Configuring a Red Hat Linux System

For Multiple RAID-1 Devices

Graham Leach

Managing Partner

TIG – The Imperators Group, Inc.

Table Of Contents

Why Do This At All?

Some people would say that enabling multiple RAID-1 devices under Linux is a non-trivial task, some would even describe it as difficult, poorly documented and not worth the trouble. So why bother?

Here are some compelling facts that support the idea that RAID-1 is a good idea.

RAID-1 Is A Good Thing

1) RAID raises the dependability of your system.
2) RAID can raise the performance of your website.

When a Redundant Array of Inexpensive Disks (RAID) is deployed on a system, the information on that system is **duplicated** on two physically different disk drives. In the event of a failure of one disk, the other is used – increasing the availability of information by 100%.

Because RAID devices can be deployed on multiple I/O channels, they can offer performance gains over disk systems that only use a single channel.

In other words, providing RAID may enhance your reputation and certainly helps to make your customers feel more secure when using your platform.

Setting up Multiple RAID-1 Devices in Red Hat 7.2

The Partitioning Strategy

The first thing that has to be determined when you are going to go with RAID is how the RAID is to be implemented. In the example used in this document, RAID is being used to mirror partitions as opposed to entire drives.

The boot and swap Partitions

Every Linux system needs somewhere to keep its boot files. On older drive (less than 540Mb) this used to be a non-important decision. Boot files were placed subdirectory of root called /boot and never thought about again, except in the case of a kernel rebuild.

With the advent of truly large drives the entire question of booting has became a bit trickier. BIOS restrictions cause a certain rule: The boot files must reside within the first 1024 cylinders of the drive. To accommodate the rule and guarantee a boot we must define a separate partition to hold the boot file. To make this easy, we define the **first** partition on the drive to be /boot.

The next partition is also **really** important. Linux systems use **virtual memory**, a system where sections of the RAM of the computer are swapped on and off the disk drive according to which program is currently active. This is happening even during installation, so the next task is to create the **swap partition** – in our case we anticipate physical RAM reaching 256Mb.

The rule of thumb is to define a swap space that is **twice** that of physical RAM, so our swap partition should be 512Mb.

Here is the partitions we will need for our "housekeeping" partitions:

Partition	Device	Size	Type
/boot	dev/hda1	20Mb	Linux
<swap>	dev/hda2	512Mb	swap

The third partition is where all of the actual files that compose the Linux system will reside – aside from the kernel itself. But there's a problem, we want to define the following partitions:

/
/home
/usr
/var

But the partition table can only old a maximum of four (4) entries. Summing up the two "housekeeping" and the "working" paritions gives us a total of six (6) partitions. We don't have enough entries to spare in the partition table to accommodate this scheme!

Fortunately there is a way to get around the four primary partition limit. The **extended** partition was invented for just this reason. Extended partitions contain **logical** partitions inside of them. In this case the **extended** partition is really nothing more than a container for other partitions.

Here is the definition of the extended partition we will need to hold the "working" partitions

Partition	Device	Size	Type
<extended>	dev/hda3	<rest of disk>	extended

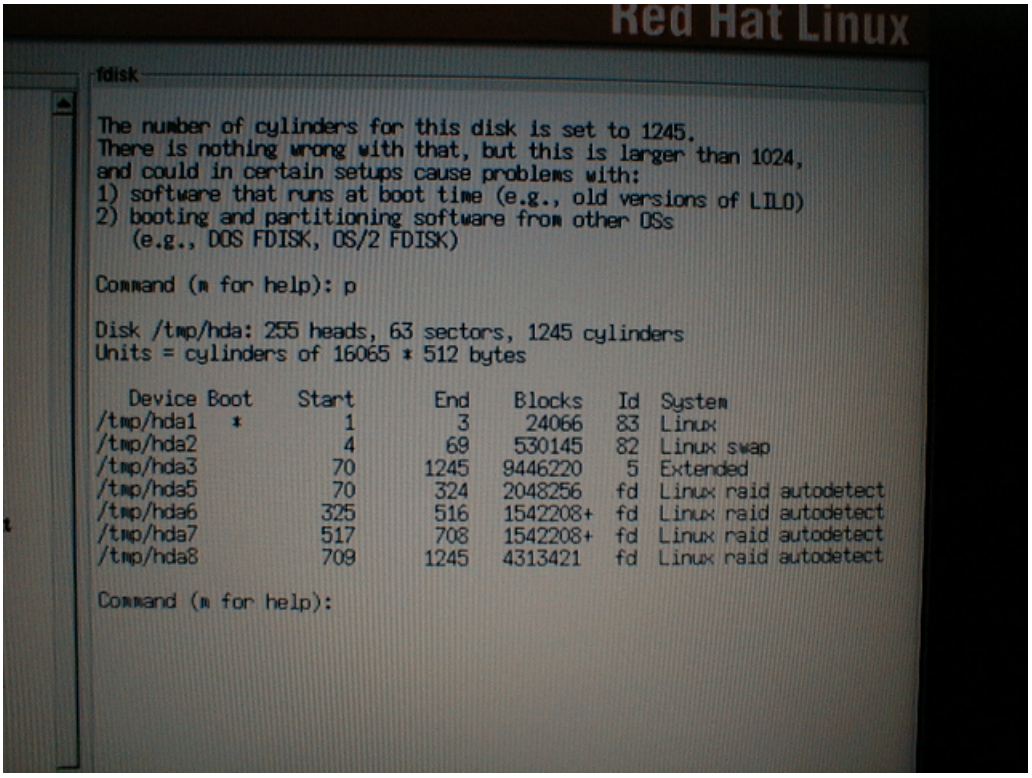
The Mirrored Partitions
Now that the extended partition has been created, **logical** drives within the partition are defined.

In this scheme we are going to partition the RAID-1 partitions according to the following scheme:

Partition	Device	Size	Type
/	/dev/md5	4000Mb	Linux RAID
/home	/dev/md6	16000Mb	Linux RAID
/usr	/dev/md7	3000Mb	Linux RAID
/var	/dev/md8	5000Mb	Linux RAID

What We're Trying To Achieve

Here's a picture of what we should arrive at when this setup process is complete:



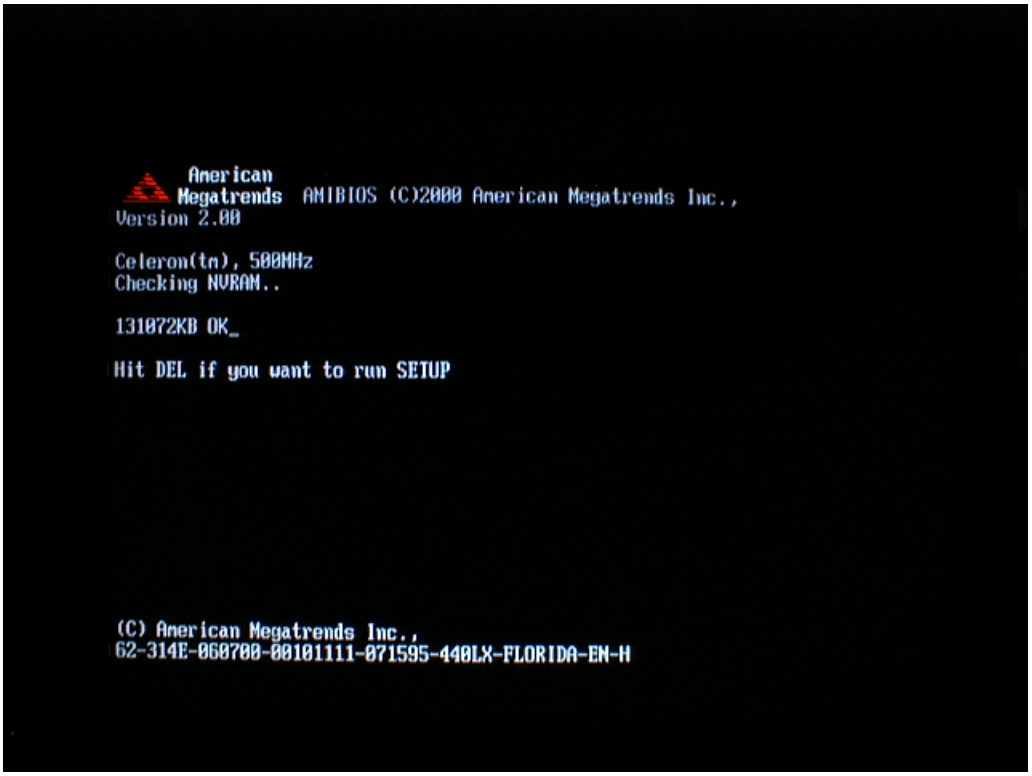
OK, now that we know what we're targeting, let's make it real!

The Distribution

The BIOS Setup

One of the most important steps in making RAID-1 under Linux a success is the proper setup of the hard disks in the BIOS. If they are not set up properly, this entire exercise fails!

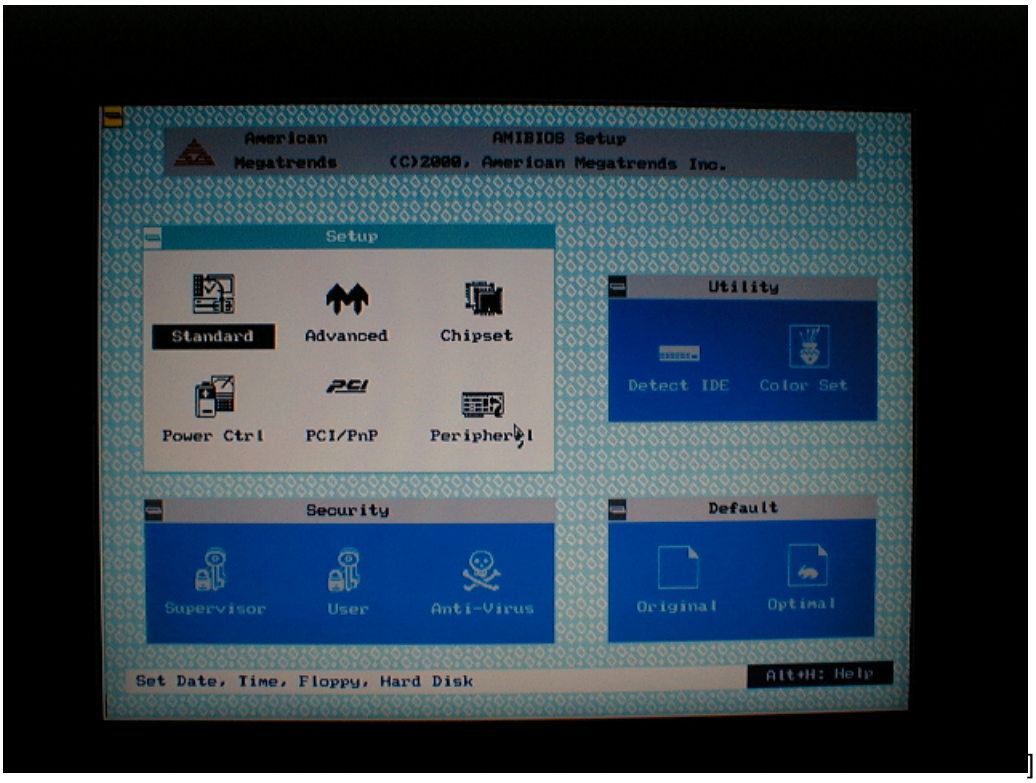
Consider the following:



In the above example the computer is booting and a prompt explains that to enter the BIOS setup the **DEL** key must be pressed.

The BIOS Setup Screen

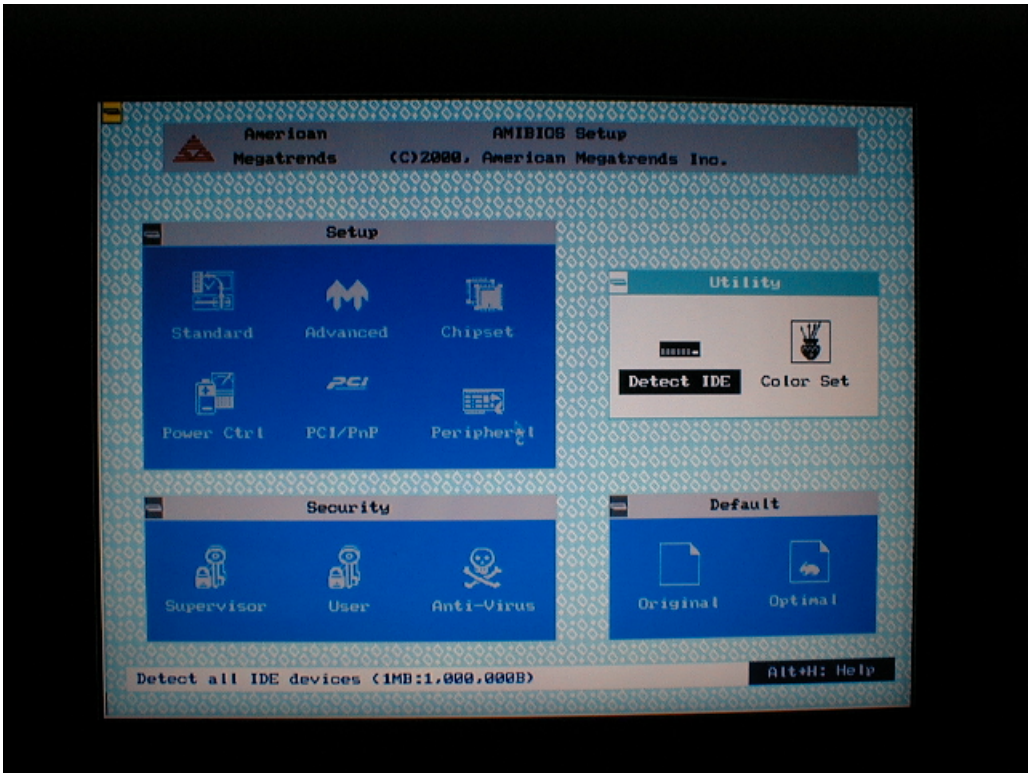
Consider the following:



In the above example, the BIOS has been entered and the initial screen is being displayed. We want to move to the screen to the **right** of this one, so we press <TAB>.

Getting Into the IDE Auto Detect Screen

Consider the following:

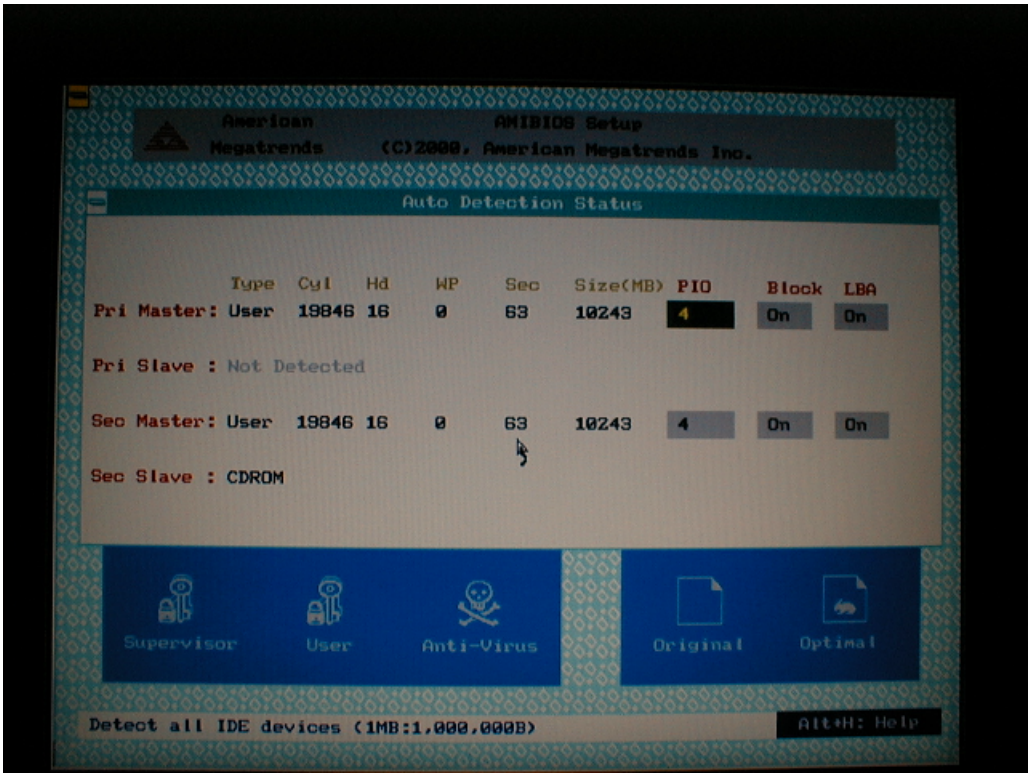


In the above example we are on the upper-right menu and about to invoke the **Detect IDE** utility. This utility will probe the hard drives and obtain their geometry information.

Run the utility by pressing the <ENTER> key.

Checking the results of the IDE Auto Detect Screen

Consider the following:



In the above example the **Detect IDE** utility has been run and the drive geometry for the two drives has been determined and is displayed. Note that each of the drives is **PRIMARY** on their respective IDE bus. I recommend that you make the drives PRIMARY on their own bus and any additional devices (such as CD-ROM) secondary.

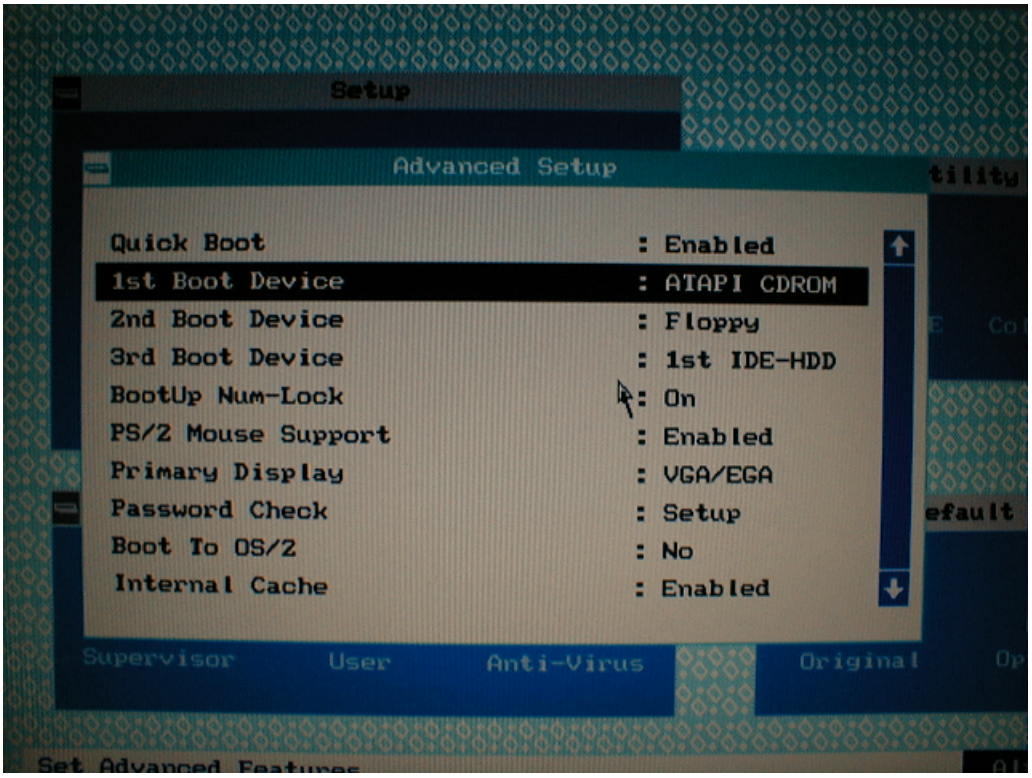
The other thing to verify is the two drives are treated **identically** by the BIOS. Make sure that they have the same translation (LBA or NONE) as well as displaying the same cylinders and heads and so forth. In my first two RAID-1 implementations the translation was AUTO – which caused /dev/had to use one kind of translation and /dev/hdb another – an incredibly frustrating and time-consuming problem that **cannot** be resolved within Linux, no matter what parameters you pass to the kernel on boot!

Do the BIOS setup properly or you will be sorry. You have been warned!

Enable boot from CDROM

In the **advanced** section of most BIOS is the ability to choose the first boot device.

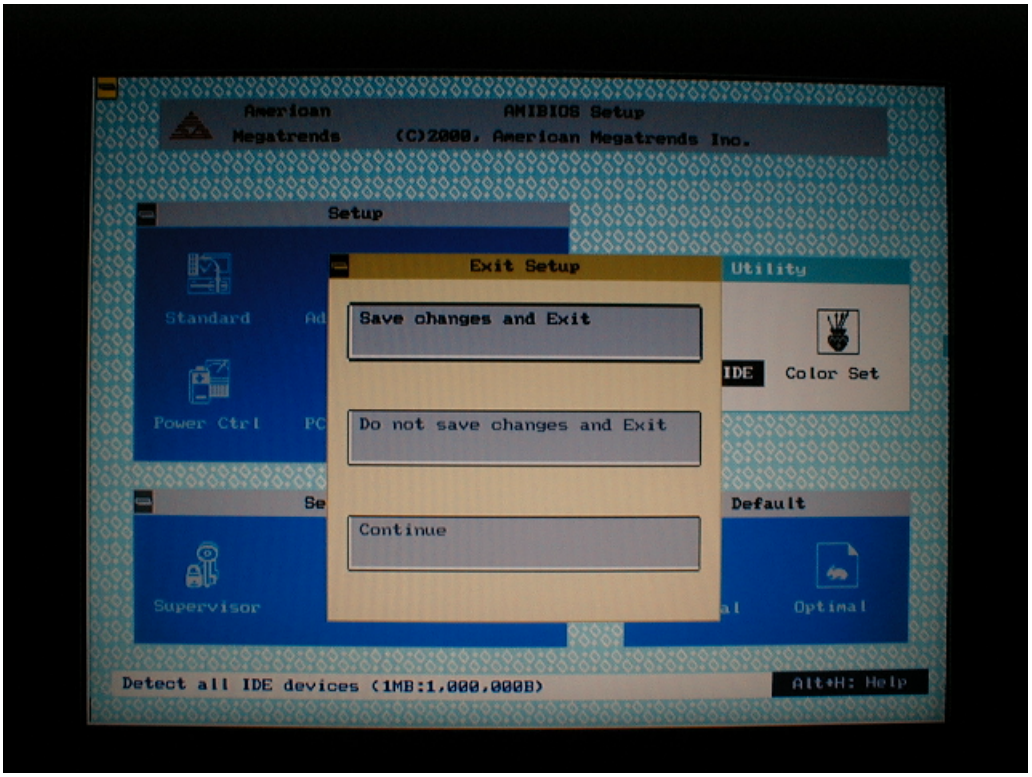
Consider the following:



In the above example the **CD-ROM** has been selected as the first boot device. This will enable the installation of Red Hat from CD.

Saving BIOS Changes

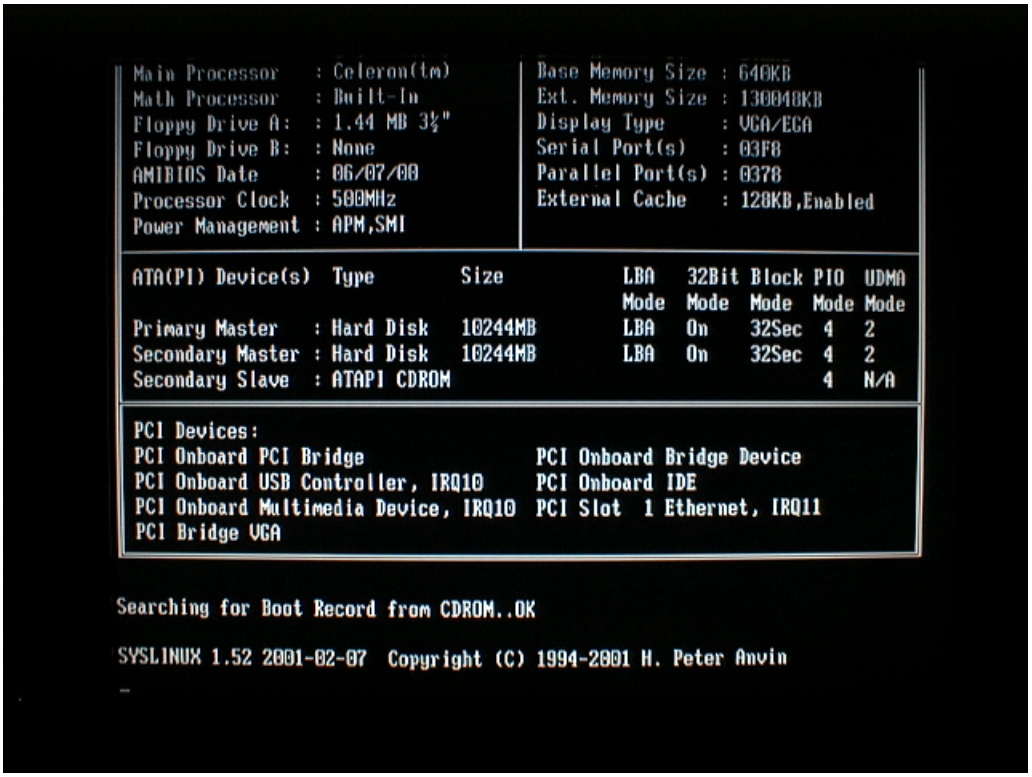
Consider the following:



After the BIOS settings have been verified (and written down) they are saved to NVRAM and the machine rebooted.

Booting From The CD-ROM

Consider the following:

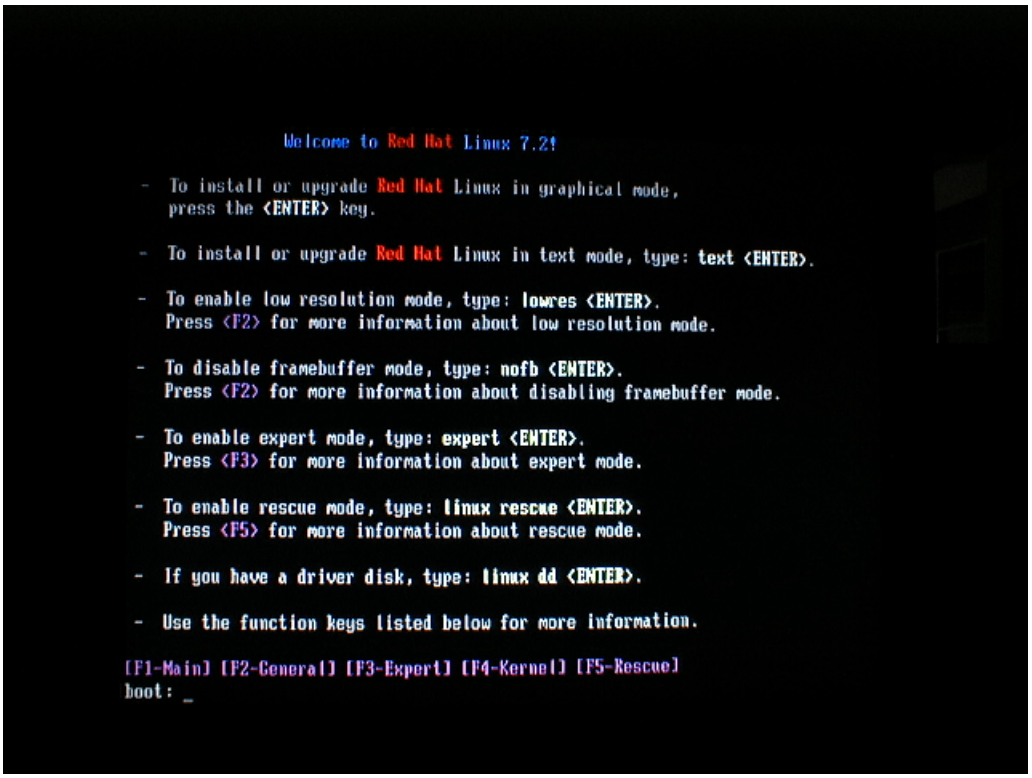


In the above example the machine is displaying its configuration – notice that the drives display the same **size**, **translation**, **mode** – in all respects their logical configuration is **identical**. If you do not have this symmetry your install may not be successful!

Note that the second to last message on the screen indicates that the CD-ROM is being booted, the last message is output from the actual boot loader.

The Red Hat 7.2 Installation Start Screen

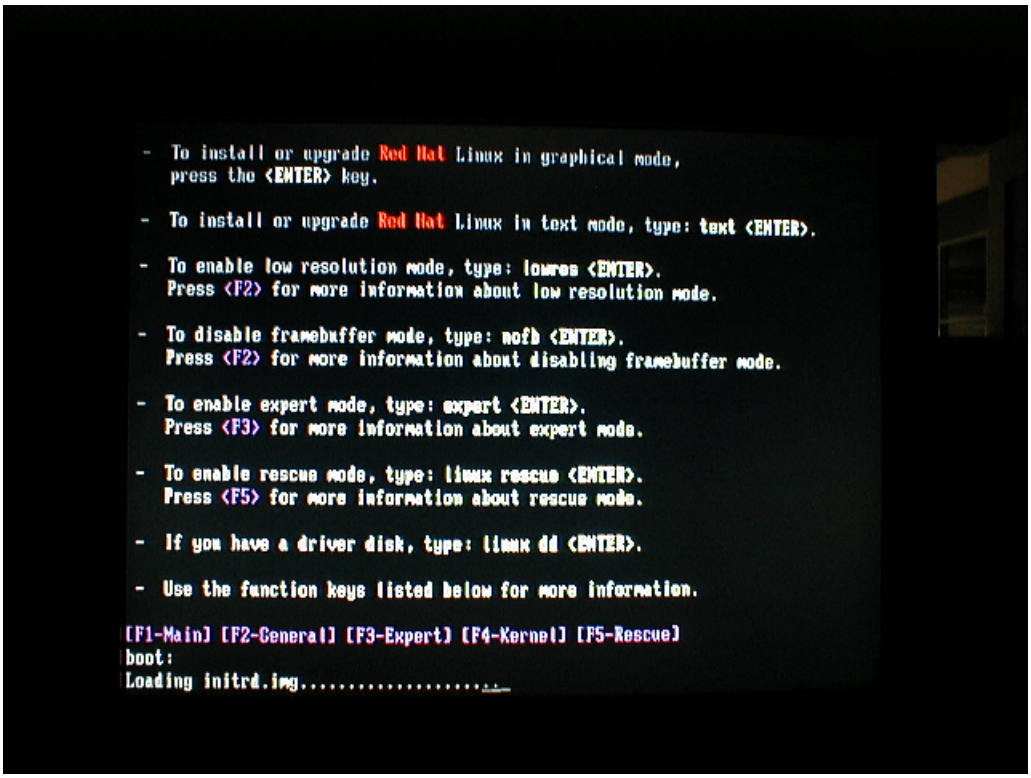
Consider the following:



In the above example the machine has booted from the CD-ROM and the installation is officially underway! The screen offers a variety of installation options, but we're going to select the default one by pressing **<ENTER>**.

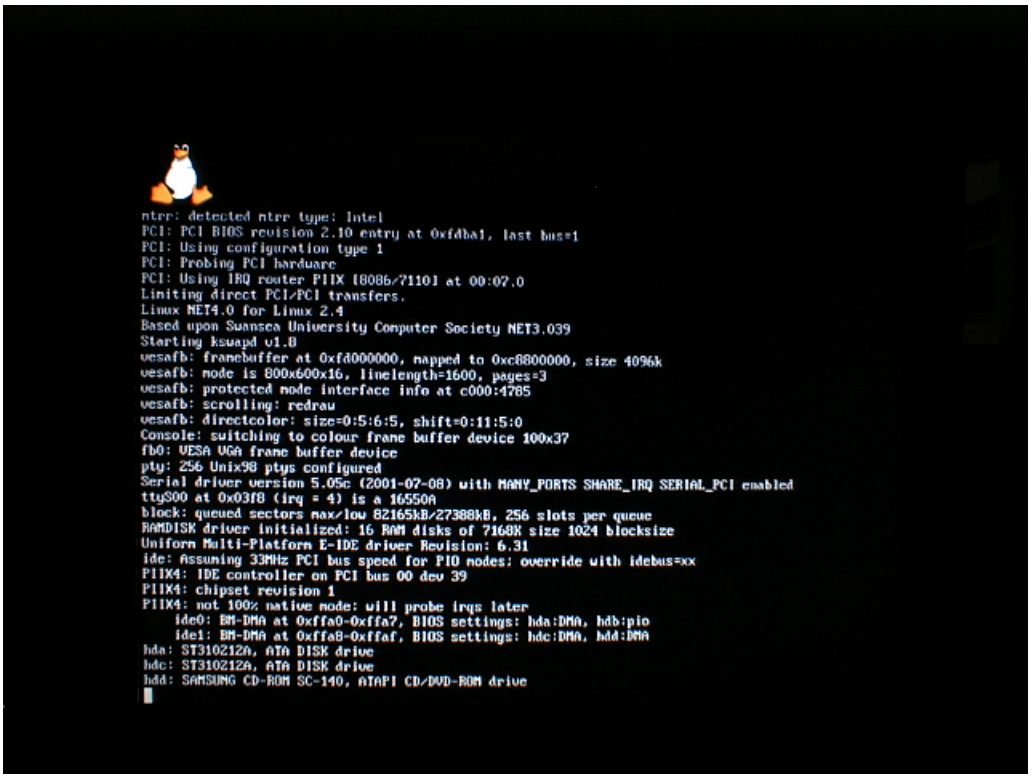
Press **<ENTER>** to continue

Loading the Linux Setup Kernel



Loading The Linux Kernel From CD-ROM

Consider the following:



In the above example the Red Hat kernel is being loaded – a cute little penguin is displayed on the top of the screen and a bunch of information about system capabilities is output to the screen – in the old days you used to be able to read this, but now it zooms by so quickly that a special utility had to be written to view it! The utility is called **dmesg** and it can be invoked from the command line at any time.

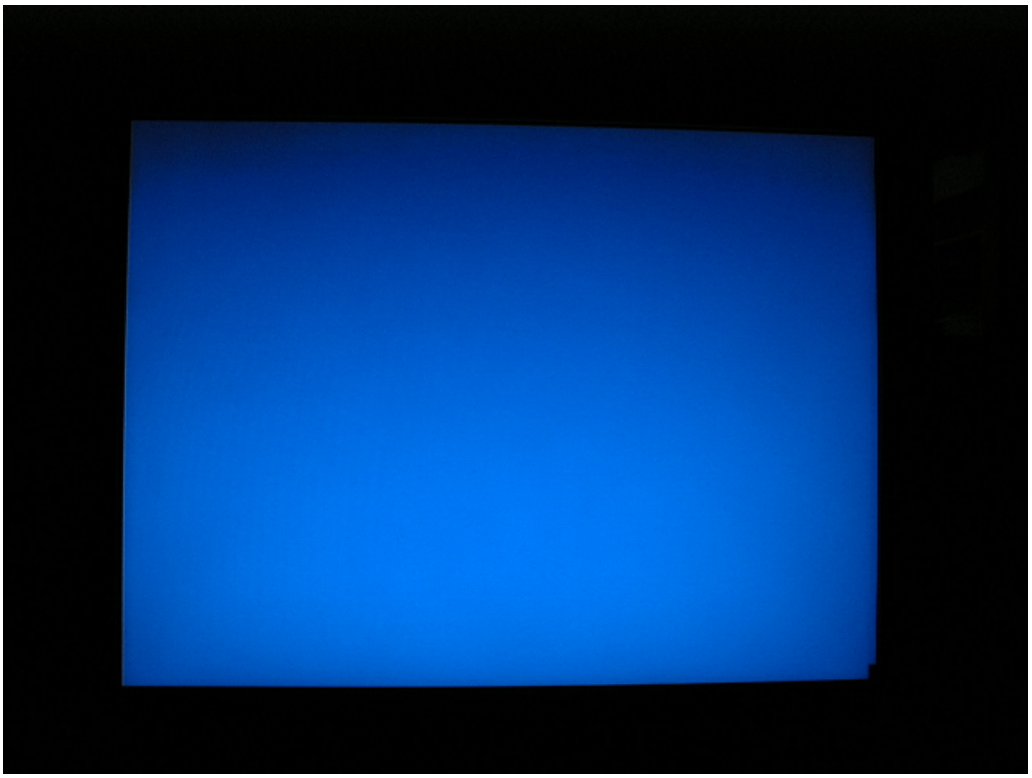
Just sit back and watch the lines fly by – do not be alarmed by anything you see, other than the phrase **kernel panic**. If you see that, there's trouble.

Starting The Anaconda Setup Program, part I

For Microsoft users, the next screen is a bit eerie. Traditionally, a blue screen did not bode well. Fortunately for Red Hat users, this just means that the initial device load is complete and the kernel is now resident in memory. The next thing the install program does after loading the kernel is to load the **anaconda** setup program.

The program is quite large and takes some time to load off of the (comparatively slow) CD-ROM. It is also probing the hardware to determine (as best it can) the monitor, mouse and keyboard attached to the computer

Consider the following:



In the above example the initial kernel load is complete and the **anaconda** program is being loaded from the CD-ROM into memory for execution.

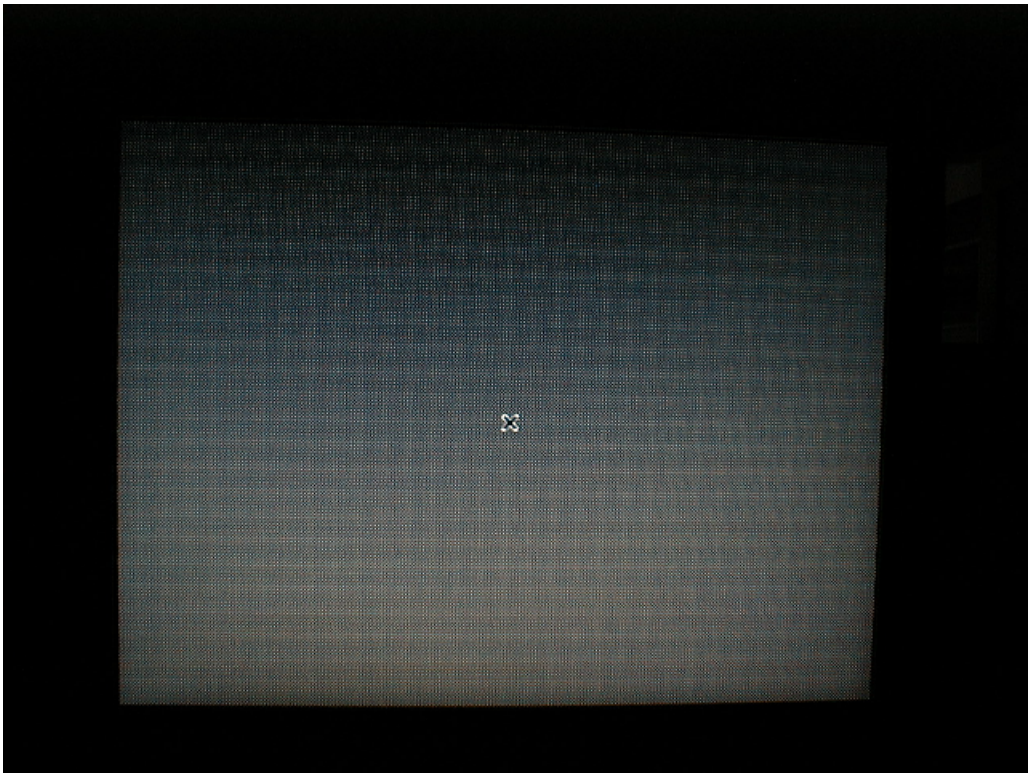
Starting The Anaconda Setup Program, part II

Consider the following:

In the above example,

Seeing X-Windows for the First Time

Consider the following:



In the above example,

The Red Hat Setup Splash Screen

For most Red Hat installers, this is a welcome but somewhat bewildering sight. Anything is better than a blue screen with a single line of text at the bottom, but this splash screen doesn't give **any** information about what's going on – at least with the **anaconda** program there was a promise of progress.

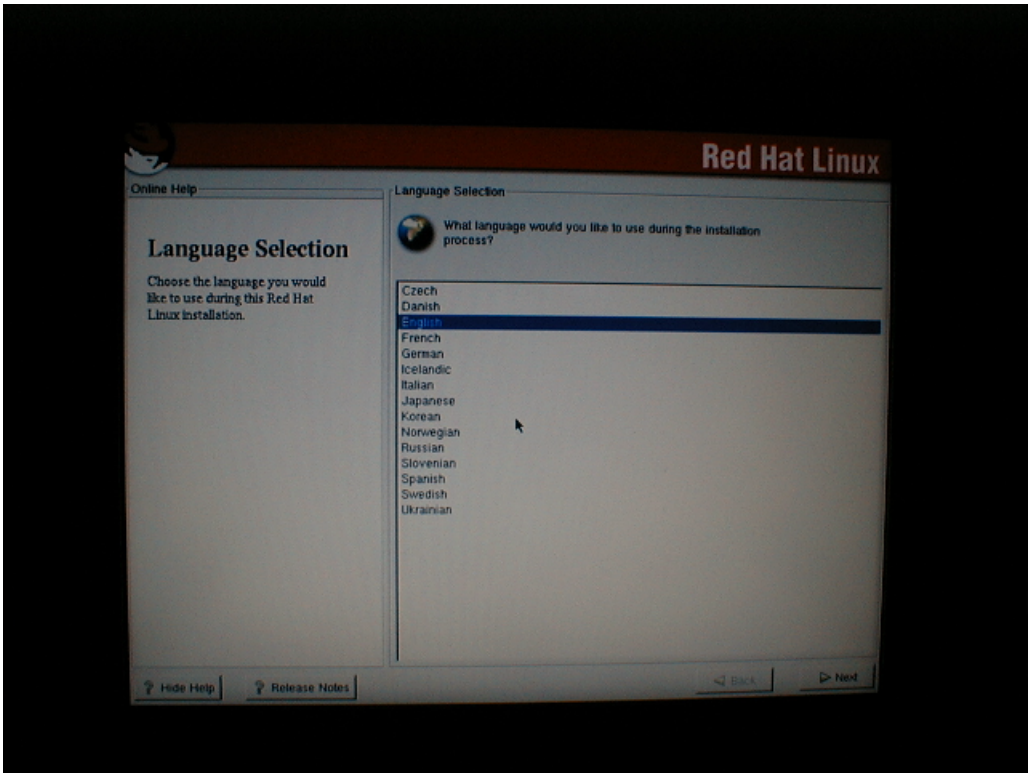
Anyway, after a few moments the splash screen goes away

Consider the following:

In the above example the **anaconda** program has been loaded into memory and the first thing it does is display a splash screen while it performs some setup tasks.

Language Selection

Consider the following:

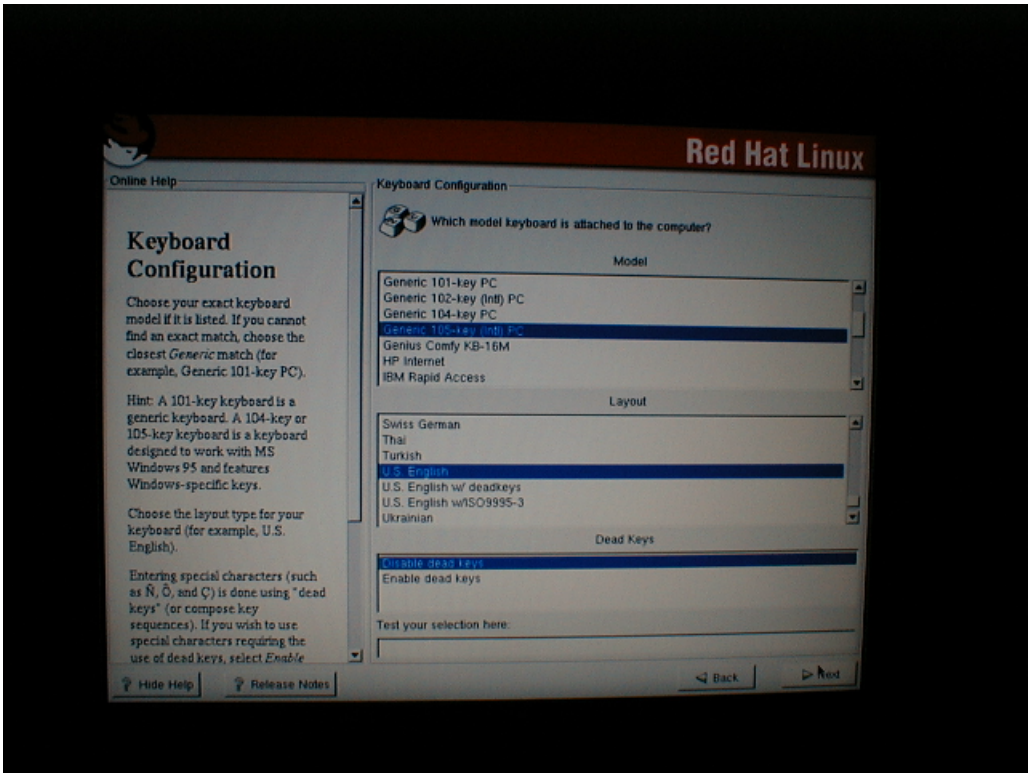


In the above example the **anaconda** program is prompting the user to supply the language they wish to have displayed during the installation.

Select **English** and the click on the **Next** button to continue. If the mouse is not working, press the **<TAB>** key until the **Next** button is highlighted and then press **<ENTER>**.

Keyboard Configuration

Consider the following:

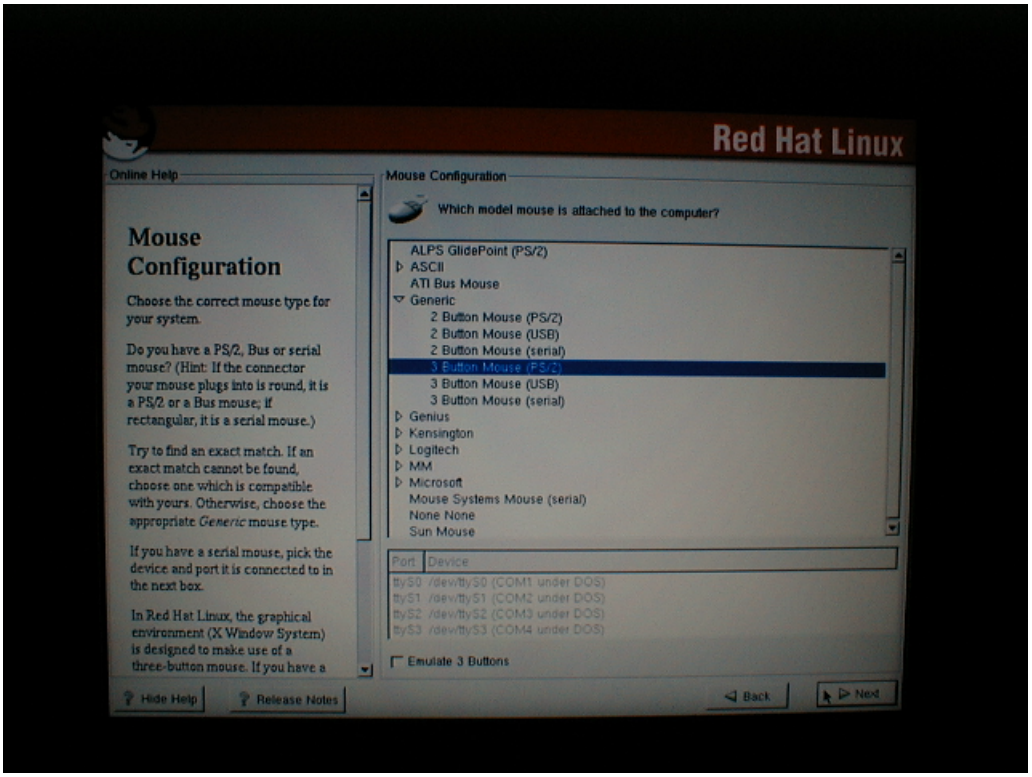


In the above example the installer is being prompted to supply information about their keyboard type and whether they want to enable extended functionality, called **dead keys**, to allow for the generation of special characters.

Pick the appropriate setting, disable **dead keys** and click on **Next** to continue. If the mouse is not working, press the **<TAB>** key until the **Next** button is highlighted and then press **<ENTER>**.

Mouse Configuration

Consider the following:



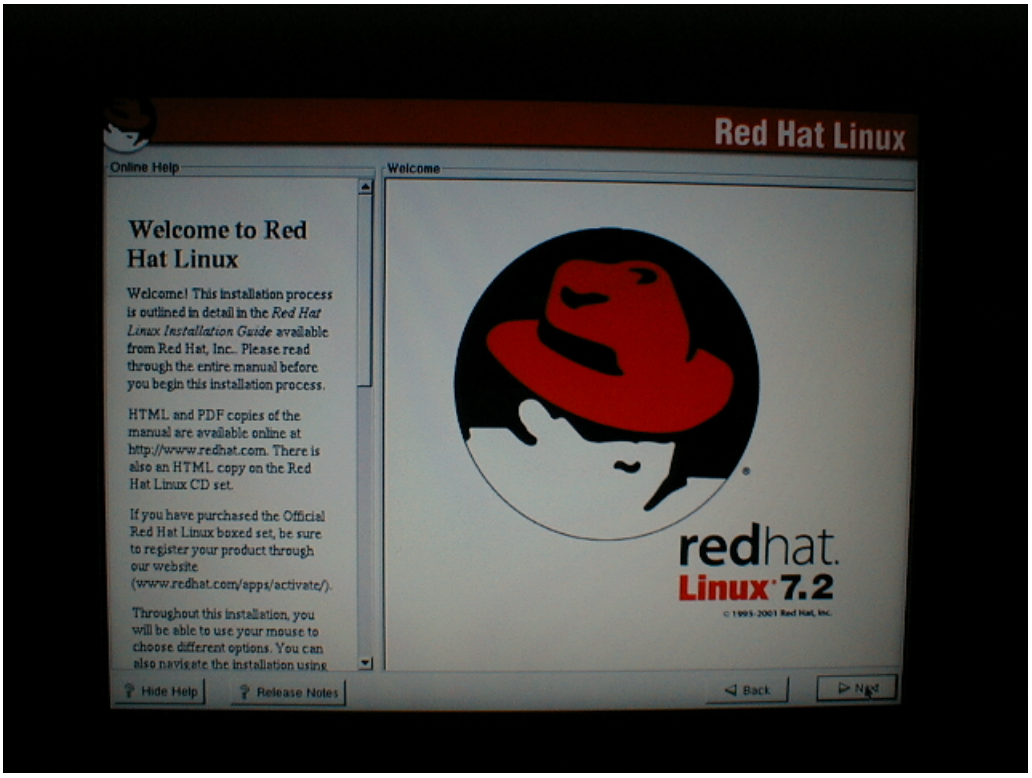
For some reason the mouse detection in **anaconda** always mis-identifies the mouse type. Most mice now are **two button, PS/2** mice – not three button. If you select a two button mouse, be sure to enable three button functionality with the check box at the bottom of the screen.

Anyway, for your situation pick the appropriate setting and click on **Next** to continue.

If the mouse is not working, press the **<TAB>** key until the **Next** button is highlighted and then press **<ENTER>**.

Commencement of System Setup

Consider the following:

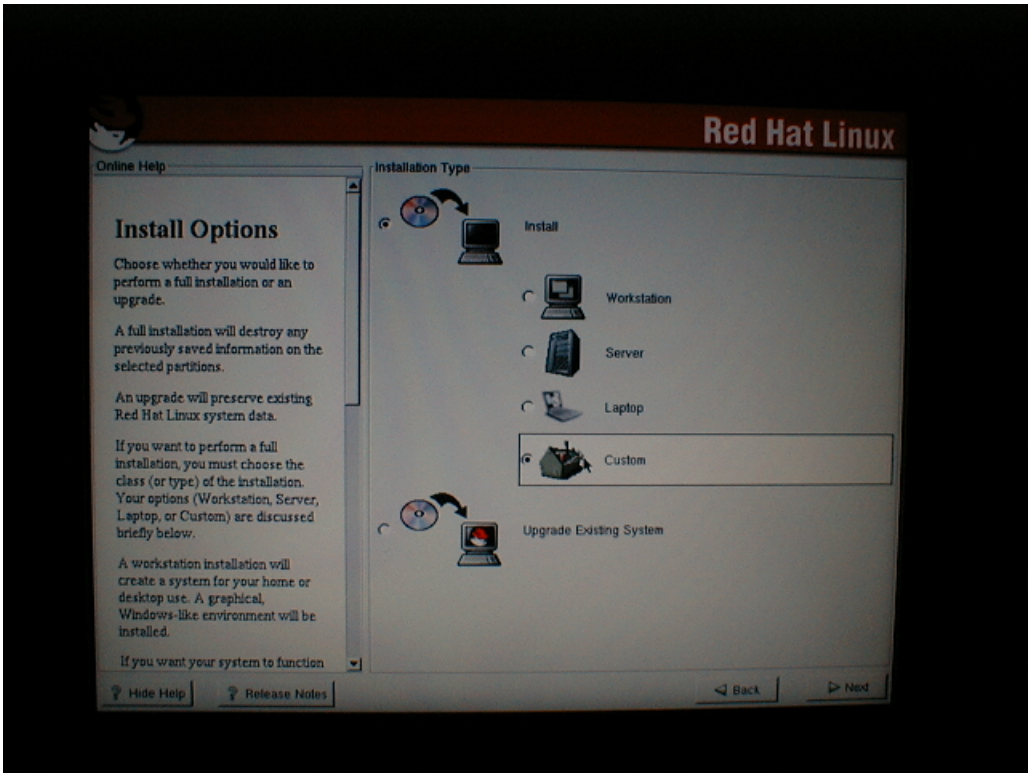


In the above example the Red Hat setup program is offering an information screen describing what is about to happen in the install, along with some other supporting information.

Click on **Next** to continue.

Select The System Profile

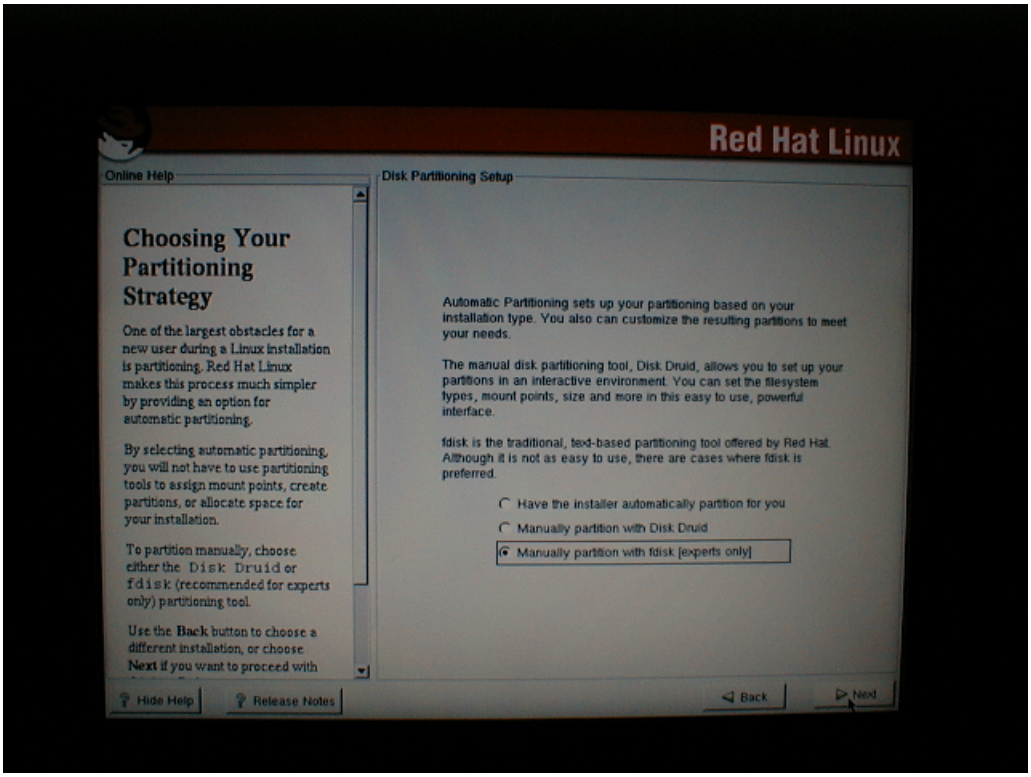
Consider the following:



Select **custom** and click on **Next** to continue.

Choosing Your Partitioning Strategy

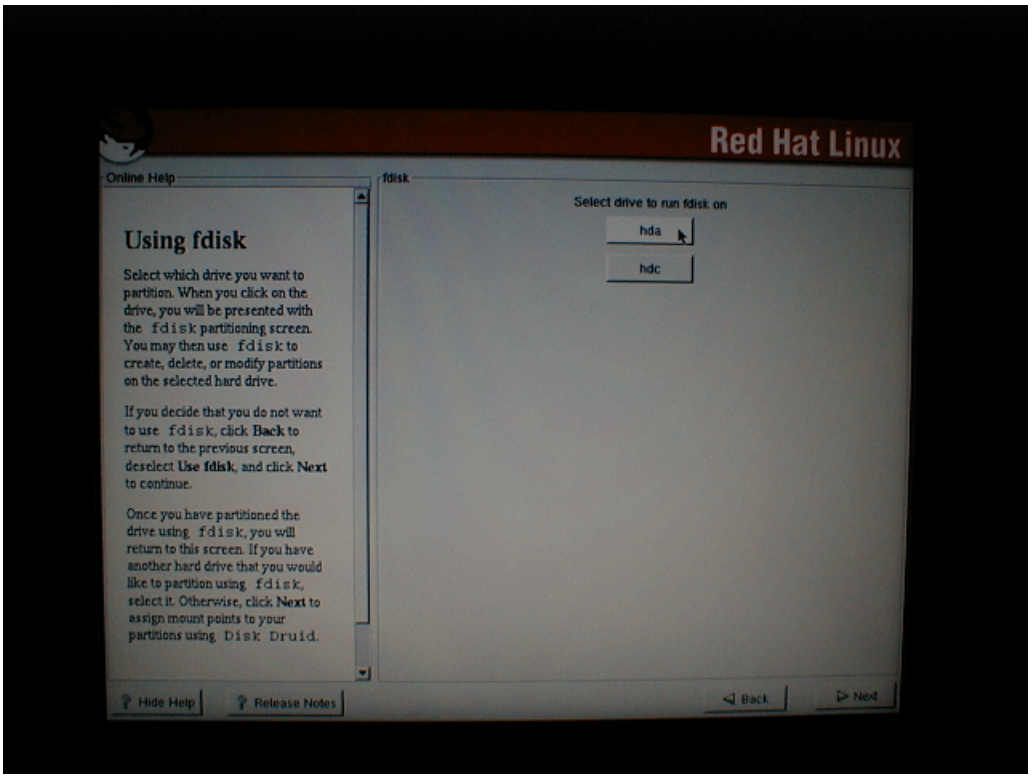
Consider the following:



Select **manually partition using fdisk** and click on **Next** to continue.

Using fdisk

Consider the following:



Select **hda** and click on **Next** to continue.

Partitioning hda

The fdisk Commands

fdisk is not the most friendly of programs, but it is quite powerful once the basic commands are mastered.

Consider the following:

NEED SHOT OF HELP SCREEN HERE

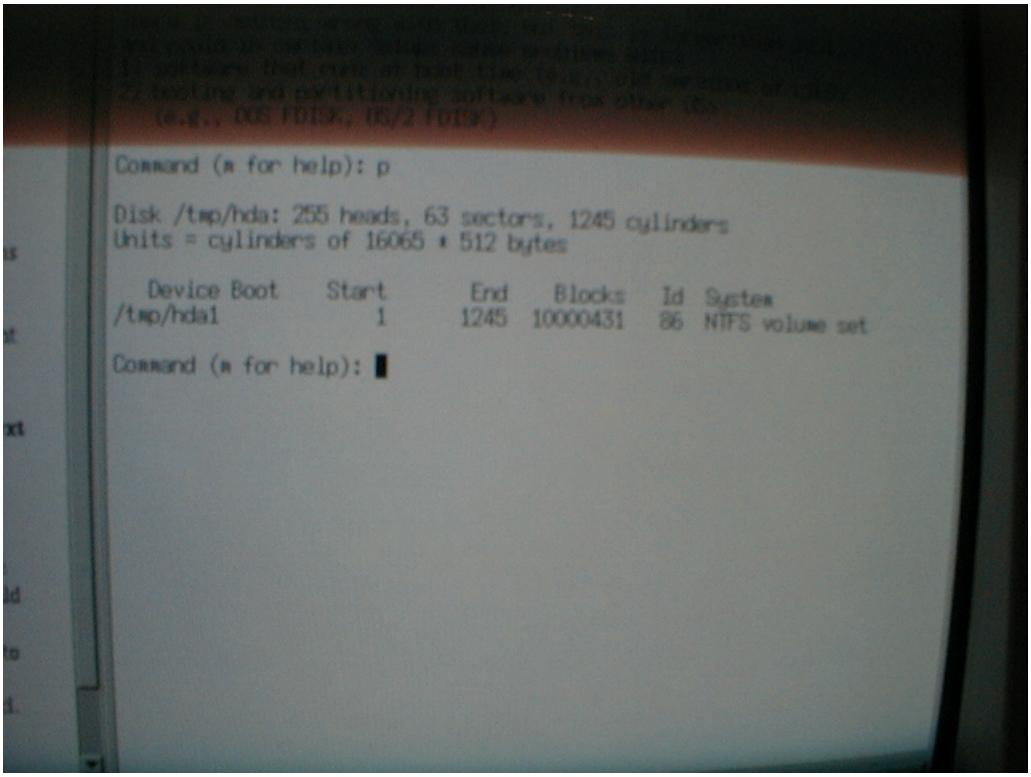
In the above example the **fdisk** program has been invoked and the program is running.

Basic fdisk Commands

Character	Function
?	Display help menu
n	New Partition
p	Print the partition table
a	Tag a partition bootable
t	Tag a parition type
w	Write the partition table and exit fdisk

How To View Partitions With fdisk

Consider the following:

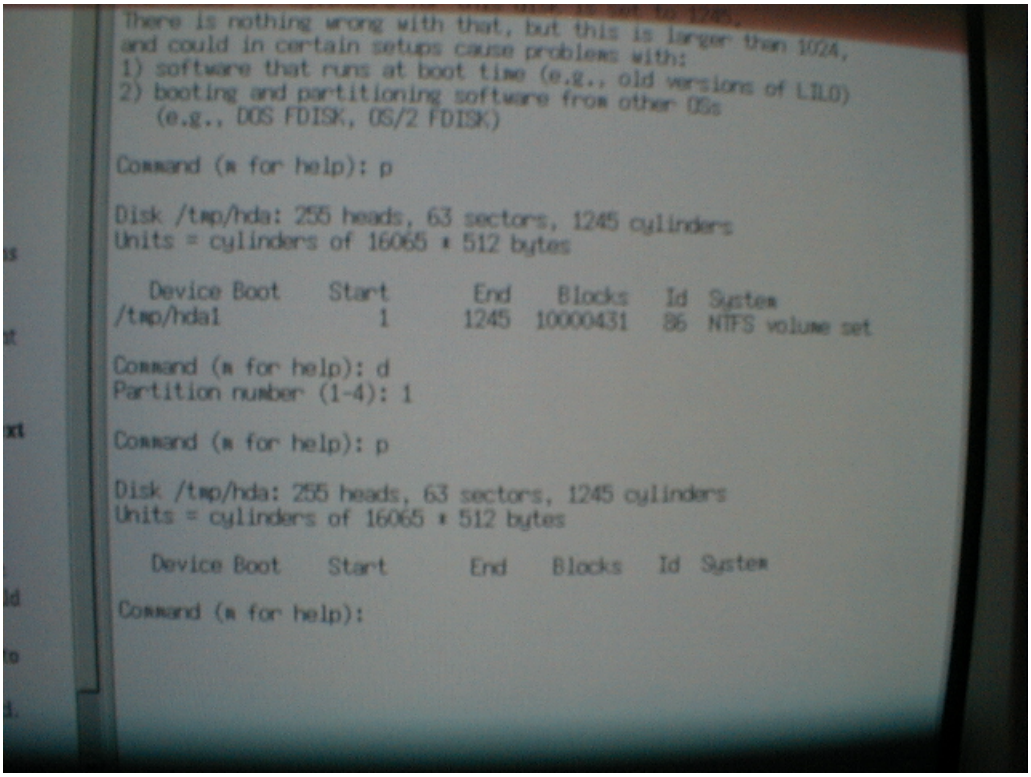


In the above example the partition table has been printed with the **p** command. In this case, a partition exists on the disk of type **NTFS volume set**. This partition spans the entire disk, from cylinder 1 to cylinder 1245.

This partition must be deleted.

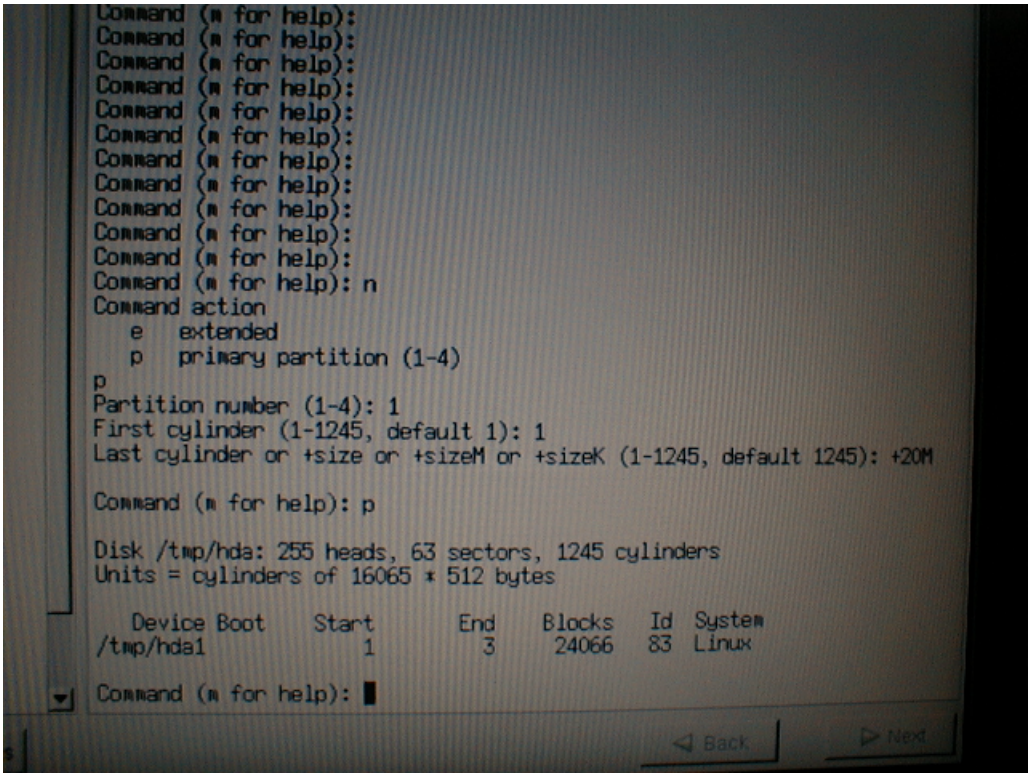
How To Delete A Partition With fdisk

Consider the following:

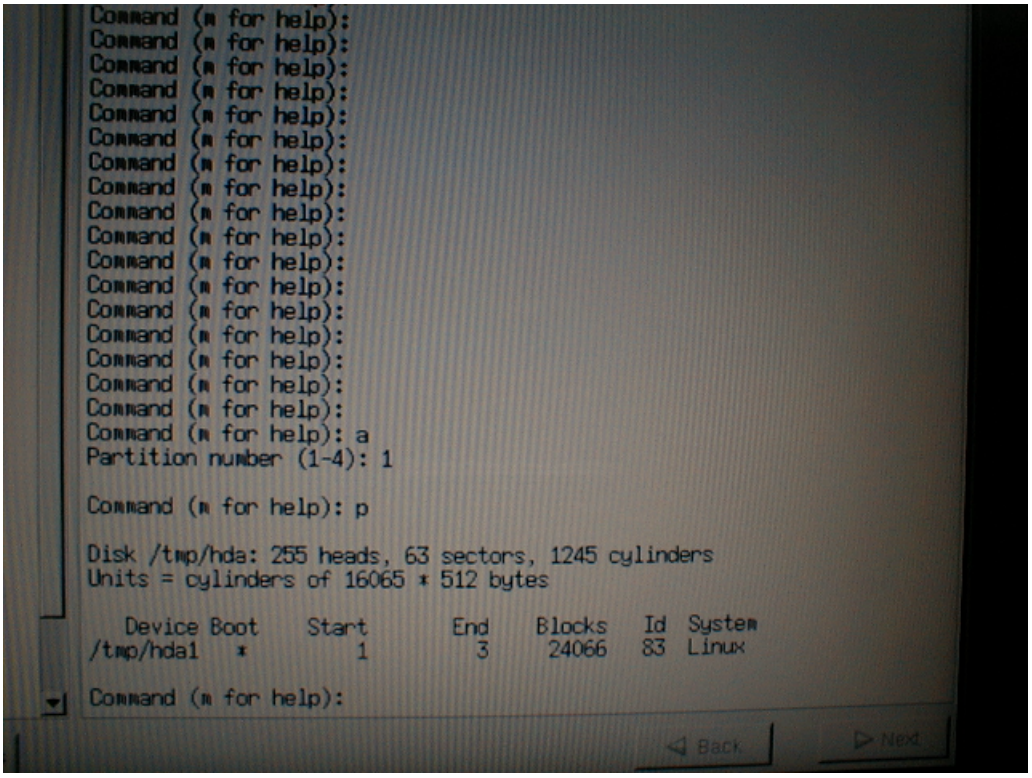


In the above example the first partition has been deleted from the disk with the **d** command.

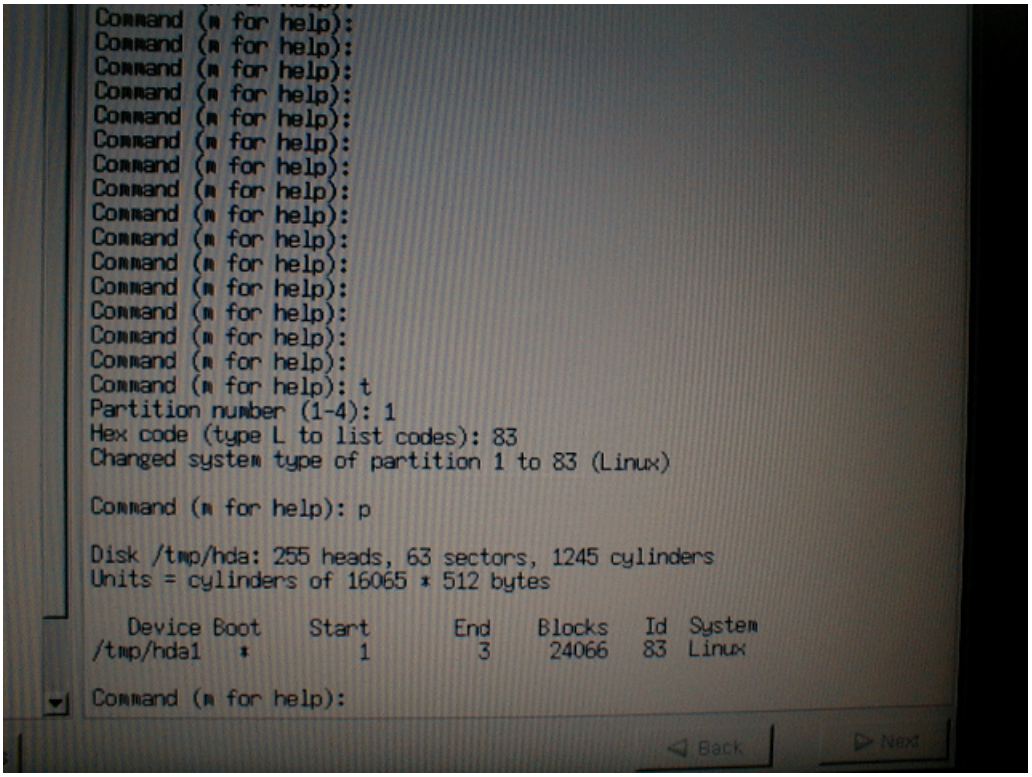
Creating A 20Mb Boot Partition



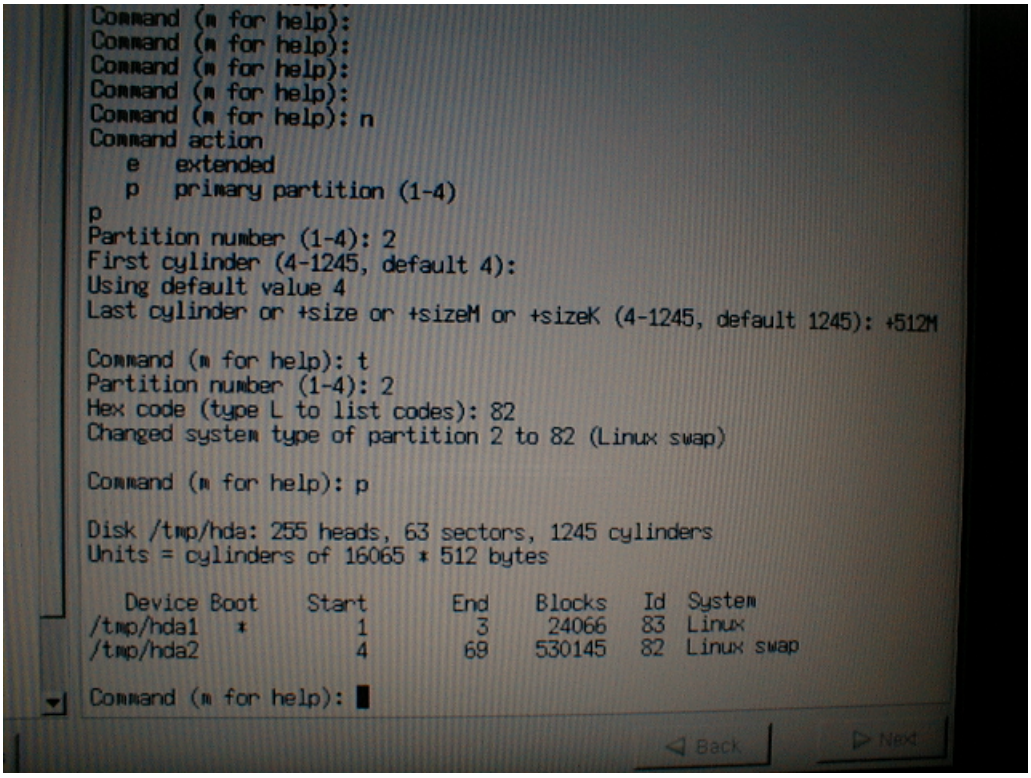
Making A Partition Bootable With fdisk



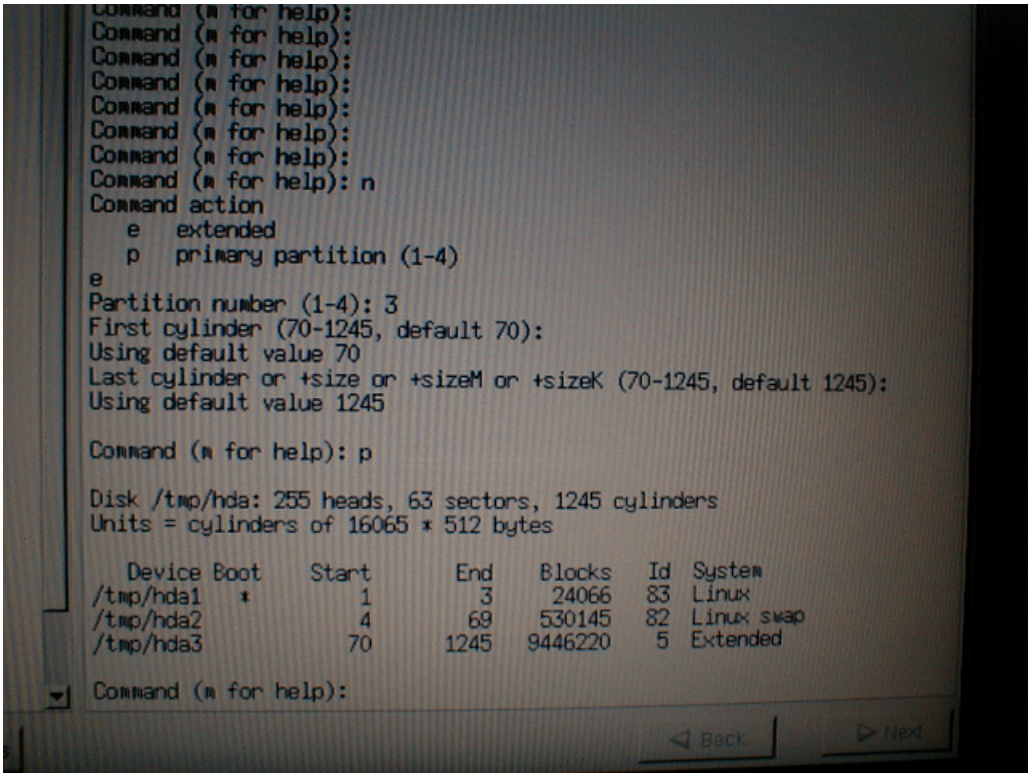
Tagging A Partition For Linux



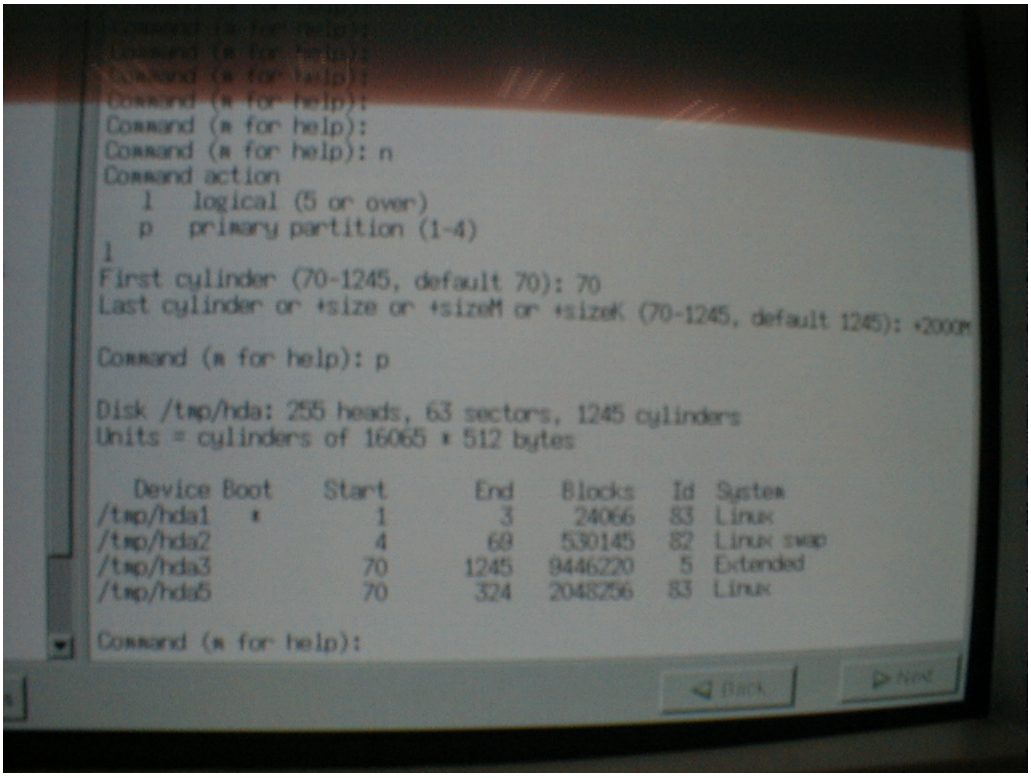
Creating A Swap Partition



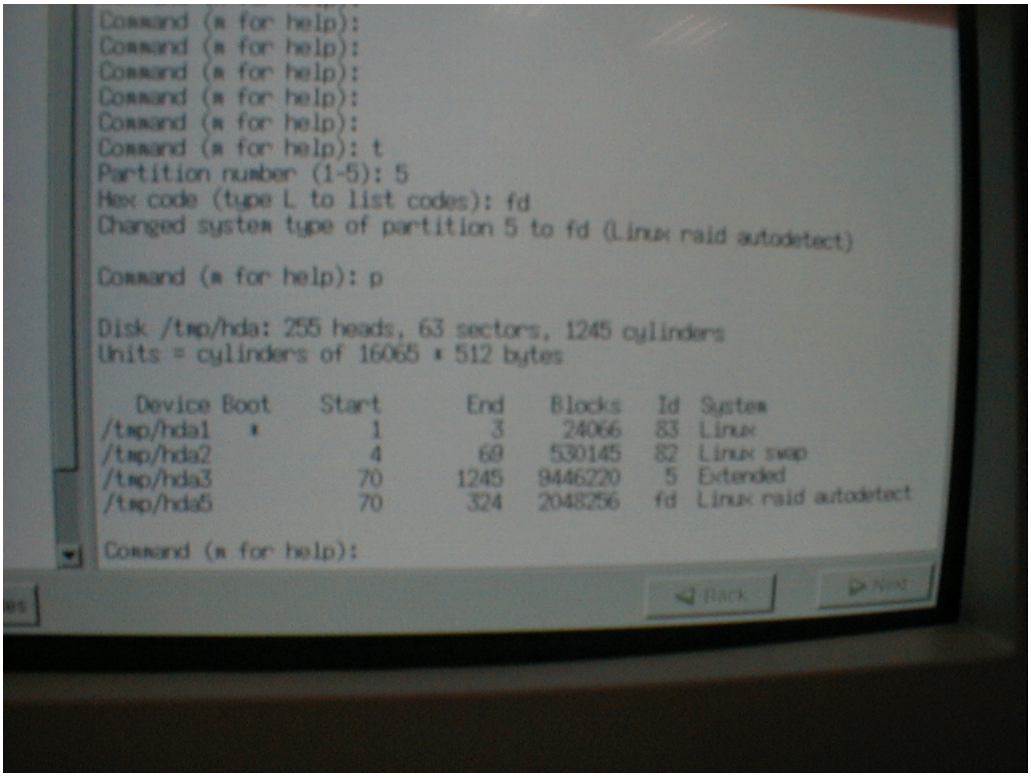
How To Create An Extended Partition For Linux



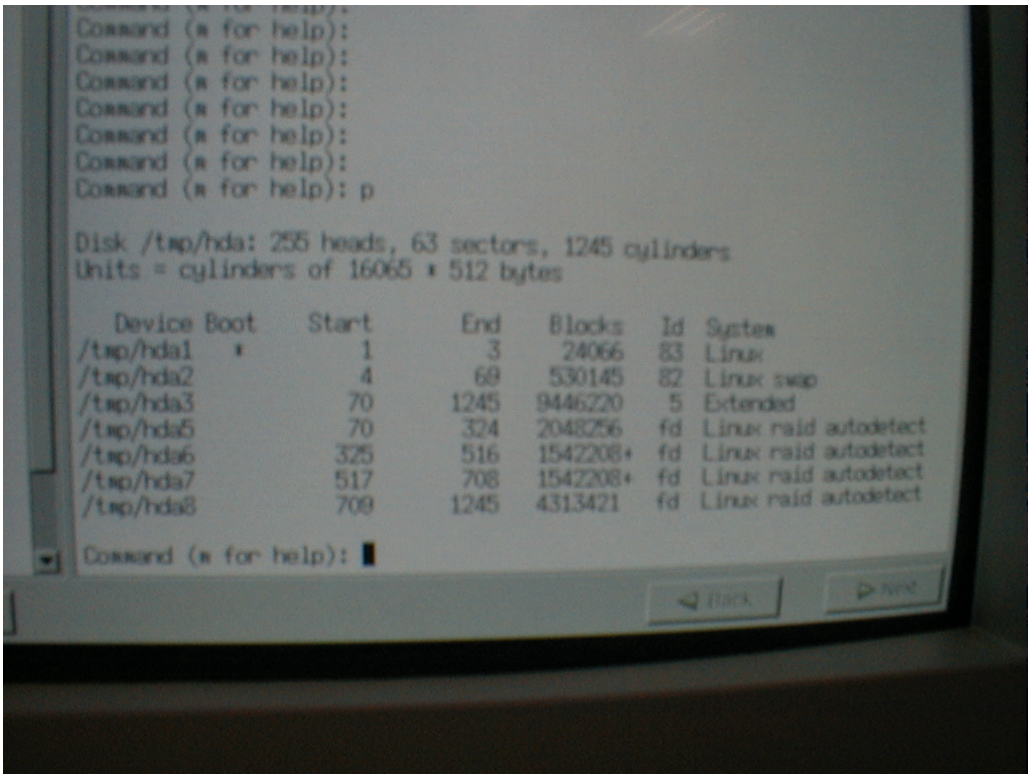
How To Create A Logical Drive In An Extended Partition



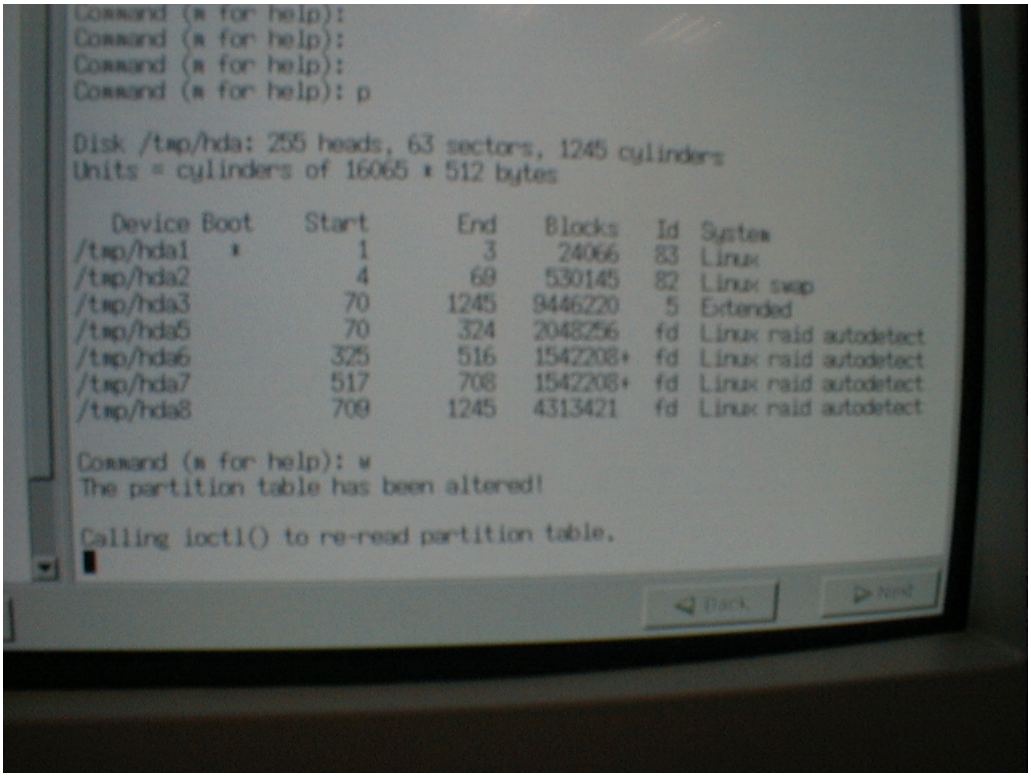
How To Tag A Partition For RAID



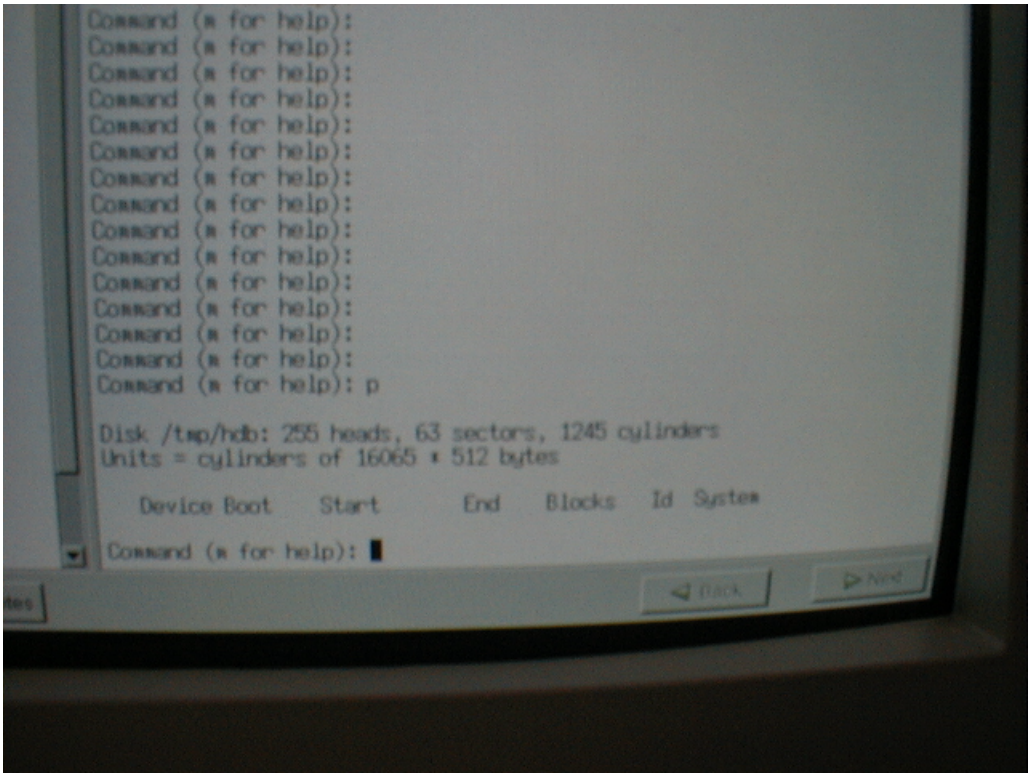
All Of The RAID Partitions Prepared



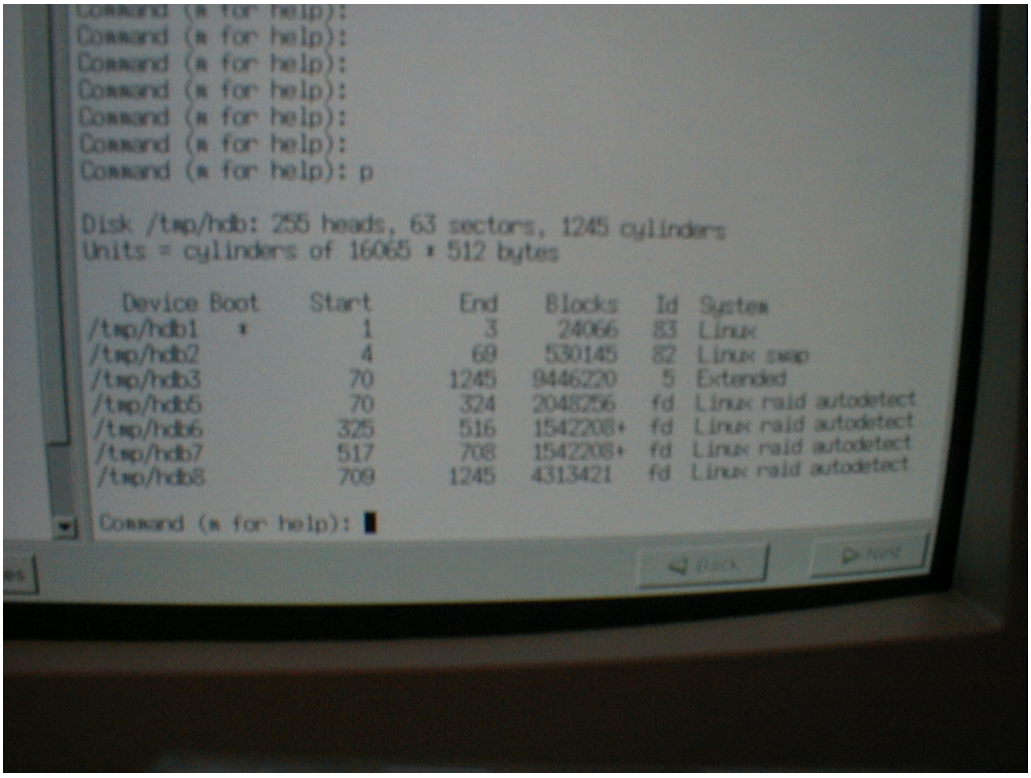
How To Write The Partition Table



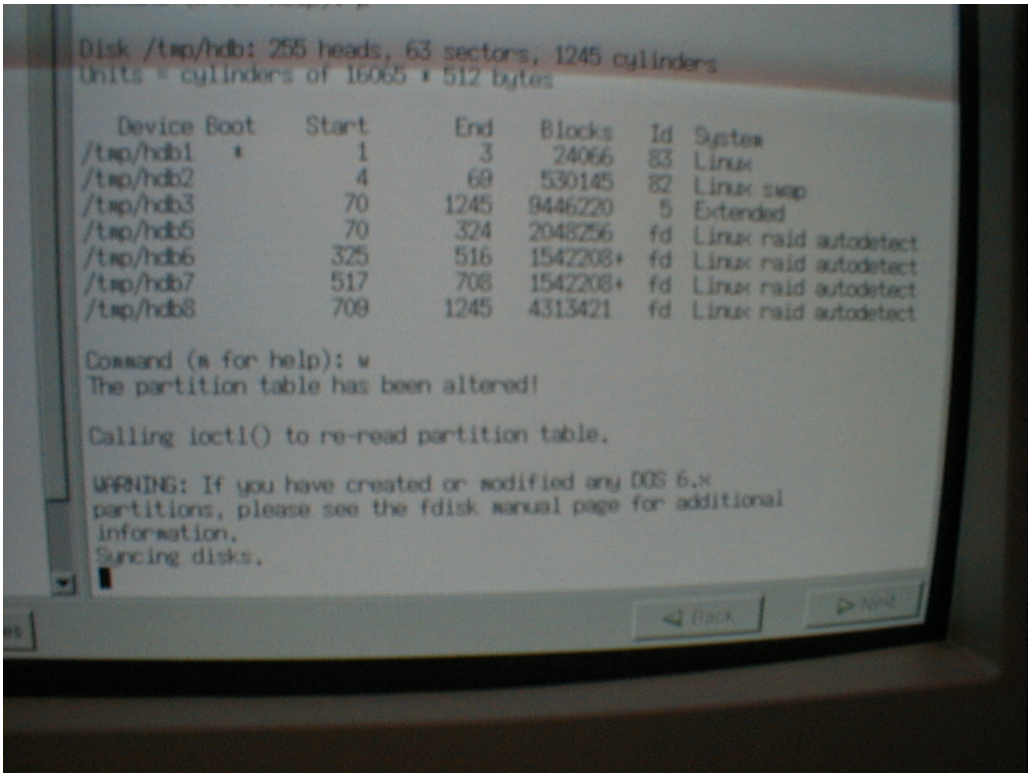
Partitioning hdb



Make hdb An Exact Duplicate Of hda

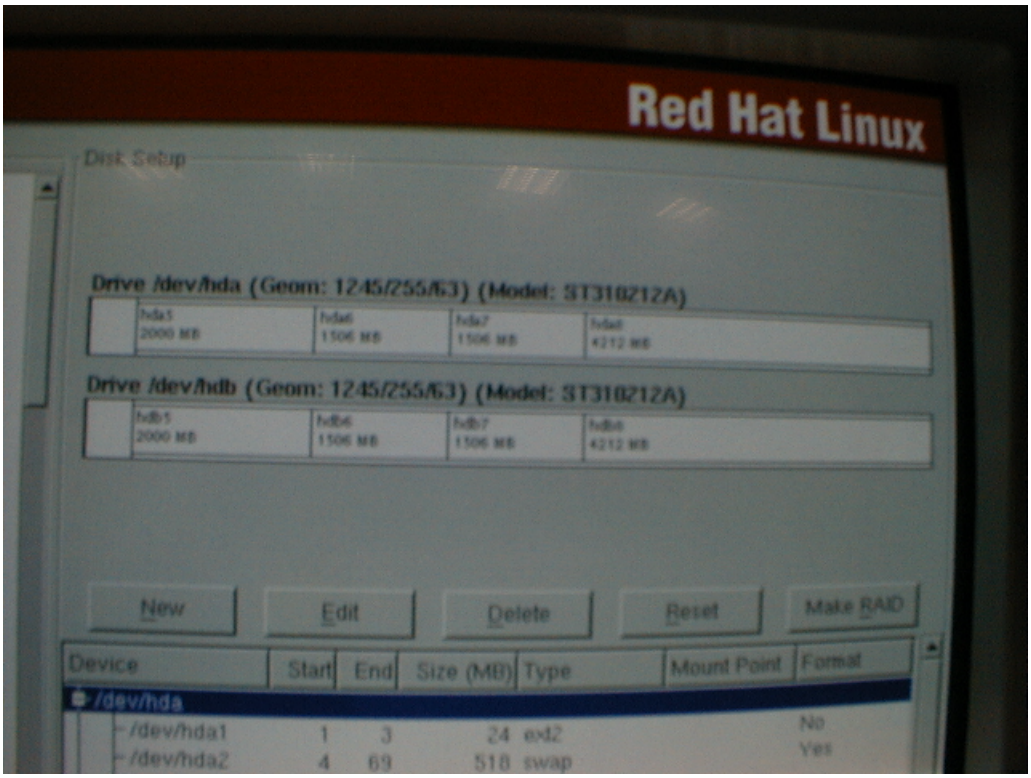


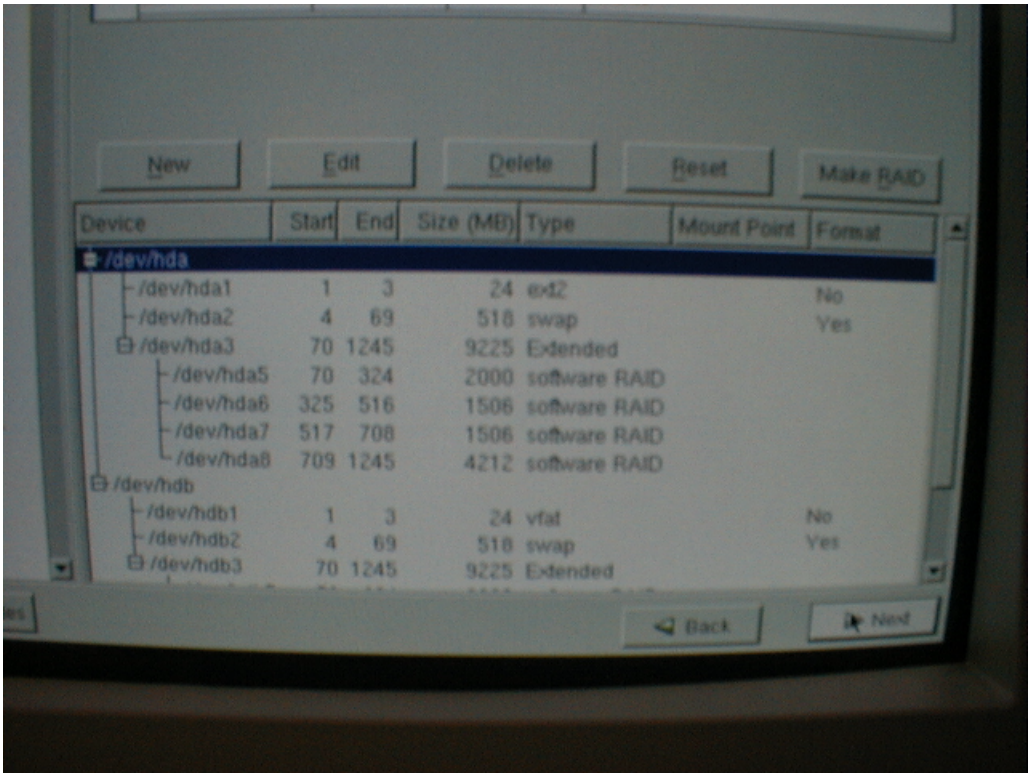
Writing The Partition Table A Second Time



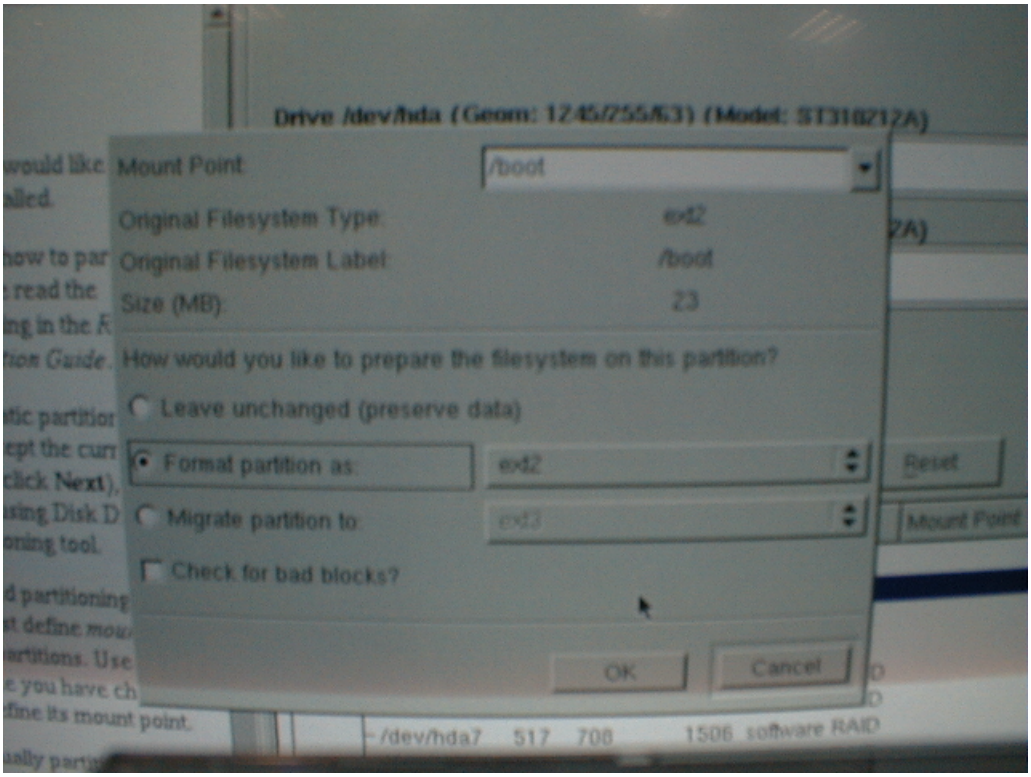
Disk Druid

Using Disk Druid To Setup Drives

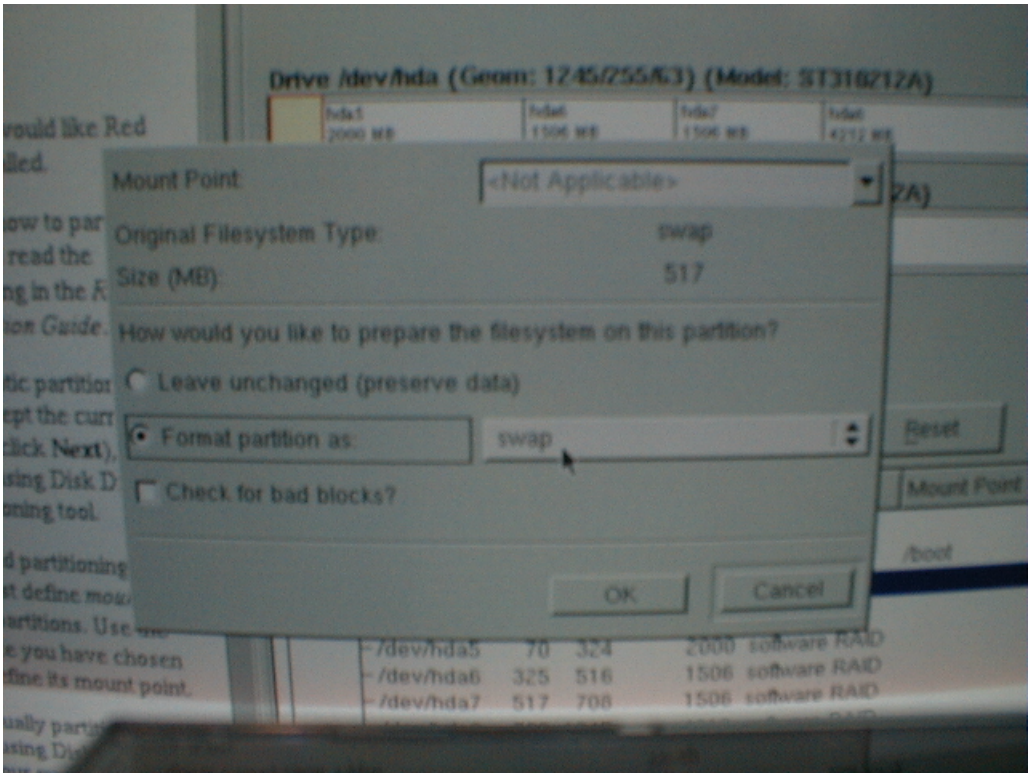




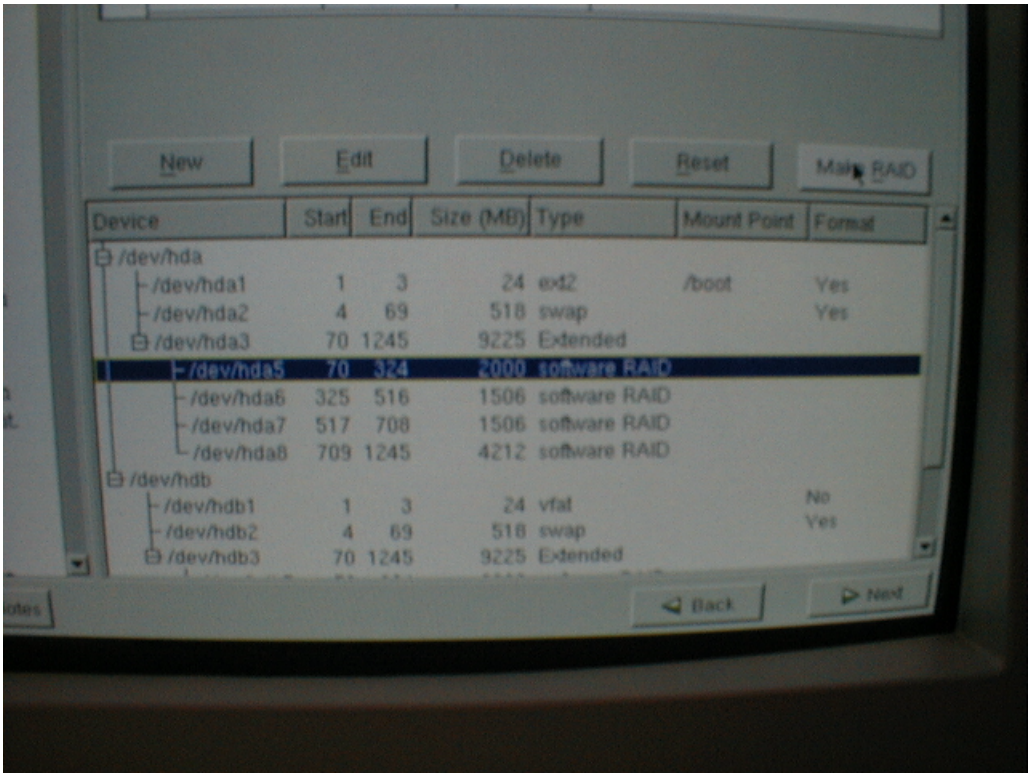
How To Configure the /boot Partition



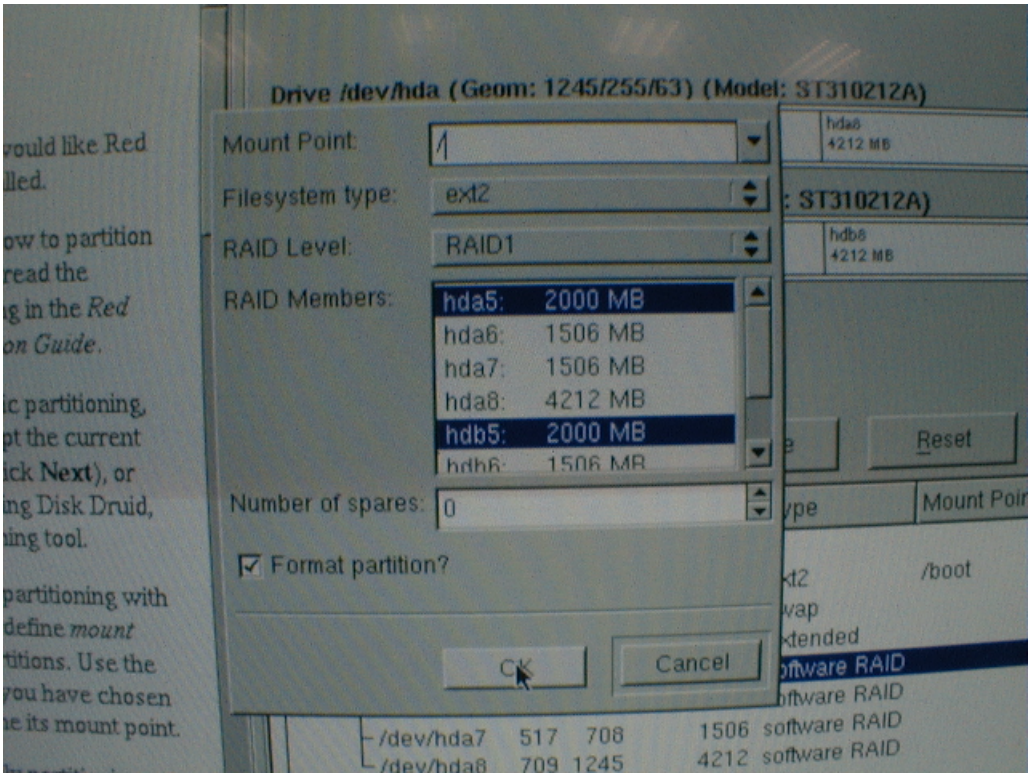
How To Configure The swap Partition



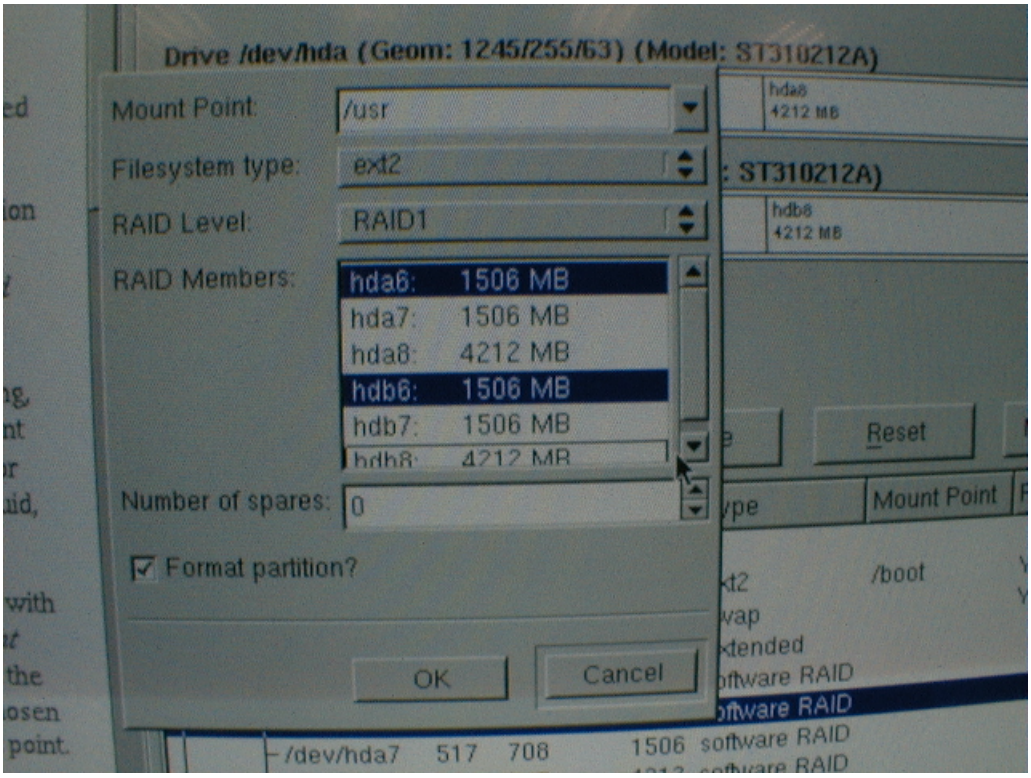
Using Disk Druid To Make A RAID Device



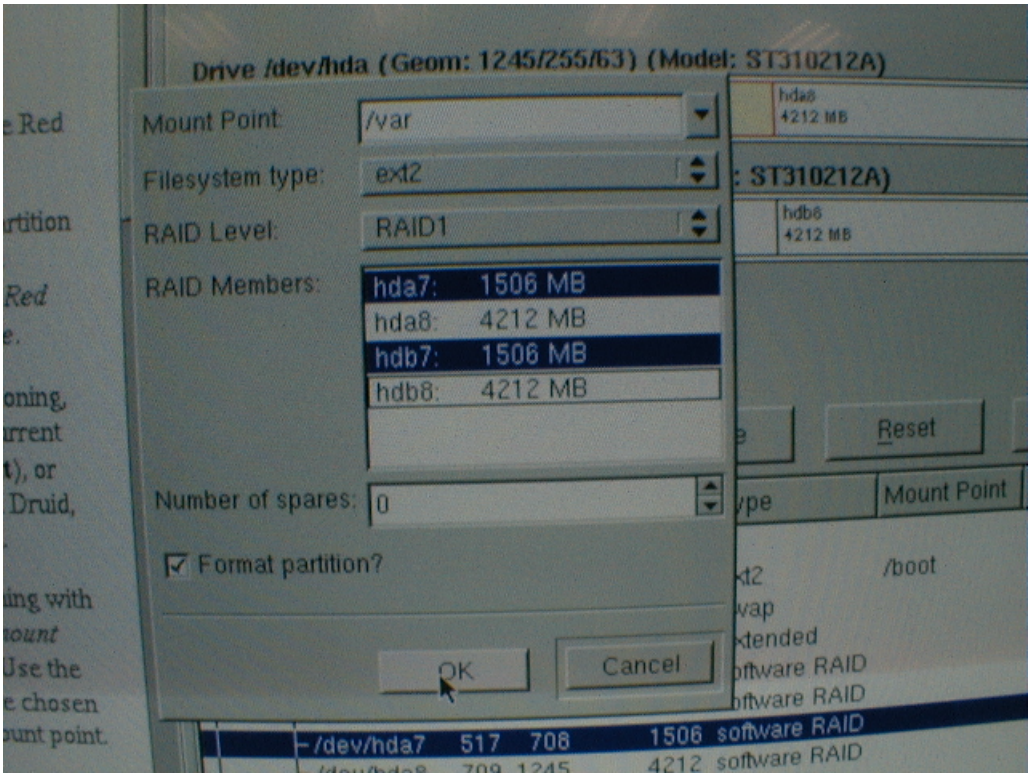
How To Select The Appropriate RAID-1 Partitions for /



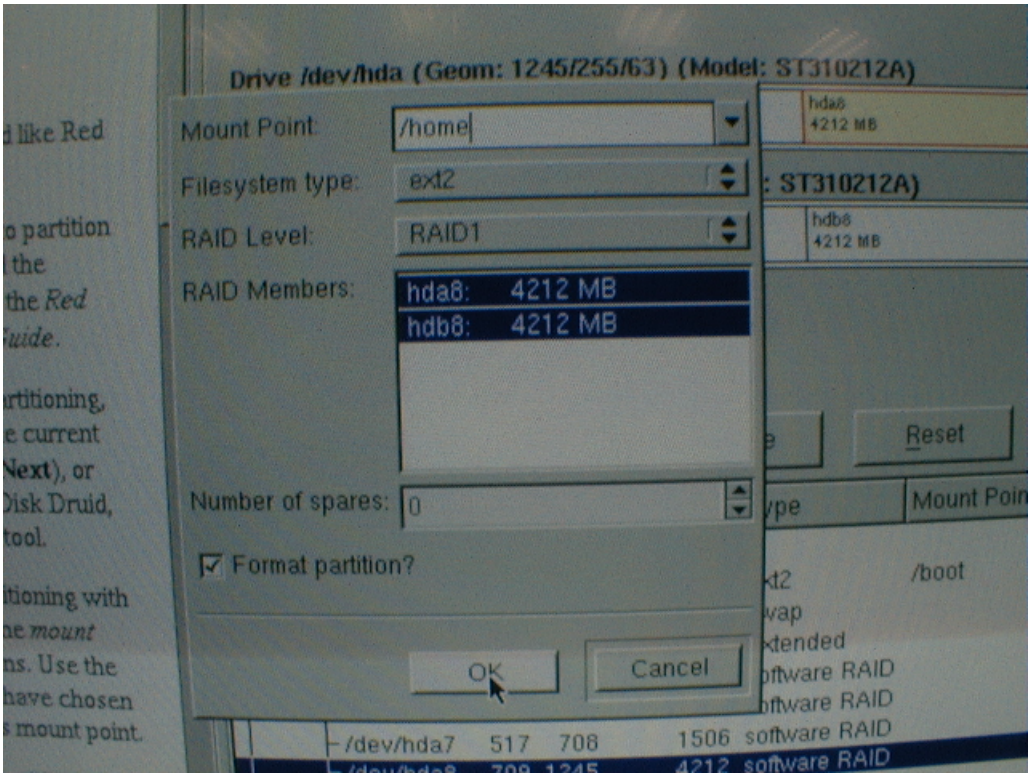
How To Configure the /usr Partition



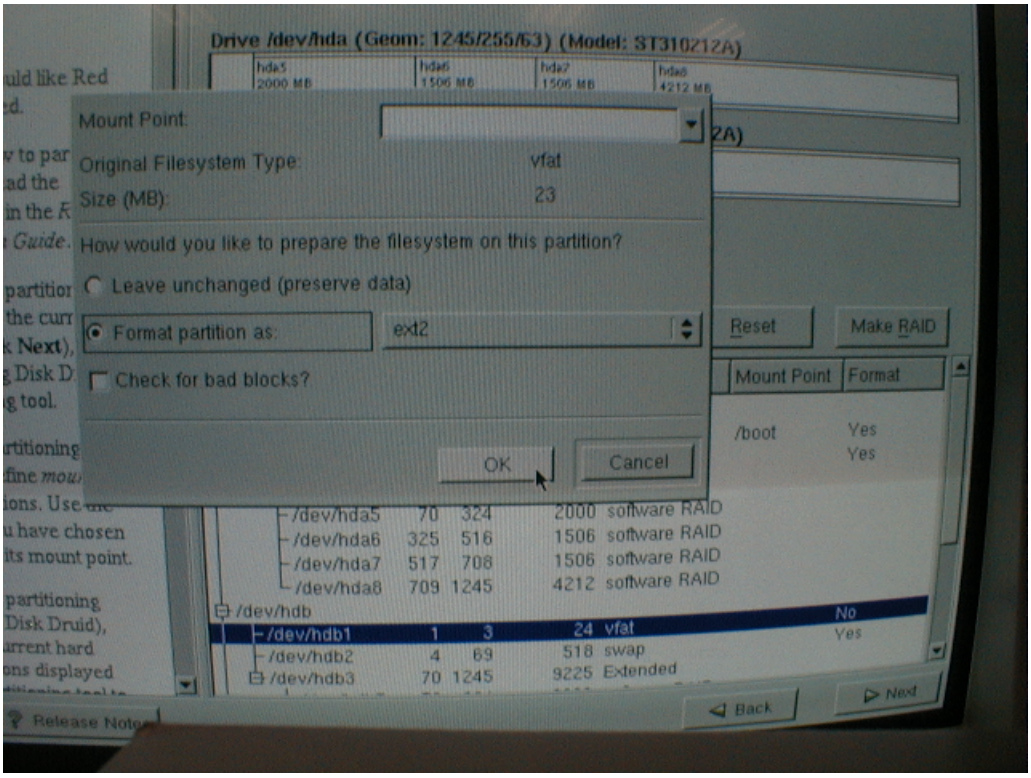
How To Configure the /var Partition



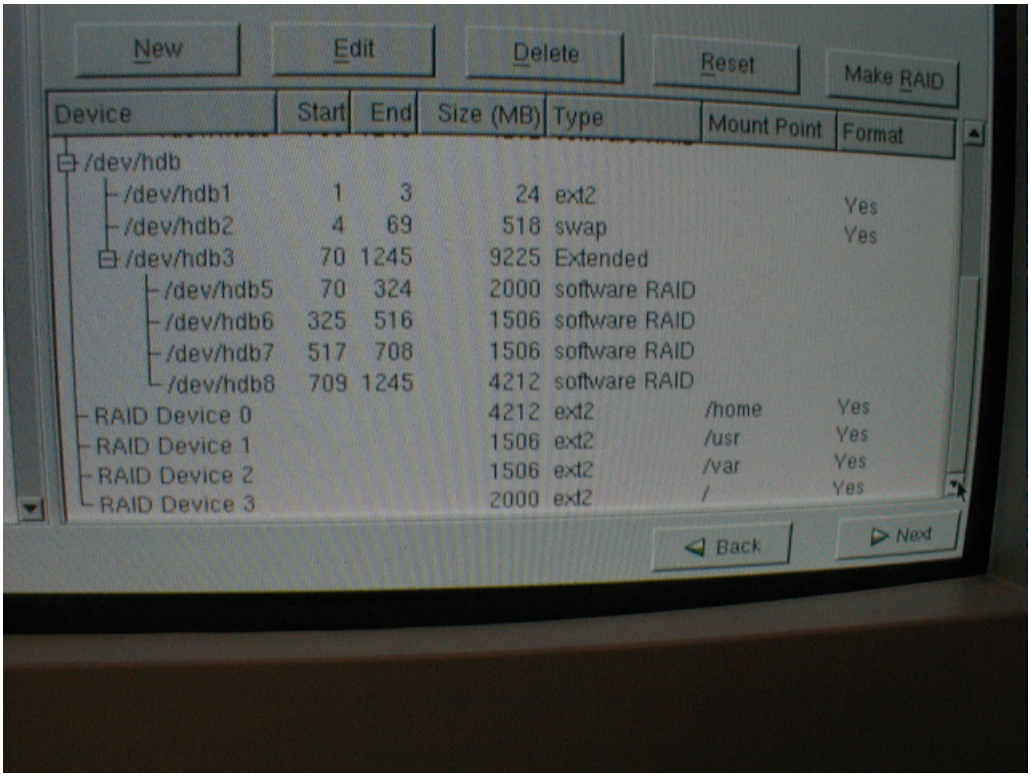
How To Configure The /home Partition



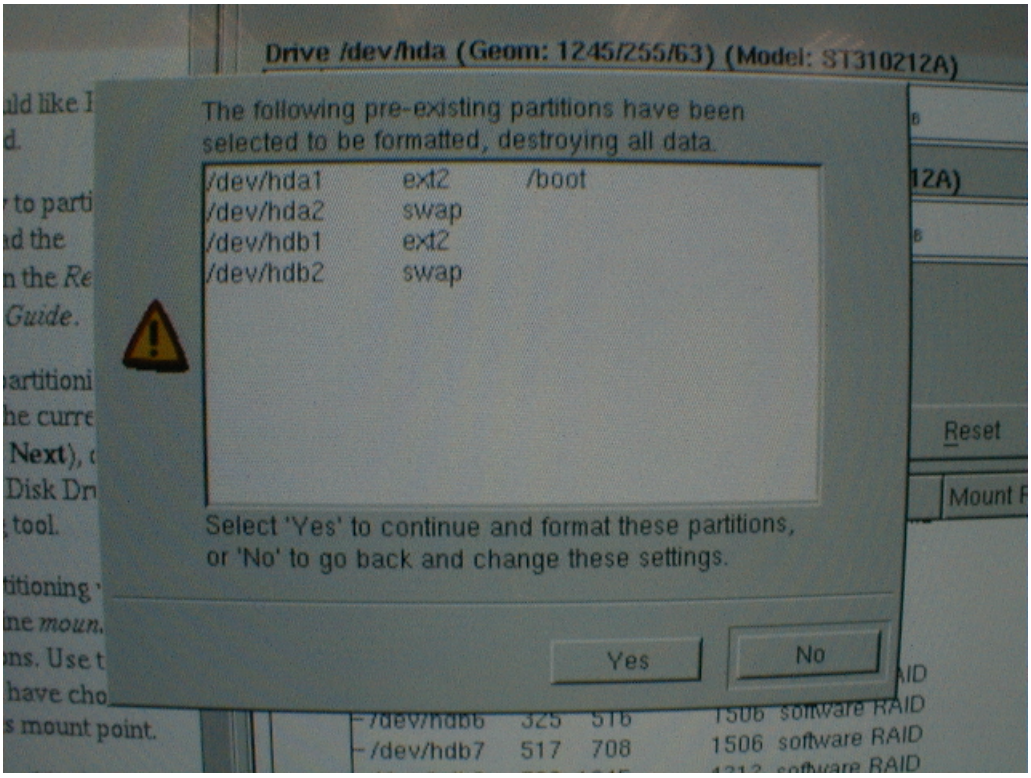
How To Configure The Duplicate /boot Partition



How To Verify The RAID Configuration

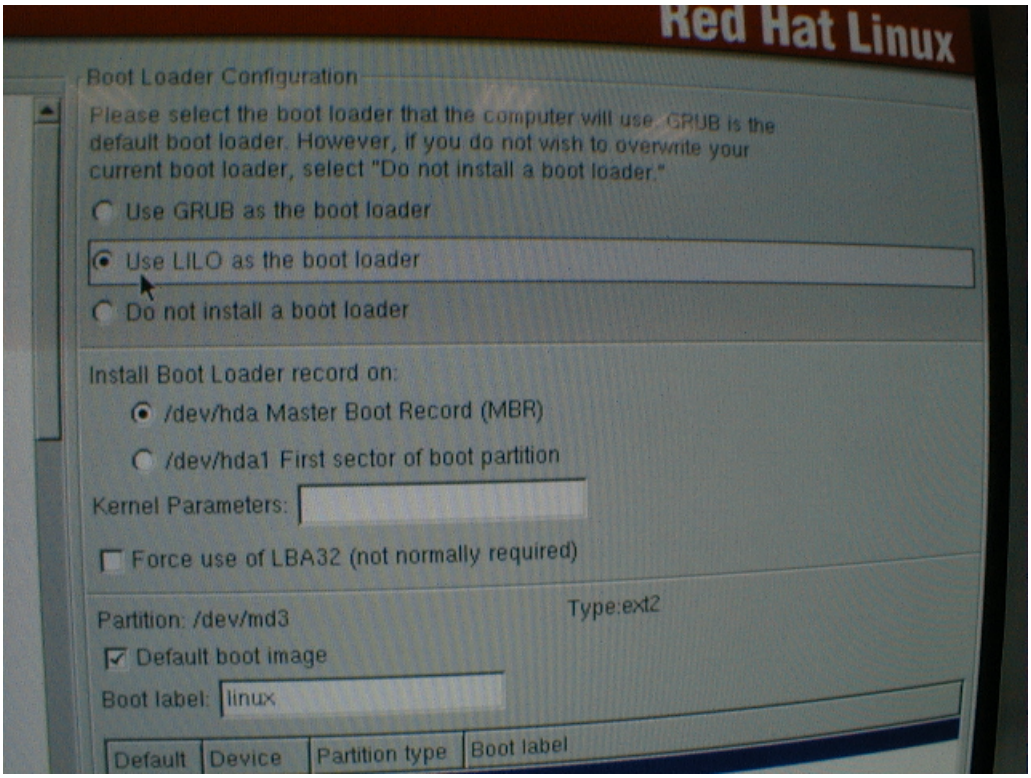


How To Confirm Partition Formatting

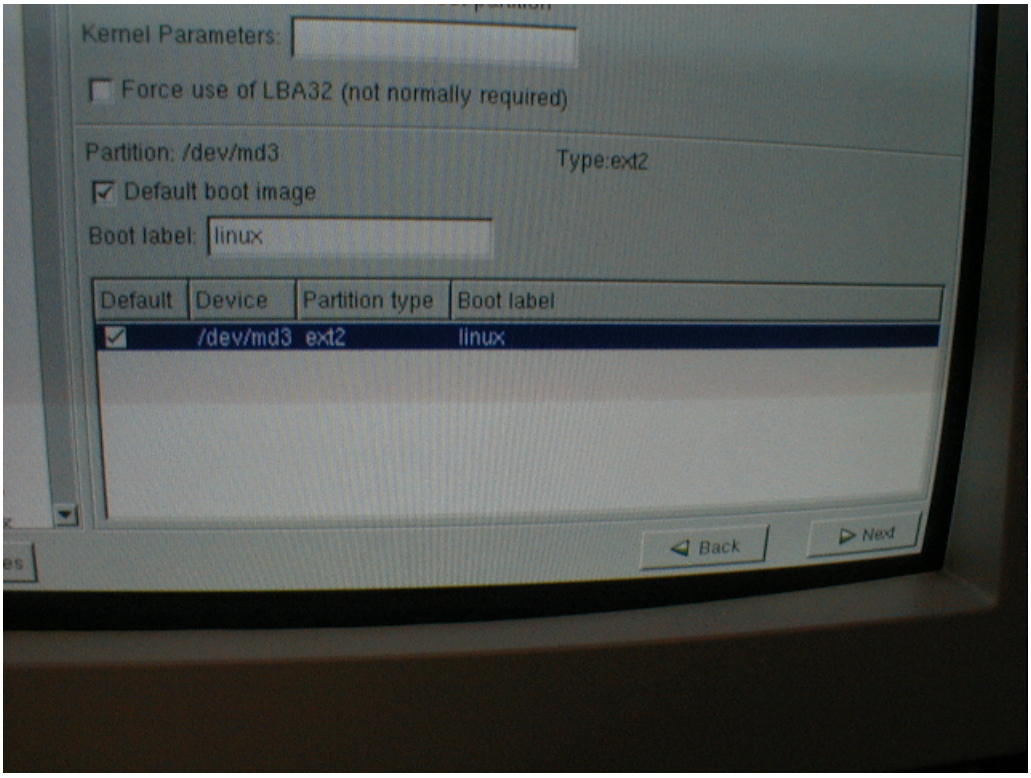


The Boot Loader

How To Configure The System To Use LILO

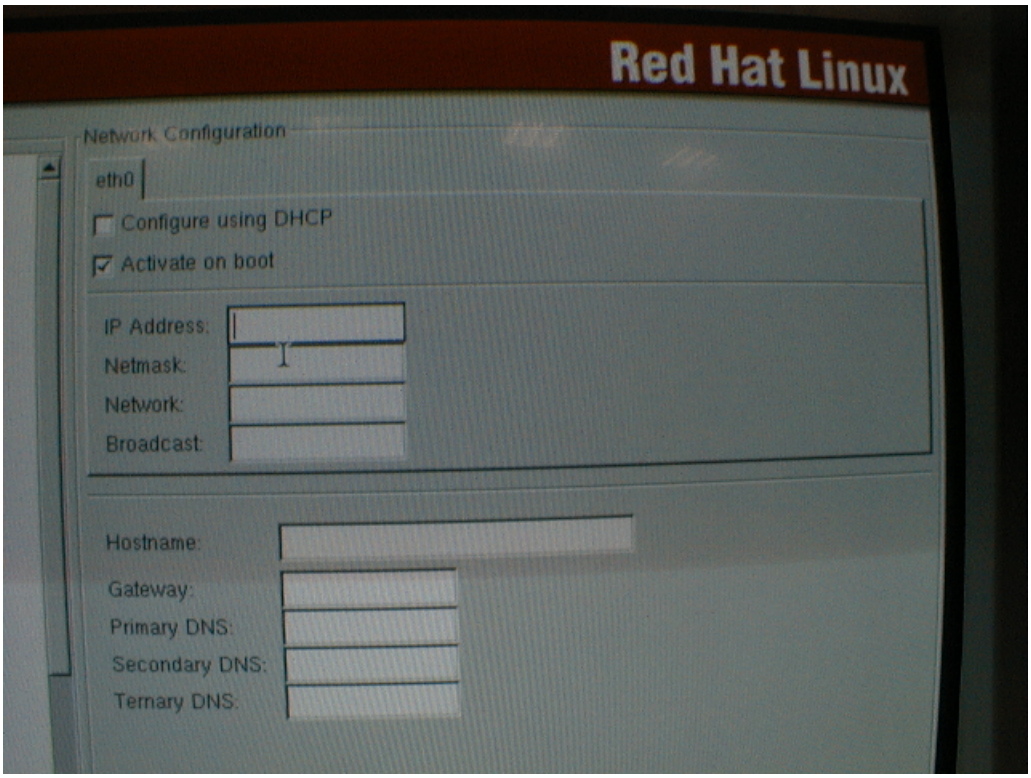


Where To Locate The Boot Code

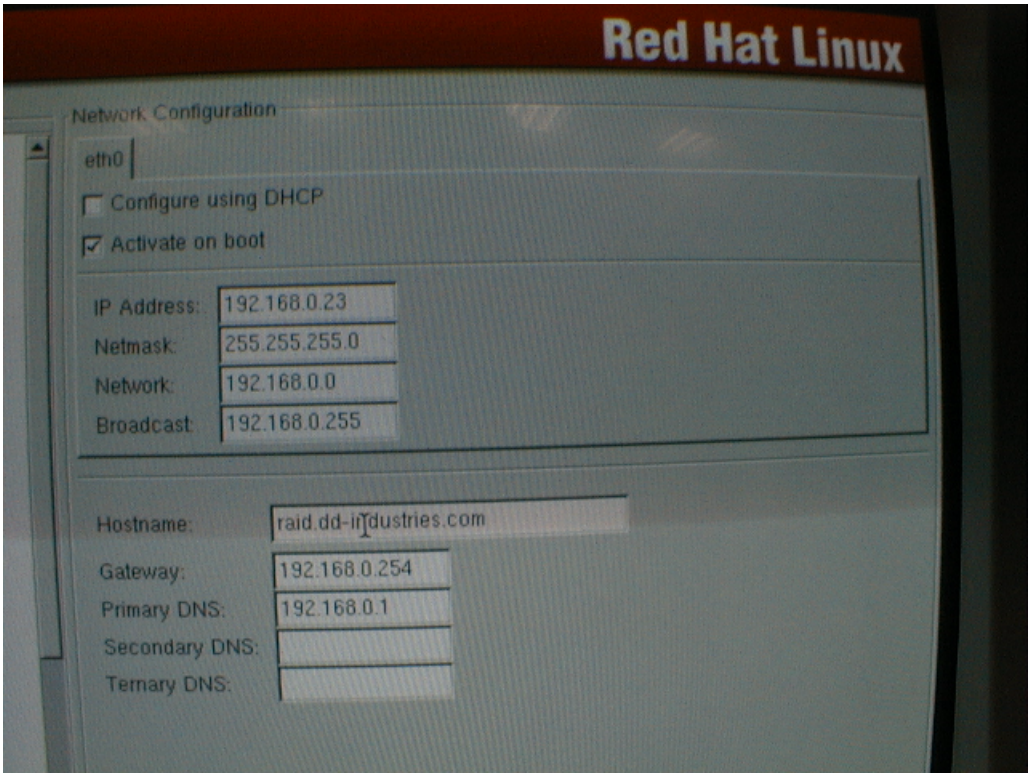


Networking

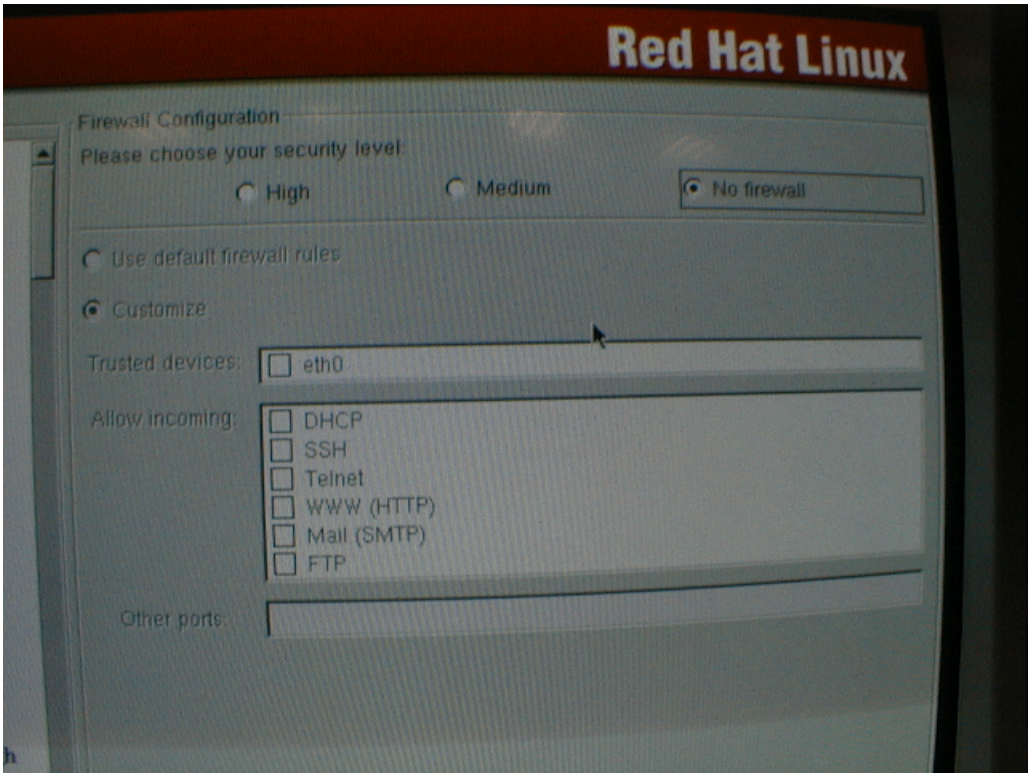
How To Configure Networking



An Example Network Configuration

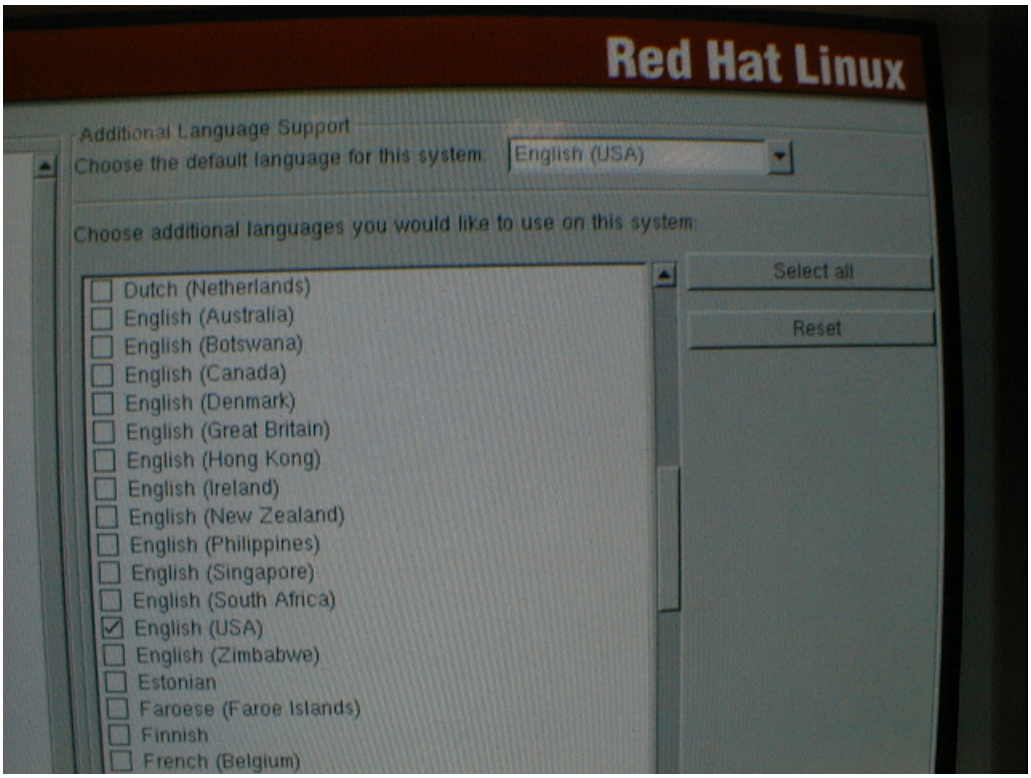


How To Defer Firewalling



Additional Language Support

How To Enable Additional Language Support

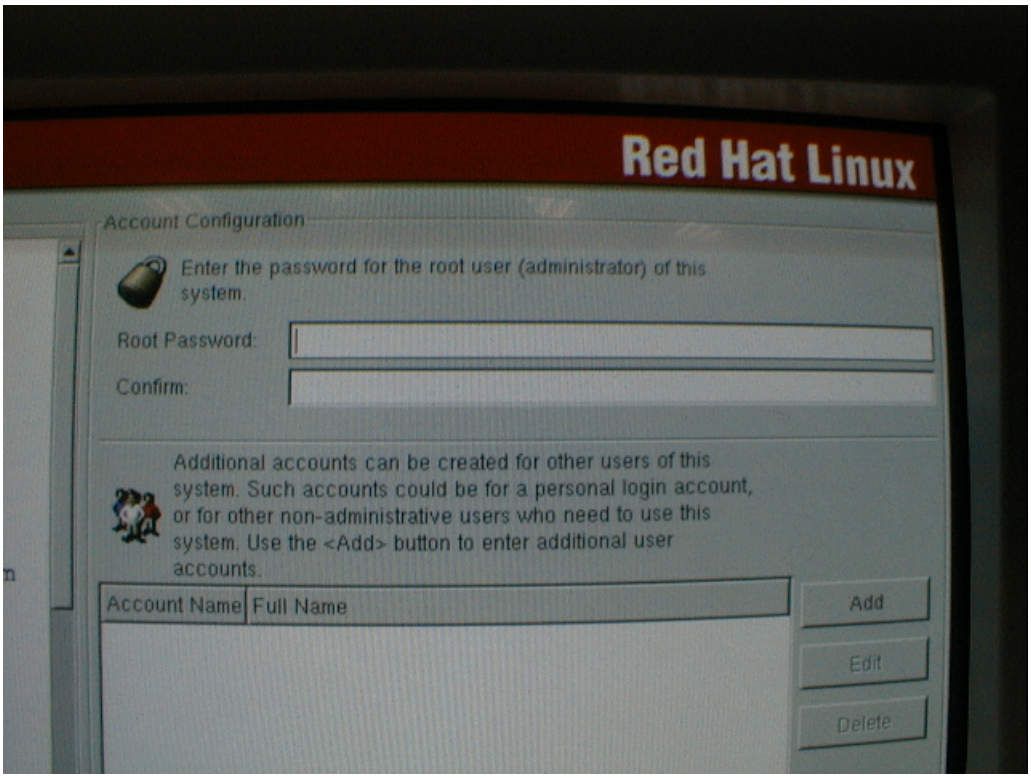


Time Zone

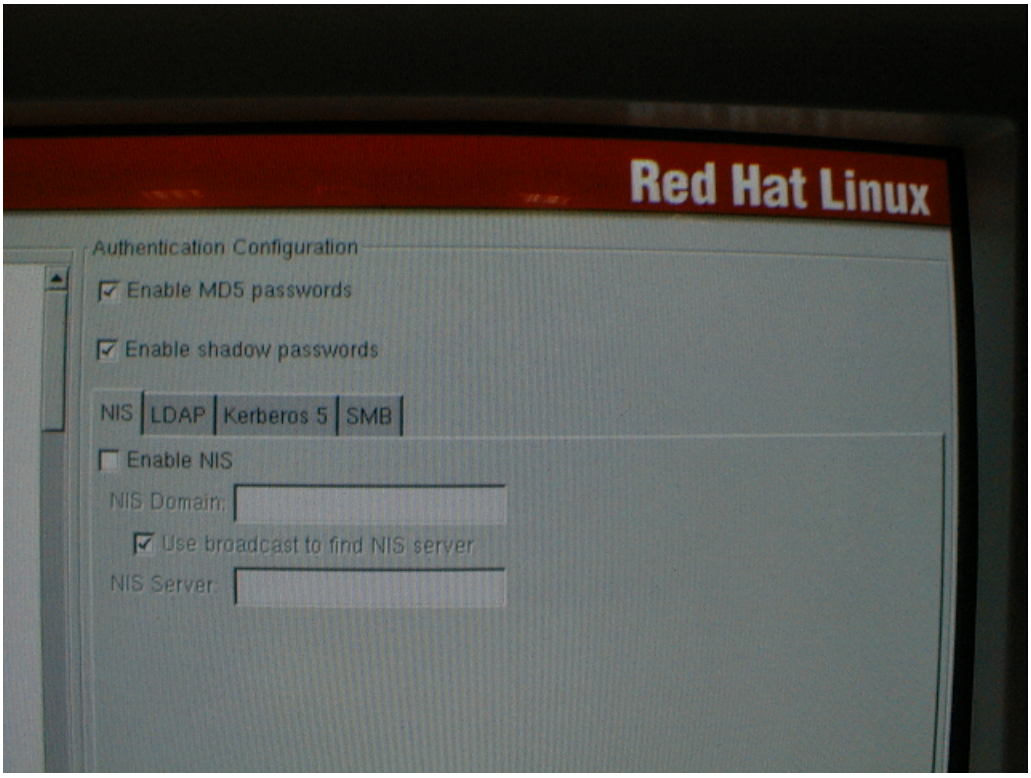
How To Configure Time Zone Support

Accounts And Passwords

Setting the root Password

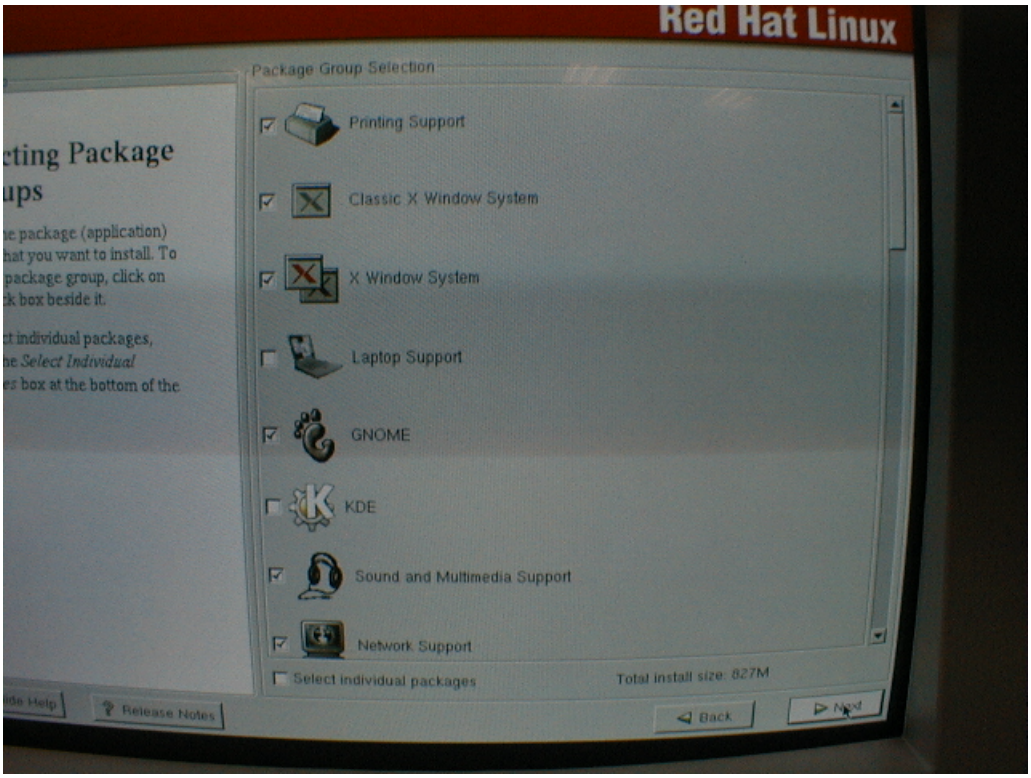


Enabling MD5 And Shadow Passwords



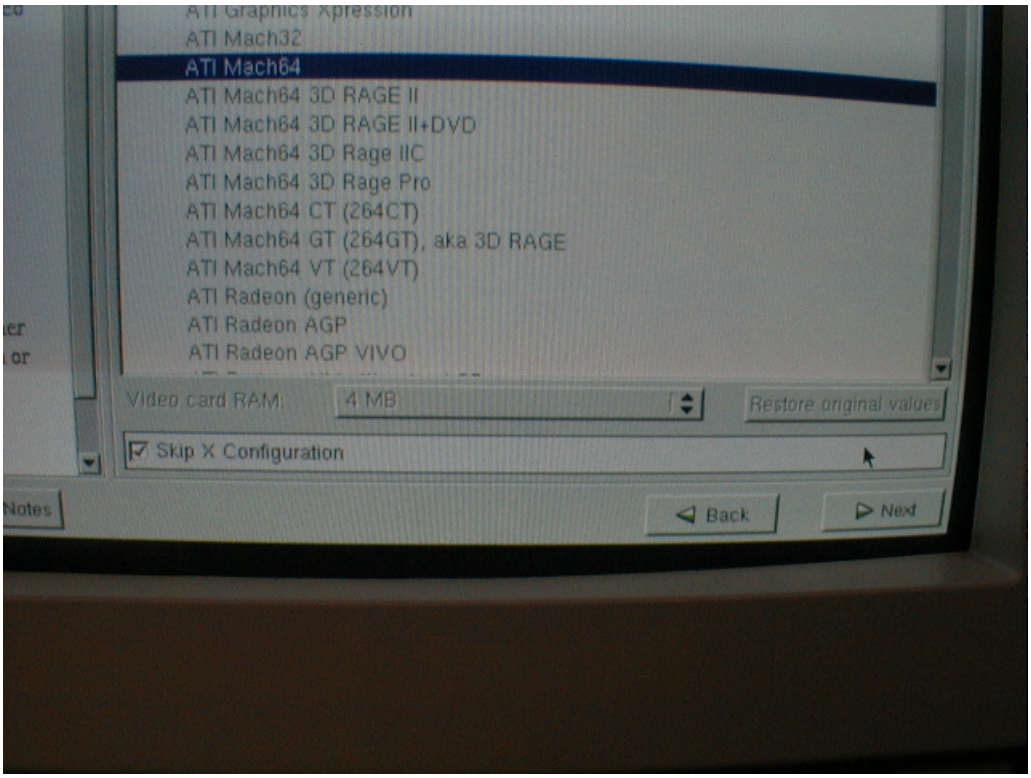
Subsystem Installation

Choosing which Subsystems To Install



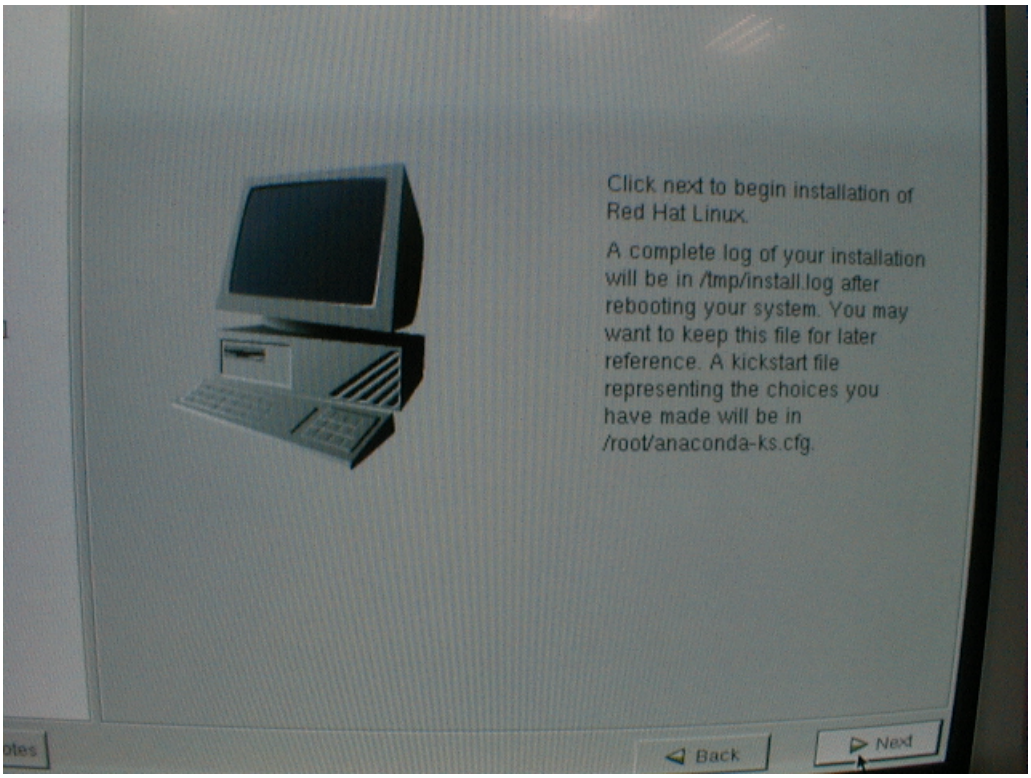
X-Windows

Skipping The X-Windows Setup



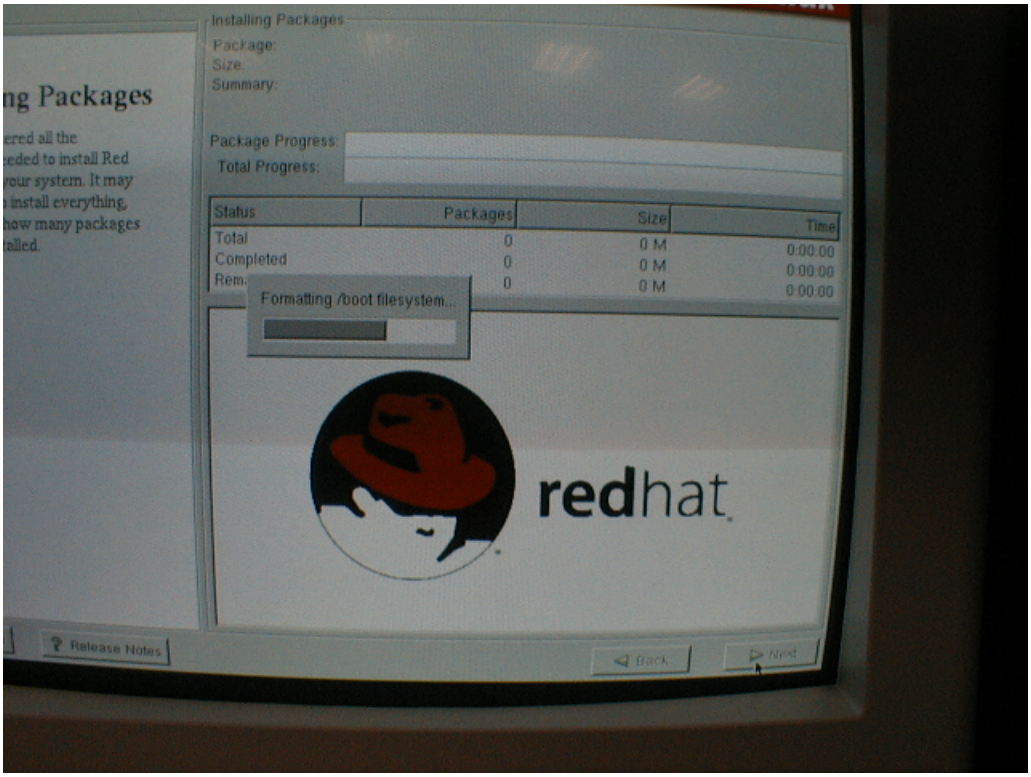
Initiating The Installation

Launching the Package Installation

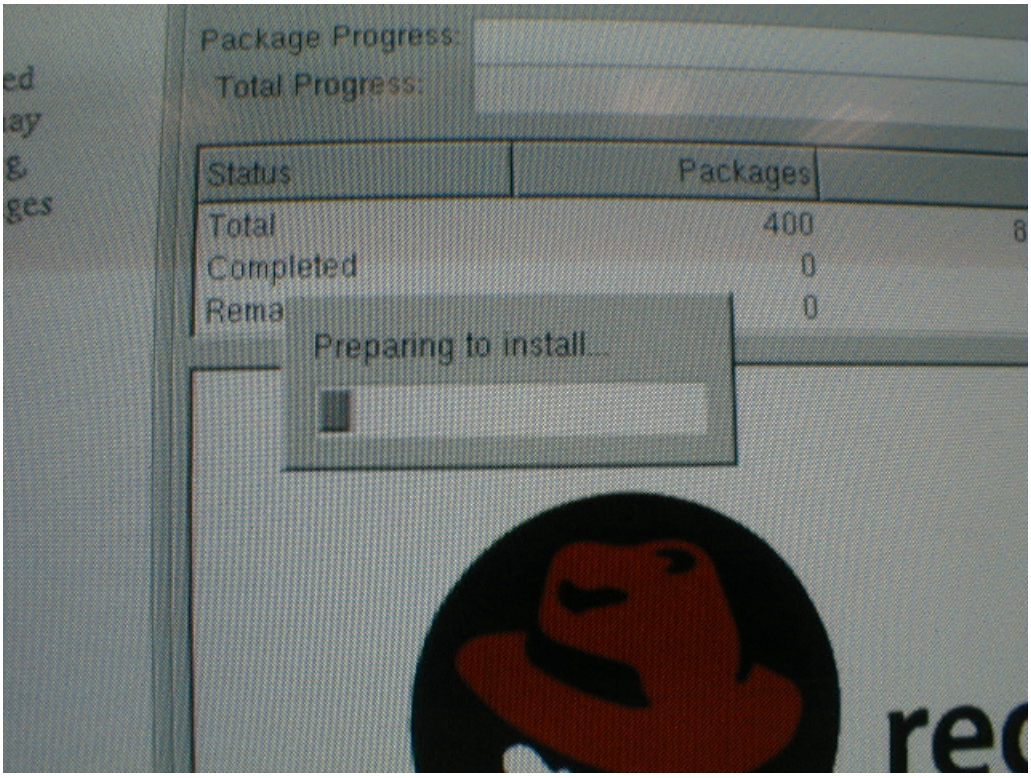


Watching The Package Installation Process

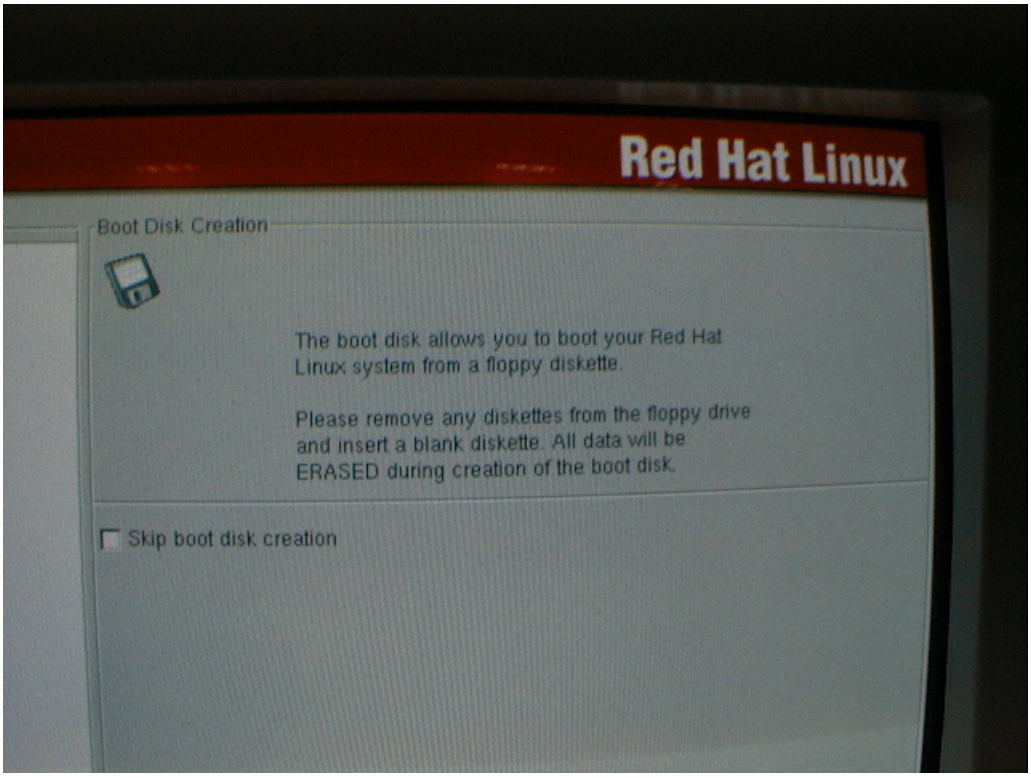
Formatting Partitions



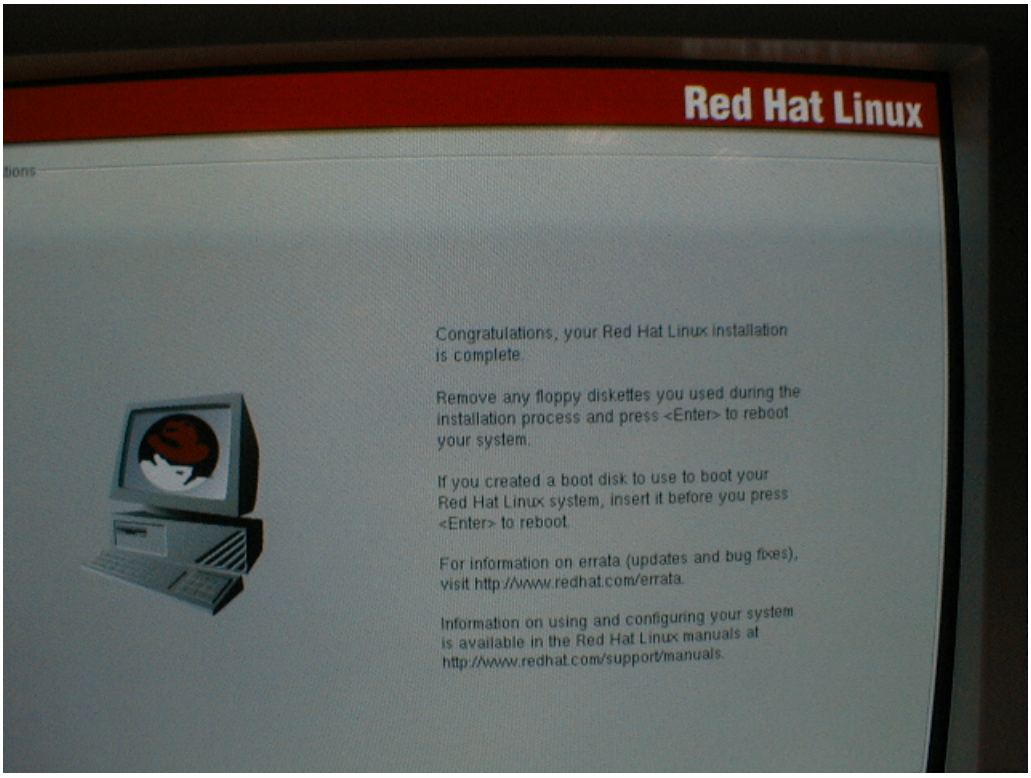
Watching The System Prepare The Installation



Creating the Red Hat Boot Disk



Concluding The Install

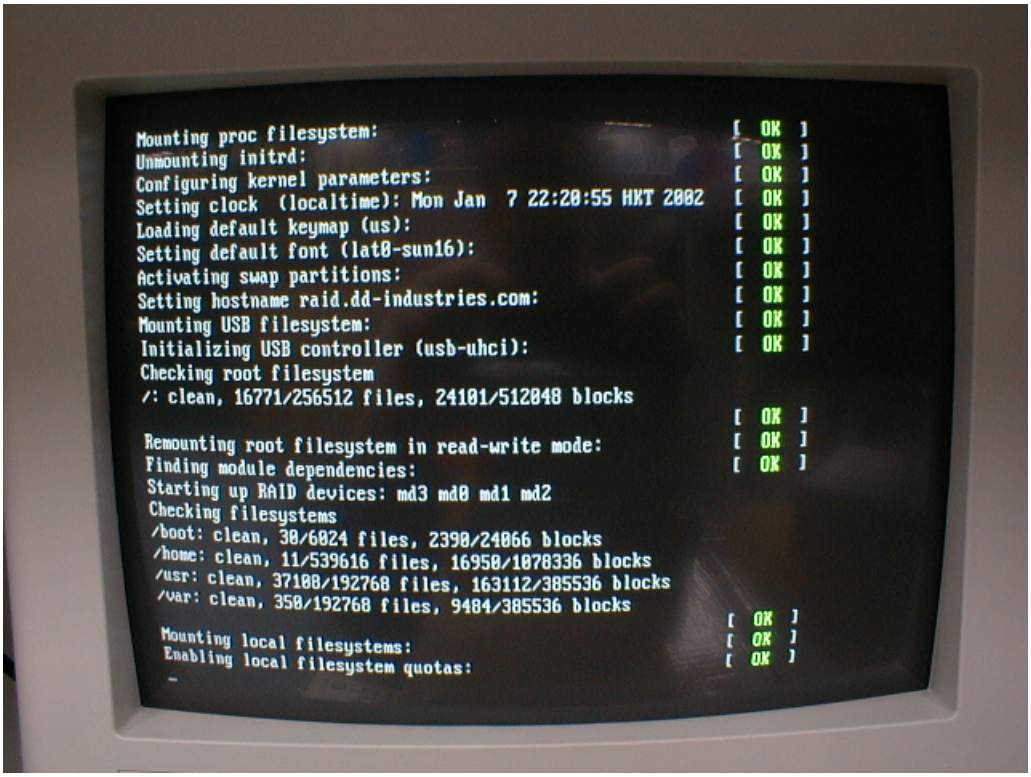


The First Boot

The First Post-Installation Boot

Watching The Kernel Boot For The First Time

Consider the following:

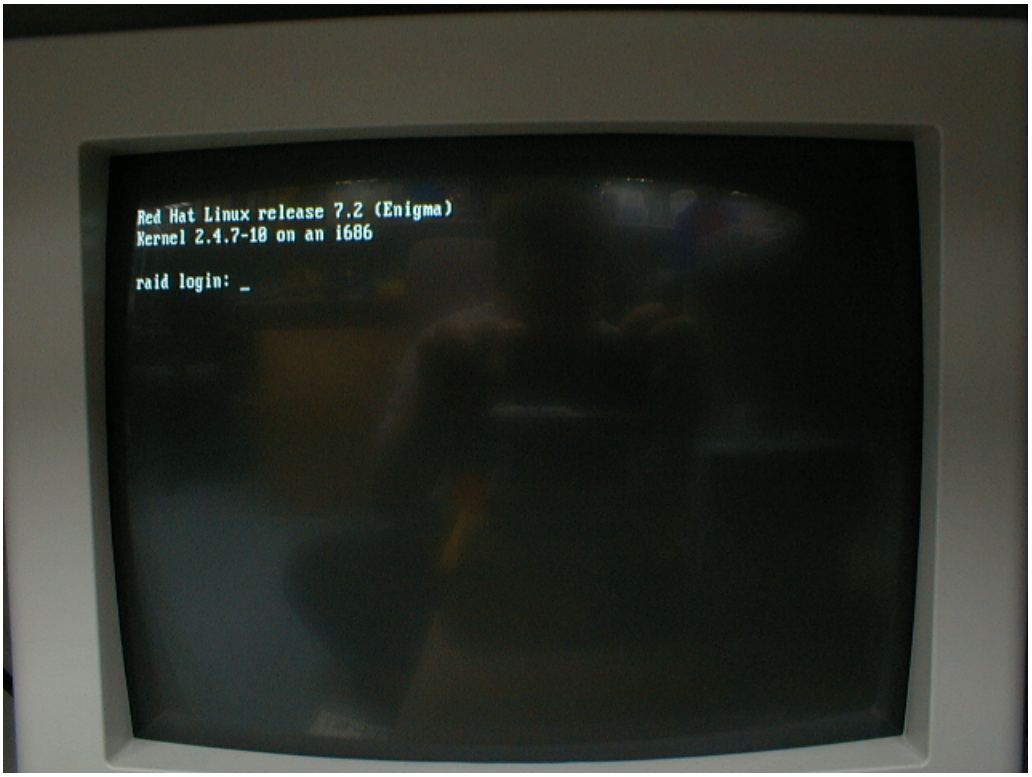


Note the RAID devices md3, md0, md1 and md2 coming up.

Note the partitions coming up clean.

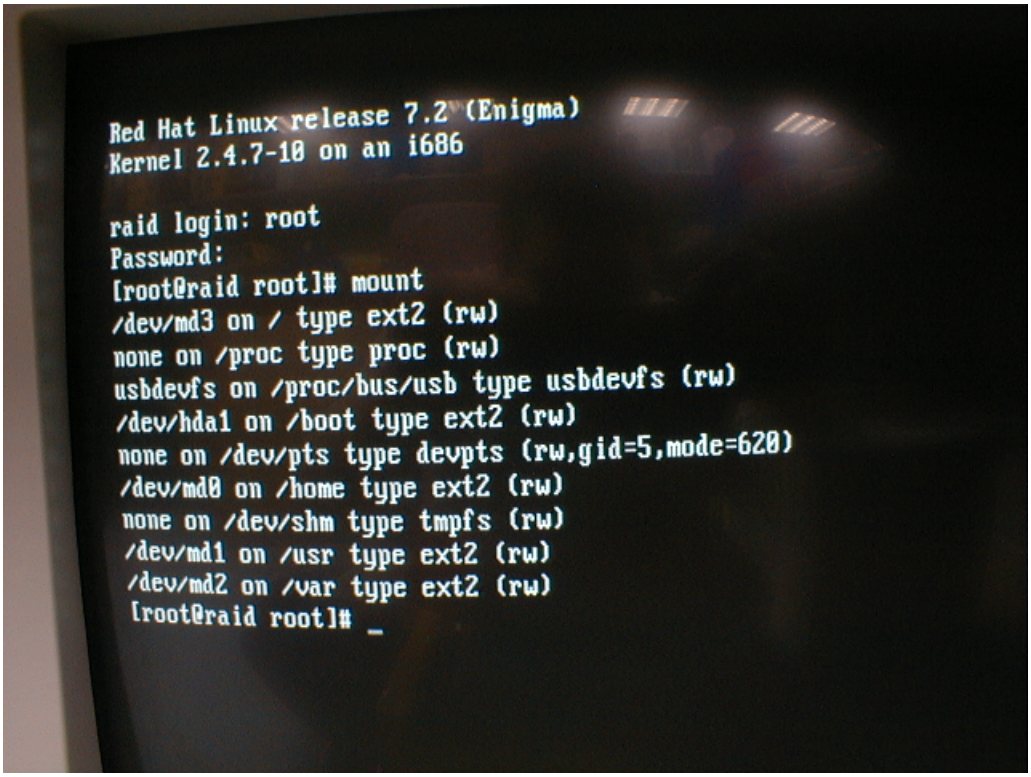
The Red Hat Login Screen

Consider the following:



Verifying the RAID Partitions

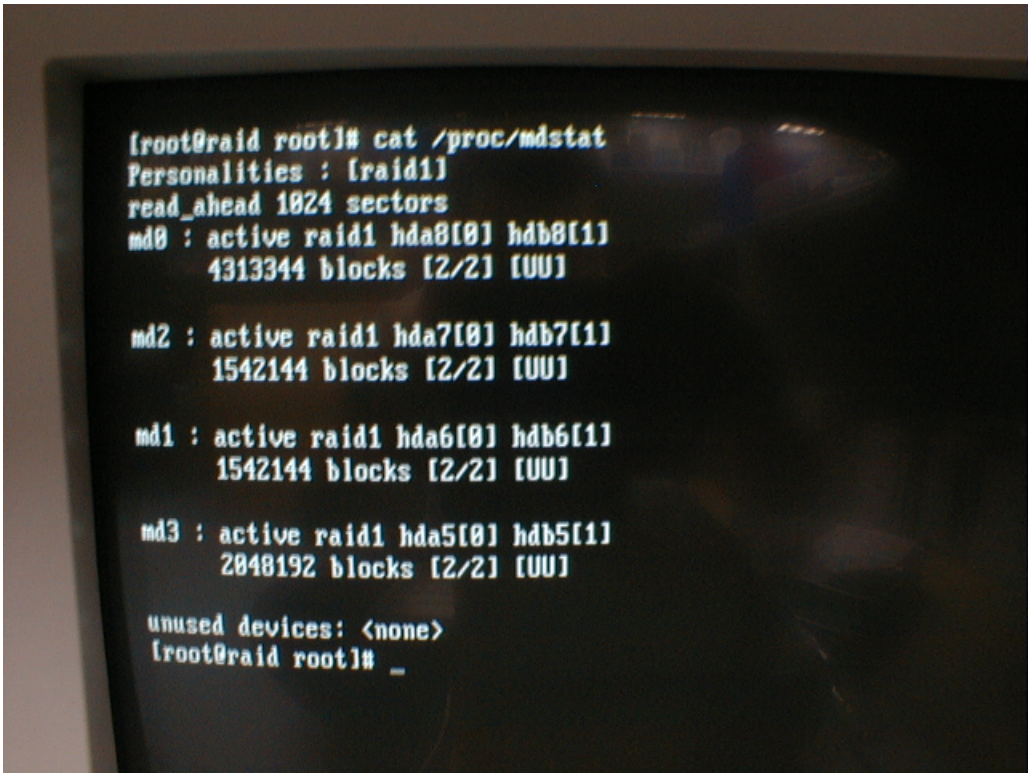
Consider the following:



In the above example the **mount** command was used to display the active partitions.

Viewing RAID Partition Status in /proc

Consider the following:



In the above example, the **cat** command was used to display the contents of the file **/proc/mdsat**, which holds the current status of the raid array.

Everything looks fine - looks like a working system.