disk (the one with `SUPP.IMG` on it). Insert the supplementary disk and you're ready to go.

Creating, Deleting, and Re-creating Your Partitions

Next, you have to repartition your hard drive. It's possible to install Linux on a second hard drive; just pick which one. The devices are as follows:

- /dev/hda is the device on the primary master
- /dev/hdb is the device on the primary slave
- /dev/hdc is the device on the secondary master
- /dev/hdd is the device on the seconary slave

When there are partitions on the hard drive that you choose, the device will be followed by a number according to its partition number. For example, if you have a partition on your secondary master hard drive, it will be `/dev/hdc2` if it is the second partition on that drive.

Linux "fdisk"

In Red Hat install, when it gives you the option of using either Disk Druid or fdisk, choose what you want (I like fdisk) and do the following:

1. Make a primary partition (n).
2. Tag it as Linux native (t).
3. Make an extended partition (n).
4. Make a logical partition (n) and tag it as Linux swap (t).

The extended partition is there so that the logical partition can overwrite it and exist within it. The number of blocks you specify determines how many megabytes that the partition will take up. If you want to use space from an already existing DOS FAT partition, you should remove the existing partition first and then make another one, then tag it as "DOS 16-bit > 32", in most cases. However, make sure the DOS partition was defragmented before you repartition. Primary partitions should be numbers 1-4, as well as your extended partition. The logical partition, though it takes up the same cylinders as your extended partition, can only exist as partition number 5 and above.

Using the Partitioning Program

The "fdisk" should present you with a menu if you press 'm' upon starting them. Among the most useful commands to me are the following:

- **Print partition table (p) -** This will display information about what partitions you have. It's not necessarily what is already written to the partition table, it's what it is configured to write to the partition table upon exit with writing (w).

TOC

- **Write table to disk and exit (w)** - This will write the changes that you've made to your hard drive. When you cha the settings, they didn't really change the partition layout of the hard drive as soon as you made the changes. It jus up for when you use this option to change your partitions.
- **Add a new partition (n)** - What this does should be pretty self-evident. However, you might have to delete a partition that's taking up the entire hard drive first, then re-create it.
- **Delete a partition (d)** - Use this to delete partitions that are already existing in order to make room for a Linux partition, or to delete partitions that you created by accident.
- **Tag a partition (t)** - Also known as changing a partition's system ID, using this on a Linux partition will get it to be recognized. You MUST use this after the Linux partition is created, or else it won't be recognized.
- **List known partition types (l)** - To be used with tagging a partition. This is so that operating systems will know what kind of partitions are on the hard drive. This helped me to figure out what number to use to tag my Linux partition as, and also my Linux swap partition.

All you have to do is re-create your DOS partition with a different size to leave room for Linux, create a primary partition, tag it as Linux native (83), create an extended partition (numbers 1-4), create a logical partition (partition number 5 and up), and then tag the logical partition as swapa (82).

When you're in installation, it will mention cylinders. The number of cylinders is proportional to the capacity of your hard drive. In other words, if you have a 2.5 gig hard drive that has 620 cylinders, 310 cylinders is equal to somewhere between 1.2 and 1.3 gigs. To find out how many bytes a cylinder is on your hard drive, divide the number of bytes there are (2,500,000,000 in my case) by the number of cylinders (620). The result for me would be about 4,032,258 bytes, or around 4 megabytes a cylinder.

Install Software Packages

After putting a new partition layout on your hard drive, Red Hat installation now leads you up to installing the software that will come with your system. If you've got the room, go ahead and do a full installation. The process is pretty straightforward, and all you do is press [Enter] a few times and all the packages you select will be installed.

If you want to compile programs yourself (or get into programming sometime), be sure to install the development packages. I would also suggest installing the networking packages. It's safe to install all the packages, but if you're short on hard drive space just choose the packages that you don't think you'll need, and if you need them later, you can install them separately. For now don't worry about it.

Install LILO, the Linux Loader

Make sure you install LILO, the bootloader, unless you really know what you're doing and/or already have another bootloader that you know can load Linux. If you don't install LILO, make sure you have `LOADLIN.EXE` somewhere, and the Linux kernel (a file called **vmlinuz** usually) which you can get from a friend who might already be using Linux. You can later use LOADLIN with the kernel you have on your Linux system, which should be copied onto your DOS/FAT partition so that you can load it with LOADLIN.

Reboot

When Red Hat installation is finished, reboot without the floppies in the drive. It will show a prompt:

```
LILO boot:
```

That's where you type 'linux' (or Linux, or LiNUX.. it's not case sensitive). When it shows a login prompt, type 'root' as the username and for the password, type in what you set the password as in Red Hat installation.

Your Linux struggle begins. Prove yourself worthy. Read other portions of this guide. :-)

TOC

Jumping Points

If you don't know what to do now, there's plenty! You can get your graphical interface set up, you can get connected to the Internet, and learn some Linux commands. And that's just the beginning. There are more pages in this guide to keep you occupied for a while. :-)

1. Accessing Your Floppy Drive, CD-ROM Drive, and Other Partitions
2. Configuring X
3. Installing Software Packages
4. Linux Commands
5. Setting Up a Dialup PPP Connection

You should also check for updates to Red Hat at http://www.redhat.com/support/docs/errata.html. These should plug up security holes and fix bugs that might have been in the software packages. To install these, read the quick guide on using RPM.

Send all feedback to jtg@computers.iwz.com. You can also use the help form or guestbook.

Accessing Your Floppy Drive, CD-ROM Drive, and Other Partitions

TOC

Last updated: July 18, 1998
Development stage: Beta

The process of "mounting" is a means of accessing other types of filesystems. That means with this, you can use this to read and write to files from media such as floppies, CD-ROMs, and other partitions of your hard drive. Linux uses `/dev` entries to access this, along with the **`mount`** command. To access your floppy drive, type this:

```
mount /dev/fd0 /mnt/floppy
```

You'll have to create the directory `/mnt/floppy` if it doesn't already exist. Linux "attaches" the contents to the mount point, otherwise known as the directory `/mnt/floppy/` here. In place of `/mnt/floppy` you can specify any other empty directory.

The equivalent of typing `a:` in DOS to get into the floppy's directory is typing **`cd /mnt/floppy`**. Since it's now part of your filesystem, you just change directories.

You usually have to be the root user to mount and unmount. According to Kevin Ng it's possible to be a normal user and access devices such as a floppy. He said he modified his `/etc/fstab` file and added the 'user' option, and to consult the mount(1) manpage (type 'man mount') for details (of course, manpages might be a little hard to understand while you're starting out with Linux, but if you can handle them, by all means read them).

CD-ROM Drive

Mounting your CD-ROM drive depends on how the drive is set up. Some uncommon CD-ROM drives might not be supported by Linux, but let's just say you have a standard drive. You might have to try all of the following:

```
mount /dev/hdb /mnt/cdrom
mount /dev/hdc /mnt/cdrom
mount /dev/hdd /mnt/cdrom
```

The `hd*` entries might go on, but chances are that it's within this range right here. If you're successful, Linux won't spit out an error; if you're not, it'll say no such device or something like that. Just use the `cd` command to move to the directory you mounted the CD-ROM's contents to (this goes for the floppy drive, too). The subdirectories of the CD will be subdirectories in the mount point directory. (Again, this goes for the floppy drive, too).

TOC

Other Partitions on Your Hard Drive

If you have multiple partitions on your hard drive and want to copy files back and forth without rebooting, use the `mount` command also. You'll have to know which partition your other operating system rests on, and it is `hda*` in your `/dev/` directory. If you have Windows 95 or MS-DOS/Win 3.1 running on your first partition, the entry will be `/dev/hda1`. So, you have to type this:

```
mount /dev/hda1 /mnt/hd/
```

Well, okay. I'm making this look too much like my computer. Here's the basic command:

```
mount /dev/hda* /mnt/hd
```

This time, you might have to create a directory in `/mnt/` to do that... try **`mkdir /mnt/hd`** as root.

Again, `/mnt/hd` is the mount point/mount directory (that's where you're viewing/reading/writing the floppy or separate partition), and `/dev/hda*` is the device you are going to access. The `*` represents which partition you are going to access through the mounting process. The '*' is a whole number, usually from 1 through 4. You might also want to change the part after 'hd' to some other letter if you're going to mount another partition on your hard drive.

Unmounting

Unmounting is pretty easy. Basically, this is what you should do.

```
umount /mnt/hd/
```

Scott Northcutt wrote in to say that when using the **`umount`** command, you should be in a directory other than where the files are mounted. For example, if you have a CD-ROM mounted on `/mnt/cdrom`, you should use **`cd /directory/somewhere/else`** to get to another directory, and then unmount. If you don't do this, Linux will complain that the device is busy.

You can also use, in place of `/mnt/hd`, the device that you're trying to unmount, such as `/dev/hda2`. In the example, `/mnt/hd` is the directory you mounted to, and you change that with whatever directory you created or used to mount the partition, floppy, or CD.

Suggestions

Basically, you should use **`mount`** to access anything from a floppy to a Zip disk. It's the only tool that I know of to easily read and write to other partitions.

The file `/etc/fstab` includes information on which devices to mount automatically at startup.

```
Device names:

The first (master) drive on the first Eide interface is /dev/hda
The second (aka slave) drive on the first Eide interface is /dev/hdb
The first (master) drive on the second Eide interface is /dev/hdc
The second (aka slave) drive on the second Eide interface is /dev/h

SCSI drives are an altogether different thing. They are /dev/sda and so
on.

From this most people should be able to determine which device names to
use.

Mounting dos partitions:

The mount command needs a parameter to access vfat (long file names)

So I mount like this:
       mount -t vfat /dev/hda1 /msdos
Also, mounting CD's should theoretically be:
       mount -t iso9660 /dev/hdd /mnt/cdrom
However, since iso9660 is default, this should not be necessary.
"man mount" is a good way to learn more.

Mans Axel Nilsson
```

And about that VFAT thing, you have to have VFAT support compiled into your kernel for it to work with Windows 95's long filenames.

Adding a New User

Last updated: June 8, 1998
Development stage: Alpha

One of the nifty things about Linux (or any other Unix) is the fact that it allows different users to use the system. The root user is commonly known as the superuser, since he/she/it has control over everything on the system. Being logged in as `root` unnecessarily is commonly known to be dangerous and a security risk, so you create a normal user account for yourself so you don't screw up the system.

The advantages of having many users and one superuser account (for the administrator) is that the normal user can't screw anything up on the filesystem; normal users don't have the ability to delete files that are essential to running the Linux machine smoothly. But since you're running your own Linux system, you're still in charge.

The Easy Way to Add Users

To easily add a new login profile (commonly known as a user), type:

```
# adduser <username>
```

Alternatively, if you're using Slackware, use the `useradd` command. I'm not sure, but the syntax is probably `useradd <username>` also. For Debian, that probably has `adduser`, same as Red Hat. Adduser is a shellscript that "automates the really boring and repetitive task of creating new user accounts". Skip the section "The Hard Way to Add Users" if this was successful.

The Hard Way to Add Users

If you don't have any utilities at all on your system, I guess you'll have to add users the hard way. I really recommend that you get a shellscript or some sort of utility and use that instead; one is available for download right here. Put the file in /usr/sbin/ and type `chmod +x /usr/sbin/adduser` so you can run it. Then you can use the instructions under "The Easy Way to Add Users" instead of this section.

If you're still reading, we're going to add a user and we'll call him "joe".

Edit /etc/passwd

First, edit `/etc/passwd` by typing **pico /etc/passwd** (or in place of pico you can use vim, emacs, joe, jed, or whatever text editor). At the bottom of the file, add this:

```
joe::500:500:Joe:/home/joe:/bin/bash
```

The first field is joe's username, "joe". The second field, between the two colons ("::"), is where the password should be, but it's not set. When you set it, that field will become garbled with the encrypted password. The two fields with "500" are the user ID and group ID. Since joe is the first user you're adding, he starts out with the 500 ID. The next user you add should have a user and group ID of 501, and it keeps going up to about 65,000. The field with "Joe" in it is what the real name of user "joe" will

TOC

appear as. "/home/joe" is user joe's home directory, which is where he keeps all his files and stuff. Finally, "/bin/bash" is the shell he uses to type in commands. With "/bin/bash" set as his shell, joe ca in and type in commands. If you don't want to let joe log in and type in commands, use "/dev/null" instead of "/bin/bash".

Edit /etc/group

Now type `pico /etc/group` and insert the line:

```
joe::500:joe
```

The first "joe" is the group name. Make sure you have two colons. Then put in the group ID of joe, which, if you can remember, is 500. Since joe is the only member of the group joe, then you type joe again. Simple enough, right?

Create Mail File

Usually the mail for users is stored in `/var/spool/mail/`. As root, type `pico /var/spool/mail/joe` and without typing anything at all, exit and save the changes.

Make joe the owner of that file by typing **chown joe.mail /var/spool/mail/joe**. We don't want anyone to read joe's mail so type **chmod 660 /var/spool/mail/joe**.

Create Home Directory

Do **mkdir /home/joe** as root. After that's done, type **chown joe.joe /home/joe ; chmod 2775 /home/joe**. Now you can set the password.

Setting the Password

Now, you shouldn't forget to set a password or else the user won't be able to log in (or maybe anybody can log in as that user). To set a password, type:

```
# passwd <username>
```

It will prompt you to type in a password twice. If it gives any errors, make sure the password isn't a common word, has 6-8 characters, or has too many of one character. This may seem limiting and insecure at first, but it actually enhances the security of each user. If you are using Red Hat 4.0, you might have to remove a userlock by typing **rm /etc/.pwd.lock**. This is a bug in Red Hat 4.0 that is not in Red Hat 4.1 or 4.2. Do **not** remove /etc/passwd or else you're really in trouble; no users will be able to log in, not even the root user, so you might end up reinstalling!

Change Finger Information

After you add users and set their passwords, you might want to do a `chfn [user]` as root to change information like the user's real name (if you choose to supply it and make it available). The `chfn` program is to **ch**ange the **fin**ger information for a user, so that anybody using "finger" protocol sees what information about a user is available, such as their real name, the time of their last login, and other

Groups

You can create groups of users that are identifiable through one name. For example, I have a group called "josh" in my system, and members of those groups are the users `jgo`, `joshuago`, and `jgo.local.net`.

So how can these be useful? Well, you can change a file to be owned by a group of users instead of just one user, so that they all can write to the file.

Edit /etc/group

If you've got pico installed as your text editor, use that for now, or use whatever other text editor you prefer (vi, vim, emacs, joe, jed, the list goes on). Anyway, start out by editing `/etc/group`, a file that contains a list of the groups.

Take the example from the line starting with "users". On my system it looks like this:

```
users::100:joshuago,jgo,jgo.local.net,juliusgo,todd
```

That's what I did to start out, taking the existing example. Then I added my group, "josh", so it would include only my accounts and not the other people's accounts.

So, I took that format and added my own user group:

```
josh::101:joshuago,jgo,jgo.local.net
```

Exit your text editor and say yes to saving the changes. That puts my `joshuago`, `jgo`, and `jgo.local.net` accounts all into one group. When I'm still logged in as any of those users though, I can't know that I'm a member of that newly created group unless I log out and log in again. Then I can type **groups** and it will show all the groups that I'm a member of.

Changing File Ownership

To change ownership of a file or directory so that members of my group can read, write, and/or execute that file, I have to log in and type:

```
chgrp josh [file_or_directory]
```

That will show up when you do an **`ls -l`**, that its owner is whoever created it, and that the group that owns it is "josh".

Changing Permissions (Access to the File)

If you want to let others in your group read from a file that you own, type `chmod g+r file`. To let other members of your group write to a file that belongs to you, do a `chmod g+w file`. To let a member of your group execute the file, type `chmod g+x file`. In case you're confused, "file" simply represents the name of the file you're trying to change permissions for. The parts after the "g+" part command line can be combined so that you let your users access a file through reading, writing, and executing. For example, to give members of your group just that (reading, writing, and executing permission), type `chmod g+rwx file`.

In place of the "+" after the "g" in the `chmod` command line, you can add a minus sign ("-") to take away those access rights. :)

This will let only the group that owns the file access it. Only the owner (a single user, not an entire group of users) can change the permissions.

Other Stuff on Permissions

In place of "g" in the chmod command line (which changes access rights for the group) you can use "u" (for **u**ser, the one who owns the file), and "o" (for **o** others, who don't belong to the group.

Comments, questions, suggestions, corrections? Send them to Joshua Go. You can also use the guestbook or help form.

Compiling a New Kernel

TOC

Created on: April 4, 1998
Last Modified: April 20, 1999
Development stage: Alpha

The kernel is a file that is the most important part of any operating system. The kernel sets up a basic interface between the hardware and software. It's also what distinguishes Linux from other Unix operating systems. Most of what people think is Linux is actually from the Free Software Foundation and other GNU authors. In addition to hardware, the Linux kernel is responsible for supporting protocols like TCP, IP, PPP, and all the filesystems out there.

Compiling the kernel should be one of the easiest things to do, but as with most things dealing with Linux, there are problems people encounter along the way. Let's just save that thought for later.

Downloading and Unpacking the Kernel Source

This is the first (and arguably most frustrating) step in the kernel upgrade process.

Downloading the Kernel

The latest Linux kernel can be downloaded from the following sites with the file name `linux-x.y.z.tar.gz`, "x" is the "2" in `linux-2.2.4`, "y" is the "2", and "z" is the "4". When the "y" value is odd-numbered, as such in "2.1.109", it is a development kernel (like beta software, but less stable) and isn't considered to be stable enough for the average Linux user. If you're reading this document, you're probably much better off with a stable kernel. *As of the writing of this document, there is no development tree. 2.2.x is your best bet!*

Kernel.org Mirror Sites

The Linux kernel source can be downloaded from the following sites.

- kernel.org (ftp.kernel.org/pub/linux/kernel/v2.2/)
- Sunsite (sunsite.unc.edu/pub/Linux/kernel/v2.2/)
- ftp://ftp.XX.kernel.org/, where XX is your ISO-country code (United Kingdom = ftp.uk.kernel.org; Canada

= ftp.ca.kernel.org; Russia = ftp.ru.kernel.org; etc)

Once you find the kernel you want to install, download it into the directory `/usr/src/`. This is a moderately large file (at least 10MB as a gzip'd tarball for the 2.2.x kernel), so go do something else while downloading it. Of course, if you've got a fast connection, then it won't take as long to download. Generally you want the latest version of the latest stable kernel: currently 2.2.zz. Remember that ftp sites go in raw numerical order - `linux-2.2.1.tar.gz` comes before `linux-2.2.12` on the list, even though `linux-2.2.12.tar.gz` is newer. This should not be an issue for several months, as `2.2.4` is the current version (25mar1999).

Unpacking the Kernel Sources

When you're done downloading it, extract it with `tar -zxvf linux-2.2.zz.tar.gz`, where linux-2.2.zz.tar.gz is the file that you downloaded. This will extract everything into the `linux` subdirectory. It might take a while, so be patient. If you have compiled the kernel before (or installed any kernel header/source packages), you will need to `mv` the old files out of the way before unpacking the kernel. Simply do `mv linux/ linux-2.0.35/`, or whatever version of the kernel the headers/source is for.

Patching

If you're downloading a new kernel and don't already have the source lying around, you can't patch yet. If you're downloading a kernel for the first time, skip this subsection and go on to the next one.

It's such a big drag to download the entire kernel source tree every time a new version comes out, so I recommend using patches. A patch contains information on the differences between a previous version and the current one. For example, if you have kernel 2.2.5 and want to upgrade to 2.2.6, you need to get the patch that contains the differences between 2.2.5 and 2.2.6.

Kernel patches have filenames such as `patch-2.2.6.bz2`. In this example, it's the patch to upgrade from 2.2.5 to 2.2.6. You should download patches into the `/usr/src/linux/` directory. Depending on what compression format you downloaded the patch in, you uncompress it differently. For bzip2, the compression format this patch uses, you use the commandline `bzcat patch-2.2.6.bz2 | patch -p1`. If you downloaded it in a gzipped format, you would type `zcat patch-2.2.6.gz | patch -p1`.

At first, the term *patch* gave me a connotation that this was only a quick fix. (I had a mental image of a piece of cloth sewed on to cover a hole on a jacket.) It's not really that. It's just something that fixes the source tree accordingly.

Checking Dependencies

When compiling and running the kernel, the system expects certain things to be in place. For example, you need to have a compiler to get from the source code to a binary format the system can execute. You also need the `modutils` if you plan on using modules, and `autofs` for automounting, etc. To know what the most recent versions are, see the file `linux/Documentation/Changes`. This file will list what packages you need, what the base version is (lowest version you need to compile--it's usually a good idea to try out the newest version of a given package), and where to get new copies of them. This file also contains vast amounts of unneeded information. What you're most interested in is the table of most recent versions and the info on getting new versions.

Here's a sample of the most recent version chart (taken from Changes for 2.2.3):

```
Current Minimal Requirements
****************************
```

```
   Upgrade to at *least* these software revisions before thinking you've
encountered a bug!  If you're unsure what version you're currently
running, the suggested command should tell you.

- Kernel modules         2.1.121                  ; insmod -V
- Gnu C                  2.7.2.3                  ; gcc --version
- Binutils               2.8.1.0.23               ; ld -v
- Linux libc5 C Library  5.4.46                   ; ls -l /lib/libc.so.*
- Linux libc6 C Library  2.0.7pre6                ; ls -l /lib/libc.so.*
- Dynamic Linker (ld.so) 1.9.9                    ; ldd --version or ldd -v
- Linux C++ Library      2.7.2.8                  ; ls -l /usr/lib/libg++.so.*
- Procps                 1.2.9                    ; ps --version
- Procinfo               15                       ; procinfo -v
```

Now, if you run a libc5 system (most older distributions), you will not have a libc6. And some of these packages are entirely optional -- if you don't currently have it, you probably don't need it, assuming your system has the ability to compile C programs.

So, you need to go through the list and double-check your dependencies. If something is old, look further down in Changes for its homepage/distribution site. Let's say your Module utilities were outdated. You'd look further down in Changes and find the lines:

```
Modules utilities
=================

The 2.1.121 release:
ftp://ftp.us.kernel.org/pub/linux/kernel/v2.1/modutils-2.1.121.tar.gz
```

Which, gives you the release number and a location to download the most recent version.

After you download the updated files, you will need to move them to an appropriate location (9 out of 10 hackers agree on `/usr/local` as a stuff-compiled-locally directory), and unpack them (`tar xzf foobar-1.2.3.tar.gz`). Most kernel packages are well-behaved and will untar into their own subdirectory. `cd` into this directory and *follow their instructions*, usually found in INSTALL or README (`less INSTALL` or `less README`). If these directions fail, mail us. But not until you've read the instructions. We'll probably grab the package and try installing it ourselves, to see if it's broken or not.

You should now have your kernel downloaded, unpacked into `/usr/src/linux`, and all the things you need installed.

Compiling the Kernel

Now that most of the hard work is taken care of, it's time to begin the dull part!

Here is a basic outline of the kernel-compilation process:

- `cd /usr/src/linux` (just in case you're in the wrong directory)
- `make mrproper`
- `make menuconfig` (If this doesn't work out, use `make config`)
- `make dep`

- `make clean`
- `make zImage (Usually the longest part of the process)`
- `make modules`
- `make modules_install`
- `cp arch/i386/boot/zImage /boot/vmlinuz-2.2.zz-new`
- (edit `/etc/lilo.conf`)
- `lilo`
- `reboot` (no need to do it immediately but the new kernel won't be loaded until you reboot)

I will not go into great detail about what every command above does; if you want to know, read the Linux Kernel HOWTO.

Start by doing `cd /usr/src/linux` and `make mrproper`. These first two steps simply clean up any cruft that might have accidentally been left in the source tree by the development team. Then read on.

Configuring the Kernel

Like all other powerful software, the Linux kernel has to be configured. This is done during the `make menuconfig` step. `make menuconfig` starts an application that allows you to browse through the options available for the kernel and make settings. To move around, use the arrow keys. Enter allows you to open up a menu, and tab gets you down to the "buttons" at the bottom. When you want a specific part of the system compiled into the kernel, hit 'y' while it is highlighted. If you realize that you don't need or don't want something, go back and hit 'n' over it. If you just want to try something out or use it occasionally, you can usually compile it as a module. All this (and more) is documented in `menuconfig`'s docs.

While doing `make menuconfig`, a lot of the things that you have to configure are pre-set, so if you don't know what it is, leave it alone. However, if you are pretty sure it won't hurt to disable support for something in the kernel, disable it, because the defaults are not always right for everybody. I usually have to change the settings a little bit to keep my kernel small.

If you use `make menuconfig`, you can hit "?" if you're not sure what an option does. This will bring up a small amount of text describing them, and usually a hint as to what the best option is if you don't know what it is (some things don't have a ? screen, so you're best off to go with the defaults on those if you're not sure what they do)

Compiling the Kernel

After you finish configuring things, do `make dep; make clean`. This does a few technical things regarding dependencies and old compiliations lying around. Ít's just something you need to do and need not worry about as to why.

Next, do `make zImage`. This may take a long time depending on your system (on my K6-2-300/64M it takes about 10 minutes with X running. On my friend's 386sx-16/8M it took several hours with nothing else running; on another friend's dual p100/48M it takes about 14mins -jbm). Usually it's a good idea to watch it compile and keep an eye on possible warning messages. In particular, `signal 11` messages are Bad News. If you get any `signal 11`s, see the note on sig11. Another common problem is the kernel being too big. If your kernel is too big, try doing `make bzImage` after the `make zImage`. A `zImage` is a gzip'd kernel--it's compressed down to a more manageble size, which makes booting a little slower. A `bzImage` is a bzip2'd kernel (bzip2 is a more compact compression format). I (jbm) have to use `make bzImage` because my kernel has gotten too large for `zImage`. If you have use a `bzImage`, please replace the references to `zImage` with `bzImage`.

`make modules; make modules_install`: these compile and 'install' any modules you selected. You also need to set up the module dependencies for these, see the Note on modules.

Notes on Compilation

Note on sig11:
A `signal 11` error during kernel compilation means that `gcc` found an error in one of the files while compiling. `sig11`s, as they are commonly referred to, are typically caused by bad hardware. In my (jbm) experience, they have been caused by bad processors, bad memory, and overclocked memory. With the popularity of 100MHz bus clock systems (ie: k6-2/3 and pII/pIII), there is an increasing number of people overclocking RAM. I know, I do it myself. I often get `sig11` messages during compiles, when the computer is getting hot. My general strategy for dealing with these is to lower system load (close x, stop listening to MP3s, etc), and recompile the whole kernel. Because this is a kernel-level file, do not simply remove the .o file and move on. This could lead to system damage.

Where to Copy the Kernel

When you're copying the kernel to where it's supposed to be, that may change from system to system. You can use what I have up there if you're on a standard Red Hat system; if you're using Slackware, it's not in `/boot`--it's in `/`. So instead of `/boot/vmlinuz-2.2.zz-new`, it's `/vmlinuz-2.2.zz-new`.

We use the vmlinuz-2.2.zz-new for safety's sake. If you forgot something essential in your kernel, your computer wouldn't be able to boot with the new kernel. Unfortunately, this makes life a little more complicated now, but it's not so bad. All you have to do is edit `/etc/lilo.conf` (a somewhat daunting task, but really pretty simple).

Editing `/etc/lilo.conf`

`/etc/lilo.conf` is the settings file for **lilo**, the LInux LOader. Editing this file can be hazardous to the health of your computer. Before you edit it, make a backup copy of it (`cp /etc/lilo.conf /etc/lilo.conf.backup-foo`, or some other name you can remember.). This backup is in case you accidentally delete a line in the file. If you need to restore your old lilo.conf, just do `cp /etc/lilo.conf.backup-foo /etc/lilo.conf` and start anew. Now then - to the real work with lilo.conf

Open up `/etc/lilo.conf` in your favorite text editor. The file should look something like this:

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
#compact        # faster, but won't work on all systems.
delay = 50
vga = normal    # force sane state
# ramdisk = 0     # paranoia setting
# End LILO global section
# Linux bootable partition config begins
image = /vmlinuz
  root = /dev/hda2
  label = Linux
  append = "ether=9,0x310,eth0 mem=64M"
```

```
# Linux bootable partition config ends
# DOS bootable partition config begins
other = /dev/hda1
  label = DOS
  table = /dev/hda
# DOS bootable partition config ends
```

What you want to do is this: copy the section from "image = vmlinuz" through to the "# Linux bootable partition config ends"(or the next "image =" line, if you have one) letter for letter right below where it currently is, like so:

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
#compact        # faster, but won't work on all systems.
delay = 50
vga = normal    # force sane state
# ramdisk = 0     # paranoia setting
# End LILO global section
# Linux bootable partition config begins
image = /vmlinuz
  root = /dev/hda2
  label = Linux
  append = "ether=9,0x310,eth0 mem=64M"
image = /vmlinuz
  root = /dev/hda2
  label = Linux
  append = "ether=9,0x310,eth0 mem=64M"
# Linux bootable partition config ends
# DOS bootable partition config begins
other = /dev/hda1
  label = DOS
  table = /dev/hda
# DOS bootable partition config ends
```

If you don't have a "delay = xx" line, or if xx is equal to 0, change it to "delay = 50" (without the quotes). By adding this line, you instruct LILO to wait 5 seconds for you to enter in a kernel image to load before it automatically boots the first one in the list (in this case Linux). Now change the second "image = /vmlinuz" (or "image = /boot/vmlinuz") line to "image = /vmlinuz-2.2.zz-new" (xx is the version number, like 2.2.4). Also change the second "label = Linux" line to "label = New". This section of the file now looks like this:

```
image = /vmlinuz
  root = /dev/hda2
  label = Linux
  append = "ether=9,0x310,eth0 mem=64M"
image = /vmlinuz-2.2.zz-new
  root = /dev/hda2
  label = New
  append = "ether=9,0x310,eth0 mem=64M"
```

Save the file to lilo.conf. Now type in `lilo` at the command prompt. It should give you output similiar to thi

```
Added Linux *
Added New
Added DOS
```

If you encounter any error when you type **`lilo`** then there the kernel is probably in a differnt locataion than the image = line says. Check that, and look for capitialization problems.

Reboot, and when the LILO: comes up, hit left-shift (or any other key that happens to please you), and then type in "new" (without quotes, again). This will boot the image labeled new - /vmlinuz-2.2.zz-new. If this boots and works well, just do `cp /vmlinuz /vmlinuz-backup` and `cp /vmlinuz-2.2.zz-new /vmlinuz`. Open up `/etc/lilo.conf` again and change the New section to be name = backup, with the image being /vmlinuz-backup. If you don't want the LILO boot prompt, comment out the delay = 50 line (stick a # in front of it). Now type `lilo` at a prompt and if lilo installs well, try rebooting to make sure both your new kernel and your backup kernel work.

PPP Support in the Kernel

Before compiling, make sure you enable PPP support during `make config` or `make menuconfig`. It's left to be NOT supported on most of the kernels I've compiled so far (maybe because the kernel developers have high-speed digital connections), but if you connect to your ISP through PPP, you better enable this or else you'll have to recompile later on to connect to the Internet.

Math Coprocessor/FPU

If you don't have a math coprocessor (if your CPU is a 386DX or 486SX without a math coprocessor), make sure you say **YES** on **kernel math emulation**. This is what stopped me the first two times; I forgot about the **EMULATION** part. If you have a tape drive, SCSI interface, or any other weird or expensive stuff then go through **`make config`** very carefully to optimize your system. If you have a 486DX, Pentium, or higher processor with normal system specs, then just browse through configuration a bit and be careful. This may get boring (especially if you have to do it three times to get it right), but still read through it before jumping to compilation. Eventually you'll know what's there to configure and you'll just zoom past it.

Unresolved Modules?

You might see some errors/complaints about "unresolved modules" when you're booting the new kernel. Whatever the cause is, try moving your old module directory in `/lib/modules` (or wherever your kernel module directory is) to `/lib/oldmodules/`. For example, when I boot up and some errors pop up about my 2.2.4 modules being unresolved, I might go into `/lib/modules/2.2.4` and move the modules that are being complained about. To avoid these errors, do `cp -r /lib/modules/2.2.yy /lib/oldmodules/; rm -rf /lib/modules/2.2.yy` before you do `make modules` and `make modules_install`. (For example, if I were recompiling Linux 2.2.4 and I already had modules for 2.2.4 installed, I would type `cp -r /lib/modules/2.2.4 /lib/oldmodules/; rm -rf /lib/modules/2.2.4` before I went and compiled the modules). The unresolved modules error is quite common for the first few times you complie the kernel and has stumped Linux newbies worldwide. Of course, you could just delete the old modules directory (`rm -rf /lib/modules/2.2.4` for example), but it is generally a bad idea to delete things you might wind up needing if the compile doesn't go quite right (also a common problem for your first few compiles ;^)). So just move things around.

I've had a problem while compiling development kernels (those starting with 2.1) that complains about unresolved modules; I thought I needed to update a few programs, and I was right. I found the page at http://www.linuxhq.com/pgmup21.html.

Comments

```
You might wish to add that when you use an RPM kernel version, it appears to
come in separate kernel-source and kernel-headers files.
Simon Hampton <simon.hampton@tvd.be>
```

(The bold is my own, to stress the importance of **RPM** here. This particular problem is due to a broken-by-design strategy on the part of the RPM package maintainers. I guess it keeps package size down, at a large convenience/stability cost. -jbm)

Authors' Notes

Some people using SCSI on their Linux box *may* have problems compiling the kernel. Since I don't have SCSI installed on my system, I don't know what to do. However, Jeffrey here seems to know. If any of you SCSI users have been having the same problem, please verify this information with me so that I'll sound more confident. :)

```
Here's a list of my in's and out's of getting the SCSI modules to boot
with your new kernel.

* After you have made the kerenel and modules i.e. "make zImage"
  and "make modules" make sure that your loopback device is
  available try "insmod loop" OR maybe it isn't a module or just
  complied in straight then you don't have RHL installed and this isn't for
  you!, This is default as a module with RHL along with everything else

* Another trap that I found is that with comfiguring the Kerenel with
  "make xconfig" at a res of 640x480 the two buttons at the bottom
  are cut off and I kept on going to just close the window and wonder why
  the kerenel wouldn't compile!

* After you have inseted the loopback device, do a "make
  modules_install" to install the new modules (including you brand new
  SCSI drivers!) (Don't forget to back up the old set of modules in case you
  stuffed on the "insmod loop" and need to reinsert it later!

* Now just type "mkinitrd /boot/initrd 2.2.3" where the
  filename points to the image file in the lilo.conf and the version of your
  new kerenel is there and present! The program mkinitrd works off your
  /etc/config.modules file!

* FEW THINGS TO REMEMBER! - Don't forget to back up your old kernels and
  old initrd image file!, you never know this MAY not work successfully! (I
  know I have stuffed it up completely about 10 times!) :( anyway this
  booted for me on both options the old and linux in LILO then copy your new
  kerenel into it's final home and RUN LILO!
```

Jeffrey Borg
jeffrey@tudogs.net.au

TOC

Comments, questions, suggestions, corrections? Send them to jbm@intertek.net - if they're my mistake I'll reply, if not I'll forward them on to Joshua.

TOC

Setting Up Sound

Created February 1, 1998
Last updated: July 20, 1999
Development stage: Beta

Personally, I do not have anything against a quiet computer, but the rest of the world seems to disagree with me. Eventually I succombed to the environment around me, and made the effort to get my sound working. It proved to be quite easily done.

I want to know what else you feel may be missing in the information presented here. An e-mail is always welcome and encouraged.

Check Your Distribution

You may already be capable of sound, especially if you have one of the newer distributions. Many distributions have sound modules included on a defualt installation so that you won't have to recompile the kernel. Chances are that your distribution does this, and if it does, you don't have to read the longer parts of this guide. After you're done here, skip on down to Sound Playing Programs.

Red Hat Linux Sound Configuration

If you have a recent version of the Red Hat Linux distribution, try to run the **`sndconfig`** program (as root).

It could ask for sound card settings such as the I/O base address, IRQ, DMA, and 16-bit DMA. Be prep to give them. It also calls **isapnptools** PnP card detection utility.

Try this command for any distribution closely related or, more commonly, based, on Red Hat Linux. (Mandrake comes to mind.)

TurboLinux Sound Configuration

TurboLinux users should use the **`turbosoundcfg`** utility. It could ask for sound card settings such as the I/O base address, IRQ, DMA, and 16-bit DMA. Be prepared to give them. It also calls **isapnptools** PnP card detection utility.

I think **turbosoundcfg** was based on Red Hat's utility anyway.

Sound Configuration on Other Distributions

If you know what sound configuration utility your distribution uses, please write in and let me know.

Otherwise, you're out of luck for now. Read the rest of this guide, or wait indefinitely until I update this page.

Configuring Your Kernel

The kernel sources could probably already be installed in your `/usr/src/linux` directory. Go there using **`cd /usr/src/linux`**. If it doesn't exist, then you need to grab yourself a copy of the kernel sources. Follow the download instructions on another page, Compiling a New Kernel. That page delves deeper into the process of compiling a new kernel, also.

Get into kernel configuration using the **`make config`** or **`make menuconfig`** command. I prefer **`make menuconfig`** because it allows me to go back and check for any mistakes, unlike **`make config`**. If you want an X interface to configure your kernel's settings, you can also use **`make xconfig`**.

Once you're in, configure all the other stuff that you need to (according to Compiling a New Kernel, and then configure sound.

Using "make menuconfig"?

If you're using **`make menuconfig`** to configure your kernel, just scroll down to "Sound" with the arrow keys and then press [Enter]. It will probably say "Sound card support" and alongside that, a checkbox. While that's highlighted, just type in "y" (standing for "Yes") to enable it. Basically, the controls are typing "y" to include support for something in the kernel, typing "m" to make something a loadable module which is separate from the kernel, and "n" to disable support for something. The controls for this interface are explained near the top of the screen.

Configuring the Soundcard and Other Stuff You Need

The configuration, no matter what you use, should ask you what type of card you have. Some that are listed include the Sound Blaster (various types... I have the SB16 PnP), the PAS16, the ESS soundcards, and other less-common cards. If your card is listed as Sound Blaster compatible you should enable support for the Sound Blaster. By default, none of these cards is supported (answered "Yes" to), so choose wisely, grasshopper. Also, only enable support for the card you need, if possible, because that will keep your ke small and will prevent potential problems when the kernel tries to detect all the supported cards.

There are some other items that you may never have heard of, such as "Generic OPL2/OPL3 FM synthesizer support" or "6850 UART MIDI support". According to Claudine Langlois, *"OPL2 / OPL3 is a standard made by Yamaha for MIDI files. It's in the chipset of the sound card and it helps to have a much much better sound when the program that reads the file knows how to use OPL2/OPL3. Believe me, it's worth to be heard. Of course, OPL3 is better than OPL2..."*. And for that 6850 UART MIDI stuff, *"6850 is a standard for a FIFO buffer for the game port on a PC. MIDI support means to plug a synthetiser in the game port of the sound card. By the way, UART means Universal Asynchronous Receiver/Transmitter."*. Thanks to Claudine and all the others who wrote in about this.

If there are options for them, enable support for **/dev/audio** and **/dev/dsp**. New kernels do not have options for these, because supporting them is automatic when one enables sound support. In **make menuconfig** this can be done by simply pressing "y" when "/dev/audio and /dev/dsp support" is highlighted.

More items you will probably want to enable are support for MIDI, FM synthesizer support, and /dev/sequencer support. If you don't have even a vague clue as to what these do, basically they're for playing files that have specific instructions to the machine to play certain sounds.

System Information

It's highly recommended that you write down or memorize the settings that your system is using. Most importantly, what I have in mind are the IRQ settings, the DMA channels, and the I/O base addresses. The kernel configuration will ask for this.

Use **cat** to print out the information about your system, such as the I/O base addresses (/proc/ioports), DMA channels (/proc/dma), IRQs (/proc/interrupts), and any devices you might be interested in (/proc/devices).

Recompiling the Kernel

Now is the time to recompile the kernel. There's more information on how to compile a kernel located in this part of this Linux guide, but if you're already familiar with compiling a kernel, then it's safe for you to read on.

Basically after you type in **`make dep`** and **`make clean`** to make the depenencies and clean out the old junk from your last kernel compilation (if any), type in **make zImage**. This should compile the kernel. After you're taken back to the prompt then you should type **`make modules`** and then **`make modules_install`** to compile all the modules, which are these little itty bitty files that are loaded for something that was not directly compiled into the Linux kernel (a single file). Then copy the kernel to the appropriate place, which is `/boot/vmlinuz` if you're using Red Hat. Do this by typing in **`cp arch/i386/boot/zImage /boot/vmlinuz`**, assuming you're using Red Hat of course. (The kernel is /vmlinuz on Slackware. I don't know about other distributions.) If you use LILO to get into Linux, type **`lilo`**. If you use LOADLIN (loading DOS/Win95 first and then using it to boot a copy of your kernel), you need to copy the kernel (arch/i386/boot/zImage) to your DOS partition. Again, there's more information on this available right here.

Did it Work?

Now, whatever you do to reload Linux totally, do it. In my case I would reboot. When your system spits out all its boot messages, here's what it should say about the soundcard:

```
Sound initialization started
<Sound Blaster 16 (4.13)> at 0x220 irq 5 dma 1,7
<Sound Blaster 16> at 0x330 irq 5 dma 0
<Yamaha OPL3 FM> at 0x388
Sound initialization complete
```

Of course, it will look different if you have a different type of card. As long as you hear a little pop from the speakers or something, that means it was configured correctly.

PnP Cards and the Problems Associated with Them

My soundcard is PnP and it cost me a lot of frustration and experimentation. That's what you have to frequently do when using Linux. But on to the information you actually care about: how to fix it if it goes wrong for you.

If you think you followed the instructions on this page but get something similar to the following error message when you boot up, then it could be due to the PnP configuration. Here it is:

```
Sound initialization started
Sound initialization complete
```

Along with error messages like that, you won't hear a pop made by the speakers, either. You probably configured everything right (you know, the IRQ, DMA, and I/O base address stuff), but the card is PnP and Linux doesn't like that (well, the stable kernel anyway). That's why you should set up your machine for PnP in the BIOS, if you can. Another method is with the isapnptools utility, which I haven't figured out yet. Information on it is available at http://www.roestock.demon.co.uk/isapnptools.

Basically all that I had to do in the BIOS to get the soundcard to work was to change a setting in the PnP configuration. I have an Award BIOS, and the option was "PnP OS Installed" and the answer was "Yes". All I had to do was change it to "No". The reason this worked, I'm guessing, is that the BIOS set up the PnP devices for me because it was told that there was no PnP-aware operating system installed. Before, it was on "Yes" so it just left the configuration of all the PnP devices (the soundcard included) up to Windows 95.

The above method should suit most people just fine. However, I also have an Ethernet card, and when I change those options in the BIOS, my Ethernet card (a jumperless one) refuses to communicate with the rest of the network. So, I have to load Windows 95 first, exit to DOS, and then boot Linux using LOADLIN and my kernel. The command line for Loadlin is **loadlin kernel_file**. Make sure that the kernel file you're using is one with support for sound. The kernel file I'm referring to is the actual kernel, which goes by the name of **vmlinuz** somewhere on your Linux filesystem (/boot/vmlinuz or /vmlinuz).

I leave the "PnP OS Installed" option to "Yes" when I want to boot into Windows 95 and then use Loadlin to load Linux. Windows sets up the soundcard and gets it working, and the Ethernet card works too. I exit from Windows by going to Start -> Shutdown and then choosing the option of "Restart in MS-DOS mode", which is just quitting Windows and going to DOS.

If I make the "PnP OS Installed" option to "No" and then try the Windows -> Linux loading method, my soundcard will still work, but my jumperless Ethernet card will not. So, I leave that option to "Yes" in the BIOS.

Sound Playing Programs

Most of the major Linux distributions (Red Hat, Slackware, Debian, and Caldera) come with utilities to play sound. I don't know about the less well-known distributions, but they probably include them also.

For `.au` formats, you don't even need to have a program installed for playing sound. All you have to have is the **cat** program and sound properly configured. At the prompt, you can type in **`cat soundfile.au > /dev/audio`**. That should work for .au formats only.

MIDI

Personally, I use the program **`playmidi`**, which was already pre-installed on my Red Hat machine. If you didn't install that package for Red Hat, it's **`playmidi-*.rpm`** from your local Red Hat mirror or your Red Hat CD-ROM. I'm using a wildcard because I'm too lazy to find out what the current version is... plus, I won't have to update this document if a new version of playmidi comes out. :-)

Playmidi is also available from SunSITE. As I'm writing this, it's available from sunsite.unc.edu/pub/Linux/apps/sound/players/. The basic syntax is **`playmidi filename.mid`**.

MP3

There are several good MP3 players that I'm aware of now. The ones that I use often are XMMS (formerly X11AMP) and mpg123. There's also GQMpeg, which is a front-end to **mpg123**.

XMMS looks very similar to Winamp. You can even use Winamp skins for it. It supports playlists as well. As usual though, watch out for high system load that will cause your MP3 playing to skip.

RealAudio

You can download RealPlayer from RealNetworks. You can configure Netscape to use that as a plugin. The README included in the Linux version might have information on how to do that. If not, let me know that it doesn't.

WAV

Paul Winkle let me know about the **play** command. I checked, and it came with my Red Hat 5.1 distribution. It uses **sox**, which can also convert between audio formats and even do basic effects.

Comments

Dave Click wrote in to say that OPL3 is his soundcard... that kinda makes sense to me now, so if you've got the soundcard that he does, then you probably want to enable support for that also. Anyway, here's his message.

Just like to say I really like your page... i'm just getting started
linux and this is helping me out a lot. I have something to add,
surprisingly... in section #9 - setting up sound, you mention:

Generic OPL2/OPL3 FM synthesizer support

OPL3 is my sound card - it's a Yamaha, and in Win95 that's my setting for
how it plays Midi.

I don't know if that helps... i don't know what opl stands for or
anything...

Sybren Stuvel wrote:

I really like to contribute to your pool of Linux-info!
Here's my addendum:

I own a Creative Labs Soundblaster 16. These are the settings:

```
PORT: 220  IRQ:5  DMA:1,5
```

Now for the problem: make menuconfig has a default IRQ 7. As you
can see, mine is 5. The menu refuses to change the IRQ to 5!!! My
solution: stick with the 7 for now. As you have configured the whole
thing, close the menu, and edit the .config-file manually. This way you
can set the IRQ to 5, and everything goes as planned!

Just leaving like that didn't work for me, but I have a strange system that has a lot of stuff that doesn't work without some extra configuration. He also wrote that doing a **`cat /dev/sndstat`** will show all the information about your soundcard.

Mutant Soundcards

If you have a weird mutant soundcard from some obscure company, take a look at this text file written by Aaron Kuhn. It was too long to include here and convert to nice HTML so I thought I'd leave it in its unmodified glory. :-)

Questions, comments, suggestions? This is currently in a development stage of Beta, which means it's not guaranteed to work yet but I'm pretty sure it will work. If it works for you, please let me know. My e-mail address is jtg@computers.iwz.com.

Playing Audio CDs

Last updated: November 26, 1998

It's quite easy to set up Linux to play audio CDs for you: all you need is a CD-ROM drive, speakers/headphones, an audio CD, and a little luck.

First, you need to have your CD-ROM drive get recognized by Linux when you run **cdplay** or **cdp**. This is all in the manual pages, but anyway, here it is. Type:

```
cd /dev
ln -sf mcd0 cdrom
```

Above, type everything exactly **except** for "mcd0". That represents where your CD-ROM drive is. In my case, I would type this:

```
cd /dev
ln -s hdc cdrom
```

This creates a link from "hdc" in /dev to "cdrom" in /dev. It's pretty simple, actually. All you need to do now is run **cdplay** or **cdp**. CD Play only starts the audio CD, but you can use the following commands to control what it does concerning the audio CD in your CD-ROM drive. Here are some extensions of CD Play you can use to control how the audio plays.

- `cdplay play <track>`
- `cdplay stop`

These are the only features I find useful to myself in CD Play. A similar extension of the program, **cdp**, gives a "GUI" (sort of). You can change tracks, stop the CD, or eject it while running **cdp**. To use the following commands, make sure you enable Number Lock (that's what the manual page says; I don't have any problem with it, number lock on OR off).

This is **directly** from the `cdp` manual. You can access the rest of the manual page by typing `man cdp` at the console (the main Linux prompt).

```
       the '9' key on the keypad is "play"
       the '8' key on the keypad is "pause/resume"
       the '7' key on the keypad is "stop"
       the '6' key on the keypad is "next"
       the '5' key on the keypad is "replay"
       the  4  key on the keypad is "previous"
       the '3' key on the keypad is "go forward 15 seconds"
       the '2' key on the keypad is "hard abort" (music stops)
       the '1' key on the keypad is "go backward 15 seconds"
       the '0' key on the keypad is "soft exit" (music continues)
       the '.' key on the keypad is "help"
       the 'enter' key is edit current song.
```

TOC

```
        the 'a' key is edit artist name
        the 'c' key is edit CD name
```

You might get an error message when you're trying to play your CD as a non-root user about `/dev/cdrom` having to be changed to mode 666. At first I thought this was some sort of satanic error message, but it appears that there really is a mode for 666. It's basically so that you can play audio CDs as a non-root user, so you might want to go ahead and do that. However, note that users logged in can eject your CD-ROM drive when you change the mode to 666.

In the X windowing system, you can also use a graphical player like **xplaycd**.

Setting Up a Dialup PPP Connection

TOC

Last updated: June 3, 1999
Development stage: Beta

Setting up an Internet connection can get tricky, even frustrating, especially if you're a new Linux user. If your ISP is like most others, it will use Point-to-Point Protocol. I will first describe setting up and using **minicom** and **pppd** separately, and then using a chatscript. One can connect to the Internet with either of these methods; if one doesn't work for you, try the other. If both work for you, then you can choose whichever one is the most convenient.

If you want to try to get set up quickly, there's a web-based configuration script that will help you. It uses CGI scripts to create files for you to paste into your configuration files in `/etc/ppp/`. The URL is http://www.linux.net.nz/pppconfig/.

Lord Pyromage mentioned the **pppsetup** program, which comes with Slackware. According to him, it makes the PPP setup process much easier. So if you've got Slackware, you might want to give it a try if you don't want to read this page.

If I haven't scared you away by now, read on. Please be patient with this document. It's long, but reading it will, in all likelihood, make things work if you've tried other documents already. Note that your ISP must use PPP in order for this document to work for you.

Using Minicom with PPPd

TOC

First you should start out by configuring minicom (a terminal dialup program). At the Linux prompt, as root, type **`minicom -s`**.

That will load `minicom` in setup mode; after this, just set up the configurations: modem initialization strings (if you have a generic modem the initialization string you might want to try using would be `AT&D2`; consult your modem manual for the fancy options), baud rate, and so on. Make sure all this is done, and then choose 'Save as dfl' (saving it as the default) from the main minicom menu.

If minicom complains about not having /dev/modem, get out of the program and type **`ln -sf /dev/ttyS1 /dev/modem`**. If your modem was on COM2 under DOS or Windows, use that. Otherwise if you have COM1, for example, you'd use ttyS0 instead of ttyS1. The number after the letters is one less, since Linux starts at 0 instead of 1.

Stuff to Watch Out For

I would try watching out for what baud rate you use. Don't use too high of a baud rate, or you might be disconnected because of an unstable connection. Also, make sure the baud rate is high enough to use your modem at its highest speed. I put the baud rate up just one setting above my modem's fastest transfer speed. Your initialization string could possibly also have something to do with unstable connections. If you don't know what you're doing (like me), I would suggest that you use a simple initialization string like AT&D2.

DNS and Nameserver Configuration

At the Linux prompt again, type **`pico /etc/resolv.conf`** to enter the nameserver addresses.

Insert something similar to the following lines, replacing them to match your own:

```
search local.net
nameserver 205.136.28.2
```

A nameserver is a machine that most providers set up to translate the hostnames of Internet hosts into their IP addresses (for example, it would resolve www.local.net to 205.136.38.10). Many ISPs have more than one nameserver, so don't be confused if you receive two (or more) addresses when requesting information about your ISP's nameserver. You might want someone who is already using Linux to do a **dnsquery** on your ISP's domain. For example, if your ISP is Local Net (mine) and their front page is at **http://www.local.net**, then have whoever is doing the DNS query type this:

```
dnsquery local.net
```

Or:

```
dnsquery www.local.net
```

Again, I'm using my own ISP as an example. There will also probably be a secondary nameserver for your ISP, so you can use either one, or even better, both. The nameservers are the last lines in the dnsquery; make sure the person doing the **dnsquery** knows that. Yours will be different if you don't use Local Net as your provider. I want the IP address of ns2.local.net, not the hostname. ns2.local.net is 205.136.38.2 so I write that down somewhere.

Just add that "nameserver" line in `/etc/resolv.conf` followed by the IP address of the actual names (205.136.38.2) and complete the rest of the process. You can also call up your ISP through the phone and them the IP address of the nameserver, if you don't know anyone who is already on Linux or who can find out. If you already know what your ISP's nameserver is, then you didn't really need to read the previous few paragraphs on the nameserver.

As far as I know, you can use anybody's nameserver as long as you have the IP address (numerical, e.g. 205.226.156.2) for it. That means you can have your /etc/resolv.conf file look exactly like mine and it would still work.

PPP Options

Now you also should edit `/etc/ppp/options` by typing **`pico /etc/ppp/options`**. This is really important in starting point-to-point protocol (PPP). If you don't fill it in you'll have to specify the options every time you type **`pppd`**, and that's would be a big hassle. Insert the following lines into the file:

```
0.0.0.0:
/dev/ttyS1
lock
crtscts
defaultroute
asyncmap 0
mtu 576
mru 576
```

The only thing you might need to change is the device entry that tells which device to use for PPP. On my computer, with my external modem using COM2, it's `/dev/ttyS1`. You should know what COM port your modem is using, either from your experience as a DOS/Windows user or as a frequent Linux user (which you're probably not... for now).

Once you find that, you can refer to that device using `/dev/modem` by making a symbolic link from `/dev/ttySX` ('X' representing whatever that number is) to `/dev/modem`. Do this by typing the following:

```
ln -s /dev/ttyS1 /dev/modem
```

Replace `ttyS1` with whatever the device that you're using is, if necessary. The basic way to remember is that ttyS0 is actually COM1 under DOS, ttyS1 is COM2, ttyS2 is COM3, and so on; the number following "ttyS" is just one number less than the number following "COM" in DOS or Windows.

The `0.0.0.0:` should be put in the PPP options file if you have a dynamic IP address (your IP address/hostname is randomly assigned to you by your ISP and usually changes everytime you establish an Internet connection). If you have a static IP address (it remains the same each time you connect), I'm not sure if you can leave this out or put in the IP address that you're assigned. If anybody can clear this up for me, please let me know.

The mru and mtu lines are your receiving and transmitting packet sizes, I think. I heard someone on IRC ask how to lower the packet size so that a large download wouldn't get in the way of other accesses to the Internet going on at the same time. The speed you can transfer is still the same; it's just smaller packet sizes. A packet size of 576 is smaller than a packet size of 1024.

After all this is done, all you'll have to do is type **`minicom`** as root and then **`pppd`** as root. Minicom can be executed by all users, but the Point-to-Point Protocol Daemon (pppd) can only be executed by the superus (root). I tried changing permissions on the file `/usr/sbin/pppd`. If you ever accidentally change the permissions on the **pppd** file/executable, just type **`chmod 755 pppd`** from the /usr/sbin/ directory, as root. If you didn't mess with the permissions, you won't need to do the **`chmod`** stuff.

Using PAP or CHAP?

More and more ISPs are switching to PAP or CHAP authentication, which requires a little extra work on your part. Fortunately, this wasn't as hard as I thought it would be. If you're sure your ISP doesn't use PAP or CHAP, you can skip on to the next section. If you use PAP or CHAP, you won't be presented with a login prompt when you dial in with **minicom**, so that's how you know.

You need to fill in `/etc/ppp/pap-secrets` if your ISP uses PAP, or `/etc/ppp/chap-secrets` if your ISP uses CHAP. Those files should already be in there, but if they're not, create whichever one you need. All that you really need to do is fill in the chap-secrets (or pap-secrets) file like this:

```
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
dork            *       unpopular
```

That's where 'dork' is the username, and 'unpopular' is the password. The asterisk (*) can be left like that. The pap-secrets file also uses the same syntax, just a different filename.

In the PPP options file (/etc/ppp/options) it should look like what it was, except with one extra line.

```
0.0.0.0:
/dev/ttyS1
lock
crtscts
defaultroute
asyncmap 0
mtu 552
mru 552
name dork
```

That extra line is `name dork`, which tells pppd to use that username along with the chap-secrets (or pap-secrets) information when asked for authentication.

I use CHAP authentication for my new ISP but instead of just the username, they want the username@domain.net (pretty weird, huh?), so instead of just "name dork" in /etc/ppp/options I would have to put in "name dork@ynn.com". That also goes for /etc/ppp/chap-secrets. Instead of "dork" under "client" I had to put in "dork@ynn.com". If anyone has to do the same thing, let me know. Otherwise, filling in just the username should work fine.

Dialing In

Now, to establish the dialup connection, run `minicom`. When it loads, it should show the initialization string with the cursor at the end of the text string. Press enter and it should say "OK". Type `ATDT #phone#`, where "#phone#" is the phone number to dial. When the other end picks up, it should prompt you for a login name and password, which you should type in (unless you're using PAP or CHAP, and if you are, just use the Alt-q

combination and answer "Yes" right there). For some ISPs it will ask whether to use PPP or not, so answe
When it shows the PPP data (the stuff that looks like junk and garbage: *&*&*!^%#!^$!%%@#&^$%&
press Alt-Q and answer 'yes' to "Exit without reset?". That should take you back to the prompt, where you
should then type **pppd**. Typing **pppd** is **critical** to establishing the connection.

Possible Routing Problems

If your ISP uses a gateway (a computer that connects you to the rest of the Internet) that's a different machine than the one you connect to for dialing in, here's what you should do. You need to change the routing configuration on your machine so that you can actually be recognized as a machine on the Internet. The way to tell if you need to change the routing is if you type `ifconfig` as root, and it shows that you have an IP address through your modem, but it still doesn't work. The output of `ifconfig` looks something like this:

```
eth0      Link encap:Ethernet  HWaddr 00:40:05:48:2E:83
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2907 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3343 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0x300

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:4364 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4364 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

ppp0      Link encap:Point-to-Point Protocol
          inet addr:207.142.2.34  P-t-P:206.53.129.11  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:576  Metric:1
          RX packets:97 errors:0 dropped:0 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
```

If it shows the `ppp0` interface, but you can't get `ping` responses from any hosts, then there might be something wrong with your routing. The way to fix it? First, type `route` and it will display something like this:

```
Kernel IP routing table
Destination     Gateway     Genmask      Flags Metric Ref    Use Iface
127.0.0.0       *           255.0.0.0    U     0      0        3 lo
205.226.158.252 *           0.0.0.0      U     0      0        7 ppp0
```

As you might expect, most of the IP addresses will probably be different on your setup. The bottom route is incorrect on my system, so to change it, I type **`route del 205.226.158.252 ppp0`** to delete it. Then I type **`route add default ppp0`** to set the new route. Or, if you don't want to delete the old route, you can just set the new route, and it will work just as well. Not deleting the old route is probably safer.

Those are the only things I had to do. For some reason unknown to me, that pointed the gateway to the correct address and I successfully received **`ping`** responses.

Other Sources of Help

That should get your dialup connection up and running. If not, keep trying, or ask for help through any of the *linux* newsgroups or the #Linux* channels on IRC. (I'd recommend #LinuxHelp on the Undernet.)

As usual, I support my work fully, so e-mail me at jtg@computers.iwz.com. Please keep it short and simple, while providing the necessary information.

Using a Chatscript

This is a very convenient way to connect to the Internet. Once you get this set up, you can connect with a single command. However, there are things you should know first. Here are things to write down:

- What does your login prompt look like? For some ISPs it's **login:**; for others, it's **username:**. If you use PAP or CHAP, there will be none.
- What does your password prompt look like? Usually it's **password:** but fill whatever your ISP prompts you for if you want this to work. PAP and CHAP users don't get a password prompt.
- How many lines are in between, say, CONNECT 38400 and the first prompt? This is how many **\n**'s you will use.
- Are there any other prompts that are presented to you when you first log in?

The best thing to have done before this is to have made sure that you could dial in with minicom and use pppd to establish a connection. That would have ensured that it was possible for you to get connected. This method is a permanent solution so you won't have to go through the tedious process of typing several keystrokes and waiting to enter the next command.

Now, go on to making the actual script. As root, create a file to use as the actual chatscript (the text file tells the `chat` program how to communicate with your modem). I suggest giving it a name with the .chat file extension; for the sake of simplicity, let's just say that we'll use `yourispname.chat` as the filename. To edit the file, type **`pico yourispname.chat`**.

Now put in the following lines into yourispname.chat:

```
""AT&D2
OK ATDT#phone#
38400 \n
ogin: foobar
word: altm.415
```

In place of AT&D2 you can plug in the initialization string that you use. In place of #phone# you should put in the number that you want your modem to call, so for me it would look like `ATDT4755868`. The '38400' represents the speed you want to connect at and '\n' is telling `chat` to expect a new line. The part saying 'ogin: foobar' is telling `chat` to expect the text 'ogin:' to appear somewhere and respond to it by sending the word 'foobar'. The same applies to the next line. Don't put in the last two lines if you're using PAP or CHAP because the passwords aren't authenticated with plain text like that.

Basically you type in any prompts that are presented to you, and then type in what to send back. What is presented and what is sent back can be separated by a space in between.

TOC

If you dial in, watch for the number of lines between your baud rate (like 38400) and the **login:** prompt; i there's one line in between, use one \n character; if there are two, use two \n characters, and so on. Be sur watch out for this.

To prevent users other than root from reading from the file and finding out your password for logging in to your provider, you should change the permissions on the chatscript so that only root can read from it. I would do this by typing **`chmod go-rwx yourispname.chat`**.

Now, you should make another script -- a normal (shell) script. Move to `/usr/sbin` as root and create the script. Let's call it `yourispname`. Now we proceed to edit the file by typing `pico yourispname` in `/usr/sbin`. Keep in mind that you can change the filenames to whatever you want.

Put the following lines into the normal (shell) script, yourispname:

```
pppd connect 'chat -f /root/yourispname.chat' /dev/modem 38400 \
0.0.0.0: asyncmap 0 crtscts defaultroute
```

You might want to change the path to `yourispname.chat`, having it correspond to the directory that you placed `yourispname.chat`. After you put in this line, make the file executable and only to be accessed by root (the head honcho on the system) by typing `chmod 700 yourispname.chat`.

To connect, all you have to do is type the name of the file that you created. All this should be done as root.

Disconnecting

The way I disconnect is I type `killall pppd`, which kills the pppd program from running and therefore shuts down communication between your modem and your ISP.

Jas has suggested this shellscript:

```
#!/bin/sh
kill -TERM `cat /var/run/ppp0.pid`
```

That means you'd put that in a file using a text editor such as **pico** or **vim**, and save it as whatever filename you want (**ppp-down.sh** for Jas' example). Then when you're back at the prompt, type **`chmod +x ppp-down.sh`** so that it can be run (executed). I guess I should try this when I disconnect...

Modem Trouble

There might be all sorts of problems you encounter that are related to your modem, and that is a major obstacle for a lot of people.

PnP and Winmodem

Linux users who have plug n' play (PnP) modems or the notorius WinModem from US Robotics might have a lot of trouble in getting Linux to recognize that their modem is present. The best solution is to use another modem, make sure that it doesn't say "WinModem", "Plug n' Play", "PnP", "Exclusively for Windows", or anything similar.

Two ways you can try to get Linux to recognize PnP devices are messing with your BIOS (set the "PnP OS Installed" option to "off") and/or loading Windows 95 first, then using a DOS program called Loadlin to boot a

TOC

copy of your Linux kernel, usually found as the file called **`vmlinuz`** in your `/boot/` directory. The file, **LOADLIN.EXE**, is usually available from your Linux distribution media, be it FTP or CD-ROM.

IRQ Conflict

Your modem might try to be sharing the same means of communicating with the system with another device that has the same interrupt request, or IRQ. Symptoms of this include an abnormally slow connection, an error about timeouts sending config-requests, and a disconnect soon afterwards.

HydroFlow from IRC on the Undernet told me that all he had to do was type **`setserial /dev/modem irq 10`** to get it working. Of course, that's his system, so change the IRQ to fit your own system settings. Then the instructions seemed to have worked.

You can check for what devices have which IRQ settings by typing **cat /proc/interrupts**.

Finding Exact Init Strings

Since some of you might be wondering about how to find exact initialization strings for your modem, FusionGyro wrote:

I would like to offer a tip for you to possibly include in your FAQ file. For people coming to Linux from Windows 9x, there is a "simple" way to find all of the exact complicated modem strings. First, make sure you have logged on to your ISP since you booted. Then just fire up the control panel, open "Modems", highlight the correct modem, press "Properties", hit the "Connection" tab, press "Advanced", and then hit "View Log". (Gee, that wasn't so easy, was it ;). You should be staring at a text file. This file will contain everything that was sent and recieved until a PPP connection was made. Here's a cutout from the example he gave:

```
12-13-1998 02:56:07.36 - Cirrus Logic 33600 bps PnP V34 in use.
12-13-1998 02:56:07.36 - Modem type: Cirrus Logic 33600 bps PnP V34
12-13-1998 02:56:07.36 - Modem inf path: MDMTI.INF
12-13-1998 02:56:07.36 - Modem inf section: Modem4
12-13-1998 02:56:07.61 - 115200,N,8,1
12-13-1998 02:56:07.62 - 115200,N,8,1
12-13-1998 02:56:07.62 - Initializing modem.
12-13-1998 02:56:07.75 - Recv: <cr>
12-13-1998 02:56:08.22 - Interpreted response: Ok
12-13-1998 02:56:08.22 - Send: ATS7=60\T0L1M1\N3-J1%C1"H3\Q3B0N1X4<cr>
12-13-1998 02:56:27.67 - Recv: COMPRESSION:V42B
12-13-1998 02:56:27.67 - Interpreted response: Informative
12-13-1998 02:56:27.67 - Recv: <cr>
12-13-1998 02:56:27.67 - Interpreted response: Informative
12-13-1998 02:56:27.67 - Recv: <lf>
12-13-1998 02:56:27.67 - Error-control on.
12-13-1998 02:56:27.67 - Data compression on.
```

While much of it is useless, it does contain the important stuff: AT &F E0 V1 W4 &D2 &C1 S0=0 -C1, etc.

Poul Petersen wrote in about a site where you can look up information about your modem, including the init string. It's at http://www.spy.net/~dustin/modem/.

TOC

Author's Notes

When you're upgrading from, say, a generic 14.4 to a 33.6, and both are external, the only settings you need to change in minicom (or the chatscript) are the baud rates and the initialization string. The standard dial string (ATDT) should work for old modems as well as new ones.

No Pico?

If you don't have **`pico`** installed, try some other text editor (`emacs`, `vi`, `joe`, or `jed`) that you might be able to use to edit that file. Use that instead of "pico" wherever "pico" appears as a command. It doesn't matter what text editor you use.

TOC

Setting Up Ethernet

Created on November 9, 1997
Last modified: April 23, 1999
Development stage: Beta

One may want to use Ethernet for a number of reasons: for a home network, for a connection to the Internet, for internal use within your business, or whatever else you can think of that a network has use for.

Due to the nearly endless possibilities of an Ethernet network, I can't cover EVERYTHING here.

Here are the sections I've divided this page up into:

Configuring Your Card and Getting it Detected

The hardest part of getting your machine set up for Ethernet would probably be getting the card detected in the first place. You'll probably have to enable support for your network card by checking with your kernel configuration, or, more conveniently, recompiling your kernel with support for your Ethernet card enabled, as well as other stuff you might need in compiling your kernel.

My Ethernet card is labeled as NE2000-compatible; it's jumperless and I had to set the IRQ and all that other stuff in its diagnostic program under MS-DOS. That was found in the `A:\UTILITY` directory on the floppy, where the information was stored. Because it was jumperless, I had no option but to use the software to set the IRQ and I/O base address stuff there.

I can't predict all the possible configurations people might have on their individual systems, but the standard IRQ for Ethernet cards is 10 and the standard I/O base address is 0x300. If you don't know what these are, please don't hit the back button on your browser; you'll find out soon enough that this isn't really all that hard.

To enable support for your Ethernet card in your kernel, use `make config`, `make menuconfig`, or `make xconfig`. I recommend `make menuconfig` so that you can go back and fix your mistakes later. And yes, this is part of recompiling your kernel. NE2000 cards should be under "Other ISA Cards" or whatever; the only thing you need to enable it is choose the card that you need and nothing else. Choosing cards that you won't need really won't hurt you, but it *will* make your kernel a little bigger. But hey, if you want to play it safe and just include support for everything there, go ahead, I'm not stoppin' ya. Note that the more devices you support in your kernel, the larger it will be, and the more memory

(RAM) it will use up when loaded.

If you're not sure whether your card is supported, read the Ethernet-HOWTO. That has mostly hardware information about Ethernet cards. Look at the index of the HOWTO for the information you need.

With my card, I had to use boot parameters to specify which IRQ, I/O base address, and device that my card used. In order to do that, I had to tell the kernel what to look for (known as **passing arguments** to the kernel). Since I had my card set to work on IRQ 10, on I/O base address 0x300, and it was my only Ethernet card, I would stick these to the "linux" part when LILO popped up:

```
ether=10,0x300,eth0
```

Up there, I'm telling Linux that my **Ether**net device is on IRQ **10**, I/O base address is **0x300**, and the device created for the Ethernet card is **eth0**. Again, you should type that when you boot with LILO; when `LILO boot:` pops up, type "`linux ether=10,0x300,eth0`". When you boot up, it should read something like this, with perhaps a few changes depending on what settings you choose to use:

```
ne.c:v1.10 9/23/94 Donald Becker (becker@cesdis.gsfc.nasa.gov)
NE*000 ethercard probe at 0x300: 00 40 05 48 2e 83
eth0: NE2000 found at 0x300, using IRQ 10.
```

Assigning an IP Address to Your Machine

Once the card is detected, you have stuff to do. The first step, I think, would be to assign an IP address (a series of 4 numbers separated by dots representing a host or a group of hosts) to your machine. Let's just say that you're going to assign the address `192.168.0.1` to your machine. You would do this by typing the following:

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

The "netmask" part is pretty standard: it doesn't change if you're using Ethernet. The "up" at the end is simply telling Linux that you're setting your machine active, so it's up. And if you're not paying attention, the "eth0" part after "ifconfig" is representing the device that your Ethernet card uses.

Routing

I worried about issues like how the card would contact the hub (because I use RJ-45 cabling--you know, the kind that has connectors that look like giant phone jacks) before I tried setting up Ethernet for the first time. At first I tried to use Red Hat's `netcfg` tool that came bundled with the rest of the distribution; that got my IP address assigned to me correctly, but I couldn't get a ping from my sister's computer, a Windows 95 machine. For a while (to me it seems like a long while) I had to load Windows 95 first, and then use LOADLIN to load Linux in order to get a ping. I think on the same day I decided to start this section of my Linux guide, I read up a bit on routing.

As stated in the NET-3 HOWTO, it is possible to write large volumes about routing; I, however, don't like reading unnecessary information so I'll try to summarize it into what you really need to know. Let's

say you use a hub. You have an IP address of `192.168.0.1` and your younger sister has her comp
set to `192.168.0.2`. In order to get a ping from her machine (and not have to load Windows 95 i
you'll have to tell your machine that a hub is ready and waiting to have information passed through it

If the machines on your network have the IP addresses 192.168.0.*, then the entire network is represented by 192.168.0.0 (192.168.0.255 officially). Since 192.168.0.0 stands for the entire network (even if you're 192.168.0.3, 192.168.0.4, 192.186.0.5, and so on), you want to contact the entire network, which, again, is 192.168.0.0. You would do this using `route` as root:

```
route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

Basically, if your network has machines that are assigned IP addresses that are 192.168.0.*, you represent the whole network with 192.168.0.0. If your network's machines have IP addresses of 192.168.1.*, then your whole network would be represented by 192.168.1.0 and you would type `route add -net 192.168.1.0 netmask 255.255.255.0 eth0` in order to get your Linux machine to contact the rest of the network.

That should also work for networks with no hubs, if you connect the computers all to each other using those RJ-58 cables (the ones that look like TV wiring).

That will allow you to contact hosts within your network. Now you have to be able to contact hosts outside your own local network, especially those on the Internet. You'll have to configure a gateway, a machine that's connected to the Internet and that is willing to share its connection with you. For example, if the machine with the IP address 192.168.0.5 is your gateway to the Internet, you would type this (as root) to send requests for addresses that are out of your network:

```
route add default gw 192.168.0.5 eth0
```

If you are connected to the Internet through a dialup connection using PPP (which most people are), you would do something different, depending on what machine the Internet connection is running off of. If the machine you're working on is the machine that's connected to the Internet through PPP, then you would type something like this:

```
route add default ppp0
```

That will configure your system to send requests for packets that are not in your network to the Internet through your modem. (I hope you're still following along; if you don't understand I'm not doing a good job...)

You might want to use IP Masquerading, a way to share your Internet connection with others on your network. The way it basically works is when a request is sent from a machine within your network, your machine takes a look at it and if it's "addressed" to an Internet address, your machine sends the packet out through your connection, whether it be Ethernet or PPP.

If you've got a host within your LAN that's your gateway to the Internet, then you would use `route add default gw 192.168.0.5`, if 192.168.0.5 is the IP address of the gateway. When your connection is shared this way instead of IP masquerading, each machine on the network (known as a **node**) has it's own IP address on the Internet.

Putting IP Assignment and Routing in Startup

The information about routing and your IP address on your Ethernet network would be very inconvenient to type in every single time you want to be connected to the rest of the network, so we'll put it in your system startup files.

One easy way is to put the following example lines in the file `/etc/rc.d/rc.sysinit`:

```
route add -net 192.168.0.0 eth0
ifconfig eth0 192.168.0.1 up
```

I have that at the bottom of the file. The other files you can modify are `/etc/rc.d/rc.local` (if you want to do it this way for Slackware) and the files in `/etc/sysconfig/network-scripts/`. I think the correct way to do it would be to change or create the appropriate files in `/etc/sysconfig/network-scripts`, but I find it much easier to just put in the lines that I already have in `/etc/rc.d/rc.sysinit`.

Something nifty I found on Slackware was that you can just edit `/etc/rc.d/rc.inet1` (or the other rc.inet* file) and it will set it for you automatically. You specify the values of certain things like your IP address and broadcast address (and more, of course).

So if you're going to edit one of those files, just go ahead and use your favorite text editor (pico, vim, vi, emacs, joe, jed; the list goes on).

Windows 95 Configuration

On the Windows 95 machine, if you have one on the network, start by clicking on the Start button, then going to **Settings**, then **Control Panel**, and finally **Network**.

In the Network dialog box, if you haven't already, add a TCP/IP service for your Ethernet card, if you haven't already. Highlight TCP/IP for your Ethernet adapter, and click on the Properties button. Fill in the information that you need, such as the gateway IP address (the Linux box if it's running IP masquerading), the nameserver (your ISP's nameserver), the IP address assigned to your machine within your network (such as 192.168.0.2), and any other stuff you might have to fill in.

You can also run **winipcfg** to see if you currently have an IP address on the network.

That's about all I know about setting up the Windows box, so let's move on. Hopefully you have whatever other machines (nodes) on the network already set up.

Testing Your Connection

To test if your connection works, try pinging another host on your network. Let's say your machine is 192.168.0.1 and the other machine, the one you're going to ping, is **192.168.0.2**. This is what you'd type to start pinging the host:

```
ping 192.168.0.2
```

TOC

If lines periodically appear and look something like this, your connection works:

```
64 bytes from 192.168.0.2: icmp_seq=0 ttl=32 time=1.2 ms
64 bytes from 192.168.0.2: icmp_seq=1 ttl=32 time=1.0 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=32 time=1.0 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=32 time=1.0 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=32 time=1.1 ms
```

To stop pinging, type Control-C (Ctrl+C) to get back to the prompt.

Other Sources of Information

- Linux Ethernet-HOWTO - Hardware info
- Linux NET-3-HOWTO - Network configuration info
- IP Masquerading mini-HOWTO - Sharing your Internet connection with other computers within your network

Related Books

If you have trouble and are wondering how to solve your problem, either solve it yourself and give me a summary of your scenario, or alternatively, ask for help. And, as usual, suggestions are always welcome.

Setting Up IP Masquerading

Author: jbm <jbm@intertek.net>
Created on: July 19, 1998
Last Modified: March 2, 1999
Status: Beta

Introduction

So - you've got your Linux up and running and you can use the LAN between Linux and Windows 95. Good for you. But you still can't get online with your Windows 95 box and your Linux at the same time. This is why they developed IP Masquerading. To begin with, you need a few things: the ability to compile your own kernel (see Compiling a New Kernel for more info on this), a working subnetwork (probably Ethernet - see Setting Up Ethernet for more info. There are other ways to create a subnetwork, but if you can get those working, you probably don't need this guide ;^), and a way to get at the internet while on your Linux (either Ethernet or dial-up, there are special instructions for using two NICs at once, see sunsite.unc.edu:/pub/Linux/docs/HOWTO/Ethernet-HOWTO). A discussion of networking is beyond the scope of this document, as is setting up dial-up connections (discussed in Setting Up an internet Connection.), so if you don't have those working yet, go ye forth and fix ye thee. If you're still with me, go grab the IP-Masquerading mini-howto, from your local sunsite mirror:Linux/docs/HOWTO/mini/IP-Masquerade or at sunsite.unc.edu:/pub/Linux/docs/HOWTO/mini/IP-Masquerade really quick. It's a more in-depth discussion of what this document covers. Also, http://ipmasq.cjb.net/ is the official homepage of Linux IP Masquerading. hint hint.

Before You Begin...

I'm no expert when it comes to this. I just got my setup working well enough, and I saw a definite need for a document like this one. The IP-Masquerading mini howto is too in-depth for the average Windows 95 --> Linux --> Internet setup. If you can add anything to this - please do! I'm currently working on on-demand dialup that's transparent to Windows 95. Any info that you need that's not covered here will most likely be found at http://ipmasq.cjb.net/. hint hint.

This document is based on my personal setup - a Windows 95 box connected via eth0 to a Linux box which is connected to the internet by ppp0. I use Slackware, with kernel 2.0.34. I'll try and make everything usable under RedHat, but I can't make any guarantees. I take no responsibility if this document messes up your boxen. Or causes your dog to shed all over the couch. Feel free to mail me(jbm@intertek.net), but please only send me questions dealing with IP masquerading and/or this document (misspellings, etc). Please no questions about setting up PPP or Ethernet.

Begin

Make sure your ethernet works ('ping' back and forth), make sure that your PPP dialup works ('ping' somebody on the net), and make sure they both work at once ('ping' back and forth locally and some internet site while online). If this is all ok, move on. If not, you need to fix it before you proceed. Check the related docs on this site, then try re-doing things (if you need to recompile your kernel, **don't** inc the IP Masq changes. It's best to change one or two things at once, so you can find exactly what's no working. After you get ethernet and PPP working side-by-side you can try to get them working hand-in-hand.)

Setting Up Linux

To get Linux ready for IP Masquerade, you only need to do three things:

- Remake your kernel
- Set up `/etc/<rc>/rc.modules` (more on this later)
- Set up `ipfwadm` in the Right Place (rc.local)

Kernel Stuff:

If you're not comfortable recompiling your kernel, stop now. You really need to be able to do this to be a Linux user, so go learn how to at Compiling a New Kernel. Go through and configure your kernel for all the things you normally need (*modules*, filesystems, SCSI if you need it, PPP/SLIP, networking, etc), and then add the following things (in older kernels you may need enable experimental things):

- Networking Support (CONFIG_NET) (required)
- Network Firewalls (CONFIG_FIREWALLS) (required)
- TCP/IP Networking (CONFIG_INET) (required)
- IP: Forwarding/Gatewaying (CONFIG_IP_FORWARD) (required)
- IP: Masquerading (CONFIG_IP_MASQUERADE) (may be experimental) (required)
- IP: ipautofw (CONFIG_IP_MASQUERADE_IPAUTOFW) (may be experimental) (recommended)
- IP: ICMP masquerading (CONFIG_IP_MASQUERADE_ICMP) (optional, i use it ;^)
- IP: always defragment (CONFIG_IP_ALWAYS_DEFRAG) (highly recommended)
- Dummy Net Driver Support (CONFIG_DUMMY) (recommended)

Now do the whole kernel building process... `make dep; make clean; make zImage (go watch tv); make modules; make modules_install`. The modules part is required because certain protocals (ftp, irc, realaudio to name just a few) need special configuration to work correctly through masquerade.

rc.modules Fun:

For this, you need to edit your rc.modules file - `/etc/rc.d/rc.modules` in Slackware and `/etc/rc.d/rc.local` in Red Hat - and add the following lines:

```
depmod -a  #if there's already a line containing this, don't add it.
/sbin/modprobe ip_masq_ftp
/sbin/modprobe ip_masq_raudio
/sbin/modprobe ip_masq_irc
```

```
/sbin/modprobe ip_masq_cuseeme
/sbin/modprobe ip_masq_vdolive
```

And any other modules you see in `/lib/modules/2.0.xx/ipv4` that start with ip_masq. According to the mini-howto, `kerneld` won't work. Sorry to those of you who use it.

'ipfwadm':

Note: if you are using a 2.2.x series kernel (or late 2.1.xx), you need to use IP chains, see below for more details. Skip this section and go on to the next.

You need to stick

```
ipfwadm -F -p deny
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

in your /etc/<rc.d>rc.local file, it only needs to be run once (i was mis-informed at last writing. My apologies). Due to the nature of this file, these lines won't automatically be executed until you reboot. You can, however, just paste these into the commandline using `gpm` and set it up on a running system.

This should complete the Linux side of the setup.

IP Chains

IP chains is the "new" way to set IP masq things up. If you are using a 2.0.xx series kernel, you don't need to worry about it just yet; if you are using 2.2.x, however, you do.

The use is just like for '`ipfwadm`', except you place

```
ipchains -P forward DENY
ipchains -A forward -j MASQ -s 192.168.0.0/24 -d 0.0.0.0/0
```

in your /etc/<rc.d>rc.local file. Due to the nature of this file, these lines won't automatically be executed until you reboot. You can, however, just paste these into the commandline using `gpm` and set it up on a running system.

Configuring Windows 95

This is by far easier. If you've got the ethernet adaptor installed right, just open up Start->Settings->Control Panel, then go to Networking. Open up TCP/IP -><name of your ethernet adaptor>. Go to the Gateway Tab, and enter the Subnet IP address of your Linux (probably 192.168.1.1). Add the appropriate settings under the DNS Configuration tab. You don't need the suffix search thing, but it's kinda nice. Click OK through all the dialogs and restart Windows. This should be all you need to do.

This should complete the Windows side of the configuration.

TOC

Setting Up Other OSs

See the IP Masquerade mini-HOWTO for instructions on setting up other OSs capable of TCP/IP networking (or UDP/IP. but i think UDP is more complicated setup...).

Test it

Well.. that should be it. Try it out - reboot your Linux box, start up your PPP connection, run the ipfwadm script (if you need one), and trying getting onto the net with Windows. If it doesn't work, make sure you ran the `ipfwadm` stuff after you connected with PPP (ie - after you actually got an IP address assigned).If that doesn't fix things, try going through the IP Masquerade mini-howto. It'smuch more in-depth and thorough, so your problem will likely be addressed there.

PPP Stops Working After You Install IP Masquerade

This confused me very much so. If you compiled PPP as a module, make sure you do `/sbin/modprobe slhc.o` before `/sbin/modprobe ppp.o`. Try doing `depmod -e ppp` to see what error messages your kernel is having problems with. I personally recommend compiling PPP into the kernel, as it's used fairly often. If that looks ok, try recompiling it, after printing out the configuration information above and double check all your settings. If it's still broken, triple check your settings. If it still doesn't work, try setting up PPP by itself. If that's broken, see Setting Up an Internet Connection. After you get that working, try the IP masq setup again. This should solve most problems.

Thanks to Tom M. Schenkenberg for pointing out the new ip-masq site, and keeping me from getting dead link complaints =).

TOC

Setting Up a Mail Server

Last updated: April 21, 1999
Development stage: Beta

A mail server is a server that is running one or more of the following: an IMAP server, a POP3 server, a POP2 server, and an SMTP server. For now, I'm only going to cover installing IMAP4, POP3, and POP2, which all come in a single package. A few examples of SMTP servers are Sendmail (widely used) and QMail (more secure and configurable). Sendmail is installed on many Linux distributions by default so it might not be necessary at all to install it.

You may already have this installed on your system. As root, type `which ipop3d` and if it shows up, you don't need to read on unless you want to reinstall or perhaps to upgrade.

Download a file at ftp.cac.washington.edu called **`imap-4.1.BETA.tar.Z`**. It's about 1.3 megabytes in size, so if you want to back it up on a floppy disk later on, go ahead. Remember also, the version might change, so just look for a file in `imap*.Z` format. Also remember to set the transfer mode in FTP to binary so that you don't download the entire file as plain text. Do that by typing `type binary` or `type image` when you're logged into the FTP server. Also, downloading the file from your web browser will probably automatically transfer it in binary format.

Now, I'm assuming you've downloaded that archive already. If you haven't already, place it in `/usr/local/src/` (or any directory you want, actually... but let me keep it simple) and extract it using **`tar -zxvf imap-4.1.BETA.tar.Z`**.

You should know by now that **imap-4.1.BETA.tar.Z** is only the name that I use as an example. There probably will be different and newer releases for that. Move into the directory it has extracted; in my case, that would be `imap-4.1.BETA`. Instead of just **`make`** from there on, you have to use **make lnx** to tell it how to set up the compiling and file formats. The "lnx" stands for Linux, and if you're using some other Unix implementation, read "Makefile" to find out which abbreviation you should use. If you're using shadow passwords on your server, you'll probably have to use `make slx`. Thanks to Charles Roth for figuring out and sharing this information.

Once you run **`make lnx`** from the `imap-4.1.BETA` directory (and yes, it will change from time to time), all the binaries should be compiled already and copied to `/usr/sbin`. The files it should have produced are **imapd, ipop3d, and ipop2d**. Those files should also be in the subdirectories, `imapd` and `ipopd` after imap-4.1.BETA. Do 'which imapd' and a 'which ipop3d' and see if they're in `/usr/sbin/`. Remember to do this as root if you plan to use the machine as a mail server. :)

This should already be present, but if it's not, put it in. Make sure that somewhere in `/etc/inetd.conf`, there are lines that read:

```
pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop2d
pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
```

TOC

Common Problems

Here's a common error that people face: when someone sends a message to a user on your machine, the message is sent back to the sender and a copy of it is given to `root` also. This is what the error message looks like:

```
   ----- The following addresses had permanent fatal errors -----
<jgo@d67.local.net>

   ----- Transcript of session follows -----
553 d67.local.net. config error: mail loops back to me (MX problem?)
554 <jgo@d67.local.net>... Local configuration error
```

This was actually a `sendmail` configuration problem. I found out on IRC that I had to add a line to `/etc/sendmail.cf` in this format: **`Cw<host.name.net>`**. Even though I have a dynamic (constantly changing/randomly assigned) IP address, it works. I just entered the whole range of my possible IP addresses at the end of `/etc/sendmail.cf` (you can put this anywhere in sendmail.cf, but I just decided to put it at the end of the file because of this habit that I have...). This is what the end of my `/etc/sendmail.cf` file looks like:

```
Cwd50.local.net
Cwd51.local.net
Cwd52.local.net
Cwd53.local.net
Cwd54.local.net
```

... and so on. With those entries, I can accept incoming mail when I'm d50.local.net through d54.local.net. You can also just put in all your IP addresses and hostnames in `/etc/sendmail.cw`. I think that putting them in `sendmail.cw` in `/etc/` is much easier, not to mention more convenient, than putting it in `sendmail.cf`, since `sendmail.cf` is the main Sendmail configuration file, which should be kept free of clogging up.

More information on Sendmail can be accessed at http://www.sendmail.org.

Related Books

1. Sendmail - the most comprehensive Sendmail book out there.
2. Sendmail: Theory and Practice - a recommended companion to the "Bat book" (mentioned above).

TOC

Setting Up a Nameserver

Created on July 24, 1998
Last updated: January 23, 1999
Development stage: Alpha

A nameserver is a server that resolves hostnames to IP addresses. Instead of having to type in "209.81.10.250", I can just type in "www.penguincomputing.com" to get to a site. BIND is nameserver software that runs on many types of machines, originally written by Paul Vixie and now maintained by the Internet Software Consortium (ISC).

This was one of the trickiest things I've had to try to figure out so far. I didn't talk to anybody as to how to go about this, unlike PPP setup. But I finally found the Linux DNS HOWTO and used that. Hopefully this will save you some trouble. Here I have instructions for BIND 8, which is newer and more advanced than BIND 4 (which some distributions still use).

Removing Old Nameserver (BIND 4)

In the event that your distribution already comes with BIND 8, then all you need to do is find out how the configuration works, and/or put in entries for machines that the Linux box will be the nameserver for. I think Slackware comes with BIND 8. I don't know about Debian. Most new distributions (including Red Hat 5.2) will already have BIND 8 so you'll probably have to look for the configuration file and how it has been pre-configured.

I had to remove the old nameserver first, so it wouldn't get in the way of the new one. It's fine to remove it and replace it with a new one; there aren't any other packages that *really* depend on it.

Using Red Hat 5.1, I typed **`rpm -e bind bind-utils caching-nameserver`**. That removed all the old packages. Be careful about this, especially about bind-utils, because bind-utils contains the tools such as **dnsquery** and **nslookup** that you might be using fairly often. Red Hat 5.2 already has BIND 8.

Installing New Nameserver (BIND 8)

First, download BIND from ftp.isc.org. If you've got it on your system already, then you don't need to get BIND 8 unless you want to make a minor upgrade. The filename is something like bind-8.1.2-src.tar.gz, which says that it's BIND version 8.1.2 in source format (which you have to compile on your system). I'll work with the source version since that's what I usually do anyway.

After you have it on your system, type **`tar -zxvf bind-8.1.2.tar.gz`**. It will extract a directory called `src/` and that's where you go into. Simply type "make" and have it compiled for you. If you need anything to be tweaked then read the INSTALL file (**`less INSTALL`**) and find what you need. After it finishes compiling, type **`make install`**.

Configuration: /etc/named.conf

Configuring the nameserver was probably the hardest part of the process that I had to go through. Hopefully, what I've written up about what I learned will save the reader some trouble.

The main BIND 8 configuration file is in `/etc/named.conf`. The configuration syntax for the file is documented at http://www.isc.org/bind8/config.html. Here's a sample of a configuration file (as /etc/named.conf), with an explanation below.

```
/*
 * A simple BIND 8 configuration
 */

options {
        directory "/var/named";
};

zone "penguincomputing.com" in {
        type master;
        file "master/penguincomputing.com";
};

zone "0.0.127.in-addr.arpa" in {
        type master;
        file "zone/127.0.0";
};

zone "." in {
        type hint;
        file "named.cache";
};
```

In "options" I only had one option: where the directory for **zone files** were. Zone files are where information about domains is stored, and each file has information about a zone. It's a section to cover, I guess, so that's why they're called zones. I have `/var/named/` as my named data directory.

The "penguincomputing.com" section is pretty straightforward. It just indicates the location of the penguincomputing.com zone files and tells **named** that this server is a master nameserver for the penguincomputing.com zone.

The "0.0.127.in-addr.arpa" zone is for mapping localhost to 127.0.0.1, basically. It has its own zone file.

The "." zone indicates a caching nameserver; that is, someone can actually use your machine to resolve hostnames (including you). I've heard that is is efficient especially when using PPP connections, but I don't know for sure. Read the "Caching Nameserver" section to read up on how to create one.

Caching Nameserver

First you need to get a "named.cache" file. I'm not sure if you can name it anything else, but let's just use that filename. In /var/named/ (or wherever you put your nameserver's data files), type **`dig @a.root-server.net > named.cache`**. This will ask for the addresses of the main DNS servers of the Internet and direct them to a file. I'm *guessing* that the purpose of this is to give your machine an idea of which machines on the Internet to ask about hosts.

Periodically, like once a month, update the named.cache file by running that command once in a while. You can use a cron job for that. If you don't know what I'm talking about here, don't worry about it. Just be sure to update it using **dig** once in a while, that's all you have to do.

TOC

You have `/etc/named.conf` point to wherever your named.cache file is under the "." zone.

Zone Files in /var/named/

In `/var/named/`, I created directories for every type of zone I had. The directories I have in there are: `master`, `slave/`, and `zone`. With the domain name system, there is a server for each domain that is the main server (the master). I suppose that the slave server is there in case the main (master) server is down. For each domain there should be *at least* 2 servers, one master and one slave. That's just the way it goes.

While interning at Penguin Computing I set up both the master and slave DNS servers. The master's information should go in the `master` directory. You should be able to figure out where the slave's information goes. The information they store is the same, but since one machine is the main one that keeps the information (master) and the other simply follows the master's information (slave), you need to stay organized and make sure you're configuring the right machine for its right place in the nameserver system.

Note that the slave nameserver for one domain can also be the master nameserver for another domain. There just can't be two masters for a single domain, though I think there can be several slaves.

Examples of Zone Files

To figure something like this out, I was looking hard for examples. And examples really help, so hopefully you won't be too confused by my examples. Hey, I try.

Domains

The information for each domain is put in a single file. This file contains valuable information for each domain, such as machines that are under that domain (like, for the penguincomputing.com domain, the nameservers would have the information for what IP address pasta.penguincomputing.com gets and the one that antarctica.penguincomputing.com gets). Here's an example of a domain's records:

```
@       IN      SOA     penguincomputing.com.   root.penguincomputing.com.
(
                        1998082403      ;       serial
                        4H              ;       refresh, seconds
                        2H              ;       retry, seconds
                        1W              ;       expire, seconds
                        1D      )       ;       minimum, seconds
                NS      pasta.penguincomputing.com.
                NS      rice.penguincomputing.com.
                MX      10 penguincomputing.com. ;  Primary Mail Exchanger

localhost       A       127.0.0.1
router          A       140.174.204.2

penguincomputing.com.   A       209.81.10.250
ns              A       209.81.10.250
www             A       209.81.10.250
ftp             CNAME   penguincomputing.com.
mail            CNAME   penguincomputing.com.
news            CNAME   penguincomputing.com.
pasta           CNAME   penguincomputing.com.
slashdot        CNAME   penguincomputing.com.
rice            CNAME   antarctica.penguincomputing.com.
antarctica      A       209.81.10.252
antarctic       CNAME   antarctica.penguincomputing.com.
```

```
www.antarctic   CNAME   antarctica.penguincomputing.com.
www.antarctica  CNAME   antarctica.penguincomputing.com.
zork            A       209.81.10.253
tux             A       209.81.10.146
xfce            A       209.81.10.252

@               TXT     "Penguin Computing"
@               HINFO   Linux 2.0.34
```

There's a pretty weird syntax to be used for these zone files. I never would have figured it out on my own had I not read the Linux DNS HOWTO document. Basically, it specifies information about all the machines in the domain, and it contains information about the domain itself, such as the type of machine the server is running on.

I'll start explaining what all the stuff does. In the first line, it's saying that this file specifies the zones for the penguincomputing.com domain, and to send anything about the domain to *root@penguincomputing.com*. Since the "@" character has special significance in these **zone files**, the username (root) and machine name (penguincomputing.com) have to be separated by a dot. I guess BIND just knows how to split it up. That's how you fill in stuff for your domain as well.

The line with the comment "serial" shows the serial number of that domain. The syntax is YYYYMMDDRR; that is, a four digit year, two digit month in numerical form, two digit day format, and a two digit revision number. In this example (1998082403), it shows that the zone file was last modified on August 24, 1998. It's the third revision for that day. When you're changing anything in the file, make sure to increase the revision number by one if the previous change was on the same day. If I were to change the IP of one of the hosts, I would make the last two numbers, currently 03, to 04.

The next few lines show times for certain functions such as refreshing, retrying, and expiring the information. I'm not *absolutely* sure, but my best guess is that H stands for hour, D stands for day, and W stands for week.

The "NS" line indicates all the nameservers for that particular domain, including the one this information is on. This information has to match what has been registered with InterNIC. For the hostnames of the nameservers, remember to add a dot at the end. If you don't, it will add the hostname to the current domain. For example, if you forgot the dot at the end of pasta.penguincomputing.com, you would end up with the nameserver being pasta.penguincomputing.com.penguincomputing.com, which is obviously not what it's supposed to be. Watch out for this.

The MX file is the *M*ail e*X*change record, so that mail can get through to the domain. There should also be an entry in /etc/sendmail.cw to allow messages coming in from that domain (assuming you're using Sendmail, the default on many Linux systems, for mail transfer).

The next couple of lines point to the local loopback (127.0.0.1), which all Linux systems should have even if they aren't connected to a network. The "router" line points to the IP address of where the machine's Internet connection is. I'm not sure if it's really necessary but I was playing it safe back then and trying to copy the example from the DNS HOWTO as closely as possible.

The rest of the entries use either A (address) or CNAME (Canonical Name) to point hostnames to IP addresses. Note that hostnames can be mapped to other hostnames, or they can be mapped to IP addresses. Use A to map a name to an IP address, and CNAME to map a hostname to another hostname (which must be mapped to another IP address).

Localhost

The file for mapping localhost is pretty simple. Not much explanation needed. Of course, if you want to copy and paste, be sure you make the proper changes.

```
@       IN      SOA     penguincomputing.com root.penguincomputing.com (
                        1998072401      ;       Serial number
                        3H              ;       Refresh
                        1H              ;       Retry
                        604800          ;       Expire
                        86400)          ;       Minimum TTL
```

```
                NS      pasta.penguincomputing.com.
                NS      rice.penguincomputing.com.

1               PTR     localhost.
```

Reverse Mapping

This file looks similar to the zone file for the domains, but it provides the opposite function. It points IP addresses to hostnames (as opposed to vice versa), because many servers on the Internet do this thing called reverse lookup on the IP address of your hostname to make sure that you're not doing anything sneaky.

This is for the zone "209.81.10" specified in the sample configuration file. Note that my example is not complete, nor does it work in reality, because Penguin Computing doesn't own the whole block of "209.81.10.*". But this is how you'd fill in a file to resolve your IP addresses to hostnames *if* you owned the entire block of IP addresses.

```
@               IN      SOA     penguincomputing.com.   root.penguincomputing.com. (
                                1998072002       ; Serial
                                4H      ; Refresh
                                2H      ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      pasta.penguincomputing.com.
                        NS      rice.penguincomputing.com.
;
;       Servers
;
250     PTR     pasta.penguincomputing.com.
250     PTR     penguincomputing.com.
250     PTR     ftp.penguincomputing.com.
250     PTR     www.penguincomputing.com.
250     PTR     mail.penguincomputing.com.
251     PTR     rice.penguincomputing.com.

;
;       Workstations
;
252     PTR     antarctica.penguincomputing.com.
252     PTR     antarctic.penguincomputing.com.
```

If you were to fill in an actual zone file like this, it's necessary to fill in *all* the entries in your block of IP addresses, from 1 to 255. For something like that you may want to assign the task to anyone who looks bored.

So what should you do if you only own a domain but not the block of IP addresses that it's part of? Ask the people who are in charge of that block of IP addresses to map your IP addresses to their respective hostnames for you.

Resources

The following is a list of resources I referred to in order to help me set up a nameserver for myself and write this document.

1. Linux DNS HOWTO
2. BIND Configuration File Syntax

Comments, questions, suggestions? Send them to jtg@computers.iwz.com.

TOC

Setting Up a Webserver

Last updated: February 5, 1999
Development stage: Alpha

The World Wide Web is probably the primary reason of the Internet's explosive growth as a popular communications medium. So why not add to the wealth by running your own web server? The most popular webserver on the Internet is Apache, which already comes with many Linux distributions.

Chances are that you already have Apache installed, so you may skip the downloading process. Of course, downloading and compiling a new Apache gives you the advantage of knowing how to do it. The process is quite simple, even the configuration.

This document is in alpha status, so let me know if it works for you so I can update its status. That way I'll know if it works.

Removing Old Apache

Should you choose to remove any existing Apache, try typing **`rpm -e apache`**, if you're using Red Hat (or maybe another RPM-based distribution).

Let me know how to remove the default Apache package for other distributions so I may include them here. I don't want to look ignorant for a *long* time. :-)

Download

Apache is available all around the world in many FTP sites. There's a mirror list at the Apache website. As of now, Apache is at version 1.3.3, and at the rate it constantly develops, this document will probably fall behind when a new release comes out.

You can find a list of where you can download Apache at http://www.apache.org/dyn/closer.cgi. The filename for the file you're looking for goes something like "apache_1.3.3.tar.gz".

TOC

Compile and Install

You have downloaded Apache. If it's not already, put it in `/usr/local/src/` and untar (**`tar -zxvf <apache_1.3.3.tar.gz>`**) it. It should extract to some directory called "apache-1.3.3" or similar, depending on what version number you've downloaded. Now move into the apache directory by typing **`cd apache-1.1.3`** (or whatever version you've got).

Basically you can get Apache compiled by typing **`./configure ; make`** in the Apache directory. **`make install`** will install it to /usr/local/apache by default. In /usr/local/apache/ there will be many subdirectories containing various files that you might want to tinker with.

At the least, you should make a symbolic link from `/usr/sbin/httpd` to `/usr/local/apache/bin/httpd` (assuming you installed Apache to the default location). Or, you can add `/usr/local/apache/bin/` to root or user paths from `~root/.bashrc` and `/etc/profile`, respectively.

Configuration

Since you've now got Apache installed and ready to go, it's time to configure it. Move on down to `/usr/local/etc/httpd/conf/` and copy all the "*.conf.default" files to their "*.conf" selves. That means you would copy `httpd.conf.default` to `httpd.conf`, `srm.conf.default` to `srm.conf`, and `access.conf.default` to `access.conf`. Those are all the files that you need to copy. Edit "httpd.conf" with your favorite text editor and uncomment the ServerName line. Put in your hostname or IP address in place of the fake one in the commented line, which probably changes each time you log on. For ServerAdmin, put in *your* e-mail address so that people visiting your webserver can e-mail you about any problems that arise. What you fill in is pretty straightforward.

Reading the comments and examples in the *.conf files for Apache will help you if you need to do any "advanced" tasks such as virtual domains or extra MIME types (if you don't know what these are, don't worry about them yet).

Location of Configuration Files on Distributions

If you're using Red Hat Linux, you should have the configuration files in `/etc/httpd/conf/`, if you had Apache installed already.

On SuSE 5.2, 5.3 and 6.0 the config files for Apache are located in `/etc/httpd/`. Thanks to Uwe Hermann (uh1763@bingo.baynet.de) for this information.

For other distributions, I'm not sure. Please let me know if you can provide this information. You'll probably want to look in the `/etc/` directory in the meantime.

Virtual Hosts

Many people need to run multiple web servers on one machine instead of one webserver per Linux machine. First you'll have to get a hostname assigned to the webserver, which can be a subdomain (jgo.local.net if you're on the local.net domain) or a whole new domain registered through InterNIC (such as dummy.com).

Once you get past all the domain registration stuff and you've got many hostnames assigned to the same IP address, or if the machine is running under multiple IP addresses and multiple hostnames, you're ready to set up for virtual hosts.

TOC

Start out by editing `/usr/local/apache/conf/httpd.conf`. If you have Red Hat Linux with Ap installed already, the file is `/etc/httpd/conf/httpd.conf`. You'll see an example shown, looking, like this:

```
# VirtualHost: Allows the daemon to respond to requests for more than one
# server address, if your server machine is configured to accept IP packets
# for multiple addresses. This can be accomplished with the ifconfig
# alias flag, or through kernel patches like VIF.

# Any httpd.conf or srm.conf directive may go into a VirtualHost command.
# See alto the BindAddress entry.

#<VirtualHost host.some_domain.com>
#ServerAdmin webmaster@host.some_domain.com
#DocumentRoot /www/docs/host.some_domain.com
#ServerName host.some_domain.com
#ErrorLog logs/host.some_domain.com-error_log
#TransferLog logs/host.some_domain.com-access_log
#<VirtualHost>
```

That's commented out because your machine most probably does not own the hostname `host.some_domain.com` and therefore won't accept requests for it.

The basic way that this works is when someone requests a page on, say, jgo.local.net, it won't go to squeeze.local.net, the original hostname of the machine. Although jgo.local.net *is* an alias for squeeze.local.net, somebody asking for http://jgo.local.net won't get the front page to http://squeeze.local.net because it's specified in `httpd.conf` to serve out `/home/jgo/index.html` when a request for jgo.local.net is given through HTTP. When a request for 205.136.38.27 is given, however, it won't serve out `/home/jgo/index.html` because only the requests for http://jgo.local.net will give the web surfer `/home/jgo/index.html`. Some older browsers don't support virtual hosts, such as MS Internet Explorer 2.0, which came with early versions of Windows 95.

You can assign as many virtual hosts to your machine as you want, especially if the machine is a top-level domain such as local.net. That way you can assign the hostnames jgo.local.net, something.local.net, one.local.net, two.local.net, three.local.net, and anything else you might be able to think of as a hostname. Just keep in mind that you should make sure the server can handle the load of all those domains.

To add subdomains, you need to have an understanding of BIND, the nameserver. Your domain must be registered with InterNIC, and the nameserver is in charge of getting subdomains set up. It sounds complicated, but it makes a whole lot more sense once you figure it out.

Setup of /etc/hosts

My observations about the setup of /etc/hosts is that each hostname has to be part of a higher-level domain; for example, if I wanted to name my machine `neophobia`, I'd either have to make the higher-level domain "localdomain" or "dyn.ml.org". Here's what each entry would look like:

```
127.0.0.1       localhost       localhost.localdomain
```

and:

```
0.0.0.0         neophobia       neophobia.dyn.ml.org
```

People with dynamic IP addresses (most dialup PPP users) can just use the IP address "0.0.0.0" to represent their Internet IP. So, this is what my hosts file looks like:

```
127.0.0.1       localhost       localhost.localdomain
0.0.0.0         neophobia       neophobia.dyn.ml.org
```

After you're done with all that you wanted to do, try to start httpd by typing **`httpd`** as root. It is common and normal for you to have some errors, so feel free to e-mail me concerning any other errors that come up after you execute the httpd binary.

Error Messages

When you have **httpd** already running, the error message it gives will be `bind: Address already in use, httpd: could not bind to port 80` or something similar. To fix this, do `killall httpd` and then `httpd` to restart it. Make sure you bind it to the correct hostname/IP address (read the paragraphs above).

Creating Content

By default you'll have to edit the HTML files in `/usr/local/apache/htdocs/`. The directory for CGI scripts is in `/usr/local/apache/cgi-bin/`.

On a default Red Hat installation you'll probably find what you need to edit in `/home/httpd/html/`, along with some directories. The CGI-bin will be in `/home/httpd/cgi-bin/`.

The htdocs-directory (DocumentRoot) is `/usr/local/httpd/htdocs/` on SuSE 5.2, 5.3, and 6.0, according to Uwe Hermann (uh1763@bingo.baynet.de).

Let me know of the filesystem layout for other distributions. I don't have immediate access to other distributions.

Author's Notes

There are more complex things you can do with Apache, and here I only cover the basics. There are special modules that come with Apache which I have little experience with, but I just want to let you know that they're available. Read the INSTALL file for instructions on this.

Questions, comments, and/or suggestions? You can e-mail me at jtg@computers.iwz.com.

Installing an IRC Client

TOC

Author: Joshua Go
Created on: March 28, 1998
Last modified: February 6, 1999
Status: Beta

Just in case you don't know, an IRC client is a program that lets you access Internet Relay Chat using your machine. You connect to a server and then chat with other people. It's quite popular among Linux users.

Many distributions come with IRC clients, but they're usually ircII, which isn't that great. You should get ircII-EPIC, a better client, which is what this document covers. You will need to download the source and compile it yourself; if you're not comfortable with this concept, that's fine.

If you think this document sucks, check out The Official EPIC Homepage; it has some good resources. If this doc is very old at all, check out the EPIC homepage for any updates.

Download and Compile

Download the compressed archive into your `/usr/src/` or `/usr/local/src` directory. Let's assume that the archive is named `./epic4pre2.001-NR10.tar.gz.` You can grab the source (theoretically) from any of the following sites:

- ftp://ftp.epicsol.org/pub/epic
- ftp://ftp.acronet.net/pub/ircii
- ftp://ftp.neato.org/pub/irc
- ftp://epic.doggie.net/epic
- ftp://ftp.parodius.com/pub/EPIC
- ftp://clients.undernet.org/pub/irc/clients/unix/ircii
- ftp://ftp.lice.org/pub/irc/clients/epic
- ftp://ftp.shutdown.org/pub/epic
- ftp://ftp.memax.krakow.pl/pub/ircii
- ftp://ftp.totem.tihlde.hist.no/pub/mirrors/ftp.acronet.net/pub/ircii

TOC

This list is from EPIC FTP Sites, from The Official EPIC Homepage.

So, after downloading it to `/usr/src/` or `/usr/local/src/` unpack the archive by typing:

```
# tar -zxvf epic4pre2.001-NR10.tar.gz
```

This will unpack the directories and files needed to produce your brand new IRC client. When it shows a whole bunch of files being listed, then you know it's unpacking. By the way, this is how you should unpack most archives you download. When it's done unpacking and you're back at the command prompt, **`cd`** to the `./epic4pre2.001-NR10` directory ('cd `./epic4pre2.001-NR10'`, or replace the `./epic4pree1.400` with the directory the archive unpacked into.) When you're in the new directory, either read the INSTALL file (recommended even if you don't have a different archive) or use these instructions:

- **`./configure`**
- **`make`**
- **`make install`**

Again, make sure you're doing this from `/usr/src/epic4pre2.001-NR10` or `/usr/local/src/epic4pre2.001-NR10`, 'epic4pre2.001-NR10' being the directory that you extracted from the tarred and gzipped file. The file you have can be directly executed using the command **`epic`** (or **`irc`** on older versions of EPIC) after compiling. There are other versions of ircii around, though, remember. I am currently using ircII-EPIC4pre2.001-NR10 on my last update.

Being in Two Places at Once: Opening New Windows for Channels

If you want to be in two or more channels at the same time, there's no problem with that; just type **`/join [Channel]`** everytime you want to talk in that channel. Even if you've already joined that channel, it won't be set to be the active one until you type that again.

But what about if you want to split your screen into two or more sections, or maybe open up a whole new window for each channel (if you're in X)? First I'll discuss how to split your screen, and switch between them, then discuss new windows in X.

Splitting the Screen

There are several steps that I take. First, I type **`/bind ^w next window`** so that I can type Ctrl-w to switch between portions of the screen. Then I split the screen by typing **`/win new`**. You can tell which part of the screen is active by looking for a whole bunch of characters like "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^" on its bar at the bottom. Once you have a selected active window, type **`/join [Channel]`**, where [Channel] is the channel that you want to join.

Opening Up a New Window

You should be in X to do this. You can chat through **`xterm`** or **`rxvt`**, it doesn't matter. You just have to type **`/window create`** to open up a new window in X. Then use your mouse or whatever to switch to that window and join the channel of your choice.

Putting New Window Information in Startup

You can put these lines in your startup script, called `~/.ircrc`. We'll discuss the `~/.ircrc` file in more detail later on. To split the screen, assign Ctrl-w to switch windows, and join both #LinuxHelp and Computers on startup, I have in my `.ircrc` file the following:

```
bind ^w next window
join #Computers
win new
join #LinuxHelp
```

Color

If you use EPIC4, you can see mIRC-style colors by typing **`/set control_c_color on`** if it's not already enabled. If you want to turn it off, type **`/set control_c_color off`** while you're in the program.

Note that most scripts put in the color for you. They even add additional color into things that are not colored by default.

If your color doesn't seem to be showing up, your terminal type probably doesn't support color.

Startup Script

`~/.ircrc` is a file that ircII reads (and executes each command in it) every time you start it up. It should be located in your home directory; it probably doesn't exist yet, so don't worry if it's "missing". You can create it using a text editor such as **pico**, **vim**, **joe**, **jed**, or **emacs**. Each line is read as a separate command, so if you want to automatically set colors on, for example, put in the line `set control_c_color on`. If you want to automatically join a channel (I'll use #LinuxHelp as an example), add the line `join #LinuxHelp`. Putting a slash (/) character in front of the commands isn't necessary if it's in the file.

`.ircrc` is roughtly comparable to the various *.ini files that mIRC and PIRCH (IRC clients for Windows) load; ircII-EPIC is highly scriptable (contains most of the features of mIRC and pIRCh, and a few more. Check out the ircII-EPIC Scripting Help Table of Contents for a list of all its commands/functions).

Setting Up Netscape Navigator

TOC

Last modified: November 15, 1998
Development stage: Alpha

If you're trying to keep your computer free of any and all Microsoft software, it's really easy with Linux. Netscape Communicator is available for Linux and Unix systems. Some systems (such as Red Hat 5.1) might already have this, because Netscape made the browser free and even opened up Mozilla to open development.

Simple Red Hat Installation

If you're using Red Hat, you can simply download Netscape 4 as Communicator or Navigator (standalone). For Red Hat 5, you can go to the updates directory at SunSITE and get the entire Netscape Communicator suite or just Navigator. For Red Hat 4.x, you can get it at SunSITE as well. The filename will be something like `netscape-communicator-4.06-2.i386.rpm`, if you're downloading the entire Communicator suite. If you want to get both Communicator and Navigator standalone, I think you'll have to get the netscape-common package as well.

Then to install the file(s) that you downloaded, go to the directory where it was downloaded, and as the root user, type:

```
rpm -ivh netscape-communicator-4.06-2.i386.rpm
```

And that should set everything else up in a snap. You should then be able to type "netscape" in an xterm or add it to the *.rc or menu file for your window manager. You won't need to read any further. :-)

There might be a later version of the browser after you read this.

The Hard Way

I'm not sure where Netscape keeps the Navigator 3 that still works with Linux 2, but the file to download for Netscape 4 is right here. It's pretty easy to install.

Andy Smith suggested that you put the Netscape binary in `/usr/bin/` or `/usr/sbin/` so that you can just type "netscape" from the xterm window, instead of specifying the entire path. Or, you can add it to your window manager's menu file, usually the .*rc file in your home directory (.steprc for AfterStep, .fvwm95rc for fvwm95, etc.).

A more appropriate place to put it would probably be in `/usr/local/bin/` so that it would not clutter `/usr/bin` (which already has a lot of programs that were installed with your distribution). If you have access to the root user on the system, you should install it in those directories; if not, then you'll have to keep them in directories that you have permission to write to (such as `~/bin/`, which is the `bin/` subdirectory in your user's home directory).

As you may see, this is different from a lot of Linux programs and is already in a binary executable. No compiling is necessary.

Java Problems with Netscape 3.x

I now use the newer Netscape Communicator, but in case you prefer Netscape Navigator 3, Java setup was one problem that I faced until I finally decided to ask questions in #Linux on the Efnet IRC network. I found out that I was supposed to put java_301 in `/usr/local/lib/netscape/` or `$HOME/.netscape`. As you probably know by now, the `$HOME` variable represents your home directory. **The file `java_301` goes in a `netscape` directory!**

```
netscape: '/usr/local/lib/netscape/plugins/java_301' is not an ELF file
ERROR: Not an ELF file
Cant load plugin /usr/local/lib/netscape/plugins/java_301. Ignored.
```

If your console/terminal screen gives this error when you're starting Navigator, that doesn't necessarily mean that your Java isn't loaded. Anything you put in `whatever/directory/.netscape/plugins` is to be read as a plug-in, so then Netscape thinks that the `java_301` file is an executable, which it isn't. If you don't want to see this error, I suggest that you remove `java_301` from the incorrect directory and place it in the correct one. That file is just a compressed archive that is "unzipped" when you are going to use it. I might have misread the README file that came with Netscape after I extracted it, and that could be the cause of my troubles. Oh well, if any of you have trouble with Java in Netscape, this might be the solution. If not, send me the error message and I'll try to help, despite my limitations as a mere mortal.

I've heard some reports that confirmed that there was a bug with Linux Netscape 3.0 and Java. When encountered with most Java applets, Netscape 3.0 just shuts down, no error message, nothing. It just closes. The fix is to download this library (gnumalloc.so) and put it [`gnumalloc.so`] in /lib. Then put the following lines into a script, make the script executable (**chmod 755 scriptname**). Make sure you don't overwrite the `netscape` executable.

```
#!/bin/sh
export CLASSPATH="/usr/local/netscape/java/classes/java_301:."
export LD_PRELOAD="/lib/gnumalloc.so"
exec /usr/local/bin/netscape "$@"
```

I have that written to a text file as `netscape.script`. I put that script in `/usr/local/bin` so that I could execute it from X. After putting it there, edit the rc file of the window manager you're using in your home directory (or the system-wide window manager rc file if you please). Make whichever references to the `netscape` executable point to `netscape.script` or whatever you named it as. Also, if you want to load Netscape Navigator manually, just type `netscape.script` in an **xterm** window. More information on Java and Linux is available at http://www.blackdown.org.

Plugins

Netscape doesn't distribute plugins with Navigator for Linux/X11 due to some restrictions about using commercial libraries (Motif). So, you'll have to get separate plugins by downloading from people other than Netscape. This way, you'll be able to hear embedded sounds in webpages that have become so popular. One such program that I got to work is Plugger. All I did was extract the .tar.gz file, modify some source code (to point to where the .h files were), and put it in `/usr/local/netscape/plugins`. The file you put in your Netscape plugins directory should be something like `mimeplugin.so`. That was a while ago; you might not even have to modify source code.

To set it up, in Netscape, just click on the MIME types it supports by going to the **Options** menu and going to

General Preferences. Click on `audio/x-midi` in **Helpers** as an example and you will see an option to us "Generic Linux Plug-in". Enable using the plugin so that you can hear embedded sounds in a page. On my sy when I used a standalone helper application such as **wavplay** or **playmidi**, I could hear sounds if I went directly to a sound file, but not if it was embedded as a background sound in a webpage.

Additional Stuff

I am surprised by the response of people to this page! Everyone can pitch in and help, like markmohr. Here was the message he gave, stating his self-found solution:

```
Well I finally figured out what was wrong and how to install Netscape.
After transferring the file over to the Linux Partition, I told you that I
used gunzip to unzip it, and what I got from that was just a file called
netscape.

Well I opened that file in vi and started to read it, and in
it, it tells you the commands you need to use to unzip it and have all the
files you unzip in the directory you have it in.  So I transferred it back
over in its zipped up state and unzipped this time using this command
"#gzip -dc netscape.gz | tar -xvf- " it work all the files came up
and when I typed #netscape from the xterm window bing, bang boom up popped
Netscape Navigator. Now I'll I have to figure out is how to dial out with my
modem and use the darn thing.

Thanks for all your help and you might want to add to our web page the
command to unzip the zip file.

Keep in touch

Mark
```

Randy Conn wrote in to say, "*http://www.fortify.net has a patching program that will convert the freely downloadable 56 bit version of Netscape into 128 bit operation. Then, you don't have to put your name and address into the government database the way you do if you want to directly download the 128 bit version. I have successfully used fortify to upgrade Netscape Communicator 4.06 running under Debian 2.0/Linux 2.0.34&5. A few weeks after using fortify, I found that it exists as a Debian package in the unstable/web section near the end.*".

Author's Notes

Many of the problems with Navigator 3 have been fixed in Communicator. Of course, no software is perfect, so there are still bugs lying around. It's especially hard to find and maintain a program as large as Communicator.

Comments, questions, suggestions, corrections? Send them all to jtg@computers.iwz.com.

Setting Up E-mail

Last updated: February 5, 1999
Development stage: Beta

E-mail is probably one of the most important parts of using the Internet, so it only makes sense that Linux lets you use your precious e-mail.

Your ISP *probably* uses a POP3 server for your e-mail. The program **pine** is an e-mail client that enables you to read, send, reply, sort, delete, forward, and do a lot more to your e-mail. That's what I use, although you might want to try other mail programs such as **mutt** or **elm**.

Downloading Your E-mail

To download whatever e-mail you have onto your system, you should use a program called `fetchmail`. It handles POP3 (probably what you use), IMAP, and a bunch of other protocols for e-mail transfer.

Set Up Your ".fetchmailrc" File

To use fetchmail, you should set up a file in your home directory called `.fetchmailrc`. It probably doesn't exist yet, so create it by typing **pico ~/.fetchmailrc**. Here's what mine looks like:

```
poll pop.netaddress.com proto POP3 fetchall
```

That will tell fetchmail to check the server **pop.netaddress.com** using the **POP3** protocol. Since USA.net/NetAddress marks the messages as already being seen, I have to put in the "fetchall" part to tell it to get the messages no matter what.

Note that all the options for the server are on one line.

If you want to, you can also put your password in the .fetchmailrc file. Use the "pass" option. For example, if my password were "barrel", my sample `.fetchmailrc` file would look like this:

```
poll pop.netaddress.com proto POP3 pass barrel fetchall
```

In the event that your password has strange characters in it, you will want to put it in quotation marks. I would not recommend putting your password in your `.fetchmailrc` file, because if anybody ever gets the root account on your system, they will also be able to get your password. If you're not worried about the security of your machine, and just want to get your mail with the least hassle possible, then include it.

Fill in your `.fetchmailrc` file with the options you need to (there are tons more options that you can view by typing **`man fetchmail`**). You should have at least the "poll" and the "proto" fields. Then save the file and exit. Once you're back at the prompt, type **`chmod 600 ~/.fetchmailrc`**.

Using Fetchmail

Now all you have to do is type **`fetchmail`** and it should prompt you for the password. It takes your username on your own machine and uses it as the login name for the account on the POP3 server. If you need to change it to get your e-mail, add the "-u" flag and then your username. So if I had to use the username jgo instead of joshuago to download my mail, my fetchmail command line would be **`fetchmail -u jgo`** instead. If you want to keep it in your .fetchmailrc file, use the option `user [name]`.

TOC

Reading Mail

There are four programs that I know of that you can use to read your e-mail. One is **pine**, the second is **mutt**, the third is **mail**, and the last is **elm**. I have experience through normal use only with PINE (although I like Mutt a little bit too), so that's what I'll give instructions on.

Start up PINE by typing **`pine`** at the command prompt (it works in an xterm window as well). If this is the first time that you've started up PINE, you'll have to go through the setup procedure by typing the following keys once you're in the main PINE menu: **S, C**. This will take you to a screen that looks like this:

```
  PINE 3.95   SETUP CONFIGURATION                  Folder: INBOX  42 Messages

personal-name              = <Put your name here>
user-domain                = <Part after the @ in the return address>
smtp-server                = <SMTP server>
nntp-server                = <News Server>
```

The rest you can leave blank if you want. This is how mine looks like:

```
  PINE 3.95   SETUP CONFIGURATION                  Folder: INBOX  42 Messages

personal-name              = Joshua Go
user-domain                = local.net
smtp-server                = mail.local.net
nntp-server                = news.local.net
inbox-path                 =
folder-collections         =
news-collections           =
incoming-archive-folders   =
pruned-folders             =
default-fcc                =
default-saved-msg-folder   =
postponed-folder           =
read-message-folder        =
signature-file             =
global-address-book        =
address-book               =
feature-list               =
            Set        Feature Name
            ---   ----------------------
            [ ]  allow-talk

? Help       E Exit Config  P Prev      - PrevPage    A Add Value    Y prYnt
             C [Change Val] N Next    Spc NextPage    D Delete Val   W WhereIs
```

The "user-domain" part is the part that appears after the "@" in your e-mail address. In my configuration it's configured to show my e-mail address as "jgo@local.net". As far as I know, that will just make your e-mail address appear a certain way so that people can reply. It doesn't actually use that server to send or receive mail.

Make sure the username you are using on your Linux machine is the same as the username you have on the mail provider. If you don't have that, make sure you know how to use the adduser shell script (it's a text file commands to give to Linux to add a new user). I only use "root" to take care of local system tasks, to dial in, and to start point-to-point protocol (PPP).

TOC

Comments, questions, suggestions, corrections? Send them all to jtg@computers.iwz.com. You can also try using the help form or the guestbook, which is actually a feedback form.

TOC

Installing Window Maker as Your Window Manag

Written by Joshua Go (jtg@computers.iwz.com)
Created July 18, 1997
Last updated December 20, 1998
Development stage: Beta

A window manager is software that runs along with the X window server to provide titlebars, menus, and other decorations for software. As the name implies, it "manages" the windows of your software applications. Window managers are often also responsible for the "feel" as well as the look of your desktop.

Window Maker is a good-looking, nice-feeling window manager (as you might be able to guess, it's the one I prefer). Despite its simple name, it really looks and operates very well although it hasn't reached 1.0 yet. Other cool window managers are out there, like Enlightenment, but E is still under heavy development and may crash unexpectedly. Others like AfterStep are stable, but might not suit the tastes of some people (like myself).

Installing Window Maker isn't very difficult and not much hassle, *if you know what to do*. My goal for this page is to answer the questions one probably would have when installing and using Window Maker. The Window Maker homepage is at http://www.windowmaker.org.

If you want to know what Window Maker looks like, take a look at a page of screenshots that I've taken of my X desktop.

Window Maker Installation

The official Window Maker homepage is probably where you want to look for general Window Maker-related issues.

If you want desktop tiles you can go to my tiles page, Themes.org's Tile of the Day page, or Scott Goehring's tiles page. If you don't know what tiles are, they're the square things on the side of AfterStep and Window Maker screens where icons are.

Get the Files

You should download the current file, which is 0.19.3 at the time of my last thorough update. The main Window Maker pixmaps are now distributed with the program, so you only need one file. Previous versions required you to get a Window Maker-data package as well as Window Maker itself. However, I would still recommend downloading the Window Maker-data package because I think there are some nice icons that aren't included with the main package. Here are some sites you can get the files:

- http://jgo.local.net/cool_downloads/wm - An HTTP mirror of Window Maker on my site.
- ftp://ftp.windowmaker.org/pub/wmaker - Has everything you need. Just browse a little.
- http://windowmaker.muscat.co.uk/wm
- ftp://ftp.io.com/pub/mirror/linux/windowmaker

You can download those anywhere you want; I'll just use `/usr/local/src` as an example. That means `WindowMaker-0.19.3.tar.gz` should be in `/usr/local/src`.

Window Maker

Uncompress `WindowMaker-0.19.3.tar.gz` by typing in **`tar -zxvf WindowMaker-0.19.3.tar.gz`** while in `/usr/local/src`. That should create a directory called `WindowMaker-0.19.3/`. Move to that directory by keying in **cd WindowMaker-0.19.3**.

Don't immediately run **./configure**. Untar the file called `libPropList.tar.gz` using **tar -zxvf libPropList.tar.gz**. Do a **cd libPropList** and then run **./configure ; make ; make install**. That should compile that and install it on your system for future Window Maker releases, so move back to the previous directory using **cd ..**. I think libProplist has to be downloaded separately these days.

Now is the time to start compiling Window Maker itself. Type in **./configure ; make**, which will probably take a little time. Let it finish. If you're not the root user already, type in **su**, and type in the password. Then do **make install** and that should do the trick; you've got the base Window Maker system installed.

Pixmaps

This window manager won't really look very good if you're missing a few XPM icons, so install them. Move on back to `/usr/local/src/` (or wherever you put the `WindowMaker-data.tar.gz`. Untar it with **tar -zxvf WindowMaker-data.tar.gz**. That will create a directory for you called `WindowMaker-data`, which you will move into. All you do is move the `pixmaps` directory to `/usr/local/share`. If you don't have a `pixmaps` directory there, you should do **mv pixmaps /usr/local/share**, and if you already have one, then you can do **mv pixmaps/* /usr/local/share/pixmaps**.

User Installation

Choose a user, any user. In my case, I'd choose **joshuago** because that's the username I mainly use for my X sessions. I'd log in as **joshuago** and then type **wmaker.inst**. That will install all the files and create all the directories I need to run Window Maker. In most cases you should just accept the defaults. The same goes if you want to install it for other users.

Other Places of Interest

- Icon Store
- Window Maker Themes

Comments, questions, suggestions, corrections? They're all welcome, so send them to jtg@computers.iwz.com. You can also use the guestbook or the help form.

Running Multiple Sessions of X

Created on December 11, 1997
Last updated: October 3, 1998

I think this is very, very neat. I got it off of Linux Gazette Issue 23. Let me know if it works for you.

I would think that a lot of people would want to know how to run multiple X sessions; sometimes you have a different user who wants to log in and use X, but you're tied up downloading the newest version of WindowMaker with Netscape inside your own X session. Or something like that.

The solution? Run multiple X sessions. This can be done with only a few extensions to your command line. For your first X session, all you had to do was type **startx**, or if you run in high color mode like I do, you might have to type **startx -- -bpp 24**. That defaults to running it on screen 0, which is really your first X session.

What you want to do is make X run in screen 1 for the second, screen 2 for the third, screen 3 for the fourth, and so on. Here's how you would start a second X session:

```
startx -- :1
```

Now, if you run in high color mode, you would enter the proper paramaters *after* the **:1** part of the command line. If you wanted to run in 24 bits per pixel (16.7 million colors), you would type **startx -- :1 -bpp 24** as your full command line.

If you want to start yet another X session, a third one, you would type **startx -- :2** and **startx -- :2 -bpp 24** for 24 bit color. After that, you would use the numbers 3, 4, and 5 for additional X sessions.

Switching Between X Screens

You probably have a default number of 6 virtual consoles. Those are how many logins you can have in plain text mode, basically. If you have F12, you would probably have 6 "empty" screens if you're not running X.

Your first X session, as you may already know, is accessible by pressing Alt-F7 (use the left Alt key). For your second, it's Alt-F8. For the third, Alt-F9, and so on. This is what you would do to switch from text mode to an X screen.

If you want to *switch from one X screen to another*, you'll have to use Ctrl-Alt and then whatever F-key you'd normally use. For example, if you want to access X screen number 3, you'd press Ctrl-Alt-F9 from an existing X session. From text mode, you'd only have to press Alt-F9.

Suggestions

Tim Briggs previously sent in an e-mail concerning his frustration starting more than one X. He included it with a script, but then he sent in another version which is much cleaner and has a few improvements over his old one, which was previously posted here.

```
#!/bin/sh

if [ $# -eq 0 ]      # check to see if arguements are given (color depth)
       then a=16    # default color depth
        else a=$1    # use given arguement
fi

if [ $a -ne 8 -a $a -ne 16 -a $a -ne 24 ]
        then
                echo "Invalid color depth. Use 8, 16, or 24."
        exit 1
fi
```

```
for display in 0 1 2 3 4 5    # checks for open display, starts X on next avai
        {
        if [ ! -f "/tmp/.X$display-lock" ]
                then
                        exec startx -- :$display -bpp $a
                exit0
        fi
        }
echo "No displays available."
exit 1
```

The way I manage multiple X seesions is put stuff in alias form in `/etc/bashrc`. You'll probably want to use Tim Brigg's script.

```
alias x="startx -- -bpp 24 -quiet &"
alias x2="startx -- :1 -bpp 24 -quiet &"
alias x3="startx -- :2 -bpp 24 -quiet &"
alias x4="startx -- :3 -bpp 24 -quiet &"
alias x5="startx -- :4 -bpp 24 -quiet &"
alias x6="startx -- :5 -bpp 24 -quiet &"
```

If you want to, go ahead and copy that to paste in your `/etc/bashrc`. You'll need to log out and log in again for the changes to take effect after changing that, and from then on those aliases will be carried in to all new logins.

When I type **x** it starts up my first X session. Then it starts up to 5 more X sessions. The first one is referred to as session 0, although it's actually number one. That's fairly common in Unix.

It's also worth noting that I put the ampersand ("&") at the end of the startx command line in order to make it run in the background. That way it frees up a text mode console for me instead of keeping it and making me log in again on another virtual terminal.

Related Pages

1. Remote X
2. Configuring and Troubleshooting X
3. Installing and Upgrading X

Send all comments, questions, suggestions, corrections, and complaints jtg@computers.iwz.com.

Remote X

TOC

Created May 28, 1998
Last updated June 26, 1998
Development stage: Alpha

Summary:

1. Set DISPLAY variable (`set DISPLAY="some.host:0"`)
2. Allow hosts to connect to your X server (`xhost +other.host.com`)
3. Start up a font server on the server side by typing **xfs &**.
4. If you need to, set the address of the font server on the client side.
5. On a terminal (telnet, ssh, rsh) on the remote host, type in the name of the program you want to run (gimp, netscape, and xv are a few)

This page is going to help you run X programs on another computer, but display it on your X session. This is especially helpful if you have a slow machine on your LAN or want to display X applications on a Windows machine.

Client Side Configuration

On your client side (the computer you want to display and use the program) you need something that allows the display of X programs. On a Linux/Unix machine you'll probably just need to have something like XFree86 or MetroX; any standard X server will do, as long as it displays and starts up fine on your computer. If it's a Windows machine, you need to download an X server. Two that I know of are XWin32 and MI/X. You can look for more at Tucows, under server daemons, I think.

You should allow connections from the server side so that the programs can display. If you're using Unix/Linux on the client side, that can be done by typing **xhost +server.side**, to allow the hostname "server.side" to connect. You'll probably need to change the hostname to whatever the hostname of your Linux box is. With ssh you can just connect to the server and type the name of the program **without** setting xhost or $DISPLAY. I *think* it works by exporting the DISPLAY variable and automatically setting xhost to allow applications to be run from the server side. If $DISPLAY isn't set on the client side, I don't think it will be exported to the server side so that applications display on your screen. Thanks to Todd Cohen for pointing this out. If anyone can provide more information on this please let me know.

If you're using XWin32 you need to set *Options* -> *XHost* to the IP address of the machine with the X applications in XUtil32. With MI/X I don't recall having to set the host but my memory isn't fresh.

Font Setup

If the fonts that came with your X server don't work or you want to use the server side fonts, then set the font server to be the hostname of the server on port 7100. In XWin32 it's in XUtil32's *Fonts* -> *Path* dialog box; click on "Add Font Server" and in place of "xfs" use the hostname or IP address of the font server. For me I had the font server point to 192.168.0.1 on port 7100. If you're using a Linux/Unix box as your client, I'm not sure but I think you can use the command line **xset +fp 192.168.0.1:7100**, if 192.168.0.1 is the font server. If that command works for anybody, let me know.

A font server must be running on the server side. Read on to find out how to start it up.

Let me know if you find something useful to add to the information here.

Server Side Configuration

While on the server side (the machine that has the programs that you want to run), set and export the $DISPLAY variable by typing **export DISPLAY="client.side:0"**. You can get into the server machine using telnet, rsh, or ssh.

If you're using a shell other than "bash" on the server side, set the variable accordingly. Most standard Linux distributions now use "bash".

Fonts

To have the fonts that are on your X server be used on the client side, you should start up a program called **xfs** as root. That's the X font server.

To have the X font server started up everytime you boot up your system, just put the path to the **xfs** executable in `/etc/rc.d/rc.local`. On my system, **xfs** is in `/usr/X11R6/bin/xfs`, so I have the line "/usr/X11R6/bin/xfs &" in `/etc/rc.d/rc.local`. The ampersand ("&") makes it a background process so that it doesn't tie up the server side by running in the foreground and preventing other tasks from being done.

Running the Program

After the client and server side setup, you should be able to run any program for X. You can try programs such as "xv", "xbill", or "netscape". They should run and display on the client side.

Author's Notes

Running remote X sessions through a standard dialup connection to the Internet is very slow. It will still work, but it's just way too slow.

MI/X seems to have trouble with fonts on my brother's Windows 95 system. It is missing some fonts that are commonly used, so "xgalaga" and "gimp" (as well as other Gtk+ applications) won't run. Instead, I just use XWin32, although that takes a while to start up and is crippleware (2 hour session limit). You also can't use it on more than one machine (node) on your network at the same time.

When saving files, you have to save them on the server side.

TOC

Related Pages

- Configuring and Troubleshooting X
- Installing and Upgrading X
- Running Multiple X Sessions

Comments, questions, suggestions, complaints? Send them all to jtg@computers.iwz.com using either your e-mail client, the guestbook, or the help form.

Contributing to Josh's Linux Guide

Created on July 9, 1998
Last updated August 28, 1998
Development stage: Alpha

If you feel that you would like to contribute to this Linux guide, I encourage it, since I can't handle everything. More advanced topics are also welcome, since people who are new to Linux eventually move on. I'd like to keep the audience reading as long as possible. There are just some things that I want to get across, to ensure that it works out well.

1. You must be willing to maintain your document, and this means that you must provide an e-mail address to contact you with. You must also be willing to receive any sort of feedback from readers, and even help them with any problems they encounter.
2. Have a short title on your pages, but let them be descriptive enough to let the reader know what the page is about. I've made this mistake plenty of times. Put your title in the <TITLE> and <H2> tags.
3. Don't use abbreviated Internet English. Use proper spelling and grammar as far as you can help it. There will be typos, but eliminate them to the best of your ability.
4. Have fields for when the page was created ("Created on"), when it was last updated ("Last updated"), and its development stage ("Development stage" can be alpha, beta, or complete). You can see examples of this in many of my documents.
5. Have a white background (#FFFFFF) and black text (#000000) on your documents. Yes, submissions have to be in HTML. Or even SGML. Nothing else.
6. Though it's not necessary, I highly recommend allowing people to copy, print, and re-distribute your documents. I also let people modify my documents, but they have to give credit to me. I'm no legal expert so I don't know much of the quirks involved in putting up documentation online. You may also put your work under the OpenContent if you want. I am also considering switching to OpenContent. Put your copyright notice at the bottom.
7. If it's not too inconvenient, have the page hosted on your own web space so that you can easily make updates to it. If that's not possible, then just e-mail me the updates, to jgo@local.net. Even if you have the page hosted on your own server, it's also a good idea to e-mail me the updates in case something happens or whatnot. It will be copied into a subdirectory called `submissions/`.
8. Filenames depend on what type of audience it was aimed for. If it begins with "linux-" then it should aim to be applicable to any and all distributions of Linux. If it begins with "redhat-" then it should be limited to the Red Hat distribution. It's better to write documents aimed at all distributions. I only have the "redhat-" pages there so people can install it, since the way distributions are installed varies greatly.
9. Put commands in Bold (<B>), directory and file locations in fixed width (<TT>), and make printouts preformatted (<PRE>). Remember to close the HTML tags. These are just basic guidelines to follow; you can also use italic.
10. Subsections should be in <H3> and sub-subsections should be <H4>. You should never have to use <H5>.
11. Links to any outside sites must have a target of "_top" in order to ensure that they aren't loaded in

the frames. I emphasize this strongly so that your readers don't get annoyed at you.

12. If there's anything you're still unsure of, look at the HTML source on any of my documents. If you still aren't sure, then send me an e-mail.

Following these guidelines will help tremendously in keeping things consistent. Any future changes that are required, I'll let you know, or I'll make the change myself.

On a side note, I think we need more "Informational" pages. The "Instructional" section is getting a wee bit crowded.

TOC

Setting Up Procmail

Author: jbm <jbm@intertek.net>
Created on February 28, 1999
Last Modified on March 1, 1999
Development stage: Alpha

Procmail is a great tool for sorting your mail under Linux, but it seems to have a bit of a reputation for being arcane. It can be, but with a little help, configuring things is really easy to do!

This document assumes you've got `procmail` installed - sorry to those of you who don't, but most major distributions have a package for it. It's also very easy to compile.

Please note that the authors of `procmail` use the word "recipie" to refer to filters. In deference to them, I will do likewise.

Table of Contents

Setting up the `.forward` file

Place the following line in the `~/.forward` file:

```
"|IFS=' '&&exec /usr/bin/procmail -f-||exit 75 #jbm"
```

The quotes *are* important! Replace the "jbm" with your username (this is so the file is totally unique; some systems have problems if every `.forward` isn't unique). This line should be the only thing in the file, unless you're doing some weird stuff with copying your mail (which I doubt you are).

Setting up the `.procmailrc` file

Beginning the file

Open up `~/.procmailrc` in your favorite text editor, for example '`pico ~/.procmailrc`' or '`emacs ~/.procmailrc`'. `pico` is a good option for beginners. At the top of this file need to go a few lines:

```
SHELL=/bin/sh
MAILDIR=$HOME/mail
LOGFILE=$HOME/.procmaillog
VERBOSE=yes
```

Make sure MAILDIR exists! `procmail` will replace the '`$HOME`'s with your home directory (e.g.: `/home/jbm`). The LOGFILE is optional, but I recommend it highly. It lets you see if there was mail redirected to `/dev/null` that shouldn't have been--which is a bad thing. This allows you to debug your regular expressions and/or the order of the recipes you're going to apply. There is an example in the `procmailex` manpage ('`man procmailex`') detailing a "safety net" to copy the last 32 messages to a directory for you; this will be covered later.

A recipe in `procmail` is a set of rules that define a filter. These rules are generally a regular expression to match (e.g.: .*, ^From:.*bob@host.dom.au), and then a line or two tellng `procmail` what to do when it matches that regexp.

Regular Expressions

Regular expressions are extremely powerful ways to express complex things. They are used for pattern matching (like *.* or foo??.txt, but much more complex). Which makes them really hard to learn. There is at least one book on them and many many books devote chapters and sections to them. We will smooth over things a bit with a few "Regular expression formulas," in the table below:

RegExp	Meaning
.	Matches any character (except newlines)
^	Beginning of a line
.*	Wildcard: matches anything

Yep, those are the only regular expressions you need to know! Now, this is a very simple application of regular expressions, mind you. You should go out and do a little research on them some time, probably when you're learning `perl` or `sed`, two very useful utilities.

Recipes

Now that you have regular expressions under your belt :^), it's time to construct a recipe. First, every recipe begins with the line

```
:0
```

This has `procmail` scan the headers of the document for matches, use

```
:0B
```

for body scanning.

The next line is the regular expression `procmail` should match against. These are typically of the format

```
* ^Subject:.*fm/news

or

* ^From:.*oths.k12.il.us
```

These lines all begin with "*" and continue on with a regular expression. The next line will be the "folder" to place the mail in; this is just a file in the MAILDIR. Unlike the MAILDIR, `procmail` will create this file own its own. When all three lines are combined, it looks something like this:

```
:0
* ^Subject:.*fm/news
freshmeat

:0                  # Guide Mail
* ^Subject:.*uide
jlg
```

`procmail` evaluates the recipies in beginning-to-end order in the file. `procmail` compares a message with the first recipie in a file. If the message matches the criteria of the recipie, `procmail` does what the recipie instructs it to and then stops. If it doesn't match, it tries again with the second recipie, and so on. Therefore, the order you have recipies in does matter! If you want all mail about dogs to go to a dogs folder and all mail from your Significant Other to go to another with all mail from your Significant Other about dogs going to the dogs folder, you had better have the dogs filter above the SO filter.

Comments: comments must not be within your recipies! Just put them on the :0 line, and noplace else. Trust me.

Safety Nets

In case you mess anything up, you should set up a "safety net", as described in the '`procmailex`' manpage. Begin by making a '`backup`' directory under your MAILDIR. Then add the lines

TOC

```
:0 c
backup

:0 ic
| cd backup && rm -f dummy `ls -t msg.* | sed -e 1,32d`
```

at the very top of your `.procmailrc` file. That way, *every* message gets sent through the recipie. This recipie will keep the last 32 messages sent to you in `MAILDIR/backup/`.

An Example `.procmailrc` File

```
MAILDIR=$HOME/mail
LOGFILE=$HOME/.procmaillog
VERBOSE=yes

:0:                        # UF Mail
* ^Subject:.*Ufies-STAFF
uf

:0:                        # JLG Mail
* ^Subject:.*uide
jlg

:0:                        # FM digest
* ^Subject:.*fm/news
freshmeat

:0:                        # APCi trash
* ^Subject:.*ime Limit
/dev/null

:0:                       # yet more APCi junk
* ^Subject:.*Reached.Hard
/dev/null

:0:                       # cron output
* ^Subject: cron:
cron

:0:                       # SPAM!!!
* ^Subject: .*fast.*cash
/dev/null
```

TOC

Note that I use :0: for the first line; this sets up a "lockfile" so two `procmail`s don't try to write to same file simultaneously (that would be a very bad thing). Hopefully this makes setting up a `.procmailrc` easier for you.

Configuring Pine

Of course, it's fine and dandy that you've got your mail all sorted out, but you can't read it from pine this way. Now comes fun stuff. In Pine 4.05 (go grab a copy from http://www.washington.edu/pine/), go to your Folder List from the Main Menu, then to the Incoming-Folders if it's there, and then do Add. It will prompt you for a server, just hit enter. Then it will ask for a folder name. This is what you named it in your `.procmailrc`, ie: `$MAILDIR/foo`. Next it prompts you for a nickname, give it one.

As an example, let's say you subscribe to a homebrew list. In your .procmailrc you have

```
:0:
* ^Subject:.*PC.homebrew
homebrew
```

Your pine options would then be Folder: "mail/homebrew" and Nickname: "homebrew".

Further configuring things

[more to come here soon, on other things like WMMail and other biff-like apps]

Further Reading

Other things to check out for info:

- '`man procmailrc`': the `.procmailrc` documentation
- '`man procmailex`': examples of `.procmailrc`
- Procmail Homepage
- Procmail tips page: a very extensive collection of `procmail` info.

Filtering e-mails between users using qmail

TOC

Created on 10 April 1999
Last updated on 14 August 1999
Development stage: ver 1.1

This guide was written to be part of Josh's Linux Guide.

Introduction and objective

A number of commercial ISPs offer their customers several email aliases to their one mailbox. Many e-mail programs can filter incoming mails into seperate folders for each alias, but as Linux provides individual users with a fully seperate login and working environment, the question is: can incoming mails be seperated and delivered to users? The answer is yes, but it is not entirely straightforward, and you need to use some of Linux's most sophisticated programs to achieve it, hence configuration is a challenge.

This "howto" looks at the specific example of forwarding incoming mail for simon.hampton@tvd.be to linux account sim (home directory: /home/sim), and mail for liesbeth.devriendt@tvd.be to linux account lies (home directory: /home/lies). To collect my e-mails I need to access the so-called "multidrop" mailbox offered by my ISP, called mb10099.

Mails will be collected using fetchmail (section 1), and passed to qmail (a sendmail replacement, section 2) for delivery locally.

1. fetchmail

fetchmail has probably already been installed on your system. There appears to be little competition for fetchmail, but it is clearly the pride of the OSS community (its author, Eric S. Raymond, wrote the Chathederal and the Bazar). Fetchmail specialises in on-demand links, especially for temporary connections. Even though I have a permanent cable TV connection, I have to demand my e-mails as my ISP does not forward them automatically to customers.

fetchmail is fairly straightforward to get running, see for example:

- The e-mail guide provides a useful start
- the Fetchmail FAQ, and man pages
- documentation in /usr/doc/fetchmail-*

A simple ~/.fetchmailrc file should look something like the following:

```
set daemon 300
poll hostname.ISP.country with proto POP3
    user "mb10099" here with password "XXXXXXX"
    is * there options forcecr keep
```

The words in bold are read and acted upon by fetchmail, while the normal text is noise that can added as required and is ignored (more details are in /usr/doc/fetchmail-*/sample.rcfile). Don't forget to set it to chmod 0600.

set daemon 300	Fetchmail runs in the background and checks for emails every 5 minutes. You may want run fetchmail by hand if you have a dial-up connection.
poll hostname.ISP.country	The poll commands tells fetchmail to check the following site for emails. When poll is replaced by "skip", this entry would be ignored.
proto POP3	Use the POP3 protocol - still probably the most common, but fetchmail appears to support all others.
user username	This is the username needed to login.
password "XXXXXX"	The password to login. This is stored as plain text in the .fetchmailrc file and is the reason why this file is required to be set as read/write for the user only.
is user-name (or "to user-name")	Normally, fetchmail would pass mails to the MTA with a fully qualified RCPT TO address of type hostname@localhost (see FAQ). The "is" makes fetchmail feed mails with RCPT TO address user-name@localhost. For our purposes, however, we use "is *", which makes fetchmail pass mails without modifying the text before the @.
keep	

Setting Default Colors with X

TOC

Created: July 8, 1998
Last updated: Oct 3, 1998
Development stage: Beta

One thing that new users of Linux may find frustrating is that they can't do some familiar, simple things easily. For instance: You've installed Linux, you've started X, and you've decided all the default colors are no good, so now you want to change them. How do you do it?

Unfortunately the answer is a bit complex, so this document is a bit long. If you're totally new to X, you may benefit from reading this whole page. If you've already had some success playing with colors in X, go on and skim through this. First I'll describe how colors are specified in X and how to change things by hand. At the end, I will describe a few utilities that can make your life a lot easer.

Changing Specific Application Colors

Some modern X applications, such as Netscape Navigator, have their own interactive color-customization menus or dialog boxes. But for many applications, such as the many variants of Xterm, colors and many other default characteristics (fonts, window size, etc.) are traditionally set by writing an appropriate statement in a text file. Usually the file in question is called .Xdefaults, and it resides in your home directory. (If it doesn't, go ahead and make one! Look for a file called */etc/skel/.Xdefaults* , which you can copy and use as a template for your ~/.Xdefaults, and modify however you like.)

By the way, remember that files that start with a dot, aka "dotfiles", are normally hidden: that is, **ls** won't show them unless you give the **-a** ("list all") flag. Configuration files are usually hidden this way because you don't want to bother with them very often, and it also provides a bit of protection against carelessly deleting them, because **rm *** will only delete "visible" files.

The characteristics that are controlled by the .Xdefaults file are called "Resources." The man page or other documentation that comes with the application should tell you what resources are available for that app. Or you can get a list of the resources with the **appres** command. For instance, **appres xterm** will show you all the resources for xterm with their current settings. Sometimes the resource names have obvious meanings. For instance, `xterm*cursorColor: Green` is easy to read. Otherwise, you can find out what the resource does by reading the man page.

You can usually, if you prefer, specify the colors at the command line. For example, **xterm -bg blue** is the same as putting `xterm*background: blue` in .Xdefaults. Again, the man page for the application in question will tell you what command line options are available. This method is probably quicker for trying out changes. The disadvantage is that if you try to control a bunch of things, you end up with a very long command line. That's not such a problem if you set up the long command as an alias, but .Xdefaults is probably easier to read and modify.

But what about window managers?

A notable exception to controlling things with .Xdefaults is your window manager. Whichever window

TOC

manager you run - FVWM2, FMWM95, AfterStep, WindowMaker, Enlightenment, TWM, etc etc e probably has a separate file, or a whole separate directory, where its configuration is stored. And the variables that set colors are different for each window manager. Great. Fortunately, many modern window managers, such as WindowMaker, are adding on-the-fly configuration tools that can make it a lot easier.

What about this "app-defaults" stuff?

Since this is linux we're dealing with, there's always at least a few ways to do anything. So, just to mess with your mind, some applications have *another* configuration file: it has the same name as the application, and it goes in the `app-defaults` directory (which is `/usr/X11R6/lib/X11/app-defaults/` on my system). Some applications have their colors specified by an app-defaults file. Just find the lines that have to do with colors and start playing around. I haven't found myself editing app-defaults files very often.

Back to our story...

In X, colors can be specified in two ways: by giving an RGB (Red-Green-Blue) value, like rgb:BE/BA/BE , or by using an "abstract" color name, like gray 72. (Incidentally, these examples represent the same color.)

RGB values will seem strange to those not familiar with them. An RGB value is really three values: one for red, one for green, one for blue. Just to make things confusing, the numbers are usually hexadecimal, where the letters A - F are used to represent the numbers 10 through 16. And since we're in base 16 rather than base 10, the number "10" now represents one more than F, which means hexadecimal 10 = decimal 17.

As an example, `rgb:00/00/00` means "black", and `rgb:FF/FF/FF` means "white." What about something like `rgb:FF/00/45`? This means red = FF (the highest possible value, equal to 255 in decimal notation); green = 00 (the lowest possible value); blue = 45 (a medium-low value, equal to 68 in decimal notation if I did the conversion correctly). Lots of red, no green, a little blue... probably this is a nasty brilliant magenta color.

There's an alternative notation, too: `#ff0045` is the same as `rgb:FF/00/45`. You may have seen this notation used in web page design.

Or you could just specify the color as "IckyBrightRed" and leave it at that. Seems simpler. But there's a snag ... is IckyBrightRed a valid color name? And if so, what's it look like? Keep reading.

The Good-Old-Fashioned Trial and Error Method

Let's try a concrete example. In an xterm, type this:

```
xterm -bg rgb:20/A9/25 -fg black -ms Goldenrod
```

This starts a new xterm. As you can see, the cursor is a nice golden color, the text is black, and the background is a bit heavy on the green side. Maybe try something less bright than A9 for the green value. Usually it's easiest to change in big steps first, then fine-tune. So next we might try something like `rgb:20/80/25`.

How did I know what flags to set? I typed **xterm -h**, of course.

Once you have something you like enough to use all the time, put it in your .Xdefaults file. For example:

```
xterm*background: rgb:20/A9/25
xterm*cursorColor: Goldenrod
```

Setting the Desktop Background Color: The Traditional Way

Read **man xsetroot**. It's short and to the point. The basic idea is that if you type **xsetroot -solid rgb:33/33/55** from an xterm, your background will be instantly set to the color you've typed. You can use abstract color names if you like. Once you've found a color you like, you can have this be the default background color by putting the same command in your .Xclients file, which simply lists commands to run every time X starts. (Don't forget to put the command in the background by putting an ampersand at the end.)

There's a catch: Your window manager probably has its own commands to set the root window color, and those will take over when the window manager starts. So you will probably have to learn how to do it for your own specific window manager anyway.

But xsetroot is still a reasonably quick way to try out different colors, since you can use bash's command-line editing and just keep repeating the same xsetroot command with slightly different colors, and you'll see each one instantly. Then you can just use the mouse to select the last color you typed, and use the standard X-windows "paste" function (the middle mouse button) to paste this color into the appropriate place in the configuration file for your window manager.

There are other command-line X-background tools: for instance, if you do this:
bggen blue magenta | xv -root -quit
... you will see a background that fades from blue to magenta. Read **man bggen** for details. This method, unlike xsetroot, seems to work fine for me when started from the .Xclients file, so hopefully it will work with your window manager.

Problems with Text-Based Color Customization

There are two problems with abstract color names: First of all, you don't know if there *is* such a color until you find a list of abstract color names your system can display. Secondly, you don't really know how icky, how bright, or how red "IckyBrightRed" is until you actually see it... the names are pretty subjective.

There is a partial answer to these problems: a list of rgb values for the abstract colors exists in */usr/X11R6/lib/X11/rgb.txt* (some systems may put the file in a different place). Actually, you can get the same list by typing **showrgb**, and the list scrolls by. There's about 700 color names in there. Problem is, the values are given in decimal (e.g. 255 instead of FF), which is not what .Xdefaults wants. The other problem is, there's no way to perfectly organize a three-dimensional color map in a two-dimensional text file. So if you want something "Like Goldenrod, but a little greener"... how do you know what that's called? You don't.

TOC

The rgb hexadecimal method is at least quite tweakable: if you want "a tiny bit less red" than `rgb:FA/B2/38`, try `rgb:F9/B2/38`. (That will be such a slight difference you may not even notice it, so try `EA/B2/38` and see if that's not too drastic.) The big problem is, hexadecimal numbers take a little getting used to (remember in school when they taught you that A is one more than 9, but one more than F is 10? No? Me neither). And you will still find yourself typing a number in .Xdefaults, starting the app in question, re-typing the number, re-starting the app, over and over and over until like it. Or you give up and stare at something that hurts your eyes every time you work.

By now you're thinking, "There must be a better way."

Some Color Helpers You May Already Have

Just to get you going in a hurry, I'll mention some things you may have handy that could help a bit.

When started in X, **GNU Emacs** has an option in *Edit => Text Properties => Display Color* that will show and list 64 "abstract" color names, so you can see, for instance, what SpringGreen looks like. The problem is that only 64 colors are listed, and if you have a monitor made since the stone ages, you can probably use a lot more colors than that. But this is better than nothing.

One thing I *do* like about **RedHat**'s default window manager (**AnotherLevel**, a heavily-tweaked FVWM2) is that if you go into the *Preferences* menu, you can choose *Background* and get a nice little utility onscreen. Windows95 people will like this; it has sliders you can use to adjust Red, Green, and Blue levels. You can apply the result directly to the screen's background, or you can choose a pixmap for an image to display. There's a little box that displays the color as you adjust it, and right under that you can see the value of the color you've currently chosen as a 24-bit RGB color like rgb:003E2A

A nice thing about this is that you can, of course, write that color value anywhere it can be used, for instance I've done this to choose colors to put in my .Xdefaults file, like so:

```
nedit*text.foreground: rgb:AB/D9/8A
nedit*text.cursorForeground: rgb:FF/69/00
```

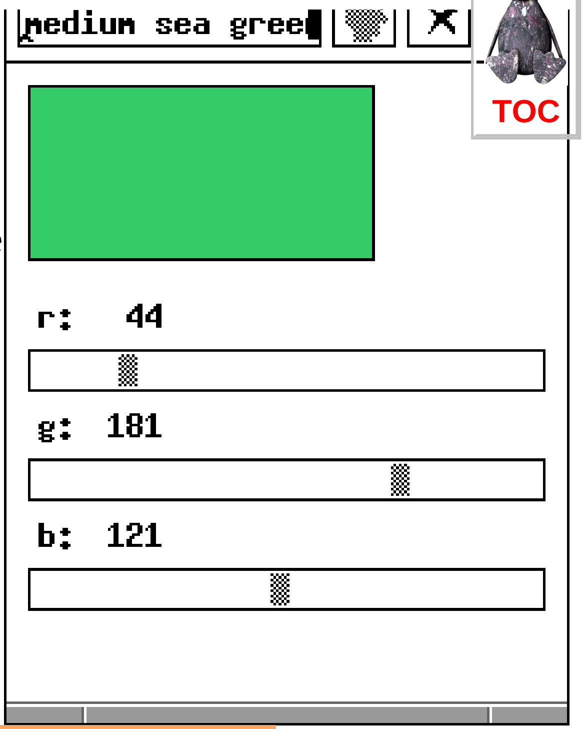
The only problem here is that this doesn't seem to be an independent application; I can't find a way to start it when using any other window manager, and lately I prefer WindowMaker. WindowMaker does have its own menu-based background selection, but there is a very limited set of background colors in the menu.

Some Color Helpers You Can Get

There's a whole bunch of things available via ftp. I found several color-picking applications at ftp://sunsite.unc.edu/pub/Linux/X11/colors/.

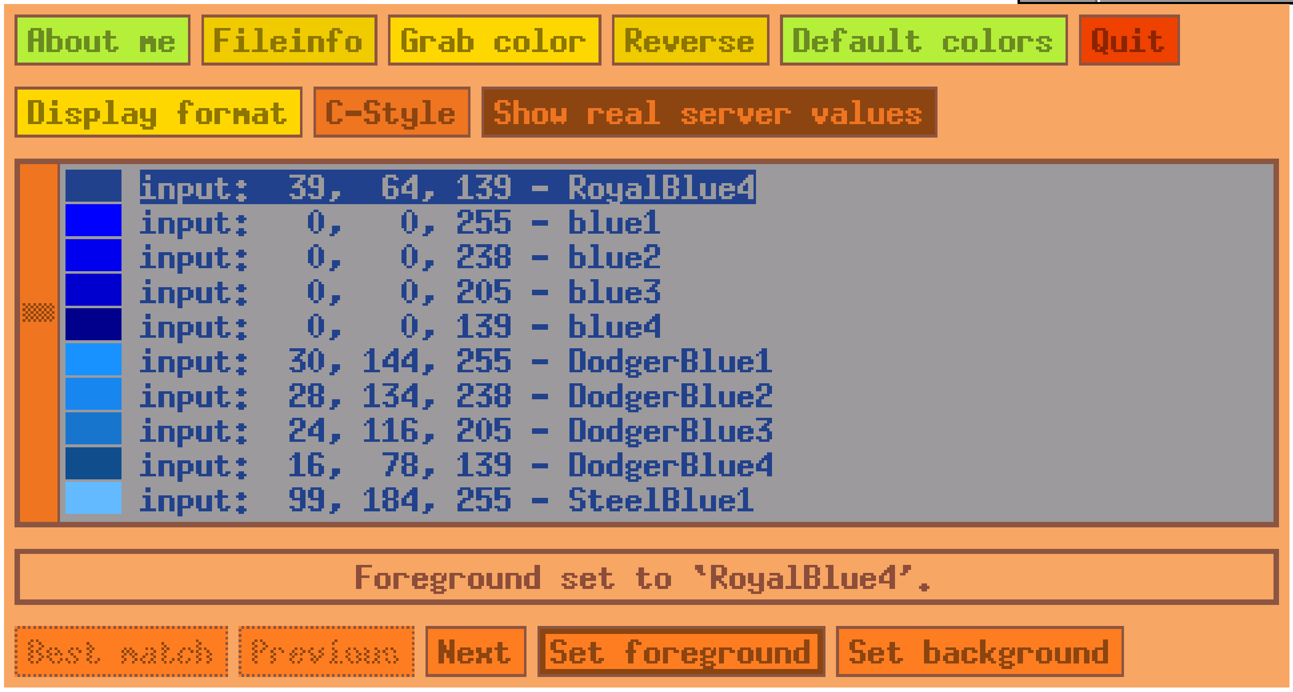
I actually downloaded five apps from sunsite, but three of the five wouldn't compile and/or run on my system, so I tossed 'em. Here's a comparison of the two I did get.

xcolsel by Markus Koelblin, available as xcolsel-1.0.tar.gz. A nice simple app with a well-written man page. There are sliders for adjusting red, green, and blue. There's a little "lookup" feature which will search the rgb.txt database for a near match, and will give you the closest abstract color name. If none is found, there's a beep. There's a command-line flag that controls how close matches must be (I found this handy, since nearly every color I tried to get a name for beeped at me. **xcolsel -near 30** beeped less, but returns some pretty inexact matches. Take your pick). You can also browse through the rgb database and see what all those weird names look like.



```
medium sea gree

r:  44
g: 181
b: 121
```



```
input:  39,  64, 139 - RoyalBlue4
input:   0,   0, 255 - blue1
input:   0,   0, 238 - blue2
input:   0,   0, 205 - blue3
input:   0,   0, 139 - blue4
input:  30, 144, 255 - DodgerBlue1
input:  28, 134, 238 - DodgerBlue2
input:  24, 116, 205 - DodgerBlue3
input:  16,  78, 139 - DodgerBlue4
input:  99, 184, 255 - SteelBlue1

Foreground set to 'RoyalBlue4'.
```

Then there's **xcolorsel** by Michael Weller (not to be confused with XColorSel by Jon Steiger). Look for xcolorsel-1.1a.tar.gz . The disadvantage compared to xcolsel is that there's no interactive color editing. The big advantage is that you can "grab" any color displayed anywhere on your monitor and find out what it is!

It has a scrollable list of colors with their names and rgb values, and each has a little swatch of the color displayed next to it. You can also quickly and easily try out a pair of colors for foreground (text) and background, which is really useful for setting up any text-based application's default colors.

Both xcolsel and xcolorsel are handy in slightly different ways; I recommend that you get both. If I should find out about a more comprehensive single solution, I'll post it here.

You know, I've just decided I'm sick of flat colors. How do I make a tiled background, like

on Windows or Mac?

TOC

Just to be nice I'll give you a quick answer. Make, or get from somewhere, a reasonably small image. OK, here, grab this one.

Now type this:

```
xv -root -quit filename
```

Note that your window manager may have its own way of doing this without using **xv**. For instance, as of version 0.17, WindowMaker can do it with one line in the GNUstep/Defaults/Windowmaker file.

How do you make your own nicely tileable images? With the Gimp, of course. Josh Go has written a nice tutorial on this very subject at http://jgo.local.net/GimpGuide/desktop.html. He also describes how to make the "tiles" that icons sit on top of in, for instance, WindowMaker or Afterstep.

Send comments and corrections to Paul Winkler.

Setting Up Xscreensaver

TOC

Created: July 8, 1998
Last updated: July 9, 1998
Development stage: Beta

This little tutorial is intended to help a Linux newbie to do something fun and hopefully learn some useful Linux commands and concepts along the way. Those more experienced with Linux may find the amount of hand-holding here to be irritating. Well, let me say right off the bat that there's nothing here you couldn't find out from the xscreensaver man page, but for the novice user, this tutorial should be a little easier to follow.

First of all, you should be running X now.

Now check to see if the screensaver is already running. Some Linux distributions may have already put it in one of the files that starts up client applications for X. (These are /etc/xinit/xinitrc, /etc/xinit/Xclients, ~/.xinitrc, and ~/.Xclients.) To see if xscreensaver is already running, do ps af and look at the result. (**ps** is a command that lists the running processes on the system... very handy.)

For instance, part of the output of ps af looks like this for me:

```
  PID TTY STAT  TIME COMMAND
    ...
  252   1 S    0:00 /bin/login -- pw
  260   1 S    0:00  \_ -bash
  271   1 S    0:00      \_ sh /usr/X11R6/bin/startx
  272   1 S    0:00          \_ xinit /usr/X11R6/lib/X11/xinit/xinitrc
--
  276   1 S    0:01              \_ wmaker
  280   1 S    0:00                  \_ xscreensaver
  281   1 S    0:00                  \_ nxterm
  286  p0 S    0:00                  |   \_ bash
  293  p0 R    0:00                  |       \_ ps af
  284   1 S    0:00                  \_ asclock -shape
```

Every process is listed with a **P**rocess **ID** number. Look at PID 280 ... there you see that xscreensaver is running on my system.

If you don't see xscreensaver listed, it's not running. To start it up, type this in an xterm: xscreensaver &. If you get an error message, you may not have xscreensaver installed on your system. Try locate xscreensaver or find / -name "*xscreensaver*" and see if that turns anything up. If not, you need to get it... it may be on your distribution CD-ROMs (it comes with RedHat, for example), or look at the Xscreensaver home page.

Choosing Screensaver Graphics

Anyway, now you've got xscreensaver running, but it probably isn't doing anything yet. The next thing you want to do is type xscreensaver-command -demo . This puts xscreensaver into an interactive demo mode, so you can see which screensavers you like.

Also at this time, you should use your favorite text editor to open the .Xdefaults file in your home directory. If you don't have one yet, go ahead and create the file. .Xdefaults is a file that contains default options for many programs that run in X. Nearly any X application will have a man page or other documentation that describes its "X resources"... that is, the options that can be set in a user's .Xdefaults file. What we're going to do now is add some entries to .Xdefaults that tells xscreensaver which graphics program(s) to run.

The xscreensaver demo is pretty easy to figure out. It blackens the screen and pops up a little window where you can select which graphics program (which are generally referred to as "hacks") to run. When you select one, it starts running and the window gets hidden. To bring back the demo dialog window, just right-click the mouse. You can scroll through the list, jump to the next or previous graphics hack, etc. When you find a graphics hack you like, add it to your .Xdefaults file. Here's what you want to put in:

```
xscreensaver.programs:  \
                        rocks -root     \n\
                        hopalong -root \n\
                        flame -root \n
```

This is just an example; I like these three graphics hacks. The backslashes are necessary so that all of this is read as one line. Each graphics hack listed must have the `-root` flag, to tell it to draw on the "root" window (i.e. the whole screen), and each hack entry must end with \n\, except the last one which must end with \n ... it looks weird to the uninitiated, but that's just how it works.

Do xscreensaver-command -demo again, and try all the graphics hacks, adding each one you really like to your .Xdefaults file. When you're done, save .Xdefaults, and click the "Reinitialize" button in the xscreensaver demo. Now only the hacks listed in your .Xdefaults are listed by the demo, and only these will be run when the screensaver starts.

Getting the Screensaver to Start

If you want the screensaver ready to go all the time when you're in X, put this in your .Xclients file:

```
xscreensaver &
```

As with anything you put in .xinitrc or .Xclients, the command shouldn't be the last thing in the file. Usually your window manager is the last thing started. (On RedHat systems, /etc/xinit/xinitrc does some auto-detect things and starts the window manager; probably everything you personally want to start should go in ~/.Xclients.) The command you add must end with an ampersand to put that process in the background; otherwise, X will exit if the process is killed, and you don't want that behavior from anything but the window manager.

You can manually start the screensaver at any time by typing xscreensaver &, and then if you want to test it out quickly, you can make it immediately take over the screen by typing xscreensaver-command -active. If the screensaver is already running and you want it to load some new resources you've just added to ~/.Xdefaults, simply type xscreensaver-command -reset.

Controlling Xscreensaver's Behavior

By default, xscreensaver will start if the user is inactive for ten minutes. At that time it will randomly pick one of the graphics hacks in your .Xdefaults file. Ten minutes later it will randomly select another one. You can change all of this in your .Xdefaults file.

For instance, if I put this in my .Xdefaults:

```
xscreensaver.timeout: 2
xscreensaver.cycle: 1
```

...Xscreensaver will now start after 2 minutes of inactivity, and will switch to another graphics hack every 1 minute after that. If xscreensaver.cycle is set to 0, then no cycling will take place. All of these options can also be set by the command line that starts xscreensaver. There's many other options. man xscreensaver for details.

Tweaking the Screensaver Graphics

We haven't mentioned this yet, but the graphics hacks that the screensaver runs are actually completely separate X programs, and they all have command-line options that change the way they look and behave. For instance, run flame. With no arguments, a new window is created and the "Flame" graphics hack runs in it. If you instead type flame -help, you'll get a listing of the options flame can take. These are described in the flame man page (yes, man flame). For instance, if we do flame -delay 100 -delay2 10000, it runs a lot faster.

You can play around with command-line options for all your favorite graphics hacks, and when you've got one you like, put the same command line flags into the appropriate entry in your .Xdefaults file. For instance, I've got this line:

```
                  kaleidescope -root -nsegments 5 -ntrails 50 \n\
```

...which looks a bit different than the default kaleidescope.

Locking the Screensaver

You can easily set up xscreensaver to "lock" ... that is, it won't let you back in to your system without your password. This could be useful if you have stuff on your system you want to keep private from anyone who might wander by, like at work or something I guess. In the demo, you can turn it on by clicking "Edit Parameters" and then "Require Password." Then type xscreensaver-command -activate to turn the screensaver on, and you'll notice that it asks for your password when you try to do anything.

If you like this feature, put this line in your .Xdefaults file:

xscreensaver.lock: True

TOC

One More Cool Trick

This one is guaranteed to SURPRISE and AMUSE YOUR FRIENDS!TM

This isn't really part of xscreensaver, but it's fun anyway. You can run any of the graphics hacks on your root window all the time, and it will just stay there merrily animating away behind all the stuff you work on. For instance, put this line in your .Xclients or .xinitrc if you really like to get dizzy and distracted while you work:

```
kaleidescope -root -nsegments 5 -ntrails 50 &
```

Of course, you can use any of the hacks you like instead of kaleidescope. You could even pick one that doesn't give you motion-sickness, but where's the fun in that?

That's all!

Send comments and corrections to Paul Winkler.

Sharing files and printers over a home network (Samba)

TOC

Created on 30 September 1998
Last updated on 15 April 1999
Development stage: ver 1.1

This guide was written to be part of Josh's Linux Guide

Introduction

This guide was written while I was setting up my own home network as a means of remember the steps involved, but I trust it may prove useful to others too. It assumes a TCP/IP ethernet connection has been established (ping works, /etc/hosts has been completed), although it is not necessary that all the computers run the same operating system (OS) - OS/2 and Windows 95/8 can easily be configured to share with Linux (as I suspect can NT but this is not covered below). My experience in this area is based on RedHat 5.1, but I think the guide is applicable for other distributions.

Warning: I should emphasise that the following is for a home network, where I presume security issues are not pre-eminent. But if you also connect to the net, you will want to stop any unwelcome atempts to access your shares. A few tips I've found are mentioned below, but if security is crucial, please check the Samba documentation more extensively.

As the ultimate network OS, Linux can act as the server in a client/server network but, for home purposes, it is simpler to aim for a 'peer' network, where each computer is a server and client simultaneously. Although the computers can talk to each other quite straightforwardly, there appears to be no harmonisation of the terminology: we shall be using Samba (and the SMB protocol) on Linux, and connecting to other Linux machines or peers running other OSs with NETBIOS, NETBEUI, LAN Manager or (ports of) Samba.

I am no expert in these matters, and would welcome comments and suggestions from those more knowledgeable.

Installing Samba

The Samba rpm is included with RH5.x, but you may need to install it manually - the latest version is available from here. Samba includes both the server and a client program. Alternatively collect the latest version in tgz format, and install it by hand.

Next you wil wish to ensure that the daemons (**smbd**, the Samba server, and **nmbd**, the NETBIOS server) are started at bootup by inetd (this can be cofigured in RedHat using /usr/sbin/ntsysv)

Warning: If smbd is started every time you boot up and this may (I'm not sure exactly what the risk is) expose parts of your system when you are on the net. **Be warned!**

TOC

Samba server: the /etc/smb.conf file

The directories and files available to peer computers are configured in the file /etc/samba.conf. The man pages for smb.conf run to 66 pages, but the following should be enough to edit the smb.conf installed by the rpm and enable your peers the simplest, relatively unfettered access to the samba server.

Note: the smb.conf file is case insensitive, is regularly checked for changes (useful while setting things up) and that, if it is not obvious, lines beginning with ; are ignored.

The [global] section

This section sets certain general parameters. If you want to enable browsing of your shares (e.g. using *Windows Network Neighbourhood*), you will need use the following and ensure that you set the same workgroup for all computers on your network - this is done differently in each operating system - and that an account "pcguest" (you may be able to use the nobody account set up by some distributions, but I don't undertsand this concept yet) has been created (with no password):

```
    workgroup = MYGROUP
    guest account = pcguest
```

For a home network, a couple of parameters can be put in the config file which help shut off any unintended access.

```
    hosts allow = 10.2.1.2/255.0.0.0
    ; hosts allow = 10.2.1.
    security = USER
```

Note: with version 2.0 of samba you can use the syntax in the commented line instead, and without the "/subnet mask" part too.

The [user] sections

In contrast with the global settings which affect all shares, the user settings are specific to each share. Consider the following extract from my smb.conf.

```
    [sim]
    Comment = "Simon's Home Directory"
    path = /home/sim
    valid users = sim
    public = no
    writable = yes
    ; wide links = no
```

The section name (line 1) is the name of the share that will be reported to the client - comment (line 2) adds more detail if necessary. Line 3 identifies the directory to be served. For the security conscious, line 4 specifies exactly which users (which will need to have already been set up) are allowed to access to this service (note that you may need to check with your client program what userid it passes to Samba, and its treatment of capital letters etc), and line 5

TOC

confirms (rather superfluously) that this is not a public service. Line 6 permits the user to w
to this directory (and its subdirectories). Line 7 would stop users following symlinks outsid
this specific part of the directory tree.

The [printers] section

The following is what I use successfully in my smb.conf.

```
[printers]
comment = All Printers
path = /var/spool/samba
printable = yes
read only = true
; public = yes
; browseable = no
; writable = no
```

Many of the parameters are discussed in the [users] section, but this section must be set to "printable". If included, users are able to connect to any printer specified in the server's /etc/printcap file.

The [homes] section

Completing this section will allow the Samba server to create user accounts on-the-fly - in a workplace environment, Samba may need to be configured to provide flexibility for a large, and potentially changing, set of clients. That is to say, it will open a default service for anyone using a known userid in the /etc/passwd file.

Smbclient

smbclient has been developed, and is distributed, with the samba package. It provides a command-line, ftp-like interface to shared resources. To quickly check whether the connection is basically working, try:

```
smbclient -L NETBIOS_Name
smbclient \\\\NETBIOS_Name\\ServiceName
```

Note: the extra backslashes (\) are needed because this is a reserved character under Linux.

You can use the -N flag if you are connecting to a public share and don't want to be prompted for a password.

In order to resolve names, smbclient looks in /etc/lmhosts first and then /etc/hosts.

Mounting remote services

In order to ensure seamless (meaning GUI) access to remote services, you will want to mount them in just the same way as local hard disk partitions. The two programs available which do this are *not* part of the Samba project, although smbmount is currently part of the Samba rpm (but see section 16 of the Current samba-bugs FAQ, which claims that the code is reported not to work well). To make things more complicated they use similar, but nonetheless different, command syntax to smbclient.

Note: I understand, and this is certainly my experience, that neither of these two solutions currently work with 2.1.x kernels.

smbmount (part of the SMBFS package)

In my own own experience, smbmount was not straight forward. I did, however, mount a simple, anonymous Win98 remote filesystem to the /mnt/samba:

```
smbmount //NETBIOS_Name/ServiceName /mnt/samba -[n]I
TCP/IP_address
```

Notes:

- the forward slashes are deliberate
- The 'n' can be inserted if the share has no password protection;
- smbmount, unlike smbclient, only tries to resolve names in /etc/lmhosts. In practice, even having a lmhosts entry did not work for me, hence the explicit inclusion of the IP address in the command line.

In order to login to a samba linux machine, I needed to use the -U parameter:

```
smbmount //NETBIOS_Name/ServiceName /mnt/samba -U login_name
```

as I have not found a way of passing on the login name used to login to the client machine. Subsequently, I am prompted for the password - in other words, I need to login twice, whereas for other OS once is enough but perhaps I've missed something?

Sharity and Sharity-light (formally rumba)

Sharity is commercial product (although free for academic use) and may well represent the future for Linux clients, but I have not tried it yet.

Sharity-light, however, is a workable product and does the same thing as smbmount (indeed both have their roots in smbfs). However, as it runs as a user level program, the the author admits that it is slower (although not noticeably). Nonetheless, sharity-light is what I now use to mount remote file systems and I am perfectly satisfied. The command syntax is the same as for smbmount, but here I find I need to force it pass the password:

```
shlight //NETBIOS_Name/ServiceName /mnt/samba -P <password>
```

LinNeigborhood

A really superb little app in the making that gives a Windows like network neighbourhood.

Samba and KDE

Two KDE aware programs are available to help ease smb.conf configuration:

- KNetMon
- KSAMBA

OS/2 setup

Since this is a Linux guide, I'll be brief when talking about other OSs. For OS/2 Warp 4 you will want to install TCP/IP, File/Print Services, and NETBIOS over TCP/IP. Robert Thomas and Frank R. Field have written excellent guides in English about setting up these services (although they had NT in mind as the server/peer), while Thomas Baumann covers similar ground in German.

Note: to use the OS/2 client software, it is important that you do ***not*** have lm announce = false in smb.conf (the default, auto, is fine).

There is also an OS/2 port of Samba available, Samba/2, which I presume would work easily although I have not tried it.

Windows setup

I'll say even less about BillyBoy's Win 9X - try this guide for basic TCPIP configuration, and see here for handling encrypted passwords.

As for **NT**, I receviced the following from Jerry Sternesky.

"I had to enable password encryption and can not log in as administrator. Linux equates this with root and if you have remote login for root turned off it will not happen. Enabling this encryption for the password did not knock the win 98 box out, other than those 2 caveats NT appears to work the same as Win 9X."

Apple

This is really not my domain, but try this

Things "To do"

As and when time permits, I plan to add:

- check conformity of advice with Samba 2.0 - note that this distributions has an excellent help file if things are not working (see /usr/doc/samba-2.0.0/docs/txtdocs/DIAGNOSIS.txt)
- a section on Winpopup using message command =
- discussion of available Gnome/E software
- discussion of name mangling

TOC

Links

An(other) excellent guide can be found at: http://www.sfu.ca/~yzhang/linux/samba/toc.html

---

Using SOCKS Proxy Firewalls on Linux

TOC

Written by: Alex Feinberg (alex@freethinkers.net)
Created on: December 24, 1998
Last updated: December 29, 1998
Development stage: Alpha

Many companies are now turning to Linux as a solution for cheap and reliable workstations. Many times they set those machines up right at employee's houses, so they can "tele-commute" to work. If you are in that situation, chances are your network is behind a *firewall*. Usually companies use *socks*-type proxy firewalls. Those firewalls allow for safety, monitoring yet reasonable privacy as well. However, you can't use many of those neat network applications such as IRC clients, e-mailers, news readers, FTP clients, telnet to outside the corporate network. This is a guide for those wanting to use the firewall to access outside machines with programs that don't readily support socks. This is also a guide for system administrators for configuring new machines, as all of this would apply to most any other Unix as well, with few differences.

First part is find out whether we have socks5 or socks4. Good idea is to ask the system administrator, or just do trial tests. Socks5 is the best protocol, let's try it first if we don't know whether we have socks5 or socks4. Also, if your firewall is *wingate*, it's socks5. If it's a unix machine, it's probably socks5, but it might be socks4 as well. Now we need to determine the hostname of the firewall machine. Just check the settings in Netscape or other web browser on your work machine.

Now we know the name of the server. We should now get the package itself so it all works. Like I said earlier, first let's try socks5, because it's better and most likely to be there. Download socks5 libraries with Netscape (which readily supports socks without any work) from ftp://ftp.nec.com/pub/socks/socks5. Download the tar.gz package extract it (with tar zxvf file.tar.gz), cd to the directory formed. There run ./configure; make; make install. All done! If these steps went fine, we've compiled the libraries! Now we need to configure the libraries to work with your network. All we do is edit `/etc/libsocks5.conf`. Here's what mine looks like:

```
noproxy - 1.2.3. - -
socks5 - - - - 1.2.3.4
```

Now let's try to understand what all this means. 1.2.3. is the mask of the network I am on. This way if I'm using a socksified client to connect to a host inside the network, it won't be going through the firewall. 1.2.3.4 is the firewall itself. Now let's try out the configuration. Use some of the premade clients built with socks5, such as rtelnet. Type rtelnet netscape.com. Wait for a reasonable time (up to ten minutes). Socks connects might be slow. Once it's connected you've done it! Now we can use runsocks to connect "socksify" anything else we want to, simply type runsocks program args. Simple huh?

In case we're still socks4, procedure is rather similar. Download socks4 from `ftp://ftp.nec.com/pub/socks/socks4` and build it the same way. Configuring it, however, **is** different. Here's an example configuration file (`/etc/socks.conf`):

TOC

```
direct firewall 255.255.255.255
sockd  @=1.2.3.4 0.0.0.0 0.0.0.0
```

Not rocket science, huh? Firewall is the hostname of the firewall server, 1.2.3.4 is it's ip. Test it out and use it similarly.

For more interesting configuration examples, use examples, problems other things check out http://www.socks.nec.com.

About GNU/Linux and Josh's Linux Guide

TOC

- GNU/Linux
- Josh's GNU/Linux Guide

About GNU/Linux

GNU/Linux is a free operating system very much like UNIX, an operating system developed for multiple users on supercomputers in the late '70s. Inspired by an earlier UNIX-like operating system called Minix, Linux was developed by a University of Helsinki (in Finland) student by the name of Linus Torvalds (he now lives in California's Silicon Valley which is where I live also... this came in handy when I got to see him in real life when he spoke for the Silicon Valley Linux Users Group). Linus began work on it in late 1991 and along with the help of thousands of programmers the world over, it is continually being developed up to now. The portion that Linus is responsible for is putting together the system kernel, not user applications.

Although UNIX (and other operating systems derived from it) are usually thought of as servers, there are countless client applications, most of them easily interchangeable between different Unix platforms. There are even Windows/Mac emulators that are available and/or currently in development that will let the user run programs for those platforms.

GNU/Linux is known as a hacker's operating system due to its non-commercial development method. Source code for programs is made available so that patches to the code itself as well as bug reports may be sent to the author or maintainer of the software. (The reason I refer to it as GNU/Linux instead of simply "Linux" is because the Linux kernel itself makes up only a tiny portion of the operating system. Software written by the GNU Project makes up a significant portion but are often overshadowed by the Linux kernel.)

Co-existing with other operating systems is not a problem for GNU/Linux. It is easily set up on computers that already have Microsoft Windows 95, OS/2, Microsoft Windows NT, other implementations of Unix, and most others. In fact, it is very common for people to have Microsoft Windows and GNU/Linux on the same computer. Linux supports all major filesystem types as well as some lesser known ones.

Many programmers use UNIX-like operating systems to write their software, and Linux is no exception. There are free compilers that are bundled with most standard Linux distributions such as GCC and G++, the standard C and C++ compilers, respectively. All one has to do to install these is enable installing development packages in the distribution's installation program. Sun Microsystem's Java Development Kit (JDK) is also available for GNU/Linux. It is no wonder GNU/Linux is the choice platform of development for many programmers.

The graphical user interface that comes with many Linux distributions is called the X windowing system (most people just call it X). It's used by people who run graphical programs such as Netscape Navigator or people who don't like typing in commands at a command prompt. The look of the windows (titlebars, menu, borders, etc.) are highly configurable and are handled by software packages called window managers.

Networking support is already quite mature under Linux. The main protocol Linux is used for is TCP/IP. A GNU/Linux box can be set up as a firewall, a gateway, a webserver, a mail server, any combination of these, and many other things that I will not be able to list because I want to live my life. The average user who doesn't want to run their GNU/Linux box as a server can just as easily set up a dialup PPP connection to his ISP.

Printing under Linux has been achieved by many of its users, most of which are former Microsoft Windows users. Dot matrix, inkjet, and laserjet are supported through printer drivers and filters. A warning for those of you who have yet to buy a printer for Linux: Do **NOT** buy a printer that says "Exclusively for Windows" if you intend to use it for Linux!

Josh's GNU/Linux Guide: A Brief History

I started writing this GNU/Linux guide around January of 1997. The guide was originally called "Josh's Linux Page", and not surprisingly, started out as a single page. The first section was a guide on how to install the CD-ROM distribution of Red Hat Linux 4.0, my first Linux distribution. The inspiration that started the guide was frustration during my first attempted installation. The motivation that keeps this guide growing is the reader feedback received, which in my opinion is a fairly large amount (more than 3 messages a day, for me, is a fairly large amount).

Ironically, when this guide was first being written, I was writing it under Windows 95. I had to work with a 386 at that time, so I couldn't view the contents in Netscape under Linux (mostly because it took about 10 minutes for Netscape 3 to start up).

I use **vim** for this site. I used to use **pico** just because it had automatic paragraph wrapping, but I found that I liked the pretty colors of **vim**'s syntax highlighting enough to ditch **pico**.

This guide is mirrored in case the main server might be down. I also keep multiple copies of this guide around so that in case one server crashes, I can just get the copies off of another server.

This GNU/Linux guide is intended for beginners, but some of the more advanced users may learn a few things in my attempt at successful exchange of information.

Comments, questions, suggestions, **corrections**? Send 'em to jtg@computers.iwz.com. You can also use the guestbook or help form.

Authors

TOC

Joshua Go is the main author of this guide. His homepage is at http://www.local.net/~jgo/. His interests include writing, gardening, reading sci-fi and legal novels, basketball, and displayed graphics. He can be reached at jtg@computers.iwz.com. He is often known to be lazy and tends to choose the easy solution to problems. He has not yet decided what he is going to do for a living, but he is considering being an author, going into the ministry, going into the medical field as some sort of practitioner, or, write Open Source software.

Shorti (whose real name also happens to be Josh), has somehow managed to contribute pages to the guide along with some (obsessively) detailed tips, despite his vast laziness. He can be reached via e-mail at jbm@intertek.net. His interests include: mucking about with Linux - programming, writing documentation, creating graphics, and anything else with a good hack value; music; physics; electronics; and spending time with his friends (most seem to be almost normal people, aside from the fact that they associate with him). He uses parens in funny (and always Wrong) ways a lot and tends to use such obscenely technical terms as "*thingy*". He is far too lazy to write much here, so go to his homepage, Shorti's Place, to find out more about him... you creepy people who actually read the authors page ;^)
