Linux Administration Made Easy

by Steve Frampton, <frampton@LinuxNinja.com>

The "Linux Administration Made Easy" (LAME) guide attempts to describe day–to–day administration and maintenance issues commonly faced by Linux system administrators. Part of the Linux Documentation Project.

Table of Contents

Table of Contents

Chapter 1. Preface

1.1. Acknowledgements

I would like to thank the Linux community; particularly those members who have participated in USENET and mailing lists with lots of helpful tips, answers, and suggestions on how to use Linux at its best. Your contributions have benefited us all.

This document was written in the DocBook SGML format, and then rendered using SGMLTools 2.x to a variety of document formats, including HTML, postscript, Rich−Text−Format, and PDF. For more information on SGMLTools, see the project web site at http://www.sgmltools.org/

1.2. Copyright Information and Legal Disclaimers

I've written this documentation and am providing it free to the Linux community as a public−service. I have made every attempt to ensure that the information contained herein is timely, accurate, and helpful, but in no way will I be held liable for any damage(s) caused directly or indirectly by any misinformation contained herein.

I will not appreciate being flamed for any errors or omissions. However, if you notice a glaring inaccuracy, or have suggestions for further improvement, please, let me know. However, please check the version number and date of this document (see the table of contents) to ensure you are looking at the most recent version. If this document is more than three months old, please check the Linux Documentation Project home page at http://metalab.unc.edu/LDP/ in case a newer version is available.

This document, currently, should be considered moderate−beta. I began writing it in 1997, and continue to update it as time permits. Development in the Open Source community continues at a rapid pace, and at times it is a challenge to keep this document up to date. As such, this document may have one or more sections which contain obsolete information.

In short, I make no guarantees for any of this information to be correct. If it helps you out, that's great!

1.3. A Plea for Help

If you find this document useful and would like to express your appreciation for it, please consider donating a food item or two to your local food bank.

Chapter 2. Introduction

Linux 2.2.0, released 25−Jan−99: Onwards to World Domination...

Perhaps you are fairly new to Linux and were hoping to find a summary of the kinds of configuration and administrative tasks that may be required of you from time to time. If this sounds like you, perhaps this document is just what you've been looking for!

2.1. Scope

This documentation will attempt to summarize the installation and configuration, as well as the day−to−day administrative and maintenance procedures that should be followed to keep a Linux−based server or desktop system up and running. It is geared to an audience of both corporate as well as home users. It is not intended to be a full overview of Unix operations, as there are several good texts available as well as on−line documentation which can be referred to in cases where more detailed information is required.

In general, your Linux system can operate with a minimum of user maintenance. Routine tasks, such as rotating and discarding of system logs, are automated. Therefore, for the most part, even with very little user intervention, Linux will hum along doing its job. However, in cases of custom needs or system failure this documentation may prove useful.

I currently use Linux both at home and at my place of employment. It has served me well, and has worked as a reliable Internet and file/print service for my employer for over four years now.

2.2. Choosing a Linux Distribution

There is quite a variety of Linux distributions from which to choose from. Each distribution offers the same base Linux kernel and system tools, but differ on installation method and bundled applications. Each distribution has its own advantages as well as disadvantages, so it is wise to spend a bit of time researching which features are available in a given distribution before deciding on one.

The following is a list of a few web sites you can visit, which will describe a given Linux distribution as well as provide information on how you can download or purchase it:

http://www.redhat.com/

> The Red Hat distribution, by commercial vendor Red Hat Software, Inc. is one of the most popular distributions. With a choice of GUI− and text−based installation procedures, Red Hat 6.1 is possibly the easiest Linux distribution to install. It offers easy upgrade and package management via the ``RPM'' utility, and includes both the GNU Network Object Model Environment (*GNOME*) and the ``K Desktop Environment'' (*KDE*), both popular GUI window managers for the X Window System. This distribution is available for the Intel, Alpha, and Sparc platforms.

http://www.debian.org/

> The Debian distribution, by non−profit organization known as "The Debian Project" is the darling of the Open Source community. It also offers easy upgrade and package management via the ``dpkg''

utility. This distribution is available for the Intel, Alpha, Sparc, and Motorola (Macintosh, Amiga, Atari) platforms.

http://www.suse.com/

The S.u.S.E. distribution, by commercial vendor S.u.S.E., is another popular distribution, and is the leading distribution in Europe. It includes the ``K Desktop Environment'' (*KDE*), and also offers easy upgrade and package management via the ``YaST'' utility. This distribution is available for both Intel and Alpha platforms.

http://www.caldera.com/

The OpenLinux distribution, by commercial vendor Caldera, is aimed towards corporate users. With the new OpenLinux 2.2 release, Caldera has raised the bar with what appears to be the easiest to install distribution of Linux available today. In addition, it comes standard with the ``K Desktop Environment'' (*KDE*). This distribution is available for the Intel platform only.

http://www.linux−mandrake.com/

The Mandrake distribution, by commercial vendor MandrakeSoft S.A., integrates the Red Hat or Debian distributions (your choice) with additional value−add software packages than those included with the original distributions.

http://www.slackware.com/

The Slackware distribution, by Patrick Volkerding of Walnut Creek Software, is the grandfather of modern distributions of Linux. Offers a fairly simple installation procedure, but poor upgrade and package management. Still based on the libc libraries but the next version will probably migrate to the newer glibc. Recommended for users who are more technical and familiar with Linux. This distribution is available for the Intel platform only.

Listing all the available distributions is beyond the scope of this document, so I've listed only the most popular. However, further information on the available distributions can be found in the ``*Distribution−HOWTO*'' guide, available at http://metalab.unc.edu/LDP/HOWTO/Distribution−HOWTO.html

Tip: Tip: If you decide to buy your distribution on CD−ROM, you might be able to find better pricing at other resellers (for example, I've been quite satisfied on several dealings with Internet−based software vendor http://www.cheapbytes.com/). On the other hand, you may wish to pay the higher price to the distribution vendors to ensure that their offerings continue to improve.

My distribution of choice is Red Hat Linux (it also happens to be, unarguably, the most popular distribution among Linux users). For almost three years, I was a die−hard Slackware fanatic (before that I had messed around a bit with a small distribution from tsx−11 way back in the kernel 0.90a days), and although I've tried Red Hat in the past, I never could bring myself to say anything good about their distributions. Then, I tried Red Hat 5.1, and found myself quickly converted! In my opinion, with 5.1, Red Hat finally "got it right".

Some of the reasons I have become a fan of the Red Hat distribution include the ease of installation, multi−platform support (until recently, Red Hat was the only distribution vendor to provide its distribution for Intel, Alpha, and Solaris platforms), and, above all, the RPM package manager. In addition, they put updates to included RPM's on their FTP site (at ftp://ftp.redhat.com/redhat/updates/) as they become available, which is a good way of keeping one's system up to date and free of any bugs or security problems that are discovered from time to time.

Since first loading Red Hat 5.1 on an otherwise unused computer at work for testing purposes, I have converted two of our main Internet/File & Print servers over from Slackware to Red Hat and haven't regretted it. I've also loaded it on my system and home, and installed it on three other systems as light servers as well. In addition, I have had the opportunity to not only play with the Intel−based versions but with Alpha− and Sparc−based versions as well. Recently, I've moved all the Linux systems I am responsible for over to Red Hat 6.1.

Therefore, this document has a definite Red Hat "feel" to it, and is most relevant for the Intel−based 6.1 version. However, hopefully most or at least some of the information contained in this document will be useful to users of other distributions.

Chapter 3. Linux Overview

Welcome to Linux!

3.1. What is Linux?

Linux is a true 32−bit operating system that runs on a variety of different platforms, including Intel, Sparc, Alpha, and Power−PC (on some of these platforms, such as Alpha, Linux is actually 64−bit). There are other ports available as well, but I do not have any experience with them.

Linux was first developed back in the early 1990s, by a young Finnish then−university student named Linus Torvalds. Linus had a "state−of−the−art" 386 box at home and decided to write an alternative to the 286−based Minix system (a small unix−like implementation primarily used in operating systems classes), to take advantage of the extra instruction set available on the then−new chip, and began to write a small bare−bones kernel.

Eventually he announced his little project in the USENET group comp.os.minix, asking for interested parties to take a look and perhaps contribute to the project. The results have been phenomenal!

The interesting thing about Linux is, it is completely free! Linus decided to adopt the GNU Copyleft license of the Free Software Foundation, which means that the code is protected by a copyright −− but protected in that it must always be available to others.

Free means *free* −− you can get it for free, use it for free, and you are even free to sell it for a profit (this isn't as strange as it sounds; several organizations, including Red Hat, have packaged up the standard Linux kernel, a collection of GNU utilities, and put their own "flavour" of included applications, and sell them as distributions. Some common and popular distributions are Slackware, Red Hat, SuSe, and Debian)! The great thing is, you have access to source code which means you can customize the operating systems to your *own* needs, not those of the "target market" of most commercial vendors.

Linux can and should be considered a full−blown implementation of unix. However, it can not be called "Unix"; not because of incompatibilities or lack of functionality, but because the word "Unix" is a registered trademark owned by AT&T, and the use of the word is only allowable by license agreement.

Linux is every bit as supported, as reliable, and as viable as any other operating system solution (well, in my opinion, quite a bit more so!). However, due to its origin, the philosophy behind it, and the lack of a multi−million dollar marketing campaign promoting it, there are lot of myths about it. People have a lot to learn about this wonderful OS!

3.2. Breaking the Myths

I've been using Linux for several years, and I like to think I know a bit about the operating system and what it can and cannot do. As I'm an avid USENET reader, I follow the latest developments and of course, the various flame−wars that invariably crop up (those darn cross−posting advocacy people! ;−)). I've seen my share of myths (often called *FUD* −− "Fear, Uncertainty, Doubt" which seems to be a common tactic used by commercial technology vendors to frighten their market away from competing technologies) that more than a few people believe. So, let me try to run down a few of the more common ones and attempt to shatter them.

:−)

* Linux is freeware, hence, it is a toy.

Some people seem to have the notion that, because a piece of software was written by volunteers with no profit motive in mind, that the results must clearly be inferior to commercial−grade offerings.

This may have been true in the past (I mean, there *was* a lot of freeware which was absolute garbage in the DOS and early Windows world), but it is most certainly not true in recent days.

The power of the Internet has made it possible to bring together some of the brightest minds in the globe, allowing collaboration on projects they find interesting. The people who have put a hand into developing Linux or the thousands of GNU utilities and applications packages are from a diverse background, and all of them have different personal reasons for wanting to contribute.

Some are hard−core hackers who develop for the love of coding, others have a need for something (for example, a network traffic monitor for a LAN at work) and decide to write it themselves, others are academics and computer scientists who are using Linux for its research qualities.

Unlike a commercial offering where a package is developed and sold, source code excluded, to the end−user, code used in Linux is scrutinized, debugged, and improved upon by anyone who has the interest and ability. This act of peer−review is one of the reasons that Linux offers the high reliability and high performance that it does.

Don't forget: The Internet itself was built and runs almost exclusively on Open Source projects. The e−mail you exchange on a daily basis with people around the world has an 80% chance of being handled on one or both ends by Sendmail, the web pages you browse while "Surfin' the Web" are served to you by Apache on over 50% of the world's web sites. Reliable enough for you?

* There is no support for Linux.

Hearing this myth somewhat sickens me. And supposedly the "other" vendors *do* offer support? I've had personal experience with one very popular commercial operating system, where the vendor's so−called "support" was completely useless.

First of all, there *is* support for Linux. Yes, commercial support. There are some companies that can provide as much support as you are willing to pay for; offering telephone and e−mail support, many offering to come right to your door to deal with the problem!

However, in 99% of the situations you will run into with Linux, you will be able to accomplish what you wish if you can simply get the answer to a question or two. This is easily accomplished on USENET or on any of the many mailing lists available!

I've never had a problem I couldn't find a solution to, by either searching on http://www.dejanews.com/, or by asking in one of the comp.os.linux.* newsgroups. Normally I can receive an answer to any of the support issues I ask about within three to twelve hours of my posting.

Another interesting aspect of Linux is that, because the source code for the entire kernel and most of the other operating system components is freely available, key−support issues such as security, denial of service, or CPU bugs (such as Intel's *F00F* fatal exception) are tracked down and solved *very* quickly −− usually an order of magnitude faster than solutions offered for similar or identical problems on the commercial offerings. So, where's the commercial support!?

There are countless others that I would like to debunk, but that is beyond the scope of this document. However, for further myth debunking, check out the "Linux Myth Dispeller" at http://www.KenAndTed.com/KensBookmark/linux/index.html as well as "The Linux FUDfactor FAQ" at http://www.geocities.com/SiliconValley/Hills/9267/fud2.html

3.3. One User's Perspective

I use Linux both at work and at home.

At my place of employment, we are using Linux to provide Internet services for hundreds of users. These services include TACACS (dial−in modem user) authentication, web page hosting and proxy caching, as well as SMTP and POP services. In addition, we are using Linux to provide NFS services, and also for providing and mounting SMB−protocol (WfW/Win95/WinNT) file & print and FAX services using the Samba package.

At home, I use Linux for my personal needs, such as Internet services, software development, and of course game playing (seeing Quake II running on a Linux box is a thing of beauty)! One of the things I love about Linux is, no matter how hard I pound on it, it does *not* crash! It's also a great way to learn, develop, and maintain my Unix skills.

I am using the Red Hat 6.1 distribution of Linux (see http://www.redhat.com/ for more information). This distribution includes all the necessary software for a full−blown unix system −− shells, compilers & interpreters, networking support, the X Window System, and all Internet services (eg. Mail, news, web server, telnet, etc.). The distribution comes standard with Linux kernel 2.2.12.

At my place of employment, the Linux−based system we use as our primary Internet server has the following configuration:

- Kernel: 2.2.12
- Machine: Pentium II @ 300 MHz (bogo−mips 299.83) with PCI−bus, 256 Mb RAM
- one 3 Gb Fujitsu IDE hard drive (/dev/hda)
- four 4.4 Gb Quantum Fireball SCSI hard drives (/dev/sd0 through /dev/sd3),
- 24x speed SCSI CD−ROM (/dev/scd0),
- Adaptec AHA−131 SCSI controller
- HP SCSI DAT tape drive (/dev/st0 and /dev/nst0),
- Intel EtherExpress Pro 10/100 Ethernet card

We have a second system −− an even nicer Intel box −− also running Red Hat 5.2, running in another office location. It provides networked file & print services via Samba, local web caching via Squid, and secondary DNS services. Unfortunately, this box is over 50 km away from where I usually work, and therefore it's left pretty much on its own −− yet this baby is really my pride and joy! Here are some specs:

- Kernel: 2.2.12
- Machine: Pentium II @ 350 MHz (bogo−mips 349.80) with PCI−bus, 256 Mb RAM
- one 4.1 Gb Quantum Fireball SCSI hard drive (/dev/sda)
- four 9.4 Gb Quantum Fireball SCSI hard drives (/dev/rd/c0d0, /dev/rd/c0d1) as hardware RAID level 5 array,
- 36x speed SCSI CD−ROM (/dev/scd0),

- BusLogic BT−948 SCSI controller
- Mylex AcceleRAID 250 (DAC960) RAID controller,
- HP SCSI DAT tape drive (/dev/st0 and /dev/nst0),
- Intel EtherExpress Pro 10/100 Ethernetcard

Having an incredible 24+ Gb of available storage space, with redundant storage configured as a hardware RAID5 array is a humbling feeling. The Mylex RAID controller works great, and I would not hesitate to recommend it to others seeking a hardware RAID solution! (If you are interested in configuring your Linux system with a RAID array, see Section 11.2 for details.)

We have four other Linux systems in place; an Alpha, a Sparc, and two Intel boxes; two of which are being used in production, and then there is my own personal system at home, but I won't bore you with the details.

This document will attempt to remain as hardware independent as possible but it may be helpful to you if you know where I am coming from as far as hardware is concerned.

Chapter 4. Installation and Hardware Configuration

This chapter will detail the procedures needed to install Red Hat 6.1 onto an Intel system; the procedures are similar whether you choose to install using either GUI− or text−based installation. Since much of this information is already well documented in the Red Hat User's Guide (provided as a paper manual in the "Official" boxed sets, included in the ``/doc'' directory on the CD, as well as available online at ftp://ftp.redhat.com/pub/redhat/redhat−6.1/i386/doc/rhinst/index.htm), I've skimmed over much of the details. However, there are a few things which I think are lacking in the Red Hat guide, and therefore I will attempt to cover those items in greater detail.

4.1. Creating an Installation Diskette

The first step in getting Red Hat's distribution of Linux onto a system, you need to find a way of starting the installation program. The usual method of doing so is to create an installation disk, although if you are installing from CD−ROM, and your system's BIOS supports it, you should be able to boot directly into the installation program from the CD.

Otherwise, to create an installation diskette, you'll need to copy the ``boot.img'' (which is simply an image of an ext2−formatted Linux boot diskette with an additional installation program) onto a floppy diskette.  The ``boot.img'' file can be obtained from the /images directory of the Red Hat CD−ROM disk, or downloaded via FTP from ftp://ftp.redhat.com in the /pub/redhat/redhat−6.1/i386/images directory (assuming you are installing Linux on an Intel box).

You can create the boot diskette either from a DOS or Windows system, or from an existing Linux or Unix system. For your destination diskette, you can use either an unformatted or a pre−formatted (for DOS) diskette −− it makes no difference.

Under DOS: Assuming your CD−ROM is accessible as drive D:, you can type:

```
d:
cd \images
..\dosutils\rawrite
```

For the source file, enter ``boot.img''.  For the destination file, enter ``a:'' (assuming the diskette you are created is inserted into the A: drive). The ``rawrite'' program will then copy the ``boot.img'' file onto diskette.

Under Linux/Unix: Assuming the ``boot.img'' file is located in the current directory (you may need to mount the CD−ROM under /mnt/cdrom and find the file in /mnt/cdrom/images), you can type:

```
dd if=boot.img of=/dev/fd0
```

The ``dd'' utility will copy, as its input file ("if"), the ``boot.img'' file, onto the output file ("of") /dev/fd0 (assuming your floppy drive is accessible from /dev/fd0).

Unless your Linux or Unix system allows write permissions to the floppy device, you may need to do this command as the superuser. (If you know the root password, type ``su'' to become the superuser, execute the ``dd" command, and then type ``exit" to return to normal user status).

With either of the above schemes, you should now have a bootable Red Hat 6.1 installation diskette that you can use to install your new Red Hat Linux system!

4.2. Booting Linux Installation Program

To begin setting up your new Red Hat system, either boot from the installation CD, or insert the installation diskette in the system's A: drive, and reboot or power−on the system. After a few moments, the Red Hat installation program screen should appear.

In most cases, you can just press <Enter> to begin the installation process, but if you are a more experienced user who knows exactly how your hardware devices should be set up, you can enter ``expert" for the additional information and prompts this feature provides. (If you do nothing, the default installation procedure will start in about 10 to 15 seconds after the installation screen first appears.)

You will then be asked to choose your language (usually "English") and your keyboard type (even in Canada I choose "US 101−key"), as well as where you are installing from (such as from your CD−ROM or over the network). Red Hat is very flexible in where it can be installed from.

Most likely you will choose ``Local CDROM" to install from your Red Hat CD−ROM (which should be inserted into your CD−ROM device). However, if your system is not equipped with a CD−ROM device, there are a number of other installation methods you can choose.

If you have another Linux system (or any other operating system that supports NFS file mounting), you can use ``NFS" to install from an NFS mount. To do this, you'll need to have your CD−ROM mounted in the other system (or otherwise have the Red Hat distribution tree somewhere on the other system −− it is possible to download everything via FTP and then install from your other system's hard drive), make sure you have an entry in your /etc/exports file allowing access by the new system to the appropriate directory (see Section 7.6 for details on how to set up and use NFS), and then enter the appropriate details. Here's an example walk−through:

- Insert the Red Hat CD into the other system (eg. a system called ``spock").
- To mount the CD, type:
  ```
  mount /dev/cdrom /mnt/cdrom -t iso9660
  ```
- Edit, as the superuser, your ``/etc/exports" file and put an entry like:
  ```
  /mnt/cdrom newsys.mydomain.name(ro)
  ```

  (This says that the new system at newsys.mydomain.name is allowed read−only access to the directory ``/mnt/cdrom/" and any subdirectories under it).

  If your new system does not yet have a domain name assigned to it, you can instead use its IP address:

  ```
  /mnt/cdrom 10.23.14.8(ro)
  ```

  (Assuming your new system has 10.23.14.8 as its IP address).

- Again, as superuser, type:
  `killall -HUP rpc.nfsd ; killall -HUP rpc.mountd`

  This will restart your NFS and mountd daemons, which is necessary before your new NFS export will work.

- Now, from your new system, you can choose ``NFS'' as your installation source. You'll be asked to provide information on your network card, as well as your IP settings. You'll likely use static IP settings if your system is sitting on a local LAN, or DHCP settings if, for example, your system is connected to a cable modem. Enter the settings as appropriate for your new system.
- You'll then be asked to enter the NFS server name and Red Hat directory. For our example system, we would type in ``spock'' as the NFS server name, and ``/mnt/cdrom/'' as the Red Hat directory.

There are other ways of installing Red Hat, such as using a Samba (Windows−style networking) connection, from an existing partition (such as your DOS or Windows 95 partition) on your hard drive, or via FTP. Check the Red Hat users guide for more details on installing using these methods, or just try to struggle through them (the procedures are really not very difficult!)

Once you have chosen your installation source, Red Hat will ask you if you wish to "Install" or "Upgrade" your system. As you are installing a new system, you should choose "Install". (As an aside, I'm a fairly anal person who *never* upgrades new distribution releases over existing systems −− I guess having suffered through so many problems with Microsoft products I have developed a significant mistrust for upgrading systems as a whole. I prefer to install from scratch, and simply restore from backup my personal/user and local site files.)

The installation program will then ask you if you have a SCSI adapter. If you answer yes, you'll be asked to choose the appropriate driver. In some circumstances, Red Hat will be able to detect your adapter automatically.

Next, you'll be asked to set up your file systems (ie. partition one or more drives for Linux). There are two tools available for setting up these partitions, including the Red Hat−supplied "Disk Druid", and the standard Linux "/fdisk" utility.

Both tools are similar in function, allowing you to specify the partition types and sizes. However, Disk Druid seems to be a bit more "user friendly", and a bit more complete than fdisk. In fact, if you use fdisk to partition your drives, you'll then be presented with the Disk Druid screen for specifying your mount points *anyway*. That being said, as an ex−Slackware user, I personally always use fdisk −− force of habit, I guess! :−)

The next section will detail how and why you should set up your partition information.

4.3. Partitioning Hard Drive(s)

Why partition, anyway? Well, although it is possible to get a perfectly functioning Linux system running on a single−partition system, and, in fact, is a bit easier to configure this way, there are a number of benefits from partitioning one or more of your storage devices into multiple partitions.

While it is true that Linux will operate just fine on a disk with only one large partition defined, there are several advantages to partitioning your disk for at least the four main file systems (root, usr, home, and swap). These include:

First, it may reduce the time required to perform file system checks (both upon bootup and when doing a manual fsck), because these checks can be done in parallel. (By the way, *NEVER* run an fsck on a mounted file system!!! You will almost certainly regret what happens to it. The exception to this is if the file system is mounted read−only, in which case it is safe to do so.) Also, file system checks are a lot easier to do on a system with multiple partitions. For example, if I knew my /home partition had problems, I could simply unmount it, perform a file system check, and then remount the repaired file system (as opposed to booting my system with a rescue diskette into single−user mode and doing the repairs).

Second, with multiple partitions, you can, if you wish, mount one or more of your partitions as read−only. For example, if you decide that everything in /usr will not be touched even by root, you can mount the /usr partition as read−only.

Finally, the most important benefit that partitioning provides is protection of your file systems. If something should happen to a file system (either through user error or system failure), on a partitioned system you would probably only lose files on a single file system. On a non−partitioned system, you would probably lose them on all file systems.

This little fact can be a big plus. For example, if your root partition is so corrupted you can't boot, you can basically boot from the rescue diskette set, mount your root partition, and copy what you can (or restore from backup; see Chapter 8 for details on how files can be backed up and restored), to another partition such as home, and then reboot once again using the emergency boot disk, typing "`mount root=/dev/hda3`" (assuming the partition containing your temporary root file system is on the third partition of hda) and boot your fully functional Linux box. Then you can run an fsck on your unmounted corrupt root partition.

I *have* had personal experience in file system catastrophies, and I was very grateful for having had the damage limited due to the use of multiple partitions.

Finally, since Linux allows you to set up other operating system(s) (such as Windows 95/98/NT, BeOS, or what−have−you), and then dual− (or triple−, ...) boot your system, you might wish to set up additional partitions to take advantage of this. Typically, you would want to set up at least one separate partition for each operating system. Linux includes a decent boot loader (called LILO on Intel−based systems, although much the same thing is available as MILO on Alpha, and SILO on Sparc) which allows you to specify which operating system you want to boot at power on, with a time−out default boot of your favorite operating system (probably Linux, right?)

You should partition a disk (or disks) according to your needs. In my experience on Intel, Alpha, and Sparc platforms, for a fairly loaded system (feature−wise), doing a fair amount of tasks (as a desktop system at home, or as an Internet server at work), I have found the following approximation of space works pretty effectively for determining a partition size.

```
Given:

A given disk of X Mb/Gb          (eg. 2 Gb)
(Or, more than one disk with a combined total of X Mb/Gb)

Calculate:

(swap) about double main RAM     (eg. 64 Mb system gets 128 Mb swap)
```

```
/ (root)  about 10% of available (eg. 200 Mb)
/home about 20% of available     (eg. 400 Mb)
/usr any remaining space         (eg. 1272 Mb)

/var (optional -- see below)
/boot (optional -- see below)
/archive (optional -- see below)
```

Of course, the above amounts are approximate guidelines only. Obviously you are going to want to juggle those percentages around a bit depending on what you are going to use your Linux system for. If you are going to be doing stuff like adding lots of bulky applications such as WordPerfect or Netscape, or perhaps adding Japanese character support, you would probably benefit from a bit more /usr space.

I *always* seem to have a lot of space available on /home, so if your users aren't doing much (or you have imposed strict quota sizes), or you aren't offering shell accounts and personal web pages, etc., you probably could lower /home space and raise /usr.

Here is a description of the various mount points and file system information, which may give you a better idea of how to best define your partition sizes for your own needs:

- */ (root)* – used to store things like temporary files, the Linux kernel and boot image, important binary files (things that are needed before Linux can mount the /usr partition), and more importantly log files, spool areas for print jobs and outgoing e–mail, and user's incoming e–mail. It is also used for temporary space when performing certain operations, such as building RPM packages from source RPM files. Therefore, if you have a lot of users with a lot of e–mail, or think you will need plenty of temporary space, you might want more space available. The partition type should be left as the default of 83 (Linux native). In addition, you'll probably toggle the bootable flag on this partition to allow boot information to be stored here.
- */usr/* – should be the largest partition, because most of the binary files required by Linux, as well as any locally installed software, web pages, Squid proxy cache, Samba share services, some locally–installed software log files, etc. are stored here. The partition type should be left as the default of 83 (Linux native).
- */home/* – typically if you aren't providing shell accounts to your users, you don't need to make this partition very big. The exception is if you are providing user home pages (such as school web pages), in which case you might benefit from making this partition larger. Again, the partition type should be left as the default of 83 (Linux native).
- *(swap)* – Linux provides something called "virtual memory" to make a larger amount of memory available than the physical RAM installed in your system. The swap partition is used with main RAM by Linux to accomplish this. As a rule of thumb, your swap partition should be at least double the amount of physical RAM installed in your system.

  If you have more than one physical hard drive in your system, you can create multiple swap partitions. This can improve the performance of swapping by taking advantage of parallel disk access. For example, on a 256 Mb system with four drives, I would probably create four 128 Mb swap partitions, for a total of 256 Mb RAM, 512 Mb swap (for a combined total of 768 Mb available as virtual memory). The partition type needs to be changed to 82 (Linux swap).

  > **Note:** Note: It is a common misconception that Linux has a 128 Mb swap size limit. This was true in the past, but in modern Linux distributions, the size depends on your architecture (for example, Intel systems can have swap sizes as large as 2 Gb). Type ``man mkswap'' for more information.

- */var/* (optional) − You may wish to consider splitting up your / (root) partition a bit further. The /var directory is used for a great deal of runtime storage, including mail spools (both ingoing and outgoing), print jobs, process locks, etc. Having this directory mounted under / (root) may be a bit dangerous because a large amount of incoming e−mail (for example), may suddenly fill up the partition. Since bad things can happen (eg. system crash?) when the / (root) partition fills up, having /var on its own partition may avoid such problems. I've had success in taking whatever space I've allocated to / (root), perhaps doubling it, and then creating separate partitions for / (root) and for /var. The partition type should be left as the default of 83 (Linux native).
- */boot/* (optional) − In some circumstances (such as a system set up in a software RAID configuration) it may be necessary to have a separate partition from which to boot the Linux system. This partition would allow booting and then loading of whatever drivers are required to read the other file systems. The size of this partition can be as small as a couple Mb; I recommend approximately 10 Mb (which should give you plenty of room to store the kernel, initial RAMdisk image, and perhaps a backup kernel or two). The partition type should be left as the default of 83 (Linux native).
- */archive/* (optional) − If you have any extra space lying around, perhaps you would benefit from a partition for a directory called, for example, /archive. You can then use the /archive directory to store backup material, large or infrequently accessed files, samba file services, or whatever else you can find a use for it. The partition type can be left as the default of 83 (Linux native), or if you want to access it from both Linux as well as from another operating system, you could change it to a different ID, such as 6 (DOS 16−bit >=32M).

As extra drive(s) are added, further partitions can be added to the new drives, mounted at various mount−points as required −− this means a Linux system never needs to worry about running out of space. As an example, if in the future it is clear that sda6 is starting to get filled up, we could add another drive, set a nicely sized partition with a mount−point at /usr/local −− and then transfer all the information from /usr/local over to the new drive. But no system or application component would "break" because Linux would see /usr/local no matter where it was located.

To give you an example of how one might set up partitions, I have used the following partitioning scheme on an Intel system (dual boot, Windows 95 and Linux):

```
   Device Boot    Begin     Start       End   Blocks   Id  System
/dev/hda1   *         1         1       254  1024096+   6  DOS 16−bit >=32M
/dev/hda2           255       255       782  2128896    5  Extended
/dev/hda5           255       255       331   310432+  83  Linux native
/dev/hda6           332       332       636  1229728+  83  Linux native
/dev/hda7           637       637       749   455584+  83  Linux native
/dev/hda8           750       750       782   133024+  82  Linux swap
```

The first partition, */dev/hda1*, is a DOS−formatted file system used to store the alternative operating system (Windows 95). This gives me 1 Gb of space for that operating system.

The second partition, */dev/hda2*, is a physical partition (called "extended") that encompasses the remaining space on the drive. It is used only to encapsulate the remaining logical partitions (there can only be 4 physical partitions on a disk; in my case I required more than 4 partitions, therefore I had to use a logical partitioning scheme for the others).

The third through fifth partitions, */dev/hda5*, */dev/hda6*, and */dev/hda7*, are all e2fs−formatted file systems used for the / (root), /usr, and the /home partitions, respectively.

Finally, the sixth partition, */dev/hda8*, is used for the swap partition.

For yet another example, this time an Alpha box with two hard drives (sole boot, Linux only), I have chosen the following partitioning scheme:

```
   Device Boot    Begin     Start       End    Blocks   Id  System
/dev/sda1             1         1         1      2046    4  DOS 16-bit <32M
/dev/sda2             2         2       168    346859   83  Linux native
/dev/sda3           169       169       231    130851   82  Linux swap
/dev/sda4           232       232      1009   1615906    5  Extended
/dev/sda5           232       232       398    346828   83  Linux native
/dev/sda6           399       399      1009   1269016   83  Linux native
/dev/sdb1             1         1       509   2114355   83  Linux native
/dev/sdb2           510       510      1019   2118540   83  Linux native
```

The first partition, */dev/sda1*, is a DOS−formatted file system used to store the MILO boot loader. The Alpha platform has a slightly different method of booting than an Intel system does, therefore Linux stores its boot information in a FAT partition. This partition only needs to be as large as the smallest possible partition allowed −− in this case, 2Mb.

The second partition, */dev/sda2*, is an e2fs−formatted file system used for the / (root) partition.

The third partition, */dev/sda3*, is used for the swap partition.

The fourth partition, */dev/sda4*, is an "extended" partition (see previous example for details).

The fifth and sixth partitions, */dev/sda5*, and */dev/sda6*, are e2fs−formatted file systems used for the /home and /usr partitions, respectively.

The seventh partition, */dev/sdb1*, is an e2fs−formatted file system used for the /archive partition.

The eighth and final partition, */dev/sdb2*, is an e2fs−formatted file system used for the /archive2 partition.

After you finish setting up your partition information, you'll need to write the new partition to disk. After this, the Red Hat installation program reloads the partition table into memory, so you can continue on to the next step of the installation process.

4.4. Setting up Swap Space

Once you've set up your partition information, and have assigned "mount points" (ie. /usr is the mount point for the /usr file system), the installation program will ask you which partition(s) it should used for swap space. Since your swap partitions should already be identified as such (partition ID # 82), you can press `<Enter>` to begin formatting those partition(s) for swap usage. I recommend you enable the "`Check for bad blocks during format`" to ensure the partition is free of potentially damaging problems. It does slow down the formatting process substantially but I believe it is worth the tradeoff.

4.5. Choosing Partitions to Format

Now, the installation program will display a list of the partitions you have assigned to Linux, and ask you to select which, if any, of these partitions you want to format as new file systems. Likely, you will want to format all of them, except if you are upgrading your system or perhaps have some information (eg. on /home) that you don't want to lose.

Again, I recommend you enable the "`Check for bad blocks during format`" option.

4.6. Choosing Desired Packages to Install

Next, you'll be presented with a list of system components and asked to specify which ones should be installed. If you are an experienced Linux user, you can pick and choose according to your needs. If you are new to Linux, you'll likely want to select the bottom option, "Everything".

What I usually do is select the components I know I'll need, and then enable the "`Select individual packages`" option, which allows me to control the installation in finer detail.

Once you have chosen your desired components, select "`Ok`" to begin installation. If you have enabled the "`Select individual packages`", you'll be asked the specify which individual packages should be installed. This is fairly straightforward, and if you are unsure of what a given package is for, you can press the <**`F1`**> key for a brief description of what it does.

Don't worry if you make a mistake choosing (or not choosing) a package or two. After all, all the packages are on your CD−ROM (or other source media), so you can use the handy Red Hat RPM tool to make adjustments after your system is up and running (see Section 10.1 for details).

After you have chosen the packages you wish to install, the installation program will now format the partitions you have defined. This may take several minutes, especially for larger partitions or if you've enabled bad block checking, so please don't think your system has frozen during this procedure!

After the format completes, Red Hat Linux will begin installation of the selected packages. This should take between five and fifteen minutes to complete, depending on the speed of your system.

4.7. Hardware Configuration

After package installation, Red Hat will begin configuring the devices on your system. In most cases, except with very new hardware that may not be fully supported by Linux, the installation program does a good job of automatic configuration.

The prompts you will see are very straightforward:

- Detection of your mouse (including choosing between 2− and 3−button models. If you have a 2−button mouse you'll likely want to enable 3−button emulation.)
- Detection of your video card
- Choosing your monitor
- Running of ``` ``XConfigurator ```" to configure the X Window System (you'll want to "Probe" your card. If you get an error here, don't worry as you can take care of X configuration later, after your system is up and running; see Chapter 5 for details.)
- Selection of video modes (you can choose the defaults, or you can fine− tune the video modes you'll want to use under the X Window System)
- LAN configuration
- Clock and timezone configuration

- Startup services (the default selection is probably best, but again, you can press <**`F1`**> for a description of what a given service does)
- Printer configuration
- Assignment of root password (choose something secure!)
- Creation of a boot disk [don't be lazy! Make one! :−)]

4.8. Booting with LILO

Next, the installation program needs to write a boot loader to your hard drive. The boot loader (*LILO* on Intel systems) is responsible for booting Linux along with any other operating systems if you have set up your system for multi−boot (see Section 4.8.1 for details on this).

The "`Lilo Installation`" dialog box will ask you to choose where the boot loader image should be written to. You'll likely want to install it on the master boot record of your first drive (usually /dev/hda for IDE, /dev/sda for SCSI).

Once you have selected the location for writing the boot loader, a second dialog box will appear, allowing you to enter extra boot−time configuration parameters. Usually you don't need to enter anything here, but if you have more than 64 Mb of RAM you'll need to enter a special parameter in order to have Linux make use of the extra RAM (otherwise, it will only use the first 64 Mb). For example, if your system has 128 Mb of RAM, you should enter:

```
append="mem=128M"
```

If your system has SCSI drives, or you wish to install LILO on a partition with more than 1023 cylinders, it may be necessary to enable the option to "`Use linear mode`". If it is not, enabling this option shouldn't hurt anything, so it is probably a good idea to do so.

4.8.1. Multi−boot with Other Operating Systems

Finally, if you've set up your system to multi−boot Linux with other operating system(s), you'll be presented with a third dialog box which lists the available partitions. Here, you can assign names to your other operating systems (which you enter at the "LILO" prompt at boot time to boot your desired operating system. The installation program does assign default names to each bootable partition, so it isn't necessary to change them unless you don't like the defaults.

The default operating system that will boot upon system start up will, of course, be Linux. However, if you wish, you can change the default to any of the other operating systems you have defined.

After installing the boot loader on your hard drive, the installation program should hopefully present you with a "Congratulations" dialog box, indicating that Linux has been successfully installed. Remove the installation floppy diskette (if any), and press <**`Enter`**> to reboot your system...into Linux!

Linux will boot, and if all goes well you should see a "login" prompt. From here, you should be able to log in as "root" using whatever password you have assigned during the installation process.

4.9. Downloading and Installing Red Hat Updates

Red Hat has produced some pretty impressive versions of their distribution so far, but seems to have a history of releasing them when they are not quite "ready for prime time". Therefore in order to take full advantage of your Linux system, it is necessary to download and apply updated packages. These packages, also called "rpm files" are applied using the RPM utility (for details on this utility, see Section 10.1).

This will prove to be one of the more time−consuming parts of getting your Linux system ready (unless you have a stellarly fast Internet connection). However, take the time to do this! You will likely save yourself a *lot* of grief!

First, download all files from:

ftp://ftp.redhat.com/redhat/updates/6.1/i386/

(The above assumes you are using Linux on an Intel box).

You should probably download everything into a single directory, and then you can simply type: ``rpm −Uvh *'' which will upgrade all the packages. If you've downloaded any kernel rpm files, you should probably move them to another directory for now. Upgrading or customizing your kernel is a bit more complicated and needs to be done with great care (see Section 10.4 for details on this). Therefore before you apply the upgrades, you may wish to consider moving all the kernel−*.rpm files out of your temporary upgrade directory.

To apply the upgrades, you can simply run ``rpm'' against all the packages at once (ie. "rpm −Uvh *"), or if you prefer, you can upgrade them one at a time (ie. "rpm −Uvh file_to_upgrade.rpm"). The latter method is for us anal types who wish to ensure that each update is applied correctly without error. :−)

Perhaps you are curious to see if a given package is installed before you attempt to upgrade it. Or perhaps you wish to find out what version of a given package is installed. All this can be done with the RPM utility; see Section 10.1 for details.

Chapter 5. Configuring the X Window System

The X Window System, aka "X" (commonly and incorrectly known by many as "X−Windows") is a GUI which sits on top of Linux. Unlike Microsoft Windows, the X Window System can look and operate in a large variety of different ways. It can operate very primitively or very advanced, look beautiful or ugly, be sleek and fast or bloated and slow (each of which are subjective qualities which cause as many arguments among users as the "Linux vs. Microsoft NT" debate seems to).

Getting X working properly can range from simple to hair−pulling complicated! It is a common complaint among users who are new to Linux, and I've fought with configuration settings countless times myself, so I'm completely empathic about this. Fortunately, such configuration is becoming easier and more automated in the newer distributions of Linux. In fact, if you are using Red Hat 6.1 you will probably not have to worry about this issue.

Although in a majority of cases X can be configured automatically, there are exceptions; I would recommend you know or find out the type of video card and amount of video RAM your system has installed, as well as the type of monitor and its horizontal and vertical synch rates (this information is usually available in the back pages of the monitor's users guide, or can be found on the WWW).

5.1. Getting the X Window System Working with X−Configurator

There are two main methods of getting X working under Red Hat's distribution of Linux. The first and easiest method, is to use Red Hat's own ``Xconfigurator'' utility. The utility tries to detect your hardware and installs the applicable X software with the appropriate configuration settings.

If you are still unsuccessful after trying out various settings with Xconfigurator, you may have better luck with the ``xf86config'' utility. Although certainly not as user−friendly or attractive as Xconfigurator is, it gives you finer control over the configuration process.

Finally, if you are *still* out of luck you may have to resort to editing the ``/etc/X11/XF86Config'' file by hand and tweaking various settings. If this is the case, you may need to get help from the Linux community (see Section 13.3 for details). Relax, however −− in a majority of cases Xconfigurator does an adequate job!

After getting X working properly, you may be disappointed in the lack of rich colours. This is because X uses a default 8−bit per pixel (``*bpp*'') colour depth. You can use higher colour depths, however, assuming your video hardware will support them.

The various colour depths are listed in your ``/etc/X11/XF86Config'' file, and look like this:

```
Subsection "Display"
    Depth       24
    Modes       "800x600" "1024x768"
    ViewPort    0 0
    Virtual     1024 768
EndSubsection
```

The above section shows the possible resolutions which are available when using the 24−bit colour depth (800x600 and 1024x768, as listed in the "Modes" line); these resolutions can be switched between "on−the−fly" using the <**Alt**><**+**> and <**Alt**><**−**> keys.

> **Tip:** Tip: As a default, when X starts up it does so using the lowest resolution. If you dislike this behaviour as much as I do, simply edit the ``/etc/X11/XF86Config" file and reverse the resolutions (ie. "1024x768" "800x600").

When you are getting things set up, you can test each colour depth manually by typing, ``startx -- -bpp 24" (for the 24–bit depth) and make sure X is working properly for the colour depth you wish to use.)

If you are able to successfully use a higher colour depth and wish to use it as the default, you will need to create a ``/etc/X11/xinit/xserverrc" file as follows:

```
exec X :0 -bpp 24
```

The above change will allow X to use 24 bits per pixel (if you have problems with this, try 16 or 32 instead of 24).

Assuming you have configured X properly, starting it is a simple matter of typing ``startx" as any user. The X GUI will start, and after you have finished your session and quit X, you will be returned to the regular Linux console.

Optionally, X can start up at system boot, and *always* run (see Section 5.2 for details on how to accomplish this). This can be handy for those users who dislike seeing the "boring" black & white console, or for those who wish to avoid dealing with command line shells as much as possible.

5.2. Using the X Desktop Manager

If you wish, you may use the X Desktop Manager (``xdm") to start up the X Window System automatically at system boot time. This allows your Linux system to always run under X (although you can switch from the GUI to the regular consoles with <**Ctrl**>–<**Alt**>–<**F1**>, and then back again to the GUI with <**Alt**>–<**F7**> as needed). This is a nice way of providing an attractive and friendly environment for you and your users, and avoid having to type ``startx" all the time.

To enable xdm, simply edit the ``/etc/inittab" file and change the line that reads "id:3:initdefault:" to the following:

```
id:5:initdefault:
```

The above change will switch Linux to run level 5 upon system boot up; this run level, by default, will start xdm. You may also wish to check your ``/etc/inittab" file, probably near the bottom, to ensure the following line is present:

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

If you have enabled xdm and wish to use a higher ``bpp" value than the default of 8 (and your video card and monitor will support it), you will need to modify the ``/etc/X11/xdm/Xservers" file as follows:

```
:0 local /usr/X11R6/bin/X -bpp 24
```

The above change will start the xdm at 24 bits per pixel.

You may also wish to edit the ``/etc/X11/xdm/Xsetup_0" file and with a ``#" character, comment out the line that starts ``xbanner" as shown:

```
#/usr/X11R6/bin/xbanner
```

This will prevent the default xdm banner screen from displaying for a split second between KDE sessions. Aesthestics, I know, but...

> **Tip:** Tip: Sometimes you may find it necessary to switch back to the console (for example, certain games run under the console but not under X). There are two ways of doing this. To temporarily switch away from X to the console, press <**Alt**><**F1**>, and to switch back to X again, press <**Alt**><**F7**>. Or, if you wish to terminate X altogether (thus freeing up your available memory), you can type ``/sbin/telinit 3" as "root" to switch the system run–level; this tells XDM to terminate. To switch back, type ``/sbin/telinit 5".

5.3. Improving Font Appearance Under X

Quite frankly, X has never been known for having particularly attractive fonts. In fact, many people resign themselves to the notion that having ugly, nasty fonts is an unfortunate fact of life under X.

Fortunately, it *is* possible to dramatically improve the look of, and increase the number of fonts you can use, under X. In fact, if you own a copy of Windows, you can even copy over the TrueType fonts from that platform and use them under X as well! Such font support is accomplished by using a font server such as ``xfstt" or ``xfs".

Red Hat 6.1 now includes support for ``xfs" built in, and as a result provides attractive font support right out of the box. Therefore, if you're using this version of Linux, you may be satisfied with the way things are. However, there are a couple of things you can do to improve things still further, as well as make use of your TrueType fonts if you have them available.

To enable TrueType font support, create a directory (eg. ``/usr/local/share/ttfonts") and copy any or all of the font files from your Windows system (where they can be found in the ``c:\windows\fonts" directory) into the new directory.

> **Tip:** Tip: If you do not have any TrueType fonts available to you, you can download them directly from Microsoft at http://www.microsoft.com/typography/fontpack/default.htm.

To make use of the fonts, from within your new ``ttfonts" directory, type the following (as root):

```
ttmkfdir -o fonts.scale
mkfontdir
```

Next, edit the ``/etc/X11/fs/config" file, add add your new font directory to the list of existing directories. Also, change the default-point-size from 120 to 140, which will give you larger, more readable fonts.

Finally, exit X (if you haven't done so already), and restart your xfs server as follows:

```
/etc/rc.d/init.d/xfs restart
```

Finally, restart X and enjoy your glorious new fonts!

For more detailed information on improving font support under X, there is a very excellent resource called the ``*XFree86 Font Deuglification Mini HOW−TO*'' at http://www.frii.com/~meldroc/Font−Deuglification.html.

5.4. Choosing a Window Manager for X

Now, you should decide on a window manager. The X Window System is simply the environment which allows graphics to be displayed on your system's hardware; the window manager is responsible for how X looks and how it interacts with you and your applications.

The Red Hat distribution of Linux contains several window managers, including fvwm, olvm, twm, AfterStep, and others. The default one that you will probably see when starting up X for the first time is fvwm95, a Win95−like environment.

Personally, I find the usual offerings differ from my own tastes, and I recommend using either GNOME or KDE (or both!), whose installation are covered in the next two sections.

5.5. GNOME Installation and Configuration

The GNU Network Object Model Environment (GNOME) is a windowing environment that enhances your X window environment. It is full−featured, including a large selection of applications you may find useful. However, at the time of this writing, GNOME still has a few minor bugs, meaning you may have to put up with errant behaviour at times. However, it is fairly stable and definitely usable!

If you're using Red Hat 6.1, the latest version of GNOME (at least, the latest at the time of this writing!) is included with the distribution. Otherwise, you will need to download the latest RPM distribution of the package. At the time of this writing, the RPM files for Red Hat 6.0 i386 systems can be found at ftp://ftp.gnome.org/pub/GNOME/RHAD/redhat−6.0/i386/ (or from a mirror site).

> **Note:** Note: If you're using Red Hat 6.0, you should be aware that it was shipped with a fairly buggy version of GNOME. You should download the latest RPM's from the FTP site as described above.

After you have all the necessary files, the GNOME package can be installed with a simple command, typed as "root":

```
rpm −Uvh gtk*.rpm *.rpm
```

(The above command ensures the GTK libraries are installed first, to avoid dependency errors from occurring).

Contrary to popular belief, GNOME is actually *not* a Window manager, but instead sits on top of your favorite one, providing added functionality to it. Therefore, once you have installed GNOME, you should decide which window manager you wish to use, and create a ``.xinitrc'' file in your directory which loads the appropriate window manager and starts GNOME. The file should look something like this:

```
afterstep &
exec gnome-session
```

The above file will load AfterStep for the window manager, and then run GNOME on top of it.

More information on the GNU Network Object Model Environment can be found on the GNOME web page at http://www.gnome.org/. Don't forget to check out the screen shots, located at http://www.gnome.org/screenshots/.

5.6. KDE Installation and Configuration

The K Desktop Package (KDE) is another popular window manager that is somewhat more mature than GNOME is at the time of this writing. However, it seems to require a bit more memory resources than GNOME does, so take into consideration the amount of RAM you have available on your system (if you have anything less than 64 Mb of RAM and 128 Mb of swap, you might be better off using GNOME).

The first step for installing KDE is to download the latest RPM distribution of the package. To do so, locate an FTP mirror at http://www.kde.org/mirrors.html. Try to choose a mirror that is close to your geographic location, but make sure whichever one you choose is updated often (which can be determined by looking at the list of mirrors).

When you have found a suitable mirror, download all the RPM files which are applicable to your version of Red Hat and your platform. For example, if you are using Red Hat 5.2 (or above) on an Intel platform, you will likely want to download the package from the ``/pub/mirrors/kde/stable/latest/distribution/rpm/RedHat-5.2/i386/'' directory on the FTP mirror.

After you have all the necessary files, the KDE package can be installed with the following simple commands, typed as "root" (make sure you are in the directory where all your KDE rpm files are):

```
rpm -Uvh qt*.rpm
install-kde-1.1-base
```

The above commands will install the Qt libraries first, and then install the KDE base package. Once this is done, you should log off and log back in (or if you are ``su''ed as root, just exit and ``su'' again) so that your path environment is set appropriately, then type:

```
install-kde-1.1-apps
```

The above command will install the applications programs.

This installation procedure is discussed in more detail in the file ``readme-redhat-rpms.txt'' that should have been included with the KDE files you downloaded.

If all goes well, and KDE has been installed without any error messages, you may, if you wish, configure KDE to be the default window manager for any of your users (the one they will see immediately after typing ``startx''), by typing the following, again as "root":

```
/opt/kde/bin/usekde userid
```

(Make sure you replace *userid* with an actual user id!)

More information on the K Desktop Environment can be found on the KDE web page at http://www.kde.org/. Don't forget to check out the screen shots, located at http://www.kde.org/kde2shots.html.

Chapter 6. General System Administration Issues

6.1. Root Account

The "root" account is the most privileged account on a Unix system. This account gives you the ability to carry out all facets of system administration, including adding accounts, changing user passwords, examining log files, installing software, etc.

When using this account it is crucial to be as careful as possible. The "root" account has no security restrictions imposed upon it. This means it is easy to perform administrative duties without hassle. However, the system assumes you know what you are doing, and will do exactly what you request –– no questions asked. Therefore it is easy, with a mistyped command, to wipe out crucial system files.

When you are signed in as, or acting as "root", the shell prompt displays '#' as the last character (if you are using bash). This is to serve as a warning to you of the absolute power of this account.

The rule of thumb is, never sign in as "root" unless absolutely necessary. While "root", type commands carefully and double–check them before pressing return. Sign off from the "root" account as soon as you have accomplished the task you signed on for. Finally, (as with any account but especially important with this one), keep the password secure!

6.2. Creating User Accounts

Warning
(WARNING: SLACKWARE–CENTRIC. NEEDS UPDATE FOR RED HAT)

This section assumes you are using the Shadow password suite on your Linux system. If you are not, you should consider doing so, as it helps to tighten up security somewhat. The Shadow suite is fairly easy to install and will automatically convert your non–shadow password file format over to the new shadow format.

There are two steps to creating a new user account. The first is to actually create the account itself, the second is to provide an alias to their e–mail address (at my place of employment, we follow the convention of "Firstname.Lastname@our.domain.name".)

To create the account, decide on the username you are going to assign to the user. The username is at most 8 characters long, and wherever possible you should choose their last name, or last name and first initial if a user account already exists (the adduser script will detect and prevent you from adding duplicate account names).

You will then be prompted to enter other information: full name of user, user group (usually the default value), a user id # (automatically assigned), home directory (automatically assigned), a user shell, some password expiration values, and finally the desired password (which won't echo to the screen; you should have the user choose a password between 6 to 8 characters in length for security reasons).

Please note that *everything* should be entered in lowercase, except for the full name of the user which can be entered in a "pleasing format" (eg. Joe Smith) and the password. Case is sensitive, so inform your user(s) they must use identical case when entering their username and password.

Here is a sample session where we will add a user named Joe Smith:

```
mail:~# /sbin/adduser
User to add (^C to quit): smith
That name is in use, choose another.
User to add (^C to quit): smithj
Editing information for new user [smithj]
Full Name: Joe Smith
GID [100]:
Checking for an available UID after 500
First unused uid is 859
UID [859]:
Home Directory [/home/smithj]:
Shell [/bin/bash]:
Min. Password Change Days [0]:
Max. Password Change Days [30]: 90
Password Warning Days [15]:
Days after Password Expiry for Account Locking [10]: 0
Password [smithj]:</ FL1539
Retype Password:</ Fl1539
Sorry, they do not match.
Password:</> FL1539
Retype Password:</ FL1539

Information for new user [smithj]:
Name: Joe Smith
Home directory: /home/smithj
Shell: /bin/bash
Password: <hidden>
Uid: 859        Gid: 100
Min pass: 0     maX pass: 99999
Warn pass: 7    Lock account: 0
public home Directory: no
Type 'y' if this is correct, 'q' to cancel and quit the program,
or the letter of the item you wish to change: Y
```

The next step is to create the alias for the person's e–mail account. This gives people the choice of using their account name for their e–mail address, or their full name (First.Last combination) to make it "easier" for the outside world to guess their e–mail address when trying to contact them for the first time.

To add the e–mail alias, edit the ``/etc/aliases'' file as follows:

```
mail# pico -w /etc/aliases
```

Add the new alias at the bottom of the file. The format for an alias is:

```
First.Lastname:username
```

You should ask the user what preference they have for this (eg. Joseph.Smith or Joe.Smith). For our new Joe Smith user, the entry would be as follows:

```
Joe.Smith:smith
```

When finished adding the alias, press <**Ctrl**>–<**X**> and save the file. Then, type ``newaliases'' to update the aliases database.

At this point the user account has been created and is ready for use. It is a good idea to remind the user that his username and password must be entered in lowercase characters, and what their e−mail address would be (eg. ``*Joe.Smith@mail.mydomain.name*'').

6.3. Changing User Passwords

To change a password on behalf of a user, first sign on or "su" to the "root" account. Then type, ``passwd user'' (where user is the username for the password you are changing). The system will prompt you to enter a password. Passwords do not echo to the screen when you enter them.

You can also change your own password, by typing ``passwd'' (without specifying a username). You will be prompted to enter your old password for verification, and then a new password.

6.4. Disabling User Accounts

To disable a user account, edit, as root, the ``/etc/shadow'' file (assuming you're using shadow passwords; if not, edit the ``/etc/passwd'' file instead), and replace the password (which is stored in its encrypted form) with a ``*'' asterisk character. All Unix passwords, regardless of length (up to a maximum of 8 characters), are stored in the password file as encrypted strings of 13 characters. Therefore, by replacing the password with a single ``*'' character, it is impossible for the user to sign in.

> **Note:** Note: This method will require you to assign a new password to the user if you re−enable the account, since the encrypted password field will have been replaced. One solution to this which seems to be popular among system administrators is to simply prefix the ``*'' asterisk character in front of the encrypted password to disable the account, and simply removing the asterisk to enable it.

For more information on the ``/etc/passwd'' and ``/etc/shadow'' files, see Section 6.6 below.

6.5. Removing User Accounts

On occasion, you may wish to remove a user's access from your server altogether.

If you are a Red Hat user, the easiest way to remove an unneeded user account is with the ``userdel'' command, which must be typed as ``root''. An example follows:

```
/usr/sbin/userdel baduser
```

The above command will remove the entry matching the username ``*baduser* from the ``/etc/passwd'', file, and, if you're using the Shadow password format (which you should be; see Section 6.6 for details), the ``/etc/shadow''.

> **Note:** Note: The ``/etc/group'' is *not* modified, to avoid removing a group that other user(s) may also belong to. This isn't much of a big deal, but if this bothers use, you can edit the group file and remove the entry manually.

Should you wish to remove the user's home directory as well, add the ``-r" option to the ``userdel" command. For example:

```
/usr/sbin/userdel -r baduser
```

I recommend not removing an account right away, but first simply *disable* it, especially if you are working with a corporate server with lots of users. After all, the former user may one day require the use of his or her account again, or may request a file or two which was stored in their home directory. Or perhaps a *new* user (such as an employee replacement) may require access to the former user's files. In any event, make sure you have backups of the former user's home directory, "just–in–case". See Section 6.4 for details on disabling an account, and Chapter 8 for details on how to perform backups.

6.6. Linux Password & Shadow File Formats

Traditional Unix systems keep user account information, including one–way encrypted passwords, in a text file called ``/etc/passwd". As this file is used by many tools (such as ``ls") to display file ownerships, etc. by matching user id #'s with the user's names, the file needs to be world–readable. Consequentally, this can be somewhat of a security risk.

Another method of storing account information, one that I always use, is with the shadow password format. As with the traditional method, this method stores account information in the /etc/passwd file in a compatible format. However, the password is stored as a single "x" character (ie. not actually stored in this file). A second file, called ``/etc/shadow", contains encrypted password as well as other information such as account or password expiration values, etc. The /etc/shadow file is readable only by the root account and is therefore less of a security risk.

While some other Linux distributions forces you to install the Shadow Password Suite in order to use the shadow format, Red Hat makes it simple. To switch between the two formats, type (as root):

```
  /usr/sbin/pwconv       To convert to the shadow format
  /usr/sbin/pwunconv     To convert back to the traditional format
```

With shadow passwords, the ``/etc/passwd" file contains account information, and looks like this:

```
smithj:x:561:561:Joe Smith:/home/smithj:/bin/bash
```

Each field in a passwd entry is separated with ":" colon characters, and are as follows:

- Username, up to 8 characters. Case–sensitive, usually all lowercase
- An "x" in the password field. Passwords are stored in the ``/etc/shadow" file.
- Numeric user id. This is assigned by the ``adduser" script. Unix uses this field, plus the following group field, to identify which files belong to the user.
- Numeric group id. Red Hat uses group id's in a fairly unique manner for enhanced file security. Usually the group id will match the user id.
- Full name of user. I'm not sure what the maximum length for this field is, but try to keep it reasonable (under 30 characters).
- User's home directory. Usually /home/username (eg. /home/smithj). All user's personal files, web pages, mail forwarding, etc. will be stored here.
- User's "shell account". Often set to ``/bin/bash" to provide access to the bash shell (my personal favorite shell).

Perhaps you do not wish to provide shell accounts for your users. You could create a script file called ``/bin/sorrysh'', for example, that would display some kind of error message and log the user off, and then set this script as their default shell.

> **Note:** Note: If the account needs to provide "FTP" transfers to update web pages, etc. then the shell account will need to be set to ``/bin/bash'' –– and then special permissions will need to be set up in the user's home directory to prevent shell logins. See Section 7.1 for details on this.

The ``/etc/shadow'' file contains password and account expiration information for users, and looks like this:

```
smithj:Ep6mckrOLChF.:10063:0:99999:7:::
```

As with the passwd file, each field in the shadow file is also separated with ":" colon characters, and are as follows:

- Username, up to 8 characters. Case–sensitive, usually all lowercase. A direct match to the username in the /etc/passwd file.
- Password, 13 character encrypted. A blank entry (eg. ::) indicates a password is not required to log in (usually a bad idea), and a ``*'' entry (eg. :*:) indicates the account has been disabled.
- The number of days (since January 1, 1970) since the password was last changed.
- The number of days before password may be changed (0 indicates it may be changed at any time)
- The number of days after which password *must* be changed (99999 indicates user can keep his or her password unchanged for many, many years)
- The number of days to warn user of an expiring password (7 for a full week)
- The number of days after password expires that account is disabled
- The number of days since January 1, 1970 that an account has been disabled
- A reserved field for possible future use

6.7. System Shutdown and Restart

To shut down the system from a terminal session, sign in or "su" to the "root" account. Then type ``/sbin/shutdown -r now''. It may take several moments for all processes to be terminated, and then Linux will shut down. The computer will reboot itself. If you are in front of the console, a faster alternative to this is to press <**Ctrl**>–<**Alt**>–<**Del**> to shut down. Please be patient as it may take a couple of minutes for Linux to terminate.

You can also shut down the system to a halt (ie. it will shut down and not reboot the system). The system will be unavailable until power–cycled or rebooted with <**Ctrl**>–<**Alt**>–<**Del**>. This can be useful if you need to power down the system and move it to a different location, for example. To do this, type ``/sbin/shutdown -h now'' when signed into or "su"ed to the "root" account. Linux will shut itself down then display a message, "System halted". At this point you can power down the computer.

It is probably a good idea to only shut down the system when you are at the console. Although you can shut it down remotely via a shell session, if anything goes wrong and the system does not restart properly, the system will be unavailable until action is taken at the system unit. (I haven't experienced any problems doing this myself, however).

Upon system bootup, Linux will start automatically, and load all necessary services including networking support, and Internet services.

> **Tip:** Tip: If you wish to provide some kind of warning to any online users (online meaning logged in to shell accounts), you can substitute a time value instead of the "now" keyword. You can also customize the shutdown warning message. For example, ``/sbin/shutdown -r +5 Hardware upgrade'' would inform users that the system was about to shutdown for the given reason. They are then given periodic warnings that they should close files and log off before the big moment arrives.

Chapter 7. Custom Configuration and Administration Issues

For both personal use as well as at work, I was able to start with a standard installation of the Red Hat Linux distribution and provide services "out–of–the–box" with little or no changes to default configuration settings.

However, there were a number of small changes and extra services that were necessary to provide all the Internet, file & print services, and other services that are in use at my place of employment. The local administrator should be aware of the following:

- The ``/etc/rc.d/rc.local" file is executed upon system start–up and contains any extra services you have added to your server that should be executed upon bootup.
- Look in /etc for any site–specific changes that may be required. These may include:
    - ``/etc/inetd.conf" (you should ensure unnecessary services were disabled such as finger, echo, chargen; as well as add or change any services you may need)
    - ``/etc/exports" (contains a list of hosts allowed to mount NFS volumes; see Section 7.6 for details)
    - ``/etc/organization", ``/etc/nntpserver", ``/etc/NNTP_INEWS_DOMAIN" (set as appropriate)
    - ``/etc/lilo.conf" (contains information for the LILO boot loader –– the process which loads the Linux kernel upon bootup; see Section 4.8 for details)
    - ``/etc/sudoers" (a list of users who should be given special privileges, along with the commands they are allowed to execute)
    - ``/etc/named.boot" (for DNS use; see Section 7.2 for details)
- Anything in ``/usr/local/" (and subdirectories) are extra packages or modifications to existing ones that you have installed here, if you have installed from things like tarballs instead of using RPM. (Or at least, you should have installed them here.) These files, particularly in /usr/local/src/, should be kept up–to–date. See Chapter 10 for details.

7.1. Web Server and HTTP Caching Proxy Administration

Warning
(WARNING: DISREGARD THIS SECTION!)

1. Create an Internet user as per normal. The "shell" account should be ``/bin/bash" (as FTP requires a valid shell).
2. ``cd /home ; chown root.root theuser" This makes "theuser"'s directory belong to root, for security reasons.
3. ``cd /home/theuser ; mkdir www ; chown theuser.theuser" This creates their "www" directory, and sets ownership so they can read/write to it.
4. ``echo "exit" > .profile" This creates a ``.profile" file with the single line ``exit" in it. If the user tries to log in via telnet, they will get disconnected immediately.
5. Do an ``ls -l" and make sure there are only 2 files in the directory (not including ``.." and ``."):
    - *.profile* (owned by root.root)

♦ *www* (owned by theuser.theuser)

All other files can be deleted (eg. ``rm .less ; rm .lessrc")

6. If the user needs to have e–mail forwarding enabled you could create a .forward file which simply has the proper e–mail as the first and only line in the file.

That's it. The user can use FTP to update the pages.

7.2. Domain Name Server (DNS) Configuration and Administration

At my place of employment, we are using Linux as a DNS server. It performs exceptionally well. This section will address configuration of DNS tables for these services using the BIND 8.x package which comes standard with the Red Hat distribution.

> **Note:** Note: Red Hat versions 5.1 and earlier used the BIND 4.x package, which used a slightly different format for its configuration file. BIND 8.x offers more functionality over that offered by BIND 4.x, and as 4.x is no longer being developed, you should probably consider upgrading your BIND package to the latest version. Simply install the BIND RPM package (see Section 10.1 for details on using the RPM utility), then convert your configuration file to the new format.
>
> Fortunately, converting your existing BIND 4.x configuration file to be compliant with BIND 8.x is easy! In the documentation directory provided as part of BIND (for example, ``/usr/doc/bind-8.1.2/" for BIND version 8.1.2), there exists a file called ``named-bootconf.pl", which is an executable Perl program. Assuming you have Perl installed on your system, you can use this program to convert your configuration file. To do so, type the following commands (as root):
>
> ```
> cd /usr/doc/bind-8.1.2
> ./named-bootconf.pl < /etc/named.boot > /etc/named.conf
> mv /etc/named.boot /etc/named.boot-obsolete
> ```
>
> You should now have an ``/etc/named.conf" file which should work with BIND 8.x "out–of–the–box". Your existing DNS tables will work as–is with the new version of BIND, as the format of the tables remains the same.

Configuration of DNS services under Linux involves the following steps:

1. To enable DNS services, the ``/etc/host.conf" file should look like this:

```
# Lookup names via /etc/hosts first, then by DNS query
order hosts, bind
# We don't have machines with multiple addresses
multi on
# Check for IP address spoofing
nospoof on
# Warn us if someone attempts to spoof
alert on
```

The extra spoof detection adds a bit of a performance hit to DNS lookups (although negligible), so if you're not too worried about this you may wish to disable the "nospool" and "alert" entries.

2. Configure the ``/etc/hosts" file as needed. Typically there doesn't need to be much in here, but for improved performance you can add any hosts you access often (such as local servers) to avoid performing DNS lookups on them.
3. The ``/etc/named.conf" file should be configured to point to your DNS tables according to the example below.

   Note: (Note: IP addresses shown are examples only and must be replaced with your own class addresses!):

```
options {
        // DNS tables are located in the /var/named directory
        directory "/var/named";

        // Forward any unresolved requests to our ISP's name server
        // (this is an example IP address only -- do not use!)
        forwarders {
                123.12.40.17;
        };

        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
        // query-source address * port 53;
};

// Enable caching and load root server info
zone "named.root" {
        type hint;
        file "";
};

// All our DNS information is stored in /var/named/mydomain_name.db
// (eg. if mydomain.name = foobar.com then use foobar_com.db)
zone "mydomain.name" {
        type master;
        file "mydomain_name.db";
        allow-transfer { 123.12.41.40; };
};

// Reverse lookups for 123.12.41.*, .42.*, .43.*, .44.* class C's
// (these are example Class C's only -- do not use!)
zone "12.123.IN-ADDR.ARPA" {
        type master;
        file "123_12.rev";
        allow-transfer { 123.12.41.40; };
};

// Reverse lookups for 126.27.18.*, .19.*, .20.* class C's
// (these are example Class C's only -- do not use!)
zone "27.126.IN-ADDR.ARPA" {
        type master;
        file "126_27.rev";
        allow-transfer { 123.12.41.40; };
};
```

   Tip: Tip: Make note of the `allow-transfer` options above, which restricts DNS zone transfers to a given IP address. In our example, we are allowing the host at

123.12.41.40 (probably a slave DNS server in our domain) to request zone transfers. If you omit this option, anyone on the Internet will be able to request such transfers. As the information provided is often used by spammers and IP spoofers, I strongly recommend you restrict zone transfers except to your slave DNS server(s), or use the loopback address, ``127.0.0.1'' instead.

4. Now you can set up your DNS tables in the ``var/named/'' directory as configured in the ``/etc/named.conf'' file in step three. Configuring DNS database files for the first time is a major undertaking, and is beyond the scope of this document. There are several guides, online and in printed form that should be referred to. However, several examples are provided below.

   Sample entries in the ``/var/named/mydomain_name.db'' forward lookup file:

```
; This is the Start of Authority (SOA) record.  Contains contact
; & other information about the name server.  The serial number
; must be changed whenever the file is updated (to inform secondary
; servers that zone information has changed).
    @ IN SOA mydomain.name.  postmaster.mydomain.name. (
        19990811        ; Serial number
        3600            ; 1 hour refresh
        300             ; 5 minutes retry
        172800          ; 2 days expiry
        43200 )         ; 12 hours minimum

; List the name servers in use.  Unresolved (entries in other zones)
; will go to our ISP's name server isp.domain.name.com
        IN NS           mydomain.name.
        IN NS           isp.domain.name.com.

; This is the mail-exchanger.  You can list more than one (if
; applicable), with the integer field indicating priority (lowest
; being a higher priority)
        IN MX           mail.mydomain.name.

; Provides optional information on the machine type & operating system
; used for the server
        IN HINFO        Pentium/350     LINUX

; A list of machine names & addresses
    spock.mydomain.name.    IN A    123.12.41.40   ; OpenVMS Alpha
    mail.mydomain.name.     IN A    123.12.41.41   ; Linux (main server)
    kirk.mydomain.name.     IN A    123.12.41.42   ; Windows NT (blech!)

; Including any in our other class C's
    twixel.mydomain.name.   IN A    126.27.18.161  ; Linux test machine
    foxone.mydomain.name.   IN A    126.27.18.162  ; Linux devel. kernel

; Alias (canonical) names
    gopher      IN CNAME        mail.mydomain.name.
    ftp         IN CNAME        mail.mydomain.name.
    www         IN CNAME        mail.mydomain.name.
```

   Sample entries in the ``/var/named/123_12.rev'' reverse lookup file:

```
; This is the Start of Authority record.  Same as in forward lookup table.
    @ IN SOA mydomain.name.  postmaster.mydomain.name. (
        19990811        ; Serial number
        3600            ; 1 hour refresh
```

```
        300             ; 5 minutes retry
        172800          ; 2 days expiry
        43200 )         ; 12 hours minimum

; Name servers listed as in forward lookup table
        IN NS           mail.mydomain.name.
        IN NS           isp.domain.name.com.

; A list of machine names & addresses, in reverse.  We are mapping
; more than one class C here, so we need to list the class B portion
; as well.
    40.41       IN PTR    spock.mydomain.name.
    41.41       IN PTR    mail.mydomain.name.
    42.41       IN PTR    kirk.mydomain.name.

; As you can see, we can map our other class C's as long as they are
; under the 123.12.* class B addresses
    24.42       IN PTR    tsingtao.mydomain.name.
    250.42      IN PTR    redstripe.mydomain.name.
    24.43       IN PTR    kirin.mydomain.name.
    66.44       IN PTR    sapporo.mydomain.name.

; No alias (canonical) names should be listed in the reverse lookup
; file (for obvious reasons).
```

Any other reverse lookup files needed to map addresses in a different class B (such as 126.27.*) can be created, and would look much the same as the example reverse lookup file above.

5. Make sure the named daemon is running. This daemon is usually started from the ``/etc/rc.d/init.d/named'' file upon system boot. You can also start and stop the daemon manually; type ``named start'' and ``named stop'', respectively.
6. Whenever changes are made to the DNS tables, the DNS server should be restarted by typing ``/etc/rc.d/init.d/named restart''. You may then wish to test your changes by using a tool such as "nslookup" to query the machine you have added or changed.

More information on configuring DNS services can be found in the ``*DNS−HOWTO*'' guide at http://metalab.unc.edu/Linux/HOWTO/DNS−HOWTO−5.html.

7.3. Internet User Authentication with TACACS

At my place of employment, for TACACS authentication of dial−up Internet users (who are connecting to our modem pool which are in turn connected to a couple of Cisco 250x access servers), we are using the Vikas version of "xtacacsd".

After compiling and installing the Vikas package (latest versions are available from ftp://ftp.navya.com/pub/vikas; I don't believe the package is available in RPM format), you should add the following entries to the ``/etc/inetd.conf'' file so that the daemon will be loaded by the inetd daemon whenever a TACACS request is received.

```
# TACACS is a user authentication protocol used for Cisco Router products.
tacacs dgram udp wait root /etc/xtacacsd xtacacsd -c /etc/xtacacsd-conf
```

Next, you should edit the ``/etc/xtacacsd−conf'' file and customize it, as necessary, for your system (however you will probably be able to use the default settings as−is).

Note: Note: If you are using shadow passwords (see Section 6.6 for details), you will have some problems with this package. Unfortunately, PAM (Pluggable Authentication Module), which Red Hat uses for user authentication, is not supported by this package. One workaround that I use is to keep a separate ``passwd'' file in ``/usr/local/xtacacs/etc/'' which matches the one in /etc/ but is non−shadowed. This is a bit of a hassle, and if you choose to do this make sure you set permissions on the other password file to make sure it is only readable by root:

```
chmod a−wr,u+r /usr/local/xtacacs/etc/passwd
```

If you do indeed use shadow, you will most certainly need to edit the ``/etc/xtacacsd−conf'' file and location of the non−shadowed password file (assuming you are using the workaround I have suggested above).

The next step is to configure your access server(s) to authenticate logins for the desired devices (such as dial−up modems) with TACACS. Here is a sample session on how this is done:

```
mail:/tftpboot# telnet xyzrouter

Escape character is '^]'.
User Access Verification
Password: ****
xyzrouter> enable
Password: ****
xyzrouter# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
xyzrouter(config)# tacacs−server attempts 3
xyzrouter(config)# tacacs−server authenticate connections
xyzrouter(config)# tacacs−server extended
xyzrouter(config)# tacacs−server host 123.12.41.41
xyzrouter(config)# tacacs−server notify connections
xyzrouter(config)# tacacs−server notify enable
xyzrouter(config)# tacacs−server notify logouts
xyzrouter(config)# tacacs−server notify slip
xyzrouter(config)# line 2 10
xyzrouter(config−line)# login tacacs
xyzrouter(config−line)# exit
xyzrouter(config)# exit
xyzrouter# write
Building configuration...
[OK]
xyzrouter# exit

Connection closed by foreign host.
```

All TACACS activity log messages will be recorded in ``/var/log/messages'' for your perusal.

7.4. Windows−style File and Print Services with Samba

Linux can provide SMB services (eg. WfW, Win95, and NT−style network file & printer sharing), using the Samba package. This section will describe how to configure shares, and how to access them from client machines.

The Samba package is included with the Red Hat distribution, you can check if it is installed and what version you have by typing:

```
rpm -q samba
```

If it isn't installed, you will need to install it using the RPM utility. See Section 10.1 for details on how to do this.

The most important Samba files you should concern yourself with are:

/etc/smb.conf

Samba configuration file where shares and other configuration parameters are set up (see below)

/var/log/samba/

Location of Samba log files

/home/samba/

Suggested location where file shares should be set up. However, you should choose a location where you have enough space on the file system to accomodate the files you will store. Personally, I usually set up a large partition mounted on /archive/ and place my shares here.

The file ``/etc/smb.conf'' contains configuration information on file & print shares. The first few lines of the file contain global configuration directives, which are common to all shares (unless they are over–ridden on a per–share basis), followed by share sections.

The Samba installation includes a default smb.conf file which in many cases should be adequate for your needs and require only a few changes.

Here is an example of this file (which I have heavily customized to show you some of the more important and interesting options):

```
# Items common to all shares (unless over-ridden on a per-share basis)
[global]
   # Number of minutes of inactivity before client is disconnected
   # to avoid consuming resources.  Most clients will automatically
   # reconnect so this is a good idea to enable.
   dead time = 10

   # Don't let users connect as "root", just-in-case.  :-)
   invalid users = root

   # Specify the account for guest shares (shares that don't require
   # a password to connect to.  This username must be a valid user
   # in the /etc/passwd file.
   guest account = guest

   # Specify where log files should be written to.  The "%m" suffix
   # means that log files will be created in the format
   # log.machine-name (eg. "log.twixel")
   log file = /usr/local/samba/logs/log.%m

   # Maximum size of log file, in Kilobytes.
   max log size = 1000
```

```
   # Password level 3 means that case is not an issue when entering
   # passwords.  A little less secure than level 1 or 2 would be,
   # but seems to be a fair compromise for user convenience.
   password level = 3

   # Specify that all shares should appear in the browse list
   # (override any you don't want on a per-share basis).
   browseable = yes

   # If this is enabled, you can see active connections using the
   # "smbstatus" command.
   status = yes

   # The level of debugging information that is recorded in the log
   # files.  Higher values generate more information (which is
   # probably not very useful, most of the time).
   debug level = 2

   # This will send any Windows-style "POPUP" messages received on
   # the server to the postmaster by e-mail.  Not very useful, but
   # an interesting demonstration of what can be accomplished.
   message command = /bin/mail -s 'Message from %f on %m' postmaster < %s; rm %s &

   # This is a form of caching that, when enabled, may improve
   # performance when reading files.
   read prediction = true

   # A list of services that should be added automatically to the
   # browse-list.
   auto services = cdrom

   # The location of your "printcap" file, a text file containing
   # definitions for your printers.
   printcap name = /etc/printcap

   # If enabled all printers in the /etc/printcap file will be
   # loaded into the browse-list.
   load printers = yes

   # The print command by which data is spooled to a printer under Linux.
   print command = lpr -r -P%p %s

   # The print command by which job queue information (printer status)
   # can be obtained.
   lpq command = lpq -P%p

   # The print command by which unwanted print jobs can be deleted
   # from the queue.
   lprm command = lprm -P%p %j

   # The level at which Samba advertises itself for browse elections.
   # Currently set to a high value to give it an even "foot-hold" with
   # any swarmy NT servers on the network.  :-)
   os level = 34

# These are user's personal shares.  If the client's username matches on the
# server, they can access their home directory (provided they enter the
# correct password).
[homes]
   # The comments appear in the browse list.
   comment = Home Directories
```

```
   # This matches the username of the client to that of the share.
   # If they do not match, no share will be displayed in the browse
   # list, or available to connect to.
   user = %S

   # The path to the share.  For example, "smithj" would map to
   # "/home/smithj"
   path = /home/%S

   # If enabled, allow read/write access to the shares.
   writeable = yes

   # Just an inverted synonym for "writeable".  We don't *really* need
   # to use both.  :-)
   read only = no

   # Keep this disabled so that a password is required to access these
   # shares.
   public = no

   # We don't want this share (after all, it is private) to appear in
   # the browse-list of other users.
   browseable = no

# This is a publicly available print share, called "hp_laser".  It appears
# on the browse lists and can be accessed without a password by any client.
[hp_laser]
   # The comment that appears in the browse-list.
   comment = Main office printer (HP Laserjet 400)

   # The username that this share is accessed as (guest means all users).
   user = guest

   # All generated print files will first be created in the /tmp
   # directory.
   path = /tmp

   # Do not allow file creation except through print spooling.
   writeable = no

   # Set permissions accordingly -- root access to print jobs only.
   create mode = 0700

   # If this is enabled a password is not required to access the share.
   public = yes

   # This should be enabled to indicate that this is a printer share.
   printable = yes

# Here is a service providing access to the CD-ROM device.
[cdrom]
   comment = Shared CD-ROM drive on Linux
   user = guest
   path = /cdrom
   writeable = no
   read only = true
   browseable = yes
   public = yes
   guest ok = yes
```

> **Tip:** Tip: Recent versions of Samba, from 2.0 onwards, provide a very slick web−based configuration utility called ``*swat*'', which makes the process much more user−friendly. The utility listens on TCP port *901* of your server, so to use the utility just point your favourite web browser as follows:

```
mydomain.name:901
```

(Of course, in order to use the SWAT utility you will need to have a web server running, such as Apache. See Section 7.1 for details.)

The latest Samba versions also add considerable features in comparison with versions prior to 2.0. It is worth taking the time to upgrade this package.

A client must have a TCP/IP network stack running in order to connect to shares. Further, for browsing to work, the TCP/IP protocol must be bound to NETBEUI. Under Windows 95 this can be configured from the "Network" icon from within the Control Panel.

Assuming the client has been configured properly, you should see the server shares appear in their "Network Neighborhood" (or equivalent browsing scheme if you are not using Windows 95/NT). You can then map network drives from the network neighborhood, or type in an absolute path to the share (*eg. "\\mail\cdrom"*). If the shared service requires a password to be entered, you will be prompted for one.

More information on Samba can be obtained from the Samba Home Page at http://samba.anu.edu.au/samba/.

7.5. Macintosh−style File and Print Services with Netatalk

Linux can also provide Appleshare services (eg. Macintosh−style network file & printer sharing), using the Netatalk package. This section will describe how to configure shares, and how to access them from client machines.

In order to use Netatalk, you will need to have Appletalk networking support in your Linux kernel. Stock kernels from Red Hat usually already include this support as a module, or you can compile your own custom kernel with such support.

> **Note:** Note: Make sure Appletalk support is compiled in as a *module* and not included as part of the kernel (see Section 10.4 for details on how to upgrade or customize the Linux kernel). Otherwise you will have difficulties when stopping and then restarting the Netatalk daemon.

Once you have ensured your kernel is capable of supporting Appletalk, you will need to install the Netatalk package. Because Netatalk is not included with the Red Hat distribution, you will have to download and install a copy. The Netatalk package can be found on Red Hat's "contrib" site, at ftp://ftp.redhat.com/contrib/libc6/i386/.

After Netatalk has been installed, you may need to modify one or more configuration files in ``/etc/atalk/''. Most of the files contain sample configuration examples, and therefore are at least somewhat self−documenting. The files are:

config

This file contains configuration information for tuning your Netatalk daemon. This information is specified in environment variables, and this file is "sourced" (ie. read) by the Netatalk start up script before the service is started. You can specify the number of simultaneous connections, whether or not guest logins are allowed, etc. You will almost certainly want to modify this file according to your needs.

atalk.conf

This file contains information on which network interface to use, as well as your Appletalk routing, name registration, and other related information. You will likely not need to modify this file; the required network information is detected and added to this file the first time you start the Netatalk server. However, you may wish to add your server name.

Note: *Note: Type ``*`man atalkd`*'' for more information on this file.*

afpd.conf

This file allows you to specify additional parameters which are passed to Netatalk by means of command−line options. You can specify which port or IP address you wish to run the Netatalk server on, add a login message that is displayed to connecting users, as well as other related options. You will likely not need to modify this file.

Note: *Note: Type ``*`man afpd`*'' for more information on this file.*

papd.conf

The file contains configuration information for enabling Mac users to print to network printer shares. I haven't played with this yet, so unfortunately I can't offer any advice on it.

Note: *Note: Type ``*`man papd`*'' for more information on this file.*

AppleVolumes.default

This file lists the available file shares that a Mac user will see after logging in. To enable a share, enter the path to the file directory, followed by a textual description of it. For example:

```
~                "Home"
/archive/busdept "Business Department Common Files"
```

(The above will provide two shares to connecting Mac users: their home directory, as well as a shared area for the business department.)

Tip: Tip: A neat trick here is to set up shares with the same file paths under Samba, which would provide your users with platform− independent file shares for both your Mac as well as your Windows users. See Section 7.4 for details on using Samba.

AppleVolumes.system

This file also lists file shares just like ``AppleVolumes.default'' does, the difference being that these shares will be made available to *all* users, regardless of whether or not they log in. This file also contains file−type mappings. You will likely not need to modify this file unless you want to add general shares available to anyone; this is probably a bad idea for most people.

Once everything has been set up with appropriate configuration information, you can start the Netatalk services manually by typing:

```
/etc/rc.d/init.d/atalk start
```

(The services should start up automatically at system boot).

More information on Netatalk can be obtained from the Netatalk Home Page at http://www.umich.edu/~rsug/netatalk/. In addition, very helpful configuration information is available in the Linux Netatalk HOWTO, available at http://thehamptons.com/anders/netatalk/.

7.6. Network File System (NFS) Services

Linux can act as both client and server for file systems shared using the Network File System (NFS) protocol, which is the defacto standard for providing file system mounts among Unix systems.

> **Note:** Note: Please be aware that having an NFS service available on your system can be a security risk. Personally, I don't recommend using it.

In order to use NFS, you will need to ensure that NFS support has been included in your kernel or kernel modules. See Section 10.4 for details on how to upgrade or customize the Linux kernel.

NFS shares are configured by modifying the ``/etc/exports'' file. Here are some example entries, showing some of the options available:

```
/archive spock.mydomain.name(ro)
/archive2 spock.mydomain.name(ro)
/mnt/cdrom other.domain(ro)
/archive2 10.23.14.8(ro,insecure)
```

The first couple of lines allow the host, ``spock.mydomain.name'' access to both the ``/archive'' as well as the ``/archive2'' directories via NFS. These shares are made available read–only with the ``(ro)'' option. For security reasons, it is a good idea to do this for all of your NFS shares if at all possible.

The third line will allow any host in the ``domain.name'' domain name space to access the CD–ROM drive. Of course, it is necessary to mount the CD–ROM device to ``/mnt/cdrom'' first.

> **Note:** Note: Using the ``(ro))'' option to mark this device read–only may seem a bit redundant, however doing so will prevent a miscreant from writing to a real file system should the CD–ROM device not be mounted.

After you have made changes to the ``/etc/exports'' file, you will need to restart the NFS daemon. To do so, type:

```
/etc/rc.d/init.d/nfs restart
```

You can also configure your NFS mount points with the ``Network Configurator'' tool included in the ``Linuxconf'' utility. For more information on the Linuxconf utility, see Section 7.7.

More information on NFS can be found in the ``*NFS−HOWTO*'' guide at http://metalab.unc.edu/LDP/HOWTO/NFS−HOWTO.html, as well as in the man pages on ``nfsd'' and ``exports''.

7.7. Configuration from A−Z with Linuxconf

There is an excellent tool called ``linuxconf'' which can make many configuration issues easier to do. Linuxconf runs on whatever means of display environment it has available to it −− you can run it from the console, over a telnet session, and as a GUI−based tool under X and it will automatically start up in the appropriate manner.

If you need to adjust your system time, modify your network settings, set up file systems, perform user administration, as well as perform many other administrative and configuration duties, you should give this tool a try. The only caveat I would give is that, at the time of this writing, the GUI−based tool is still a bit "buggy" and at times may stop responding to mouse clicks. However, this tool is a promising work in progress, and future revisions should become quite usable.

Chapter 8. Backup and Restore Procedures

Performing regular backups should be considered one of a responsible system administrator's top priorities. Although Linux is an extremely reliable operating system, failures can, do, and probably will occur. They may be caused by hardware failure, power outages, or other unforeseen problems.

More likely will be those problems caused by human error, resulting in undesired changes to, or even deletions of, crucial files. If you are hosting users on your system, you will most certainly be requested to restore an inadvertently deleted file or two.

If you perform regular backups, preferably on a daily basis (at least for user files which are updated often), you will hopefully reduce the possibility of, and increase your recovery from, such file lossage.

The safest method of doing backups is to record them on separate media, such as tape, removable drive, writeable CD, etc., and then store your backup sets in a location separate from your Linux system. Sometimes this may not be practical –– perhaps you do not have a fire–proof vault in which you can store your backup tapes! Or perhaps you do not have access to such an external backup system in the first place. Nonetheless, backups can still be performed, albeit on a slightly limited basis.

At my place of employment, I perform backups on several Linux servers. Depending on the situation, some of these backup sets are written to tapes, others are written to a separate server over the network, while still others are simply written to a separate disk partition (for example, in the ``/archive/'' file system) by an automatic cron job (perhaps because the server is in a remote location, for which a daily visit to perform a tape backup is impractical or impossible).

At home, I do not have an external backup system, nor do I have massive amounts of available disk space to write a backup image. Therefore, I instead back up only my user files on ``/home/'' as well as some customized configuration files in ``/etc/'', writing the backup set to a separate disk partition.

8.1. Server Backup Procedures

There are a variety of methods of performing backups with Linux. These include command–line tools included with every Linux distribution, such as ``dd'', ``dump'', ``cpio'', as well as ``tar''. Also available are text–based utilities, such as ``Amanda'' and ``Taper'', which is designed to add a more user–friendly interface to the backup and restore procedures. There are GUI–based utilities as well, such as ``KDat''. Finally, commercial backup utilities are also available, such as ``BRU'' and ``PerfectBackup+''. Any one of these backup solutions can provide protection for your valuable data.

A brief listing of some of the tools available, including where they can be obtained, can be found on the "Linux Applications and Utilities Page", at http://www.xnet.com/~blatura/linapp2.html#back. When deciding on a backup solution, you will need to consider the following factors:

- *Portability* – Is backup portability (ie. the ability to backup on one Linux distribution or implementation of Unix and restore to another; for example from Solaris to Red Hat Linux) important to you? If so, you'll probably want to choose one of the command–line tools (eg. ``dd'', ``dump'', ``cpio'', or ``tar''), because you can be reasonably sure that such tools will be available on any *nix system.
- *Unattended or automated backups* – Is the ability to automate backups so that they can be performed

at regular intervals without human intervention important to you? If so, you will need to choose both a tool and a backup medium which will support such a backup scheme.
- *User−friendliness* − Is a user−friendly interface important to you? If so, you will likely want to choose a tool which provides a text− or GUI−based interface. The commercial utilities may provide the easiest interfaces as well as added technical support.
- *Remote backups* − Is the ability to start backups and restores from a remote machine important to you? If so, you'll probably want to choose one of the command−line tools or text−based utilities instead of the GUI−based utilities (unless you have a reasonably fast network connection and the ability to run remote X sessions).
- *Network backups* − Is performing backups and restores to and from networked hosts important to you? If so, you'll probably want to use one of several of the command−line utilities (such as ``tar'') which support network access to backup devices, or a specialized utility such as ``Amanda'' or one of several commercial utilities.
- *Media types* − Backups can be stored on a variety of medium, such as tape, an extra hard drive, ZIP drives, or rewritable CDs. Consider cost vs. reliability, storage capacitity, and transfer speed.

Caution
Caution: When backing up your file systems, do *not* include the ``/proc'' pseudo−filesystem! The files in /proc are not actually files but are simply file−like links which describe and point to kernel data structures. Backing up a file like ``/proc/kcore'', which is actually a pseudo−file containing the contents of your entire memory, seems like a pretty big waste of tape to me! :−) You will also likely want to avoid backing up the ``/mnt'' file system, unless you have the peculiar desire to back up the files from your CD−ROM device, floppy drive, network file shares, or other mounted devices.

Obviously, the procedures for performing a backup and restore will differ depending on your choice of a backup solution. However, in this section, I will discuss methods for performing backups with the two tools I use most: ``tar'' (whose name stands for "Tape ARchiver"), which is a command−line backup tool largely portable across *nix systems; as well as ``KDat'', a GUI−based tape backup utility which comes included with the KDE packages (see Section 5.6 for more information on KDE).

Finally, I should add that, depending on your choice of backup solution, even if the tool doesn't have the ability built−in to schedule automated or unattended backups, you may be able to automate such backups by using the cron facilities. See Section 9.4 for details on using cron and on creating crontab schedule files.

8.1.1. Backing up with ``tar'':

If you decide to use ``tar'' as your backup solution, you should probably take the time to get to know the various command−line options that are available; type "`man tar`" for a comprehensive list. You will also need to know how to access the appropriate backup media; although all devices are treated like files in the Unix world, if you are writing to a character device such as a tape, the name of the "file" is the device name itself (eg. ``/dev/nst0'' for a SCSI−based tape drive).

The following command will perform a backup of your entire Linux system onto the ``/archive/'' file system, with the exception of the ``/proc/'' pseudo−filesystem, any mounted file systems in ``/mnt/'', the ``/archive/'' file system (no sense backing up our backup sets!), as well as Squid's rather large cache files (which are, in my opinion, a waste of backup media and unnecessary to back up):

```
tar -zcvpf /archive/full-backup-`date '+%d-%B-%Y'`.tar.gz \
    --directory / --exclude=mnt --exclude=proc --exclude=var/spool/squid .
```

Don't be intimidated by the length of the command above! As we break it down into its components, you will see the beauty of this powerful utility.

The above command specifies the options ``*z*'' (compress; the backup data will be compressed with ``gzip''), ``*c*'' (create; an archive file is begin created), ``*v*'' (verbose; display a list of files as they get backed up), ``*p*'' (preserve permissions; file protection information will be "remembered" so they can be restored). The ``*f*'' (file) option states that the very next argument will be the name of the archive file (or device) being written. Notice how a filename which contains the current date is derived, simply by enclosing the ``date'' command between two back−quote characters. A common naming convention is to add a ``tar'' suffix for non−compressed archives, and a ``tar.gz'' suffix for compressed ones.

The ``*−−directory*'' option tells tar to first switch to the following directory path (the ``/'' directory in this example) prior to starting the backup. The ``*−−exclude*'' options tell tar not to bother backing up the specified directories or files. Finally, the ``.'' character tells tar that it should back up everything in the current directory.

> **Note:** Note: It is important to realize that the options to tar are cAsE−sEnSiTiVe! In addition, most of the options can be specified as either single mneumonic characters (eg. ``f''), or by their easier−to−memorize full option names (eg. ``file''). The mneumonic representations are identified by prefixing them with a ``−'' character, while the full names are prefixed with two such characters. Again, see the "man" pages for information on using tar.

Another example, this time writing only the specified file systems (as opposed to writing them *all* with exceptions as demonstrated in the example above) onto a SCSI tape drive follows:

```
tar -cvpf /dev/nst0 --label="Backup set created on `date '+%d-%B-%Y'`." \
    --directory / --exclude=var/spool/ etc home usr/local var/spool
```

In the above command, notice that the ``*z*'' (compress) option is not used. I *strongly* recommend against writing compressed data to tape, because if data on a portion of the tape becomes corrupted, you will lose your entire backup set! However, archive files stored without compression have a very high recoverability for non−affected files, even if portions of the tape archive are corrupted.

Because the tape drive is a character device, it is not possible to specify an actual file name. Therefore, the file name used as an argument to tar is simply the name of the device, ``/dev/nst0'', the first tape device on the SCSI bus.

> **Note:** Note: The ``/dev/nst0'' device does not rewind after the backup set is written; therefore it is possible to write multiple sets on one tape. (You may also refer to the device as ``/dev/st0'', in which case the tape is automatically rewound after the backup set is written.)

Since we aren't able to specify a filename for the backup set, the ``*−−label*'' option can be used to write some information about the backup set into the archive file itself.

Finally, only the files contained in the ``/etc/'', ``/home/'', ``/usr/local'', and ``/var/spool/'' (with the exception of Squid's cache data files) are written to the tape.

When working with tapes, you can use the following commands to rewind, and then eject your tape:

```
mt -f /dev/nst0 rewind
mt -f /dev/nst0 offline
```

> **Tip:** Tip: You will notice that leading ``/" (slash) characters are stripped by tar when an archive file is created. This is tar's default mode of operation, and it is intended to protect you from overwriting critical files with older versions of those files, should you mistakenly recover the wrong file(s) in a restore operation. If you really dislike this behavior (remember, its a *feature*!) you can specify the ``--absolute-paths" option to tar, which will preserve the leading slashes. However, I don't recommend doing so, as it is *Dangerous*!

8.1.2. Backing up with ``KDat":

If you are using the KDE desktop environment, I believe you will find the ``KDat" utility both powerful as well as user–friendly. In addition, an added bonus is that KDat uses ``tar" as its backup engine. Therefore, backup sets written with KDat can be read not only with KDat but with tar as well! This makes KDat a very nice choice for both user–friendliness as well as backup portability.

> **Tip:** Tip: Even if you choose not to use nor install the full KDE package, you can *still* use KDat as long as you have the Qt libraries installed.

The first time you run the KDat program, you will need to create a backup profile. Such a profile tells KDat which files on your system you would like to back up. If you wish, you can create more than one backup profile, depending on your needs (for example, you could create a profile called "*Full Backup*" for a full system backup, and "*Quick Backup*" for a backup of user files only).

To create a backup profile, either choose "`Create Backup Profile`" from the "`File`" option on menu bar (or right–click on the "`Backup Profiles`" folder, then choose "`Create Backup Profile`"). On the right hand side of the KDat window, you can change various settings, such as the profile name, archive name, tar options, as well as others. Click the "`Help`" menu for more information on what these settings are for.

To specify which files should be included in your backup profile, left–click the checkbox beside the ``/" directory folder. This will enable all files in and below this directory for backups. Then, left–click on the small ``+" sign beside the folder. This will expand the folder, showing a list of files in and below it. This will allow you to exclude any files you do not wish to backup; simply left–click the checkbox beside each file or directory you wish to exclude. For example, a full backup should probably have every file and directory checkmarked, with the exception of the ``/proc" (a pseudo–filesystem containing information about your running system), ``/mnt" (a directory below which CD–ROM drives, floppies, and network shares are usually mounted), and, if you are a Squid user, ``/var/spool/squid" (Squid's cache data files). Once you have selected the appropriate files, left–click on the backup profile you are creating, then left–click the "`Files >>`" button to move the selected files list to your backup profile.

> **Note:** Note: Should your server data be larger in size than can be physically stored on a tape, you will need to create separate backup profiles, one for each portion of your backup set.

To actually perform a backup, insert a tape into the drive, and then choose "`Mount Tape`" from the "`File`" menu (or left–click the icon that looks like a tape). This will "*mount*" the tape (actually, because a tape device is a character device, it isn't actually possible to mount it –– what KDat actually does is to first rewind the tape, attempts to read in header information, and if successful, find the corresponding tape index on your hard drive. Otherwise, KDat will prompt you to format the tape.

> **Note:** (Note: If KDat keeps complaining that a tape isn't in the drive and it actually *is* in the drive, you should ensure the correct tape device name is specified in the preferences; left–click the "`Edit`" option on the menu bar and choose "`User Preferences`".)

Once KDat has mounted the tape, before you start the backup you must first choose the backup profile you wish to use for the backup. To start the backup, simply right–click on the desired backup profile, and then left–click on the "`Backup`" option. KDat will first display a dialog box showing you the details of the backup profile you have selected; left–click the "`Ok`" button to start the backup.

While the backup is in progress, KDat will display a dialog box showing various statistical information (elapsed time, backup size, backup rate, estimated time remaining, as well as the number of files and total bytes written), and display a list of files as they are backed up. A full backup containing several gigabytes of data might take several hours to complete. If you find it necessary, you can left–click the "Abort" button at any time to interrupt the backup process.

Once the backup is complete, you can unmount the tape by choosing "Edit" from the menu bar, and then "Unmount Tape", or left–click on the tape icon, which will rewind and eject the tape.

8.2. Server Restore Procedures

Unarguably, the one thing that is more important than performing regular backups is having them available when it comes time to recover an important file!

Obviously, as discussed in Section 8.1, the procedures for performing a restore will differ depending on your choice of a backup solution. In this section, I will discuss methods for restoring files which have been backed up with ``tar" and "KDat".

8.2.1. Restoring with ``tar":

The following command will restore all files from the ``full-backup-09-October-1999.tar.gz" archive, which is an example backup of our Linux system (as created in the example commands shown in Section 8.1.1:

```
tar -zxvpf /archive/full-backup-09-October-1999.tar.gz
```

The above command extracts all files contained in the compressed archive, preserving original file ownership and permissions. The ``x" option stands for extract. (The other options are described in Section 8.1.1.

Caution
Caution: Extracting files from a tar archive can be a dangerous thing to do, and should therefore be done with caution. Perhaps the files were not archived without a file path prepended (a few misguided or uninformed developers distribute tarballs of their software offerings like this), meaning they will all be extracted into the current directory. Perhaps the files were archived with leading ``/" slashes (by specify the ``--absolute-paths" option when the archive was created), meaning the files will be restored to their absolute locations (even if you didn't want them to be). Or, perhaps the files were archived *without* leading ``/" slashes, meaning the files will be restored under the current directory (even if you didn't want them to

be). This of course, depends on how the backup was created. For this reason, I strongly recommend testing your ``tar" command with a ``t" (*type*) option *first,* and then replace the ``t" with an ``x" (*extract*) when you are absolutely sure the command will do what you expect it to.

If you do not need to restore all files contained in the archive, you can specify one or more files that you wish to restore, as in the following example:

```
tar -zxvpf /archive/full-backup-09-October-1999.tar.gz \
              etc/profile usr/local/bin/tolower
```

The above command restores the ``etc/profile" and ``usr/local/bin/tolower" files from the example archive.

Tip: If you are trying to restore only one or a few files from your archive, you will not be successful unless you specify the < file name and directory path *exactly* as stored in the archive. The following example might help out:

```
tar -ztvpf /archive/full-backup-09-October-1999.tar.gz \
    | grep -i profile
```

In the above example, all files contained in the archive are listed by file name. The resulting output is then piped to the ``grep" command (using grep's ``i" option to ignore mixed case), displaying any files containing ``profile" in either the directory path or file name. Once you determine the exact file name you wish to restore, you can then specify it for extraction in a regular tar command expression.

As mentioned in Section 8.1, when creating an archive file, tar will strip leading ``/" (slash) characters from file path names. This means that restore files may not end up in the same locations they were backed up from. Therefore, either change to the ``/" root directory, or use the ``--directory /" option.

Note: Note: A far safer solution is to restore the desired files under a different directory (for example, your home directory), and then compare, move, or update the files to their original locations afterward.

8.2.2. Restoring with ``KDat":

To restore one or more files from a KDat–created backup set, insert the backup tape into the drive, choose "Mount Tape" from the "File" menu option (or left–click on the icon that looks like a tape).

KDat will try to read header information from the tape, and if successful, will then try to find the tape index which matches the identification found in the tape header. This tape index is stored on your hard drive, and is a unique file created for each backup tape formatted by KDat, and is updated each time you perform a backup.

If this corresponding tape index is missing (perhaps you are restoring from a backup set created on another machine, or the index file was deleted or somehow corrupted on your hard drive), KDat will inform you of this fact, and ask you if it is okay to recreate the index by reading the tape. Because you will need to recreate it before you will be able to restore your desired files, it makes perfect sense to left–click "Yes".

Note: (Note: Once a tape is reindexed, its name is changed to "*Reindexed Tape*". You should rename the tape to its original name.)

Once the tape index has been successfully read, it can then used to select the directories or files you wish to restore from the backup set, in much the same manner you used when creating your backup profiles (see Section 8.1 for detailed instructions on the file selection process).

Once you have selected the appropriate files, you can start the restoration process by choosing "`Restore...`" from the "File" option on the menu bar (or left–click the tape restore icon). KDat will display a dialog box, allowing you to confirm which files will be restored. In addition, you have the option of specifying a directory into which files will be restored. This will allow you to restore critical system files into your home directory, and then compare, move, or update those files to their intended location later on. This is actually the safest way of restoring files.

To begin the recovery process, click the "`Okay`" button. KDat will then scan the tape and restore the selected files.

On occasion, you may find it necessary or useful to restore one or more files from a backup set created with KDat *without* using KDat to do so. Perhaps you would like to restore such files on a system that does not offer a GUI–based environment, or would like to do so over a slow network connection through which remote execution of KDat would be impractical. Fortunately, KDat writes its backup data sets using the ``tar'' tool, a command–line based tool that is available on any *nix system.

Should you wish to restore your KDat–created backup set using tar, simply do so using whatever options you would with any normal backup set created with tar itself. Bear in mind, however, that data sets are not stored in compressed format.

> **Note:** Note: You will almost certainly get an error message when trying to access the KDat backup set with tar. This is because of the header and other information that KDat added to the tape when it was first formatted. Simply repeat the tar command two or three times to skip to the beginning of the actual tar archive file.

8.3. Cisco Router Configuration Backups

At my place of employment, we have a WAN connecting several remote locations. These remote locations have Cisco routers connected via ISDN, or in some instances, Centrex data circuits, to provide Internet and WAN connectivity. Cisco router products allow using TFTP ("Trivial File Transfer Protocol") on a network server to read and write configuration files. Whenever a router configuration is changed, it is important to save the configuration file on the Linux server so that a backup is maintained.

Please note that Red Hat disables the TFTP service by default, because it can be a real security hole if not configured properly. The TFTP daemon allows anyone to read and write files without performing authentication. The way I personally set things up is to create a ``/tftpboot/'' directory, owned by root, and then modify the existing configuration line in the ``/etc/inetd.conf'' file to specify the file location:

```
tftpd   dgram   udp    wait   root   /usr/sbin/tcpd  in.tftpd  /tftpboot
```

> **Note:** Note: Adding the ``/tftpboot'' path at the end of the above line specifically indicates where the TFTP daemon is allowed to access files. Although you can actually leave this part out and allow TFTP to access files anywhere on your system, as TFTP is considered

somewhat of a security risk, this would probably be a very bad idea.

Once you have enabled the TFTP service, don't forget to type:

```
killall -HUP inetd
```

The above command restarts the INETD daemon to recognize whatever changes you have made to the inetd.conf file.

Creating a backup of a router configuration file involves a 3−step process: setting permissions on an existing file (or creating a new one) to allow writes, writing the backup file, and then resetting permissions to restrict access to the file. An example router backup session follows:

```
mail:~# cd /tftpboot
mail:/tftpboot# chmod a+w xyzrouter-confg
chmod: xyzrouter-confg: No such file or directory
mail:/tftpboot# touch xyzrouter-confg
mail:/tftpboot# chmod a+w loyola-confg
mail:/tftpboot# telnet xyzrouter

Escape character is '^]'.
User Access Verification
Password: ****
xyzrouter> enable
Password: ****
xyzrouter# write network
Remote host []? 123.12.41.41
Name of configuration file to write [xyzrouter-confg]?
Write file xyzrouter-confg on host 123.12.41.41? [confirm]
Building configuration...
Writing xyzrouter-confg !! [OK]
xyzrouter# exit
Connection closed by foreign host.

mail:/tftpboot# chmod a-wr,u+r xyzrouter-confg
mail:/tftpboot# exit
```

In case of router failure (caused, for example, by a power surge during a lightning storm), these backup files can be helpful to reload the router configuration. Again, restoring from a configuration file involves a 3−step process: setting permissions on the existing file, loading the file, and then resetting permissions to restrict access to the file. An example router restoration session follows.

```
mail:~# cd /tftpboot
mail:/tftpboot# chmod a+r xyzrouter-confg
mail:/tftpboot# telnet xyzrouter

Escape character is '^]'.
User Access Verification
Password: ****
xyzrouter> enable
Password: ****
xyzrouter# config network
Host or network configuration file [host]?
Address of remote host [255.255.255.255]? 123.12.41.41
Name of configuration file [xyzrouter-confg]?
Configure using loyola-confg from 123.12.41.41? [confirm]
Loading xyzrouter-confg from 123.12.41.41 (via BRI0): !
[OK - 1265/32723 bytes]
```

```
xyzrouter# write
xyzrouter# exit
Connection closed by foreign host.

mail:/tftpboot# chmod a-wr,u+r xyzrouter-confg
mail:/tftpboot# exit
```

Chapter 9. Various & Sundry Administrative Tasks

Linux has proven itself to be extremely reliable during the over four years I have had it in service as an Internet server and requires very little hands−on administration to keep it running. Where possible, many repetitive or tedious administrative tasks can and should be automated through crontab entries and script files. However, to ensure that Linux continues to operate in a trouble−free manner, various quick checks can be done from time to time. These include:

9.1. Checking Storage Space

It is important to check from time to time that adequate free space remains on the storage devices. Use the "`df`" command to get a report of available space. It will look as follows (information shown is from the Internet server at my place of employment):

```
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sda1            1888052  135908  1654551      8%   /
/dev/sdd1            4299828  100084  3977246      2%   /archive
/dev/hda2            3048303  897858  1992794     31%   /archive2
/dev/hda1              11677    1380     9694     12%   /boot
/dev/sdc1            4299828  350310  3727020      9%   /home
/dev/sdb1            4299828  598504  3478826     15%   /usr
/dev/sda2            1888083  700414  1090075     39%   /var
/dev/scd0             593958  593958        0    100%   /cdrom
```

These file−systems are pretty stable in that they have a fairly slow growth pattern.

The "/" (aka root) file−system, mounted on /dev/hda1, contains the Linux kernel, device drivers, and other directories. It also is where user mail messages are stored (*/var/spool/mail/*) as well as log files (*/var/adm/*) but as mail messages are received and log files are recycled, the available capacity stays fairly stable (an estimated growth of about 1% per month). Log files are rotated and purged automatically on a weekly basis, so you'll always have about a month's worth of log information available to you.

> **Tip:** Tip: If this file−system is growing rapidly, concentrate your efforts in the /var/spool/mail directory −− look for huge mailboxes (something like ``` ``find /var/spool/mail -size +1000k ```" would display a list of mailboxes larger than 1Mb in size). If you find a file much larger than 1,000,000 bytes in size, the user probably isn't retrieving their mail, is on a high−volume mailing list, or their e−mail package isn't configured to remove the mail from the server. Contact the user and/or clear the mail file, using "`> mailbox`", (eg. ``` ``>smithj ```" to clear Joe Smith's mail box). Also check the ``` ``/tmp/ ```" directory, which may need to be cleaned out on an occasional basis (usually old tin* files left over from aborted newsreader sessions, old print files, etc).

The "/usr/" (aka user) file−system, mounted on /dev/hda2, contains user−installable (user meaning user−installed by system administrator) software, things like your web site pages, etc. This is the largest file−system, and is also fairly slow−growth. The log files for the web pages may also be stored here, and grow in size; check and trim them periodically as needed. On my machines, at the beginning of each month the current web log files are moved to month summary logs (eg. access_log.11 for November's log entries). At the end of the year these logs are all deleted and the cycle starts again (which means each January 1st should see a fair improvement in available space).

> **Tip:** Tip: If this file−system is growing rapidly, check the ``/usr/local/etc/httpd/logs" and the ``/usr/local/squid/logs/" directories (if you have them). There may be log file(s) that are getting too large (if, perhaps, the web site received a high number of visits). If, however, the logs are purged automatically on a regular basis as I have them, you shouldn't run into any problems with space here (indeed, as the logs are used for statistical analysis of my site's traffic I'd rather not have to delete them if possible). Another place to check for potentially deletable files is in ``/usr/tmp/".

The "/home/" (aka user's personal home) file−system, mounted on /dev/hda3, contains all the user directories and personal files. Unless you are giving out shell accounts, most of these will be useless and inaccessible to the user (these directories are created when each users' accounts are created, and can later be used to forward the user's mail, etc.). However shell account users, as well as any non−shell accounts which have web pages (eg. personal web pages) will probably have them stored here. In addition, main server pages are stored here in the /home/httpd directory under Red Hat, while other distributions usually place them in the /usr file system (see Section 7.1 for more information).

This file−system is probably the slowest growth unless you are offering a lot of shell accounts.

> **Tip:** Tip: If this file−system suddenly grows in size, it is probably because one of your users is adding web pages or binary files in his/her personal space. Check the ``/var/adm/xferlog.*" log files for recent activity, which should show you which user has added to their web pages.

I also have an "/archive/" (aka archive files) file−system, mounted on /dev/hdb1, which is a spare 1.02 Gb hard drive that can be used for any purpose (eg. data files, software kits, etc.) I am using a good portion (approximately 70%) of this drive for disk−to−disk full current backups of the system). Generally speaking you can add your own devices and mount them as you wish.

I also have a CD−ROM drive, mounted as "/mnt/cdrom/" on /dev/scd0, which is a 24X−speed SCSI CD−ROM device that can read any ISO9660 formatted CD. It is used primarily for software installation, but DOS/Windows CD's can be mounted and then accessed from Windows 3.x/95/NT network shares as needed via a Samba service (see Section 7.4 for details).

The "rm" command will delete a file. Usage is ``rm filename". If you want confirmation of deletion, use the "−i" option (eg. ``rm −i *"). You would then be asked to confirm each file before it is deleted.

> **Note:** (Note: This is the default for normal shell users, but beware −− the root account will not confirm before deleting files unless you specify the "−i" option!)

Be careful you don't make a silly typo with this command −− particularly when logged in as "root" −− because you may end up regretting deleting the wrong file.

9.2. Managing Processes

From time to time you may wish to view processes that are running on Linux. To obtain a list of these processes, type ``ps −aux", which will look similar to the following:

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
```

```
bin         69  0.0  1.0   788   320  ?  S   Nov 30   0:00 /usr/sbin/rpc.portmap
frampton 10273  0.0  2.1  1136   664  p0 S    14:12   0:00 -bash
frampton 10744  0.0  1.1   820   360  p0 R    17:25   0:00 ps -aux
frampton 10745  0.0  0.8   788   264  p0 S    17:25   0:00 more
nobody   10132  0.0  1.8  1016   588  ?  S    13:36   0:00 httpd
nobody   10133  0.0  1.8   988   568  ?  S    13:36   0:00 httpd
nobody   10413  0.0  1.8  1012   580  ?  S    14:56   0:00 httpd
nobody   10416  0.0  1.8  1012   580  ?  S    14:56   0:00 httpd
nobody   10418  0.0  1.8  1012   588  ?  S    14:57   0:00 httpd
nobody   10488  0.0  1.7   976   556  ?  S    15:34   0:00 httpd
nobody   10564  0.0  1.8   988   564  ?  S    16:06   0:00 httpd
nobody   10600  0.0  1.8   988   564  ?  S    16:15   0:00 httpd
nobody   10670  0.0  1.8   988   568  ?  S    16:45   0:00 httpd
nobody   10704  0.0  1.7   976   552  ?  S    17:03   0:00 httpd
root         1  0.0  1.0   776   312  ?  S   Nov 30   1:13 init [3]
root         2  0.0  0.0     0     0  ?  SW  Nov 30   0:00 (kflushd)
root         3  0.0  0.0     0     0  ?  SW  Nov 30   0:00 (kswapd)
```

The list shows you the owner of the process ("nobody" for special services such as web servers), the process identification number, the % of CPU time the process is currently using, the % of memory the process is consuming, and other related information, as well as a description of the task itself.

To get more information on a given process, type ``ps pid" (where "pid" is the process identification number). Looking at our example above, "ps 10704" would display:

```
10704  ?  S     0:00 /usr/local/etc/httpd/httpd
```

This would tell you that this particular process is a web server (the Apache web server appears multiple times in the process list; for information on why see Section 7.1).

If you happen to notice a service is not operating, you can use the "kill -HUP pid" (where "pid" is the process identification number as shown in the process list produced with "ps"). For example, if Internet services (a process called inetd, process #123 in our example) are not working as they should, a ``kill -HUP 123" (or even safer, use the ``killall" command and specify the process name: ``killall -HUP inetd") should restart the process. The -HUP option to the kill command means "hang up"; the process knows that it is supposed to reload itself.

Another thing to try if you are unable to resolve the problem would be to shut the system down and reboot it (see Section 6.7 for details).

At times, you may find it necessary to temporarily suspend a process, and then resume its execution at a later time. For example, you may be running a CPU−intensive job and wish to burn an IDE−based CDRecordable. Since IDE−based devices rely on the CPU for much of the work behind input/output, they are prone to buffer starvation if your CPU is too busy, and you end up with a useless coaster instead of a properly prepared CD! The following two commands will suspend a process, and the resume it, respectively:

```
kill -STOP 945
kill -CONT 945
```

Red Hat provides a better way of starting and stopping some processes, which are covered in Section 9.3 below.

9.3. Starting and Stopping Processes

The Red Hat distribution of Linux provides a slightly more organized way of managing processes. Instead of hunting and killing them by finding their process id in the process table, Red Hat provides a collection of scripts in the ``/etc/rc.d/init.d'' directory which will allow you to start and stop processes as desired.

For example, to shut down the ``httpd'' (Apache web server) service, simply run the httpd script, as follows:

```
/etc/rc.d/init.d/httpd stop
```

In much the same manner, you can use the ``start'' option to start a service. Or, if you have made changes to a configuration file and wish to restart a service so those changes are recognized, you can use the ``restart'' option.

> **Note:** (Note: Oddly enough, the ``restart'' option does not seem to be supported for some services.)

9.4. Automating Tasks with Cron and Crontab files

Like most Linux users, you may find it necessary to schedule repetitive tasks to be run at a certain time. Such tasks can occur as frequently as once a minute, to as infrequently as once a year. This scheduling can be done by using the ``cron'' facilities.

The cron facilities as implemented in Linux are fairly similar to those available in other Unix implementations. However, Red Hat has adopted a slightly different way of scheduling tasks than is usually done in other distributions of Linux. Just as in other distributions, scheduling information is placed in the system ``crontab'' file (locating in the ``/etc/'' directory), using the following format:

```
minute hour day month year command
```

You can specify each time component as an integer number (eg. 1 through 12 for the months January through December), or specify one or more components as ``*'' characters which will be treated as wildcards (eg. * in the month component means the command will run at the given day and time in *every* month. Here are some examples:

```
# Mail the system logs at 4:30pm every June 15th.
30 16 15 06 * for x in /var/log/*; do cat ${x} | mail postmaster; done

# Inform the administrator, at midnight, of the changing seasons.
00 00 20 04 * echo 'Woohoo, spring is here!'
00 00 20 06 * echo 'Yeah, summer has arrived, time to hit the beach!'
00 00 20 10 * echo 'Fall has arrived.  Get those jackets out.  :-('
00 00 20 12 * echo 'Time for 5 months of misery.  ;-('
```

Note that commands which produce output to standard out (ie. a terminal) such as the examples above using ``echo'' will have their output mailed to the ``root'' account. If you want to avoid this, simply pipe the output to the null device as follows:

```
00 06 * * * echo 'I bug the system administrator daily at 6:00am!' >/dev/null
```

In addition to the standard ``crontab'' entries, Red Hat adds several directories:

```
/etc/cron.hourly/
/etc/cron.daily/
/etc/cron.weekly/
```

As their names suggest, executable files can be placed in any of these directories, and will be executed on an hourly, daily, or weekly basis. This saves a bit of time when setting up frequent tasks; just place the executable script or program (or a symbolic link to one stores elsewhere) in the appropriate directory and forget about it.

Chapter 10. Upgrading Linux and Other Applications

To get the most out of your Linux system, such as adding features, getting rid of potential bugs, and ensuring it is reasonably free of security holes, it is a good idea to keep your server –– including the Linux kernel, modules, and user applications –– upgraded. At times it may also be necessary to upgrade hardware components such as a larger hard drive. This chapter will address these issues.

10.1. Using the Red Hat Package Manager (RPM)

The Red Hat distribution of Linux, including kernel, libraries, and applications are provided as RPM files. An RPM file, also known as a "package" is a way of distributing software so that it can be easily installed, upgraded, queried, and deleted. RPM files contain information on the package's name, version, other file dependency information (if applicable), platform (such as Intel or Alpha, etc.), as well as default file install locations.

The RPM utility was first developed by Red Hat and provided as an Open Source product as is common in the Linux community. Other developers picked it up and added extra functionality. The RPM method of packaging files has become popular and is used not only on Red Hat's but on some other distributions as well.

Popular Linux applications are almost always released as RPM files, usually in fairly short order. However, in the Unix world the defacto−standard for package distribution continues to be by way of so−called "tarballs". Tarballs are simply files that are readable with the ``tar'' utility. Installing from tar is usually significantly more tedious than using RPM. So why would people choose to do so? Unfortunately, sometimes it takes a few weeks for developers to get the latest version of a package converted to RPM (many developers first release them as tarballs).

If you start installing or upgrading your system or applications with tar, your RPM database will become out−of−date and inconsistent. This isn't really a big deal (when I used Slackware, I used tar exclusively –– there was no other choice –– without too much discomfort), but wherever possible I try to be patient and wait until an RPM becomes available, or perhaps send a polite request to the developer of the package. (You can also build your own RPM files and distribute them to others, which is sometimes helpful to developers who don't have the ability or time to produce such files themselves.)

A really good place to check if a piece of software is available in RPM form is the RPM repository at http://rufus.w3.org/linux/RPM/. The repository provides indexed categories which can be helpful to locate a given RPM file, and contains pointers to thousands of such files.

To query a package, use ``rpm −q pkg−name'' (eg. ``rpm −q pine''). RPM will either tell you what version of the package is already installed, or that the package is not installed.

Assuming the package is installed already, and is an earlier version than the update package you downloaded (which it should be), then you should be able to apply the update with ``rpm −Uvh pkg−name''. If all goes well, the package will be automatically installed and immediately ready for use. If not, RPM will give you a pretty good reason (for example, perhaps a supporting package needs to be upgraded first). This may require a bit of thinking, but problems such as these are very straightforward to figure out.

If, on the other hand, the package is *not* yet installed, and you decide you wish to install it, type ``rpm -ivh pkg-name''. If there are any supporting packages that are required, RPM will tell you.

Sometimes, you will want to install a package that is only available in source format. In fact, unless you are installing packages from a trusted source (such as the Red Hat FTP site), you probably *should* install from source in case the binaries contain a trojan horse or other nasty thing (of course, a source RPM could also contain such a thing, but they are unlikely to because they would probably be exposed in short order by another developer).

The way to install a package from source is to specify the ``rebuild'' switch to the RPM utility. For example:

```
rpm -ivh --rebuild foo.src.rpm
```

The above command would configure and compile the ``foo'' package, producing a binary RPM file in the ``/usr/src/redhat/RPMS/i386/'' directory (assuming you are using Linux on the Intel platform). You can then install the package as you normally would.

Finally, if you are having problems getting a source package to compile (perhaps you need to modify a makefile, or change a configuration option, etc.) you can use the following steps (again, illustrating our ficticious ``foo'' package example) to compile the source, build a new binary package, and then install from the binary package:

```
rpm -ivh foo.src.rpm
cd /usr/src/redhat/SPECS
pico -w foo.spec
```

Make whatever changes you feel are needed to the ``.spec'' file, and then type:

```
rpm -ba foo.spec
```

This will rebuild the package using whatever changes you have made to the ``.spec'' file. As above, the resultant binary RPM file will be located in ``/usr/src/redhat/RPMS/i386/'', and can be installed as you normally would.

You should look at the Red Hat documentation for more information on RPM. It is an extremely powerful tool that is worth learning in finer detail. The best source of information on RPM is ``Maximum RPM'', which is available in both book form, as well as in postscript format at http://www.rpm.org/maximum-rpm.ps.gz. (If you decide to print the postscript document, be advised that you'll need a *lot* of paper to do so!) There is a smaller guide, the ``RPM-HOWTO'', at http://www.rpm.org/support/RPM-HOWTO.html available as well.

10.2. Installing or Upgrading Without RPM

Sometimes, you may find it necessary to install or upgrade an application for which an RPM package is not available. Of course, it is certainly possible to do such a thing (in fact, it is the "defacto-standard" way of doing things in the so-called "real" Unix world), but I would recommend against it unless absolutely necessary (for reasons why, see Section 10.1).

Should you need to install anything from tarballs, the general rule of thumb for system−wide software installations is to place things in your ``/usr/local/'' filesystem. Therefore, source tarballs would be untarred in ``/usr/local/src/'', while resultant binaries would probably be installed in ``/usr/local/bin'', with their configuration files in ``/usr/local/etc/''. Following such a scheme will make the administration of your system a bit easier (although, not as easy as on an RPM−only system).

Finally, end−users who wish to install software from tarballs for their own private use will probably do so under their own home directory.

After downloading the tarball from your trusted software archive site, change to the appropriate top−level directory and untar the archive by typing commands (as root, if necessary) as in the following example:

```
tar zxvpf cardgame.tar.gz
```

The above command will extract all files from the example ``cardgame.tar.gz'' compressed archive. The ``z'' option tells tar that the archive is compressed with gzip (so omit this option if your tarball is not compressed); the ``x'' option tells tar to extract all files from the archive. The ``v'' option is for verbose, listing all filenames to the display as they are extracted. The ``p'' option maintains the original and permissions the files had as the archive was created. Finally, the ``f'' option tells tar that the very next argument is the file name. Don't forget that options to tar are cAsE−sEnSiTiVe.

Caution
Caution: As mentioned in Section 8.2.1, I recommend first using the ``*t*'' option to display the archive contents to verify the contents prior to actually extracting the files. Doing so may help avoid extracting files to unintended locations, or even worse, inadvertently overwriting existing files.

Once the tarball has been installed into the appropriate directory, you will almost certainly find a ``README'' or a ``INSTALL'' file included with the newly installed files, with further instructions on how to prepare the software package for use. Likely, you will need to enter commands similar to the following example:

```
./configure
make
make install
```

The above commands would configure the software to ensure your system has the necessary functionality and libraries to successfully compile the package, compile all source files into executable binaries, and then install the binaries and any supporting files into the appropriate locations. The actual procedures you will need to follow may, of course, vary between various software packages, so you should read any included documentation thoroughly.

Again, unless it is absolutely necessary, I really recommend avoiding tarballs and sticking to RPM if you can.

10.3. Strategies for Keeping an Up−to−date System

From time to time you may hear of significant upgrades to the Linux kernel or user applications from various sources. These sources may be magazines, newsgroups, web pages, etc.

Probably the best single online resource that a Linux administrator should −− nay, *must* −− keep an eye on is the http://freshmeat.net/ web site. This site contains descriptions of new Open Source applications and projects, documentation, and other announcements of interest to the Linux community.

Another resource for keeping track of new applications announcements is through the comp.os.linux.announce newsgroup. This newsgroup contains postings of new applications, some kernel or application upgrades, web pages, etc. available for Linux. It is a moderated newsgroup and therefore has a high "signal to noise" ratio.

Not all product upgrade announcements are made to comp.os.linux.announce, however. Therefore, visiting the web pages or FTP sites for the applications you are using is probably a very good idea as well.

10.4. Linux Kernel Upgrades

From time to time it may be wise to upgrade your Linux kernel. This will allow you to keep up with the new features and bug fixes as they become available. Or, perhaps, you are running Linux on new or specialty hardware, or wish to enable certain features for which a custom kernel is needed.

This section will describe upgrading and customizing a new kernel. It isn't as difficult as you might think!

Announcements of new kernel versions can be obtained through various sources, including the comp.os.linux.announce newsgroup, as well as on the http://freshmeat.net/ and http://slashdot.org/ web sites.

Please note that there are currently two "streams" of kernel development −− one stream is considered "stable" releases, while the other stream is considered "development" releases. For mission critical applications such as an Internet server, it is highly recommended that you use the stable releases and stay away from the development kernels.

The difference between the two streams is that, with the development kernels, new as−yet untested hardware drivers, filesystems, and other "cutting edge" developments are introduced on a regular basis. These kernels are for use by hackers only −− people who don't mind having to reboot their system, should a kernel bug rear its ugly head.

The stable kernels introduce new features and drivers only after they have been thoroughly tested. Minor releases in this stream also serve to clean up any remaining bugs that are found and corrected.

The two streams use version numbers which are numbered differently to help distinguish between them. The stable kernels are numbered with the second number even (eg. 2.0.35, 2.0.36, 2.2.4) while the development kernels are numbered with the second number odd (eg. 2.1.120, 2.1.121, 2.3.0).

The latest stable kernel is always made available in source as well as pre−compiled binary formats on the ftp://ftp.redhat.com/redhat/updates/ FTP site. Download the desired kernel packages for your version and platform (for example, you would want to navigate to the ``/6.1/i386/'' directory and download the ``kernel−*.i386.rpm'' files for the 6.1 version on the Intel platform).

> **Note:** Note: You do not need to download the kernel sources file unless you are planning on building a custom kernel yourself (see Section 10.6 for details on building a custom kernel).

Sometimes, you may find it necessary to use a kernel that has not yet been made available as an RPM. In this case, you can find the latest kernels from the ftp://ftp.kernel.org FTP site, in the /pub/linux/kernel/ directory. Change to the appropriate major version subdirectory (eg. ``v2.0''), which contains all kernel releases up to the most current one. Download the desired kernel package (for example, the compressed tarball for version 2.0.36 would be called ``linux−2.0.36.tar.gz'' for the Intel platform) and untar it in the ``/usr/src'' directory.

> **Note:** Note: Most user−installed applications not installed from RPM should be untarred under the ``/usr/local/src/'' directory by convention, but this is a kernel tree so we'll make an exception in this case. :−)

Please be aware that if you decide to upgrade your kernel by downloading a tarball, you will most certainly need to configure, compile, and install it yourself. Unless you have special needs that require the very latest development kernel, I *strongly* recommend you upgrade your kernel through Red Hat−provided RPM files −− these are preconfigured and precompiled for you, although you can compile a custom kernel from RPM files as well should you wish.

10.5. Upgrading a Red Hat Stock Kernel

By far the easiest way of upgrading your kernel is to do so using a stock kernel RPM as provided by Red Hat. These RPM files contain pre−compiled binary kernel code, with support for a large variety of hardware and popular features.

Installing a stock kernel is easy to do and involves little risk. Simply type, as root, the following sequence of commands:

```
rpm −Uvh kernel−2.0.36.i386.rpm
cd /boot
ls
```

Make note of the new kernel name, as reported by the ``ls'' command above. You are interested in the ``vmlinuz'' file; for example the third RPM release of kernel 2.0.36 would look like ``vmlinuz−2.0.36−3''.

Now, use an editor to edit the LILO configuation file (type: ``pico −w /etc/lilo.conf'') and change the ``image=/boot/...'' line to point to the new kernel file. After you have done so, type ``/sbin/lilo''. If LILO reports an error message, double−check the file name in your ``lilo.conf'' file with the file name in the ``/boot/'' directory.

Caution
Caution: Do not forget this step!

(The above commands assume you are using the Intel platform and use LILO to boot your system. See Section 4.8 for details on the LILO boot loader).

After you have upgraded your stock kernel and have updated your boot loader information, you should be able to shutdown and reboot using the new kernel (see Section 6.7 for details on shutting down your system).

10.6. Building a Custom Kernel

If you are running Linux on a system with hardware or wish to use features not supported in the stock kernels, or perhaps you wish to reduce the kernel memory footprint to make better use of your system memory, you may find it necessary to build your own custom kernel.

Upgrading the kernel involves configuring desired modules, compiling the kernel and modules, and finally installing the kernel image. This is followed by a system reboot (with fingers crossed!) to load the new kernel. All of this is documented in the ``README'' file which comes with each kernel package. Further information can be found in the ``Documentation/'' subdirectory. A particularly helpful file there is ``Configure.help'' which contains detailed information on the available kernel compile options and modules.

The following is a sample session demonstrating the build of a custom kernel, version 2.0.36 on the Intel platform. While building a custom kernel is usually just a matter of configuring, compiling & installing, sometimes (usually in the case of new hardware) it is necessary to download additional driver software should your hardware not yet be supported by the kernel version you are compiling.

The first step in building a custom kernel is to download and install the kernel sources from either RPM (preferred) or from tarball. See Section 10.4 for details on obtaining the appropriate files.

Next, use the ``rpm'' utility (or ``tar'', as appropriate) to install the kernel source tree and header files. For example, to install the 2.0.36–3 kernel RPM files:

```
rpm -Uvh kernel-source-2.0.36-3.i386.rpm kernel-headers-2.0.36-3.i386.rpm
rpm -Uvh kernel-ibcs-2.0.36-3.i386.rpm
```

(If you are running Linux on a notebook, you would also likely install the ``kernel-pcmcia-cs-2.0.36-3.i386.rpm'' file, which provides power management features.)

After installing the kernel files, you should be able to find the new source tree in the ``/usr/src/linux/'' directory.

The next step is to download any additional driver files (if applicable) and install them in the new kernel source tree. For example, to add support for the Mylex DAC960 hardware RAID controller, I would download the driver software from the http://www.dandelion.com/ web site. Unfortunately, such driver software are usually only offered as tarballs and need to be installed using the ``tar'' utility. For example:

```
cd /usr/src/
tar zxvpf DAC960-2.0.0-Beta4.tar.gz
```

You should read the documentation provided with your additional driver software, if applicable. For example, the DAC960 driver includes a ``README'' file which gives instructions on where the newly downloaded files should be located, and how to apply the kernel patch:

```
mv README.DAC960 DAC960.[ch] /usr/src/linux/drivers/block
patch -p0 < DAC960.patch
```

The next step is to ensure your system's symbolic file links are consistent with the new kernel tree. Actually, this step only needs to be done once, so the following needs to be done only if you haven't compiled a custom kernel before:

```
mail:/usr/src# cd /usr/include
mail:/usr/include# rm -rf asm linux scsi
mail:/usr/include# ln -s /usr/src/linux/include/asm-i386 asm
mail:/usr/include# ln -s /usr/src/linux/include/linux linux
mail:/usr/include# ln -s /usr/src/linux/include/scsi scsi
```

Note: Note: The above step is no longer necessary for 2.2.x or higher kernel versions.

The next step is to configure your kernel settings. This is the most important step in building the custom kernel. If you disable the wrong settings, you may leave out support for features or hardware you need. However, if you *enable* the wrong settings, you will be needlessly enlarging the kernel and wasting your valuable system memory (that being said, it is probably better to err on the side of the latter rather than the former).

The best way of ensuring you compile the kernel properly is to know what features you will need to use, and what hardware is in your system that you will require support for. After you have gained experience in customizing your kernel a few times, the process will become "old hat" and won't seem so intimidating!

Type the following to begin the configuration process:

```
mail:/usr/include# cd /usr/src/linux
mail:/usr/src/linux# make mrproper
mail:/usr/src/linux# make menuconfig
```

(You could type ``make xconfig'' instead of ``make menuconfig'' if you have the X Window System running; see Chapter 5 for details on how to get X working.)

To configure your kernel, go through the various settings and select (enable) whichever ones you require, and de−select (disable) the ones you do not require. You can choose between having such support built right into the kernel, or having it built as a module which is loaded and unloaded by the kernel as needed. (If you compile a feature that is actually needed to boot your system, such as a SCSI driver, as a module, you will need to create a RAMdisk image or your system will not boot. This is done with the ``mkinitrd'' command; this procedure is described a little further down.)

When going through the configuration settings, you can select <Help> for a description of what a given kernel option is for.

After you have configured your kernel settings, type the following commands to compile your kernel:

```
mail:/usr/src/linux# make dep ; make clean
mail:/usr/src/linux# make bzImage
mail:/usr/src/linux# make modules
```

If you are *recompiling* the same kernel as you have previously (2.0.36−3 in this example), you will likely want to move the existing modules to a backup directory as with the following command:

```
mail:/usr/src/linux# mv /lib/modules/2.0.36-3 /lib/modules/2.0.36-3-backup
```

Now, type the following command to actually install the new modules:

```
mail:/usr/src/linux# make modules_install
```

The next step is to copy the kernel into the ``/boot/'' directory and use LILO to update the boot record so that the new kernel is recognized. The following commands will make a backup copy of your existing kernel, copy the new kernel over, and then refresh the LILO boot record:

```
mail:/usr/src/linux# cd /boot
mail:/boot# cp vmlinuz vmlinuz.OLD
mail:/boot# cp /usr/src/linux/arch/i386/boot/bzImage vmlinuz-2.0.36
mail:/boot# /sbin/lilo
```

Finally, you will need to edit your ``/etc/lilo.conf'' file, and make sure the "image" reference is pointing to the new kernel. You should also add a section which points to your backup kernel, called, perhaps, "OldLinux". Here is an example file:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz
        label=Linux
        root=/dev/hdb1
        read-only
image=/boot/vmlinuz.OLD
        label=OldLinux
        read-only
```

By adding your backup kernel information in this way, should your new kernel fail to boot properly (perhaps a device is not recognized, or a daemon doesn't start as it should), you can simply type ``OldLinux'' to boot from the old kernel and investigate the problem.

> **Note:** Note: As mentioned previously, if you've compiled a feature required to boot your system as a module, you will need to create an initial RAMdisk image in order to boot your system. (Don't forget to compile your kernel with support for such an initial boot image.)

The procedure to create and use an initial RAMdisk image is as follows:

- Add an entry in your ``/etc/lilo.conf'' to boot off the initial RAMdisk image; this is shown as an addition to the example configuration file shown earlier:
  ```
  image=/boot/vmlinuz
          label=Linux
          root=/dev/hdb1
          initrd=/boot/initrd-2.2.4-4.img
          read-only
  ```
- The loopback device needs to be loaded before you are able to use the mkinitrd command. Make sure the loopback device module is loaded:
  ```
  /sbin/insmod loop
  ```

  (If you get an error message about not being able to load the loopback module, you may need to specify the full path to the module for the *current* kernel your system is still running on, for example ``/lib/modules/2.0.35/loop''.)

- Use the ``mkinitrd'' command to actually create the image:
  ```
  /sbin/mkinitrd /boot/initrd-2.0.36-3.img 2.0.36-3
  ```
- Run ``/sbin/lilo'' to update your boot loader.

Now, shut down your system and boot the new kernel!

```
mail:/boot# /sbin/shutdown -r now
```

If your kernel refuses to boot altogether, don't panic. Boot off the boot disk that was created during the installation of Linux . If you don't have copies of this disks, you should be able to create one from the Red Hat CD. Insert the boot diskette into the drive and reboot the computer. When you see the "boot:" prompt, type:

```
mount root=/dev/hda1
```

The above command assumes your "/" (root) partition is located on /dev/hda1.

Linux should then boot normally (although since you are using the kernel from the boot disk, not all services or devices may operate properly for this session), and then you can restore your old kernel and reinstall the LILO boot loader information (ie. ``mv /vmlinuz.old /vmlinuz ; /sbin/lilo") and shutdown/restart. You can then try recompiling the kernel with different options and try again.

10.7. Moving to the Linux 2.2.x Kernels

The Linux kernel 2.2.0 was released on January 25, 1999, bringing with it many new features, performance enhancements, and hardware support. Any existing Linux system can be upgraded with one of these new kernels in much the same fashion as described in Section 10.4 (with caveats).

This section will describe how to upgrade your Red Hat system to the new kernels. As Red Hat 6.0 (and above) already ships with the new kernel and supporting packages by default, this section will only be useful to those of you who are still using an earlier version, such as 5.2. I will likely remove this section from future versions of this document, once I believe a majority of users have migrated to 6.0 and beyond.

Warning
Warning! If you decide to upgrade your older system to support the new kernels, be advised that as the process involves a number of package upgrades, it is possible that something will go horribly wrong. As always, have recent backups available to you in case something goes wrong. If you don't have experience with upgrading files with RPM as well as compiling kernels, perhaps you might wish to upgrade to Red Hat 6.1.

You have the choice of upgrading to either a stock kernel as provided by Red Hat, or upgrading by compiling a custom kernel. I would recommend getting things going with a stock kernel first, and then build a customized kernel later as you normally would (see Section 10.5 for details.)

In order to use the latest kernel, it is first necessary to upgrade to the newest utilities and libraries. Red Hat has identified which packages need to be upgraded to support the newest kernel, and have placed the appropriate RPM files on their FTP site at ftp://ftp.redhat.com/redhat/updates/5.2/kernel−2.2/i386/ (for Red Hat 5.2 users on the i386 platform).

A very good web page, detailing the appropriate system tools that are necessary for moving to 2.2.x is available at http://www−stu.calvin.edu/~clug/users/jnieho38/goto22.html; I will attempt to summarize the information below (items marked with a leading ``**" indicate you will most likely need to upgrade the item for Red Hat 5.2; items not indicated as such are *probably* okay but probably worth checking).

- ** *initscripts−3.78−2.4 or better* (Type ``rpm -q initscripts'' to check your version)
- ** *modutils−2.1.121 or better* (Type ``rpm -q modutils'' to check your version)
- ** *mount−2.9−0 or better* (Type ``rpm -q mount'' to check your version)
- *gcc−2.7.2.3 or better* (``rpm -q gcc'')
- *binutils−2.8.1.0.23 or better* (``rpm -q binutils'')
- *libc−5.4.46 or better* (Red Hat uses the newer ``*glibc*''. Not needed.)
- *glibc−2.0.7−6 or better* (``rpm -q glibc'')
- *ld.so 1.9.9 or better* (``ls -l /lib/ld.so.*'')
- *libg++−2.7.2.8 or better* (``rpm -q libg++'')
- *procps−1.2.9 or better* (``rpm -q procps'')
- ** *procinfo−15 or better* (``rpm -q procinfo'')
- *psmisc−17 or better* (``rpm -q psmisc'')
- ** *net−tools−1.50 or better* (``rpm -q net-tools'')
- *loadlin−1.6 or better* (Needed only if you are booting Linux from DOS using Loadlin. Not sure how to calculate the version number; download the latest version to be sure.)
- *sh−utils−1.16 or better* (``rpm -q sh-utils'')
- *autofs−3.1.1 or better* (``rpm -q autofs'')
- *nfs−server2.2beta37 or better* (``rpm -q nfs-server''; needed only if you are serving NFS file shares.)
- *bash−1.14.7 or better* (``rpm -q bash'')
- *ncpfs−2.2.0 or better* (``rpm -q ncpfs''; needed only if you are mounting Novell file systems.)
- *kernel−pcmcia−cs−3.0.6 or better* (``rpm -q kernel-pcmcia-cs''; needed only for laptops which need PCMCIA card support.)
- *ppp−2.3.5 or better* (``rpm -q ppp''; needed only if you are connecting to the Internet with a modem and PPP.)
- *dhcpcd−1.3.16−0 or better* (``rpm -q dhcpcd''; needed only if you need a DHCP client to connect to the Internet, such as with a cable modem).
- ** *util−linux−2.9.0* (``rpm -q util-linux'')
- *setserial−2.1 or better* (``rpm -q setserial'')
- *ipfwadmin/ipchains* (Only needed if you are doing IP firewalling; see the ``*IPCHAINS−HOWTO*'' guide at http://isunix.it.iltu.edu/resources/ldp/HOWTO/IPCHAINS−HOWTO.html.)

You should download and upgrade any packages using RPM as required (see Section 10.1 for details on how to use RPM).

Caution
Caution: Upgrading to the new ``modutils'' package will result in modules no longer functioning for the older 2.0.x kernels! Therefore, do not upgrade this package until you have installed the new kernel in ``/usr/src/linux''.

After bringing your system's tools up to date, you can install the kernel sources. You can find them on Red Hat's FTP site as well; I recommend downloading the ones provided as updates for Red Hat 6.1, at ftp://ftp.redhat.com/redhat/updates/6.1/i386/. To do so, type the following:

```
rpm -Uvh kernel-source*.rpm kernel-headers*.rpm
```

Now that the new kernel sources have been installed, it should be safe to upgrade your modutils package. However, the new kernel no longer uses the ``kerneld'' module for on−demand loading of kernel modules. Therefore, you should disable this module before updating modutils. To disable kerneld and upgrade the modutils package, type the following as "root":

```
/sbin/chkconfig kerneld off
/etc/rc.d/init.d/kerneld stop
rpm −Uvh modutils*.rpm
```

You should now be able to configure, compile, and install your 2.2 kernel as you normally would (see Section 10.6 for details). You may be surprised to see the dizzying amount of new configuration settings available. Take your time and read the help text for any options you are unfamiliar with!

With any luck, the next time you boot your system you will be running the latest and greatest Linux kernel version!

Much more detailed information on these procedures can be found on Red Hat's web site at http://www.redhat.com/corp/support/docs/kernel−2.2/kernel2.2−upgrade.html.

10.8. Configuring the Apache Web Server

At my place of employment, we are using the Apache package to provide web services. Apache is a full−featured web server with full support for the HTTP 1.1 standard, proxy caching, password authenticated web pages, and many other features. Apache is one of the most popular web servers available (according to a recent site survey done by Netcraft, more than 54% of all web sites on the Internet are using Apache or one of its derivatives), and provides performance equal or better to commercial servers.

(Under construction. :−p)

To keep up with added features and bug−fixes that are made to Apache, it is a probably a good idea to upgrade your server from time to time. The Apache web site is located at http://www.apache.org/ and contains information on the latest versions.

10.9. Configuring the Squid HTTP Caching Proxy Daemon

At my place of employment, we use the Squid package to provide proxy caching of web pages. Squid offers high−performance caching of web clients, and also supports FTP, Gopher, and HTTP requests. In addition, Squid can be hierarchically linked to other Squid−based proxy servers for streamlined caching of pages.

There are two versions of Squid currently available. One, the "regular" version, seems to work well on machines with lots of RAM. The second version, "*SquidNOVM*" is suitable for machines with less RAM (I recommend using this version if you have 64 MB of RAM or less). Basically, the "NOVM" version uses less memory at the expense of more file descriptors. It's the one I use, and it works well.

(Under construction :−p)

To keep up with new features and bug−fixes, it is a probably a good idea to upgrade the Squid server from time to time. More information on Squid can be found on web site at http://squid.nlanr.net/Squid/.

10.10. Configuring the Sendmail E−mail Daemon

I use the Sendmail package to provide e−mail services. Sendmail is *the* definitive mail handler; in fact it is so popular that it is estimated that over 80% of e−mail passing over the Internet will be handled at one or both ends by it. It does just about anything and I couldn't imagine running an Internet server without it (another e−mail server package called Qmail seems to be quite popular as well −− but I haven't had a reason yet to give it a try).

To keep up with new features and bug−fixes, and most importantly, for reasons of security, it is a probably a good idea to upgrade Sendmail from time to time. In addition, the very latest versions of Sendmail include powerful anti−spam features which can help prevent your mail server being abused by unauthorized users.

This section will discuss some of the things you should do if you wish to use Sendmail as an incoming e−mail server. This would be the likely scenario for server systems. If, instead, you have no need to use it for incoming mail and wish to only use it as an outgoing mail queue, you should ((need some info here)).

For this section, it is assumed that you are using the very latest version of Sendmail (8.9.3 at the time of this writing), have it installed and running.

As packaged with the Red Hat distribution, Sendmail usually contains appropriate configuration information to operate correctly in the majority of server setups. Nonetheless, you may find it necessary to edit the ``/etc/sendmail.cf'' file and customize some settings as required. This, however, is beyond the scope of this document.

One thing I find helpful, however, is to make a couple of changes to the configuration file to thwart off spammers. These include:

```
O PrivacyOptions=authwarnings
change to:
O PrivacyOptions=authwarnings,noexpn,novrfy

O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
change to:
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b  NO UCE  C=xx L=xx
```

(The first change prevents spammers from using the ``EXPN'' and ``VRFY'' commands in sendmail. I find that these commands are too often abused by unethical individuals. The second change modifies the banner which Sendmail displays upon receiving a connection. You should replace the ``*xx*'' in the ``C=xx L=xx'' entries with your country and location codes. For example, in my case, I would use ``C=CA L=ON'' for Ontario, Canada. (The latter change doesn't actually affect anything, but was recommended by folks in the news.admin.net−abuse.email newsgroup as a legal precaution.

Next, if your mail server will have a different host name than the actual machine it is running on, you can add one or more aliases in the ``/etc/sendmail.cw'' file. For example, if you have a system called "*kirk.mydomain.name*" which is set up as the mail exchanger for mydomain.name, but want incoming mail addressed in the format ``*user@mydomain.name*'' to be delivered to your users on "*kirk*", simply add this alias as follows:

```
mydomain.name
```

Finally, If you need to restrict a domain (or subdomain) from connecting to your sendmail service, you can edit the ``/etc/mail/access'' and add the domain information as well as type of restriction. For example:

```
some.domain             REJECT
hax0r.another.domain    550 Contact site administrator at (555) 555-1234.
```

The above examples would reject all e−mail connections from the ``*some.domain*'' site, as well as reject the specific machine name ``*hax0r.another.domain*'' with a descriptive message.

After making changes to this file, you will need to update the ``access.db'' file, and then restart sendmail as follows:

```
/usr/sbin/makemap hash /etc/mail/access.db < /etc/mail/access
/etc/rc.d/init.d/sendmail restart
```

> **Tip:** Tip: If you are concerned with e−mail abuse, you can get some very helpful information from the "Mail Abuse Prevention System" (MAPS) project on dealing with such abuse; see the web pages at http://www.mail−abuse.org/
>
> If you're using Sendmail version 8.9 or above, RBL support is already built in, but not enabled by default. To enable this support, add the following to your sendmail.mc file:
>
> ```
>         FEATURE(rbl)
> ```
>
> Then, reconfigure and restart the Sendmail daemon.
>
> For more detailed information, including configuration instructions for other mail transport agents, see http://www.mail−abuse.org/rbl/usage.html.
>
> Sometimes, a domain may end up in the RBL list with which you wish to continue communications with. Perhaps it is vital for you to communicate with certain users at the black−listed domain. In this case, Sendmail allows you to override these domains to allow their e−mail to be received. Simply edit the ``/etc/mail/access'' file in the manner described above with the appropriate domain information. For example:
>
> ```
> blacklisted.domain      OK
> ```
>
> Don't forget to rebuild your access.db file (described above)!
>
> If you do decide to subscribe to the RBL, it is probably a wise idea to inform your mail users, if applicable, so they can make other service arrangements if they disagree with your decision.

For more information on Sendmail, see the FAQ document located at http://www.sendmail.org/faq/.

Chapter 11. Enterprise Computing with Linux

As Linux has earned a solid reputation for its stability and reliability, it is being used for more mission critical applications in the corporate and scientific world.

This chapter will discuss issues which are most relevant to those using Linux in the enterprise, such as tuning your server for better performance under higher loads, keeping your data safe with RAID technologies, as well as discuss the general procedures to migrate across servers.

11.1. Performance Tuning

(Under construction. :−p)

11.2. High Availability with RAID

As storage needs increase, it sometimes becomes necessary to put additional drives with larger capacities online. Yet ironically, the law of probability dictates that as the number of storage devices increases, so too does the likelihood of a device failure. Therefore, a system with a single hard drive is only 25% as likely to suffer a hardware failure as a system with four drives. [Well, theoretically speaking, anyway :−)]

Fortunately, such failures can be handled gracefully, and more importantly without downtime, using a technique called "Redundant Array of Inexpensive Disks" (*RAID*) which uses one of several methods of distributing data over multiple disks. This redundancy allows for automatic recovery of data should a device fail.

This section will describe the installation, configuration, and setup of a RAID disk array using the Mylex AcceleRAID DAC960 controller. I have been very impressed with not only the performance and reliability of the controller itself, but also with the technical support I've gotten from Mylex −− they are very Linux−friendly! (However, there are a wide variety of hardware RAID solutions for Linux, and RAID can be implemented in software by the Linux kernel itself.) The type of RAID implementation that is most useful is probably RAID level 5.

The first step in getting the RAID controller usable under Linux is to build a custom kernel with driver support for the hardware. The driver for the Mylex DAC960 can be downloaded from the Dandelion Digital Linux page at http://www.dandelion.com/Linux/DAC960−2.0.tar.gz.

The final step in getting your RAID array usable under Linux is to use the ``fdisk'' utility to create valid partitions. This is done in exactly the same manner as you would use on an IDE or regular SCSI drive. See Section 4.3 for details on how to set up partition information.

> **Note:** Note: The DAC960 driver supports a maximum of 7 partitions per logical drive. If you need to define more, you will need to define multiple logical drives in the RAID configuration utility (press <**Alt**>−<**R**> at system boot time to enter the setup utility).

Once you are able to see your RAID array, you should initialize any swap areas and file systems you wish to define. The following is an example of initializing a swap area on the third partition of the second drive, as well as an ext2−formatted file system on the first partition of the first drive:

```
/sbin/mkswap -c /dev/rd/c0d1p3
/sbin/swapon /dev/rd/c0d1p3
/sbin/mkfs.ext2 -c /dev/rd/c0d0p1
```

> **Note:** Note: The ``-c'' option in the above ``mkswap'' and ``mkfs.ext2'' commands enable bad−block checking as the appropriate swap/file systems are created. This adds *substantially* to the time it takes to complete the process, but it is probably a very good idea to perform such checks.

For any new swap areas you have defined, you should make an entry in the ``/etc/fstab'' file to ensure the swap area is actually used from subsequent bootups. As per the above example, the following line should be added:

```
/dev/rd/c0d1p3   swap       swap  defaults   0 0
```

Finally, once your file systems have been initialized, you can create mount points there and move your large file systems onto the array as you desire. It is probably a good idea to test the array for a few days before using it in a production environment.

For further information on the Mylex AcceleRAID controller, visit the Mylex web site at http://www.mylex.com/ as well as the Dandelion Digital DAC960 driver page at http://www.dandelion.com/Linux/DAC960.html. For further information on RAID in general (including both software− as well as hardware−based solutions), see the Linux High Availability web site at http://linas.org/linux/raid.html.

11.3. Server Migration and Scalability Issues

With support for a diverse selection of hardware, as well as proven speed and reliability, Linux is up to the challenge of scaling up to meet resource demands as they increase. This can include moving to an SMP (*Symmetric Multi Processing*) configuration for greater processing needs, RAID levels 0 through 5 (either in software or hardware driven modes), etc.

On occasion, you may feel that your Linux server has outgrown the hardware it is running on, perform a major Linux version upgrade, or perhaps move to a different distribution of Linux. There are, of course, two ways of doing this. Either you will be leaving your server on existing or upgraded hardware (in which case you need simply shut down services, back up your data, perform the required modifications, and then restore data if needed), or in the more radical case, migrate your server to new hardware.

This section will concentrate more on the latter situation, where you will be actually migrating your various services from the old server to a new one. There are, of course, several migration strategies, however this section will attempt to provide some rough guidelines which you can follow in order to ensure your migration effort succeeds with minimal disruption to your users.

- Prepare your new server as necessary; install and configure Linux so that your new hardware devices are supported, and any required daemons and kernel−based features (such as firewalling) are enabled. See Chapter 4, as well as Section 10.6 for details.
- Set up your existing services (such as the Apache web server, Samba or Netatalk file & print services, etc.) and make use of them with test data for at least several days to ensure everything is working as desired. See Section 7.4, as well as Section 7.5 for details. Don't forget to ensure that any changes or custom scripts you have made in the ``/etc/'' directory, including anything in

``/etc/rc.d/'' have also been done on the new server as required. It is especially important that you remember to move over your user account information in the ``/etc/passwd'', ``/etc/group'', and, if you are using shadow passwords, ``/etc/shadow''!

- Shut down services on your old server, so that your file systems will see a minimal amount of file update activity. Obviously you don't want users uploading web pages and receiving e−mail on the old server, while you are restoring the data onto the new one! As root, you can shut down most services with the following command:

```
killall httpd atalkd smbd nmbd squid sendmail ftpd
```

  The above command will shut down the web server, file & print services, e−mail server, and FTP service. (You may be running less or more services than the ones I have listed above. Check your process list and terminate any other service you feel appropriate; see Section 9.2 for details.)

  You might also want to edit the ``/etc/inetd.conf'' file on your old server, and with the ``#'' character, comment out any services (such as FTP, IMAP, and POP3 services) which might result in file system updates. Then, again as root, type:

```
killall −HUP inetd
```

  The above command will reload the TCP wrappers (security wrappers to Internet services) so that future connections to any services you have disabled in the ``/etc/inet.conf'' file will not be loaded).

- Now you should be able to move over the data from one system to another. Likely, you will have prepared your new server to have everything it needs to function, including any additional software that you wish to install that did not come with your Red Hat distribution. Therefore, you will likely need to backup any data stored in ``/home'', ``/var/spool'', as well as optional file systems, such as ``/archive'', if applicable. Here is an example command that uses the ``tar'' utility to make a compressed backup file of data:

```
cd /
tar zcvpf /tmp/backup_data.tar.gz --exclude=var/spool/squid \
          home archive var/spool
```

  The above command will write a backup of your ``/archive'', ``/home'', and ``/var/spool'' file systems (or subdirectories, depending on how you have set up your system), to a file called ``/tmp/backup_data.tar.gz'' in compressed tar format. Make sure you have enough space to create the backup, or write it elsewhere!

  > **Tip:** Tip: You can use the ``du'' utility to help determine required space. For example, to determine the requirements of the ``/archive/'' and ``/home/'' directory trees, type:
  >
  > ```
  > du −h −s /archive /home
  > ```
  >
  > Bear in mind that the above command will report the actual size of your data, but if you are using tar's ``z'' option (as above) to compress the image file, your usage requirements will likely be significantly less. Consider the output from the ``du'' command a worst−case estimate of the space required.

- Now, you can restore the backup data from the tar file onto the new server. You can restore it directly over NFS (see Section 7.6 for details on how to configure NFS), or simply use FTP to transfer it over and untar it locally. Here is an example that will restore the files that were backed up as above:

```
cd /
tar zxvpf /tmp/backup_data.tar.gz
```

- Next, if necessary, swap your IP addresses so that your new server is seen on the old address.
- Finally, you may wish to shutdown and restart your server to ensure there are no unexpected error messages that appear. See Section 6.7 for details.

Once you are done, make sure everything is working as expected! If not, you can always re–enable any services you disabled on the old server and restart them so that users can continue using it until you resolve the problems on the new one (bear in mind, however, that you'll need to repeat the above steps again if you choose to do that).

Chapter 12. Strategies for Keeping a Secure Server

Linux can certainly be considered to be as secure –– or more secure –– than operating systems from other vendors. Admittedly, with Linux becoming more and more popular, it is becoming a very attractive target for crackers to concentrate their break–in efforts on. There are exploits that are discovered from time to time, however the open nature of Linux usually means that such exploits are patched quickly, and security announcements are disseminated widely, containing either temporary workarounds or pointers to updated software.

I won't pretend to be an expert on security issues, however I am at least aware of these issues, which I believe to be a large part of the battle towards making one's systems as secure as possible. Although being aware and diligent in keeping up with security updates will in no way guarantee that a system's security measures won't be circumvented, the likelihood of a break–in is greatly reduced.

Although there have been security exploits found in external services which could have been used by crackers to break into a system (for example, the IMAP daemon exploit), I believe that it is far more likely that a determined cracker will penetrate the system from *within*. Compared to the handful of services communicating with the outside world, there are *thousands* of commands and utilities available from the shell, one or more of which may contain bugs which can be exploited to penetrate security (that being said, I must admit to recently discovering one of the servers I maintain had been compromised through an external service).

For this reason, I recommend avoiding giving out shell accounts to users unless they are absolutely necessary. Even if you consider your users completely trustworthy and have no qualms in providing them with access to the shell, all it takes is just one of these users to have a weak password. An outside cracker, finding its way into your system by exploiting this weak password, will then be able to work at his or her leisure internally, looking for further weaknesses.

There are, fortunately, things you can do to greatly increase the security of your Linux system. While a detailed discussion of security issues is beyond the scope of this document, the following checklist provides some of the most important things you should do to enhance security:

- Upgrade system tools, applications, and kernel: By far the most common cause of system break–ins is by not exercising diligence in keeping an up–to–date server. Performing regular upgrades of the system kernel, tools and utilities will ensure that your system is not filled with older items for which known exploits are available. For details on keeping an up–to–date server, see Section 4.9, as well as Section 10.3.
- Shadow passwords: You should definitely be using Shadow passwords; switching to this password format is *easy*! For details, see Section 6.6.
- Smart password management: Make sure passwords, *especially* for users you are providing with shell access, are strong and changed often. Also, if you use multiple servers, resist the temptation to use the same password for all of them (otherwise, if a cracker breaks into one server using a discovered password, he or she can break into them all).
- Use secure shell (ssh): Switch to using ``ssh'' instead of ``telnet''. Telnet is insecure for two reasons: One, sessions are unencrypted, which means everything, including username and passwords, are transmitted as clear text. Second, an open telnet port is one of the first places a cracker will try to connect to.

  Ssh provides encrypted and compressed connections and provide substantially more security than telnet connections. You can run a ssh server (which allows incoming secure connections) as well as a client (for outgoing secure connections) under Linux. You can find binary RPM packages at ftp://ftp.replay.com/pub/replay/redhat/i386/. You will need the following files (newer versions may be available by the time you read this):

- ssh−1.2.27−5i.i386.rpm The base package.
- ssh−clients−1.2.27−5i.i386.rpm Clients for outgoing connections.
- ssh−extras−1.2.27−5i.i386.rpm Some handy perl−based scripts.
- ssh−server−1.2.27−5i.i386.rpm Server for incoming connections.

Note: Note: The SSH RPM files listed above are the international versions. If you reside in the U.S. or Canada, you can choose to download the U.S. packages (which may have stronger encryption algorithms); these packages have a ``us'' instead of an ``i'' suffix after their version numbers. Under U.S. law, it is *illegal* to export strong crypto products outside of the U.S. or Canada. Hopefully one day the morons in the U.S. Department of Justice will finally see the light, and remove this silly restriction (Red Hat doesn't include SSH with their distribution because of this very reason, and we *all* suffer).

Should your Windows users be up−in−arms about no longer being able to connect to your system, they will be happy to know that several free ssh clients for Windows are available:

"TeraTerm Pro" client software

http://hp.vector.co.jp/authors/VA002416/teraterm.html

"TTSSH" client software

http://www.zip.com.au/~roca/download.html

"Cryptlib" client software

http://www.doc.ic.ac.uk/~ci2/ssh

"Putty" client software

http://www.chiark.greenend.org.uk/~sgtatham/putty.html

Note: Note: If you do decide to switch to using ssh, make sure you install and use it on *all* your servers. Having five secure servers and one insecure one is a waste of time, *especially* if you are foolish enough to use the same password for more than one server.

- Restrict access to external services: Next, you should edit the ``/etc/hosts.allow'' as well as the ``/etc/hosts.deny'' file to restrict access to services to external hosts. Here is an example of how to restrict telnet and ftp access. First, the ``/etc/hosts.allow'' file:

```
# hosts.allow

in.telnetd: 123.12.41., 126.27.18., .mydomain.name, .another.name
in.ftpd: 123.12.41., 126.27.18., .mydomain.name, .another.name
```

The above would allow any hosts in the IP class C's 123.12.41.* and 126.27.18.*, as well as any host within the mydomain.name and another.name domains to make telnet and ftp connections.

Next, the ``/etc/hosts.deny'' file:

```
# hosts.deny
in.telnetd: ALL
in.ftpd: ALL
```

- Turn off and uninstall unneeded services: Edit your ``/etc/inetd.conf'' file, and disable (ie. comment out using a ``#'' character) any services that are not needed (if you're using ssh as recommended above, you might wish to disable the ``telnet'' service). After you have done so, as root type ``/etc/rc.d/init.d/inet restart'' to restart the inetd daemon with the changes.
- Install a security detection system: Consider installing security programs such as ``Tripwire'' (see http://www.tripwiresecurity.com/) which can detect intrusions, and ``Abacus Sentry'' (see http://www.psionic.com/abacus/) which can help prevent them.
- Due diligence: Keeping your eye on your system, performing random security audits (which can be as simple as checking for suspicious entries in the password files, examining your process list, and checking your log files for suspicious entries) can go a long way towards keeping a secure system. In addition, report any break–in attempts to the appropriate authorities –– it may be a hassle to do this, particularly if your system sees several of these attacks in a given week, but such reports ensures that would–be crackers are deterred by threat of punishment, as well as ensuring that others' systems (which may themselves have been compromised) are kept secure.
- Assuming you install and upgrade system tools and applications using the ``RPM'' utility, you may wish to verify the integrity of your installed packages by auditing them with the following command:

```
rpm --verify -a > /tmp/rpm-audit.txt
```

The above command will check your system's RPM database with all relevant files, and indicate any files which have been modified, by displaying a '5'. Here is some example output of such an audit:

```
S.5....T   /bin/ls
S.5....T   /usr/bin/du
......G.   /dev/tty5
.....U..   /dev/vcs5
.....U..   /dev/vcsa5
S.5....T c /etc/lynx.cfg
S.5....T c /etc/sendmail.cf
```

In the sample output above, you can see a list of seven files, four of which have been modified. Now, obviously there are going to be several, perhaps many, files which have been modified if you have customized your system at all. A brief check of the /etc/lynx.cfg and /etc/sendmail.cf files, perhaps visually or perhaps from backup, might reveal legitimate configuration changes that you have made to your system.

However, notice in the sample above, two of the modified files are *binary executable* files? It is likely that these two binaries, the ``ls'' command as well as the ``du'' command, are actually trojan binaries which a system cracker has installed to perform some nefarious purposes (a ``diff'' command performed on any modified binaries with those restored from backup or RPM might reveal significant size or other differences; further evidence of trojans.)

(For more information on ``RPM'', see Section 10.1.)

For more information on security–related issues, an excellent resource entitled, "Securing RedHat 5.x" document is available at http://redhat–security.ens.utulsa.edu/. An excellent resource for Linux crypto and related software is at http://replay.com/redhat/.

Chapter 13. Help! Trouble in Paradise!

Linux is earning a reputation world−wide for its performance and reliability. Nevertheless, no system is perfect, and from time to time you are bound to hit a snag. Fortunately, with uptimes measuring in the months (compared to those measuring in the days or weeks as with NT), such snags will likely be few and far between.

13.1. Getting Linux Installed on new, Unsupported Hardware

(Under construction :−p)

13.2. File System Corruption after Power Outage or System Crash

Although Linux is a stable operating system, should it happen to crash unexpectantly (perhaps due to a kernel bug, or perhaps due to a power outage), your file system(s) will not have been unmounted and therefore will be automatically checked for errors when Linux is restarted.

Most of the time, any file system problems are minor ones caused by file buffers not being written to the disk, such as deleted inodes still marked in use. In the majority of cases, the file system check will be able to detect and repair such anomolies automatically, and upon completion the Linux boot process will continue normally.

Should a file system problem be more severe (such problems tend to be caused by faulty hardware such as a bad hard drive or memory chip; something to keep in mind should file system corruption happen frequently), the file system check may not be able to repair the problem automatically. This is usually, but not always, the case when the root file system itself is corrupted. In this case, the Red Hat boot process will display an error message and drop you into a shell, allowing you to attempt file system repairs manually.

As the recovery shell unmounts all file systems, and then mounts the root file system "read−only", you will be able to perform full file system checks using the appropriate utilities. Likely you will be able to run e2fsck on the corrupted file system(s) which should hopefully resolve all the problems found.

After you have (hopefully) repaired any file system problems, simply exit the shell to have Linux reboot the system and attempt a subsequent restart.

Naturally, to be prepared for situations such as a non−recoverable file system problem, you should have one or more of the following things available to you:

- The boot/root emergency disk set, *AND/OR*
- The LILO emergency boot disk, *AND*
- A recent backup copy of your important files −− just in case!

13.3. Where to Turn for Help

As Linux is developed by members of the Internet community, the best place to get help is probably by posting a message to any of the following newsgroups:

Miscellaneous postings not covered by other groups

comp.os.linux.misc

Networking−related issues under Linux

comp.os.linux.networking

Security−related issues under Linux

comp.os.linux.security

Linux installation & system administration

comp.os.linux.setup

Everybody is entitled to their opinion :−p

alt.linux.sux

For non Linux−specific topics, there are a variety of groups in the comp.* heirarchy that may suit your needs. Here are just a few of them:

Cisco router/access−server line of products

comp.dcom.sys.cisco

Miscellaneous web server questions

comp.infosystems.www.servers.misc

General unix (not Linux−specific) questions

comp.os.unix

The SMB protocol (WfW/95/NT−style file/print services)

comp.protocols.smb

There are also several resources on the Web that may be useful. Do a web search for "Linux", or visit any of the following:

Linux Resources

http://www.linuxresources.com/

The Linux Documentation Project

http://metalab.unc.edu/LDP/

The RPM repository

http://rufus.w3.org/linux/RPM/

The Linux Software Map

http://www.boutell.com/lsm

Linux Applications & Utilities Guide

http://www.xnet.com/~blatura/linapps.shtml

LinuxHardware.net: Hardware Driver Support

http://www.linuxhardware.net/

Linux User Support Team

http://www.ch4549.org/lust

The Linux v2 Information Headquarters

http://www.linuxhq.com/

The Samba Home Page (WfW/95/NT−style file/print services)

http://samba.anu.edu.au/samba/

The Apache Web Server

http://www.apache.org/

The Squid HTTP Proxy Caching Daemon

http://squid.nlanr.net/Squid/

There are a myriad of mailing lists that may prove helpful in providing answers to your questions as well. These can usually be found through a simple web search (for example, searching for *``linux raid mailing list''* might help you find mailing lists devoted to RAID issues under Linux). Here are some I recommend; to subscribe to any of these lists, simply send an e−mail message to the subscription address listed with the word "*subscribe*" in the body of your message:

Red Hat Mailing Lists

Description of available Red Hat lists: http://www.redhat.com/

GNOME Mailing Lists

Description of available GNOME lists: http://www.gnome.org/mailing−lists/index.shtml

KDE Mailing Lists

Description of available KDE lists: http://www.kde.org/contact.html

Linux SCSI Mailing List

Subscription address: linux−scsi−request@vger.rutgers.edu

Linux RAID Mailing List

Subscription address: linux−raid−request@vger.rutgers.edu

Finally, you may be interested in checking out the following two sites, both of which are my personal "daily must read" favorites. SlashDot covers the latest technology news in general with a definite Linux slant, while FreshMeat provides an up−to−date listing of Open Source applications announcements.

SlashDot: News For Nerds

http://slashdot.org/

FreshMeat: Open Source Applications Announcements

http://freshmeat.net/

13.4. Pointers to Additional Documentation

There is an incredible amount of documentation available for Linux and its applications. Most of this can be found on the web and in your local bookstore, but you will probably find that a large quantity of useful documentation is already available to you, having been loaded as part of the Red Hat Linux installation procedure.

The man pages are a must−view when you are trying to figure out how a command works. For example, if you are trying to figure out how to use the ``tar'' utility, you could type ``man tar'' and be provided with a very verbose description of tar including all of its command−line options.

You can find more general information in the ``/usr/doc/'' directory. Here you will find subdirectories which include documentation on utilities and commands, Frequently Asked Questions (FAQ) documents, as well as HOWTO documents providing good instruction on a variety of topics, such as how to set up networking, or install support for the Japanese language.

You should also look in the ``/usr/info/'' directory which contains tutorials on utilities, libraries, and applications such as emacs.

Finally, you should visit the Red Hat User's Frequently Asked Questions (FAQ) document at http://www.pobox.com/~aturner/RedHat−FAQ/ which contains a lot of helpful information specific to the Red Hat distribution of Linux.