UNIT 10.
Installation and Hardware Configuration

Structure

10.0 Introduction

This unit will detail the procedures needed to install Red Hat linux onto an Intel system; the procedures are similar whether you choose to install using either GUI- or text-based installation. Updation and installation of necessary package and mutiboot concepts are also discussed .

10.1 Objectives

At the end of this unit, You would be able to

- Undestand the concept of Installing LINUX operating system
- Know creation of Bootable Diskette
- Explain how Partitioning can be done on Hard Drive
- Install multiple operating systems on a single Hard Disk
- Upgrade the packages

10.2 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.

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 installation diskette that you can use to install your new Red Hat Linux system. Alternative methods of creating boot diskette to use the command mk bootdisk, using diskcopy command of windows or dos and also another way can be to make boot disk at the end of linux installation

10.3 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 (“US/UK 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, make sure you have an entry in your /etc/exports file allowing access by the new system to the appropriate directory, 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.

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`".

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*.

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

10.4 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 the sytems /home partition had problems, you could simply unmount it, perform a file system check, and then remount the repaired file system.

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 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.

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.

You should partition a disk (or disks) according to your needs. 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 60% of available (eg. 1200 Mb)
/home about 20% of available     (eg. 400 Mb)
/usr any remaining space

/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.

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: 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. 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.

10.4.1 Using the Partitioning Program: Fdisk

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

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

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

When you're in installation, it will mention cylinders. The number of cylinders is proportional to the capacity of your hard drive. In other words, if you have a 2.5 gig hard drive that has 620 cylinders, 310 cylinders is equal to somewhere between 1.2 and 1.3

gigs. To find out how many bytes a cylinder is on your hard drive, divide the number of bytes there are (2,500,000,000 in my case) by the number of cylinders (620). The result for me would be about 4,032,258 bytes, or around 4 megabytes a cylinder.

10.4.2 Setting up Swap Space

Once you've set up your partition information, and have assigned “mount points” (ie. / for kernel or root, /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.

10.4.3 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.

10.5 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*”.

If you are an experienced Linux user, What usually you can do is to select the components that you know, for which enable the “`Select individual packages`” option, which allows 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.

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 ten and thirty minutes to complete, depending on the speed of your system.

10.6 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. Also you can use *setup* command for X window configuration.
- 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

10.7 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.

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.

10.7.1 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.

10.7.2 Dual Booting: Linux and Windows with LiLo

Well, if you're here, it's because you're not sure how to get your new linux box to cohabitate with windows. The first thing that you need to understand is how linux looks at disk partitions.
/dev/hda is the first hard disk (Primary IDE Master). The first partition on this disk is /dev/hda1 the second is /dev/hda2 and so on.
/dev/hdb is the second hard disk (Primary IDE Slave). This disk is partitioned the same as hda was, /dev/hdb1 /dev/hdb2 etc...
Your secondary master and slave are /dev/hdc and /dev/hdd, respectively. This is just for your information, as you probably won't be booting from these devices. /dev/hdc will normally be your CDROM drive.

So lets assume that we have a system with one hard disk, and linux is installed on the first partition (/dev/hda1) and windows is installed on the second partition (/dev/hda2). This is a rather simplistic way of doing it, you may have more than one linux partition, but we're going to keep it simple.

Here is a sample /etc/lilo.conf that goes along with the assumption that we made in the previous line:

```
#/etc/lilo.conf
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
```

```
default=linux

# Linux Partition
image=/boot/vmlinuz-2.2.12-20
  label=linux
  initrd=/boot/initrd-2.2.12-20.img
  root=/dev/hda1
  read-only

# Windows Partition
  other=/dev/hda2
  label=win
  table=/dev/hda
```

Once you have edited lilo.conf to suit you, you need to run '/sbin/lilo' and it looks like this:

```
[root@panic krnl]# /sbin/lilo
Added linux *
Added win
```

The asterisk indicates that the linux label is the default boot image and that after the timeout expires, your machine will automatically boot into linux.

10.8 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.

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. 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. :-)

10.8.1 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.

10.8.2 TAR's

.tar files are a type of compressed file in the Unix/Linux OS's. This is much like a .zip file in Windows. They are a way to get the actual source code of a program, compile the source and install it on your machine. This is afterall the biggest advantage to Linux, being able see exactly what is happening in program execution and tailor it to you individual needs.

Now for installing .tar files. First thing you need to do is go to the console and change directories to where you downloaded the file. For example, if you downloaded the file '**newfile.tar**' and saved it to **/downloads**, in the console you will type **"cd /downloads"** then hit '**enter**'. Now you are in the dir that the file is stored in. Next, you need to unpack or 'untar' the file. So, type **'tar -xvf newfile.tar'**(if the file is a .tar.gz do a **tar -zxvf newfile.tar.gz**). This will create a dir in the /downloads directory called 'newfile'. Next change directories to the 'newfile' directory by typing **'cd newfile'**. (Notice I didnt include the / in front of the dir name. This is because downloads (which is the current directory) is the parent directory for the 'newfile' directory. If you wanted to change to a directory not in the current directory path, you would have to include the /.) In the 'newfile' directory, you will have a number of **README** files. You will be able to identify README files because they will be in all caps. (IE. INSTALL, README, CONFIG, etc.) These files will tell you exactly how to configure and install this package. For the most part, once you are in the Newfile directory, you can type '**make**', then '**make install**' followed by '**make clean**' to install the program. Be careful though, some programs require you to change settings before you install the program.

10.8.3 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.

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).

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.

.rpm files are a breeze! You don't have to compile anything and for some newbies, that is what you want. So shut-up and tell me how to install them you say? OK. First **cd downloads**(assume you downloaded the file to a dir called downloads). Once in that directory, type **rpm -ihf newfile.rpm**. And bam, your finished! Simple huh. Even you idiots can handle that! Now RPM has more commands that are good. The **rpm -e newfile.rpm** will uninstall the complete program if needed. Yep, that's right...no need to search for all the files it installed to remove individually. Also **rpm -Uhv --force newfile.rpm** . Notice two things here. First I didnt make a typo...I meant to replace *i* with *U*. This is the **UPGRADE** tag. You can use in place of i anytime you want, it will not screw anything up. Also, the **--force** is a command that (but you can't guess) forces the install even if errors exist.

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.

10.8.4 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.

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.

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.

10.8.5 Red Hat Linux Sound Configuration

If you have a recent version of the Red Hat Linux distribution, try to run the **sndconfig** program (as root). It could ask for sound card settings such as the I/O base address, IRQ, DMA, and 16-bit DMA. Be prepared to give them. It also calls **isapnptools** PnP card detection utility.

Try this command for any distribution closely related or, more commonly, based, on Red Hat Linux.

10.9 Summary

This chapter has covered the most important work of linux operating system installation. Any Linux users first ambition shall be to install a partial user friendly operating system in his computer. A word of caution is that you shall have a backup of all important files of your entire hard disk comprising of all partitions.

Here you will get a first glimps of basics of linux installation. By experience only one can become better administrator or installer of Linux OS.

The student has to concentrate for Creation of an Installation Diskette, Partitioning Hard Drive(s), Choosing Desired Packages required to Install, Multi-boot with Other Operating Systems.

Installation Tip:

- Partition your hard drive with DOS's FDISK and make sure you leave room for other partitions.
- Format your hard drive in DOS.
- Insert a boot disk rawritten with boot.img.
- Run the installation.
- Make a primary Linux partition, then extended, and then logical. (The last two are for the swap space.)
- Tag your partitions.
- Reboot and at the " LILO boot: " prompt, type Linux.
- Login as "root" and enter the password that you set.

10.10 Check Your Progress

I. Choose apropriate answer

1. Creation of linux boot disk can be done through
   a. rawrite b. mkbootdisk c. diskcopy d. all of these.
2. The example of a compressed file used for any package installation is
   a. *.tar.gz b. *.rpm c. *.zip d. all of theese.
3. Linuxconf can be used to
   a. mount a drive b. configure a network c. administer users d. all of these
4. LILO is used for
   a. Creation of boot diskette b. Managing linux network
   c. Multiboot d. Linux configuration
5. A Linux system must contain
   a. Swap partition b. Root password c. linux native partition d. all of these
6. Setup command helps to configure
   a. Only X windows b. Mouse c. Sound card d. all of these

II. Say True or False

1. Linear mode option must be used for multi boot option with size of first partition more than 1023 cylinders. True/False
2. Formatting of all the partitions or mount points is a must for linux installation. True/False

3. /dev mount point contains all user files and information. True/False
4. fdisk command is used to create boot diskette. True/False
5. Windows NT operating system can not be installed as dual boot for linux. True/False

III. Essay type Questions

1. Explain how to create an installatin or boot diskette.
2. Illustrate the significance of /etc/exports file with respect to NFS.
3. Elucidate the different types of booting required for linux installation program.
4. What is partitioning of hard disk drive? Why it is important explain.
5. Describe the possible benefits of partitioning of hard disk drive.
6. Give an example of space works of partitions for installing linux and windows operating systems in one hard disk size of 4.3 Gb. You may use 2 Gb for windows 98 operating system.
7. Explain the usage of /, swap, /home, /usr mount points with respect to linux installation.
8. What is LILO? How it is usefull in Linux.
9. How to perform dual Booting for Linux and Windows with LiLo.
10. Explain how the updations of packages can be done in Linux.
11. Brief the function of the command linuxconf.
12. What is a tar file ? Explain its options in detail.
13. What is a rpm utility? Explain its options in detail.
14. Explain the Partitioning utility Fdisk in detail with its different options.

IV. Further readings and other activities

1. Get more information using man or info or any help command for linuxconf, fs, lilo, setup, Xconfigurator, fdisk, sndconfig, mouseconfig, netconf, neconfig,userconf, ipcalc, fdisk, sfdisk.
   EX: 1. $ man sfdisk 2. $ apropos fdisk
2. Install first on a separate hard disk (min 300 MB) Linux OS in text mode only.
3. Install the linux OS including X windows on a separate hard disk (min 1.2 GB).
4. Install the linux OS including X windows on a hard disk (min 4.3 GB) with multi-boot concept along with windows 95/98 also windows NT.
5. Try out creation of Linux boot diskette with different commands.
6. Partition a hard disk with different sizes using any partition command such as diskdruid or fdisk.
7. For further readings you can refer the following books

   1. Title: Linux System Administration Handbook - for detail information on problems of linux installation.
   2. Title: The Complete Reference Linux Foutth Edition
      Author: Richard Petersen
      Publication: Tata McGraw-Hill
   3. Title: Red Hat Linux Starter kit in 24 hours
      Author: Judith Samson etal
      Publication: Sams Techmedia
   4. Title: Unix Shell Programming
      Author: Yashawant Kanitkar
      Publication: BPB
   5. Title: Linux Kernel Internals
      Author: M. Beck etal

Publication: Addison-Wesley, Second Edition
6. Title: Unix Power Tools
   Author: Jerry Peek etal
   Publication: BPB
7. Title: Unix System Programming
   Author: Richard Stevens

Reference e-mails: raomvp@yahoo.com roopasindhe@lycos.com
URL / Web Site: http://www.raomvp.bravepages.com

Good-Luck