System Administration

Matthew West

System Administration

by Matthew West

Published 2005-01-25 19:31:24

Table of Contents

List of Figures

This page intentionally left blank

List of Tables

This page intentionally left blank

Chapter 1. General Hints for System Administrators

Introduction

To define the responsibilities of an administrator 100% at this time would be impossible for us to do, the scope of this job would have to depend on the size of the system or network that you are working with, the numbers of users involved, the amount of resources available to you and the type of applications that the users would be using.

However one thing is for sure and that is that doing the work of a system administrator is fairly time consuming - look at the shell scripting course and get a good idea of how to automate some of the processes.

If we were to itemize the type of jobs that you may have to do the first would have to be to analyze the system(s) that you are working with and the users needs.

The second could very well be to install the operating system or to install some of the desktops and the third and maybe most important to automate would be to maintain the system that you have established.

In order to ensure that your foundation installation is solid check the following guidelines and fill in the missing information where required.

Remember that "Knowledge is power" and when installing a system the more knowledge you have the better the installation will be.

Reading list for administrators

The job of a system's administrator involves continually learning and keeping up to date with technology which affects their work. Thus, it's a good idea to have a handful of resources to refer to in order to track what's happening out there.

Some good web sites to become familiar with include:

http://www.freshmeat.net/

Lots of software, both new and updated releases, are announced here.

http://www.slashdot.org/

"News for Nerds. Stuff that Matters."

http://www.sage.org/

The System Administrator's Guild, affiliated with the Usenix Association. They have several nice booklets and a hard copy magazine, called ";login:", specifically for system administrators!

Installation and Configuration Checklist

To Install the hardware

1. Take a note of device compatibility
2. Take a note of the address information (the section called “Hardware Configuration and Compatibility” [5]).
3. Make sure that you know your system thoroughly e.g. How much memory do you have?

Prepare to install the operating system and software

1. Calculate the amount of disk space that you will need for each application or additional software that you are intending to install.
2. Also note in the installation guides for this additional software if they have recommendations for allocating kernel resources or installation tips.
3. Prepare the materials required for the installation (CDROM etc.)
4. Read any installation guide that is available to you

Install Operating System

1. Consider the defaults supplied throughout the installation process. Not all are applicable to your installation
2. Determine the hardware interrupts and install the devices
3. Follow the installation process carefully and make notes of all the settings that you configure and the decisions that you make, any error messages or areas of concern that we may have to go back to.

Configure the system and software

1. Look at the security requirements that you are going to have and implement the relevant solutions.
2. Install the application software and make changes to resources if required.
3. Install users, passwords and make other changes to the system profile files.
4. Follow the process in chapter two for post installation configuration.
5. Verify your set-up.
6. Make a backup of the clean system.
7. Create a system log.

On-going

1. Follow system resources versus system needs and change if affecting

performance

2. Create and keep organizing the filesystems and files
3. Manage the users processes
4. Administrate and monitor system security and closely follow the log files
5. Maintain the network and access to the Internet if required
6. Educate the users and hand over some responsibility e.g. log in and lo out procedures and appoint/train a junior printer administrator or backup administrator or user administrator per floor or division.

Administrators file summary

If you've been put in charge of a system which was previously looked after by someone else, there are several files and commands with which it is a good idea to familiarize yourself, so that you can tell how the system has been set up.

Obviously, most of the files in the **`/etc`** directory fall into this category, specifically **`/etc/passwd`** and the associated **`/etc/shadow`** file, as well as **`/etc/group`**. On a Debian system, you should also familiarize yourself with a list of the currently installed packages and their versions; you can use the **dpkg -list** command to achieve this. This command is covered in more detail later.

Keeping a record of your system

System Log

Record all system modifications and other events

Date	System Modification	System Event
.	.	.

Diagnose any system problems

Date	System Problem	Solution used
.	.	.

Make a note of system usage and any growth

Date	Description
.	.

Make a note of the security implementation and its success, remember not to keep these records in an unsafe place.

Hardware Configuration and Compatibility

List your system hardware components and as suggested in the course material, check the hardware's compatibility with the operating system that you are installing.

For each hardware component record the following information:

Item	.
Manufacturer	.
Serial #	.
DMA	.
IRQ	.
Base Address	.
Description (any further relevant driver information)Is it compatible with the operating system that you are installing and where did you find the compatibility information. Anything to be careful of in future or in the case of a re-install?	.

Software Checklist

Application	Description	
Software-Name	Short description of the installation, problems, resources allocated, serial numbers, licenses and any other pertinent information.	

Backup Log

Decide on your backup procedure whether full or incremental, daily, weekly etc. Then create a log that you can refer to especially at restore time or if handing over the system to someone else (see Chapter 4 [107]).

A review of some of basic commands

Check in the man or info pages if you do not know these commands:

For managing files we covered **ls**, **mv**, **cp**, **rm**, **grep**, **file** and **find**.

We displayed files with **cat**, **more**, **pg**, **less**, **head** and **tail**.

Managed directory files with **cd**, **mkdir**, **rmdir**, **pwd** and **copy**.

Used input/output redirection with >, <, >> and |

Download the vi-referance card from this [http://resources/vi-ref.pdf] page

Chapter 2. Installation and Bootup

Installation

Planning and Preparation

In this course we are going to go through a complete installation of Debian Linux. You will require at least the first Debian installation CD to do a Debian installation.

Only the first CD is required to turn your computer into a fully working Linux system, although there are lots of useful packages on the other 7 (!) CDs.

You should consult the Debian web site[1] for a list of places where you can get your hands on a copy.

This document covers Debian version 3.0r2, but is applicable to most other versions.

You will need to configure your system's CMOS to tell it that you wish to boot off the CD-ROM drive.

Before attempting the installation make an inventory of the hardware components of your machine. Although the Debian installation system will attempt to automatically detect your hardware, it may not always be successful. Probably the most important pieces of information to record are the IRQ and IO addresses for your ISA/EISA cards, if you have any.

If you're going to be doing the installation onto a machine, which already contains data, and you wish to preserve this data then make a backup!

Debian requires that there be some unused partition space on which it can be installed. Please note that this is not the same as having free space on your C: drive (assuming you're running DOS or Windows).

A typical desktop Debian installation requires about 600MB of disk space, but this can obviously change depending on what additional software you intend to install, as well as the amount of data that you are going to be storing.

It is highly recommended that you also review the installation documentation available on the Debian CDs, as this will cover specific configuration problems in depth, as well as last minute "gotchas" that might have crept into the release.

Debian Linux is a community driven operating system like most open source projects. A way to get the most out of using the software is to become involved with the community. If you're running a Debian system, it is definitely recommended that

[1] www.debian.org [http://debian.org]

you subscribe to some of the Debian mailing lists, you can subscribe to these mailing lists on this [http://lists.debian.org/] page

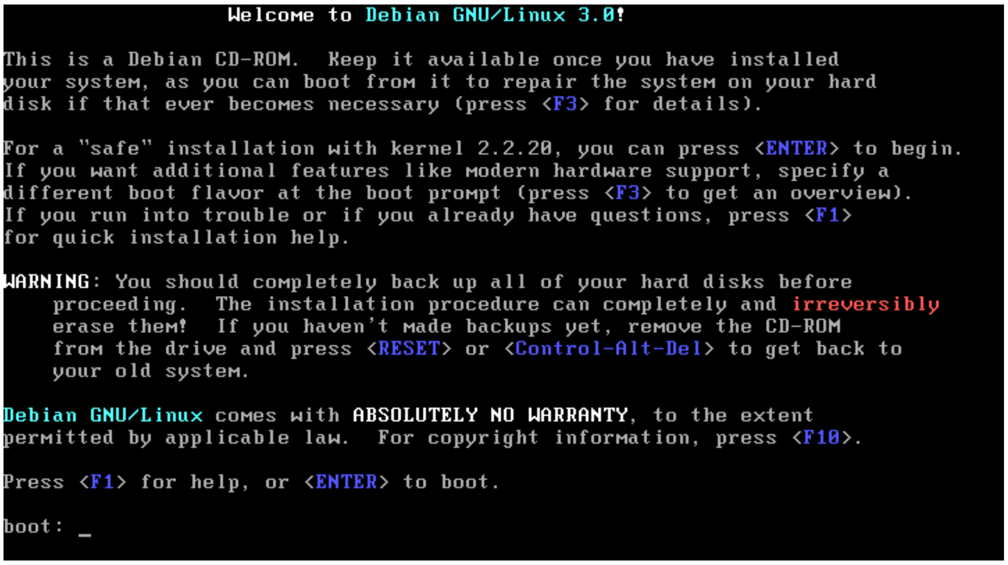
As someone looking after a Debian system, you should at least subscribe to "announce" and "security announce" mailing lists. These are low volume, but they will keep you up to date with what's happening with the Debian project, and, very importantly, about any security related information that becomes available.

During an installation of any operating system, it's a good idea to note down what it is you've done and what choices you've made, and possibly even your reasons. This can prove invaluable when reviewing a system, and especially if you have to re-install it due to a system failure!

Some of the other obvious things that you should do prior to an installation are to make an inventory of the hardware that the system contains, and then to check up online whether or not that specific hardware combination is supported by the operating system and applications that you are going to be installing.

Starting the Installation

Once you've booted off the installation CD, your screen should appear as follows:



```
                  Welcome to Debian GNU/Linux 3.0!

This is a Debian CD-ROM.  Keep it available once you have installed
your system, as you can boot from it to repair the system on your hard
disk if that ever becomes necessary (press <F3> for details).

For a "safe" installation with kernel 2.2.20, you can press <ENTER> to begin.
If you want additional features like modern hardware support, specify a
different boot flavor at the boot prompt (press <F3> to get an overview).
If you run into trouble or if you already have questions, press <F1>
for quick installation help.

WARNING: You should completely back up all of your hard disks before
    proceeding.  The installation procedure can completely and irreversibly
    erase them!  If you haven't made backups yet, remove the CD-ROM
    from the drive and press <RESET> or <Control-Alt-Del> to get back to
    your old system.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.  For copyright information, press <F10>.

Press <F1> for help, or <ENTER> to boot.

boot: _
```

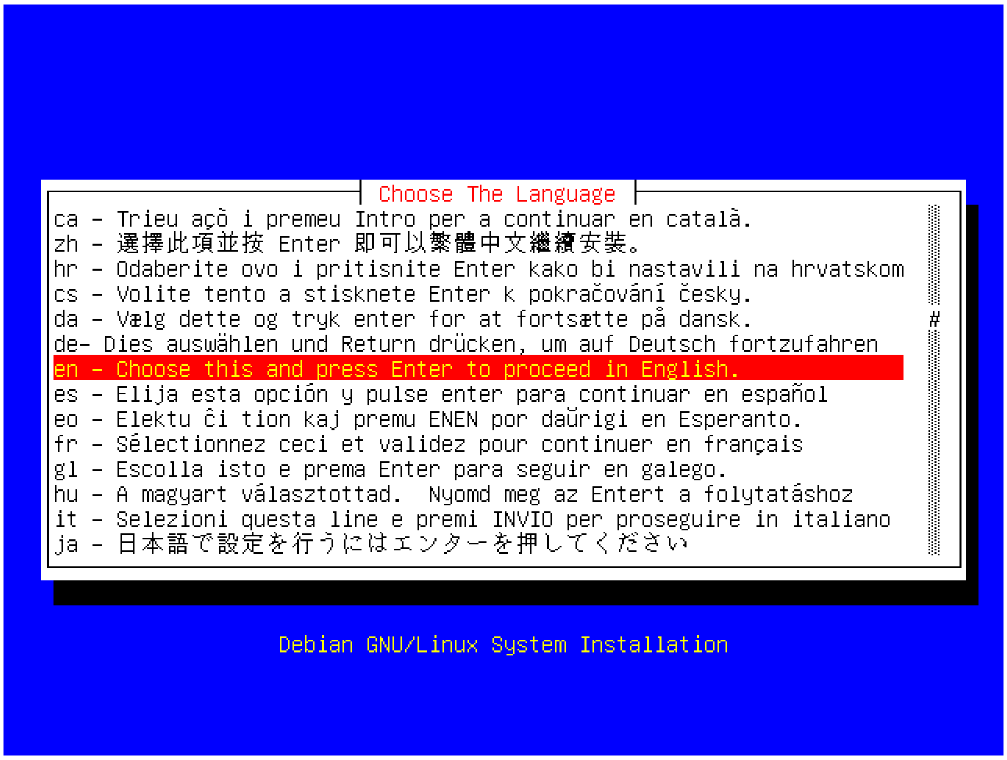
Note that it is important to read and note everything that is displayed during the installation. If you skim over something, you may miss a critical piece of information and end up having to start the installation from scratch again!

Once you've read what's on the screen, feel free to use the function keys to investigate the other screens. Once you're ready to begin press **Enter**.

You should notice a lot of text scroll off the screen. Don't worry if it goes past too quickly for you to read, we can come back to it later.

Language and "dbootstrap"

If all goes well while booting off the CD, you should be presented with the following screen:



```
Choose The Language
ca - Trieu açò i premeu Intro per a continuar en català.
zh - 選擇此項並按 Enter 即可以繁體中文繼續安裝。
hr - Odaberite ovo i pritisnite Enter kako bi nastavili na hrvatskom
cs - Volite tento a stisknete Enter k pokračování česky.
da - Vælg dette og tryk enter for at fortsætte på dansk.
de- Dies auswählen und Return drücken, um auf Deutsch fortzufahren
en - Choose this and press Enter to proceed in English.
es - Elija esta opción y pulse enter para continuar en español
eo - Elektu ĉi tion kaj premu ENEN por daŭrigi en Esperanto.
fr - Sélectionnez ceci et validez pour continuer en français
gl - Escolla isto e prema Enter para seguir en galego.
hu - A magyart választottad.  Nyomd meg az Entert a folytatáshoz
it - Selezioni questa line e premi INVIO per proseguire in italiano
ja - 日本語で設定を行うにはエンターを押してください

Debian GNU/Linux System Installation
```

The application that is now running is called "dbootstrap"; it is responsible for the installation and initial configuration of the system.

You can use the following keys to navigate in dbootstrap:

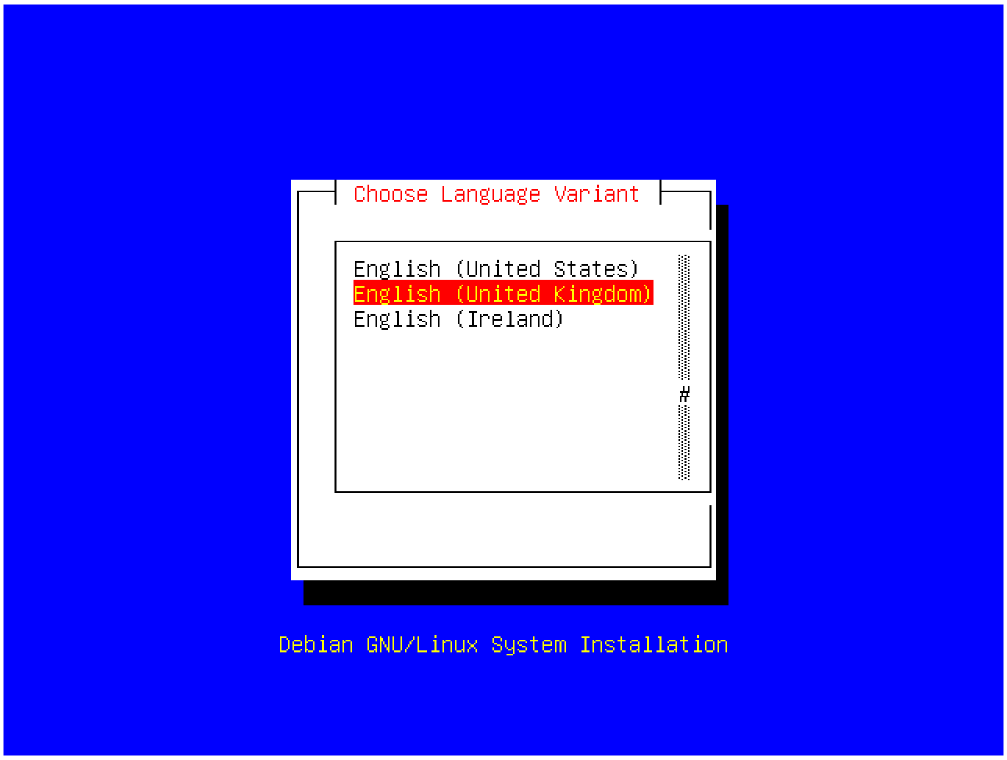
Table 2.1. Navigation Keys(dbootstrap)

right arrow, or **Tab**	move forward between buttons and selections
left arrow, or **Shift-Tab**	move backward between buttons and selections
Up/Down arrow	highlight items within a scroll list
Spacebar	select an item, such as a check box

Enter	activate current choice
Alt-F1	view dbootstrap screen
Alt-F2	view virtual console
Alt-F3	view error messages
Alt-F4	package unpacking and setup messages

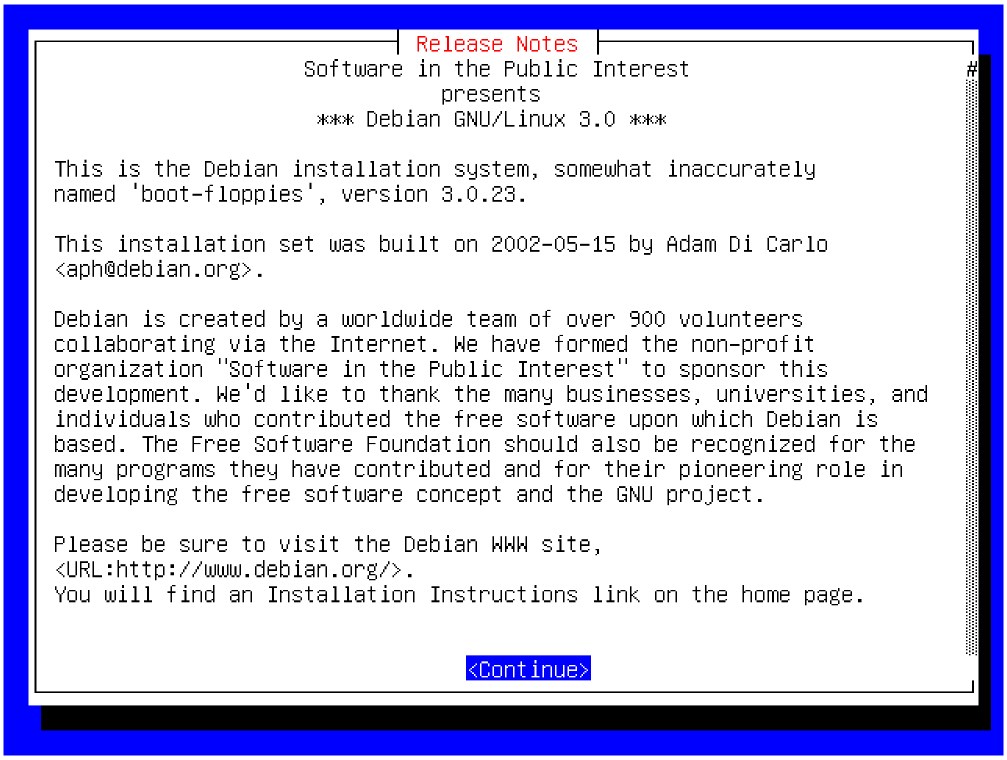
Now that you know how to find your way around, use the arrow keys to move the selection bar (the yellow text on a red background) down to your preferred language choice (probably "en"). Once your choice is highlighted, as in the screenshot, press the **Enter** key.

You may be prompted to further refine your choice, as below:



```
Choose Language Variant

English (United States)
English (United Kingdom)
English (Ireland)

Debian GNU/Linux System Installation
```

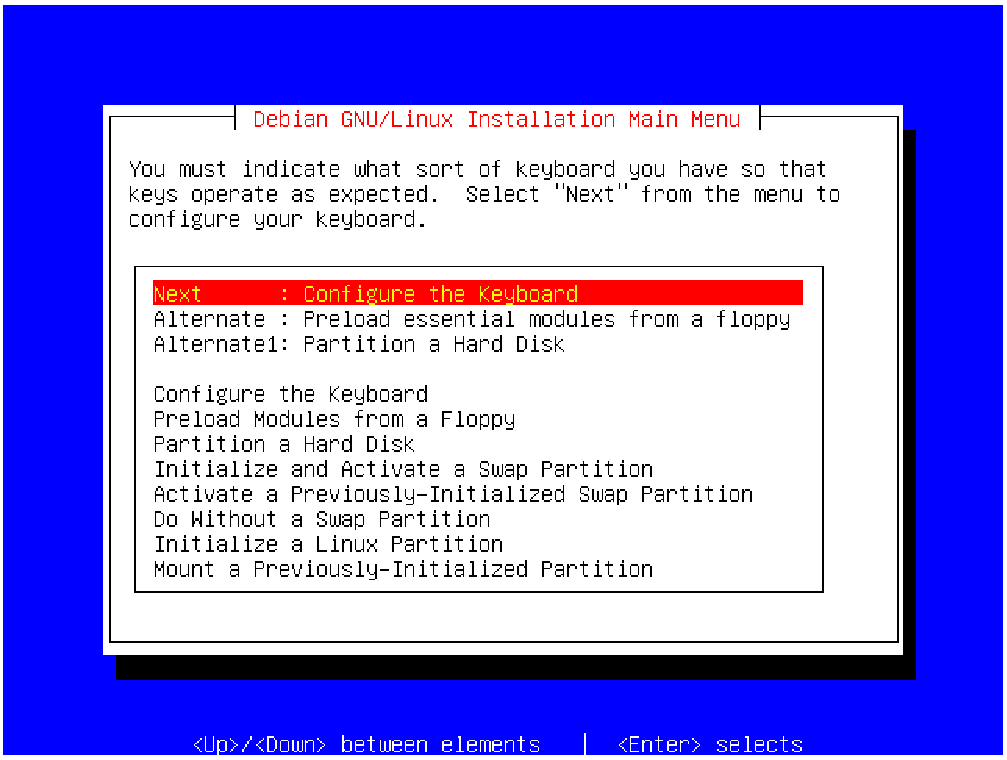
Once you've done that successfully, you will be presented with the "Release Notes" display:



```
Release Notes
                   Software in the Public Interest
                              presents
                     *** Debian GNU/Linux 3.0 ***

This is the Debian installation system, somewhat inaccurately
named 'boot-floppies', version 3.0.23.

This installation set was built on 2002-05-15 by Adam Di Carlo
<aph@debian.org>.

Debian is created by a worldwide team of over 900 volunteers
collaborating via the Internet. We have formed the non-profit
organization "Software in the Public Interest" to sponsor this
development. We'd like to thank the many businesses, universities, and
individuals who contributed the free software upon which Debian is
based. The Free Software Foundation should also be recognized for the
many programs they have contributed and for their pioneering role in
developing the free software concept and the GNU project.

Please be sure to visit the Debian WWW site,
<URL:http://www.debian.org/>.
You will find an Installation Instructions link on the home page.
```

Once you've read this, press **Enter** again to continue.

Main Menu



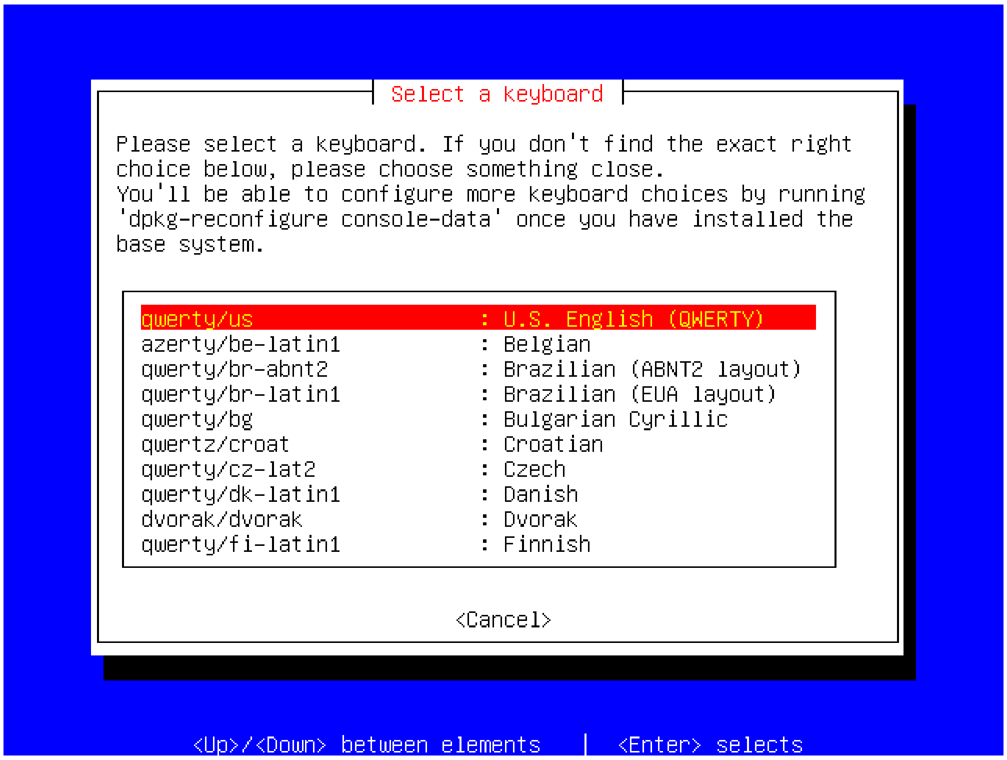
```
Debian GNU/Linux Installation Main Menu

You must indicate what sort of keyboard you have so that
keys operate as expected.  Select "Next" from the menu to
configure your keyboard.

Next       : Configure the Keyboard
Alternate  : Preload essential modules from a floppy
Alternate1: Partition a Hard Disk

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition
Do Without a Swap Partition
Initialize a Linux Partition
Mount a Previously-Initialized Partition

<Up>/<Down> between elements   |  <Enter> selects
```

From here, you can jump to most parts of the installation. For the purposes of a standard installation though, you should simply concentrate on the top grouped choices; in this case **Next**, "Alternate" and "Alternate1".

The **Next** choice is the next logical step in the installation procedure, and is usually always the one you will want. The "Alternate" options allow you to chose an alternative installation path. This is sometimes useful if you want to skip a section which doesn't make sense for your particular installation.

Make sure that the highlight bar is on "Next: Configure the Keyboard", and then press the **Enter** key.

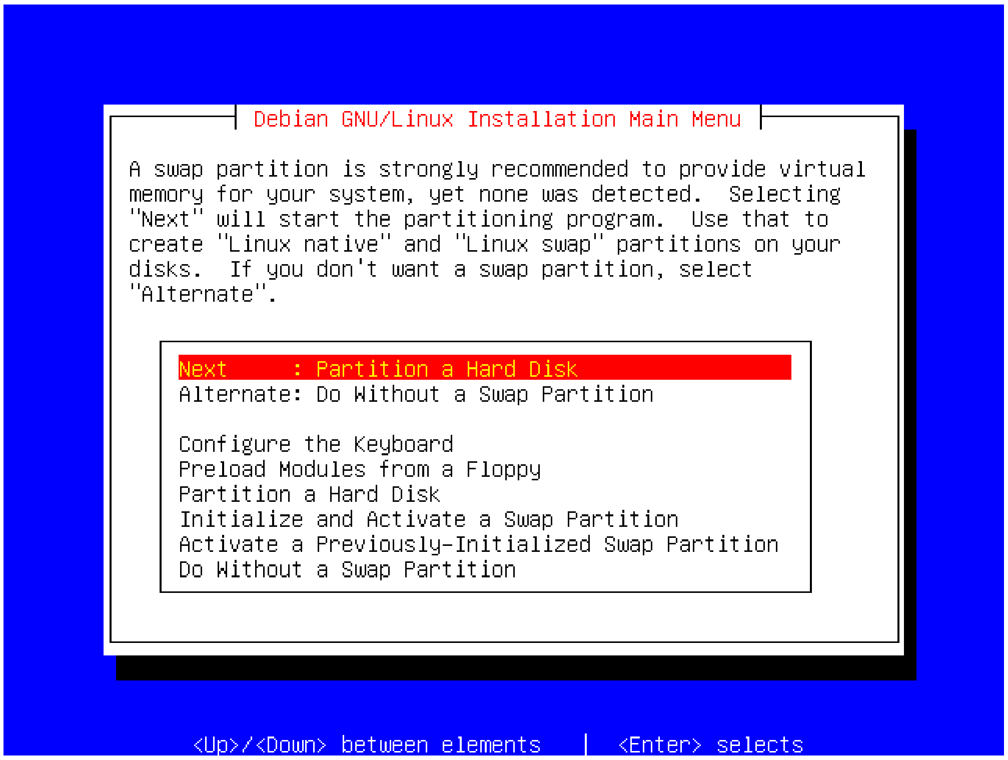
Select a keyboard



```
Select a keyboard

Please select a keyboard. If you don't find the exact right
choice below, please choose something close.
You'll be able to configure more keyboard choices by running
'dpkg-reconfigure console-data' once you have installed the
base system.

qwerty/us                  : U.S. English (QWERTY)
azerty/be-latin1           : Belgian
qwerty/br-abnt2            : Brazilian (ABNT2 layout)
qwerty/br-latin1           : Brazilian (EUA layout)
qwerty/bg                  : Bulgarian Cyrillic
qwertz/croat               : Croatian
qwerty/cz-lat2             : Czech
qwerty/dk-latin1           : Danish
dvorak/dvorak              : Dvorak
qwerty/fi-latin1           : Finnish

                     <Cancel>

<Up>/<Down> between elements   |   <Enter> selects
```

Select your keyboard type, and then press Enter. If you're not sure which keyboard you have, then go with the default "qwerty/us" selection.

Partition hard disk



```
Debian GNU/Linux Installation Main Menu

A swap partition is strongly recommended to provide virtual
memory for your system, yet none was detected.  Selecting
"Next" will start the partitioning program.  Use that to
create "Linux native" and "Linux swap" partitions on your
disks.  If you don't want a swap partition, select
"Alternate".

Next      : Partition a Hard Disk
Alternate: Do Without a Swap Partition

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition
Do Without a Swap Partition

<Up>/<Down> between elements   |  <Enter> selects
```

Debian needs at least one partition in order to run. However, you can also create a swap partition.

Why create a swap partition?

A swap partition allows the operating system to treat some of the disk space as "virtual memory". If your system has less than 16MB of RAM, then you will have to create a swap partition in order for the system to work. If you have more RAM than that, then a swap partition is not required, but is still recommended.

Generally, the cost of RAM is a lot higher than the cost of disk space. For this reason, it is often useful to be able to use part of the hard disk as if it were memory. This is what a swap partition allows the operating system to do. The kernel will be able to "swap" current unneeded data out of RAM and write it to the swap partition on the disk. When the data is needed by the kernel again, it will be read off the disk, and placed back into RAM.

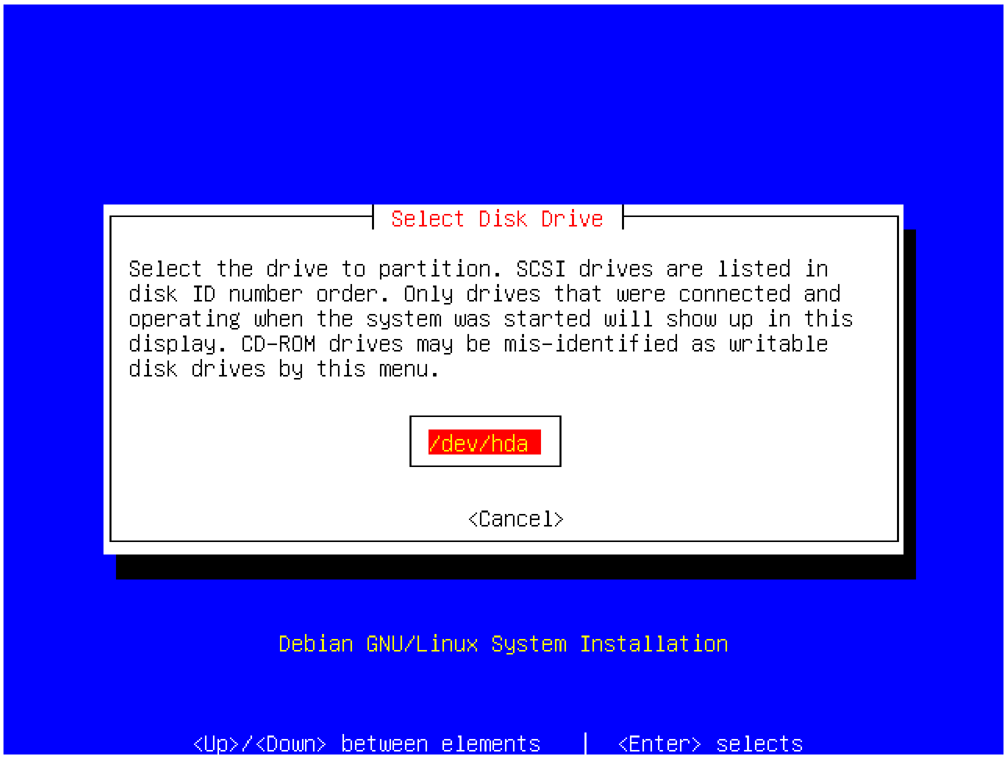
The benefit is that you get away with not having to have the full amount of physical RAM required to run a specific application. One important point to remember though is that accessing the hard drive is a lot slower than accessing RAM, and so you will incur a speed penalty.

A good rule of thumb is to have the amount of swap that you have equal to twice the amount of RAM that you have, but no more than 1GB. Obviously, this might vary depending on what you're wanting to use the system for.

In our example, we're going to create a 64MB swap partition.

Select "Partition hard disk" and then press **Enter**.

You will then be prompted to select which hard disk you wish to partition:



```
Select Disk Drive

Select the drive to partition. SCSI drives are listed in
disk ID number order. Only drives that were connected and
operating when the system was started will show up in this
display. CD-ROM drives may be mis-identified as writable
disk drives by this menu.

/dev/hda

<Cancel>

Debian GNU/Linux System Installation

<Up>/<Down> between elements   |  <Enter> selects
```

Hard drive device names

The Linux kernel assigns short names to the storage devices that it finds on your system. It's important to know what naming scheme it uses, so that you can correctly chose which disks to partition:

Table 2.2. Disk Device Names

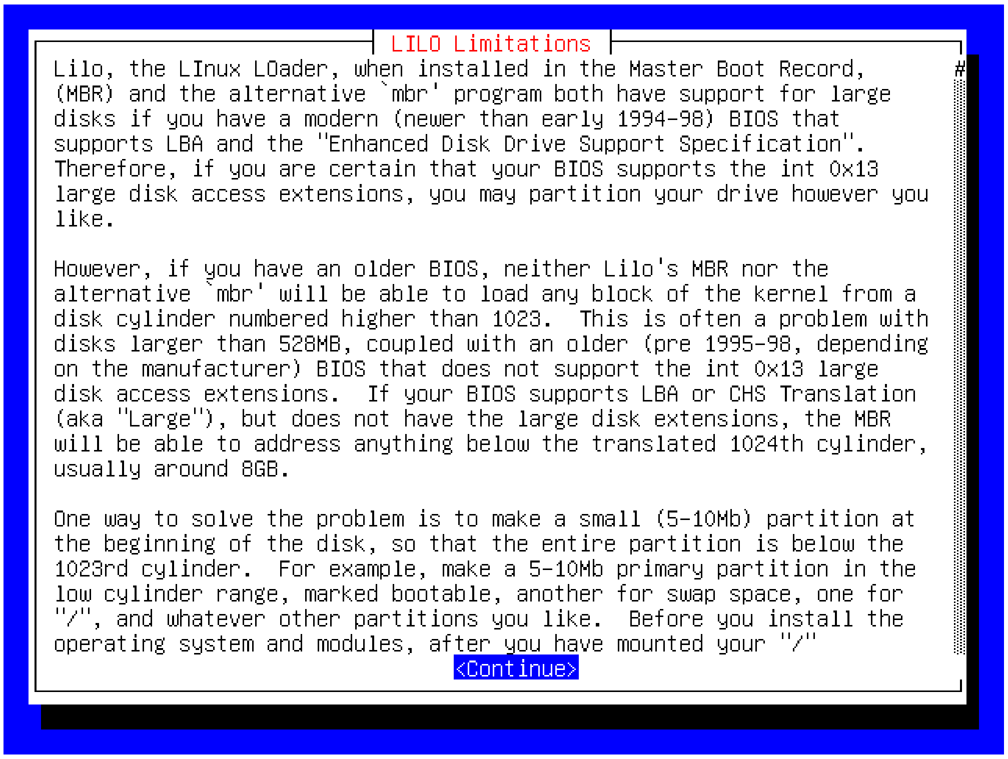
device	description
hda	primary master IDE hard disk
hdb	primary slave IDE hard disk
hdc	secondary master IDE hard disk

hdd	secondary slave IDE hard disk
sd0	first SCSI hard disk
xda	first XT disk
fd0	first floppy drive
scd0	first SCSI CD-ROM

In our example, there is only one drive installed in the system; a primary master IDE drive (hda).

Press **Enter** to select this drive, and continue.

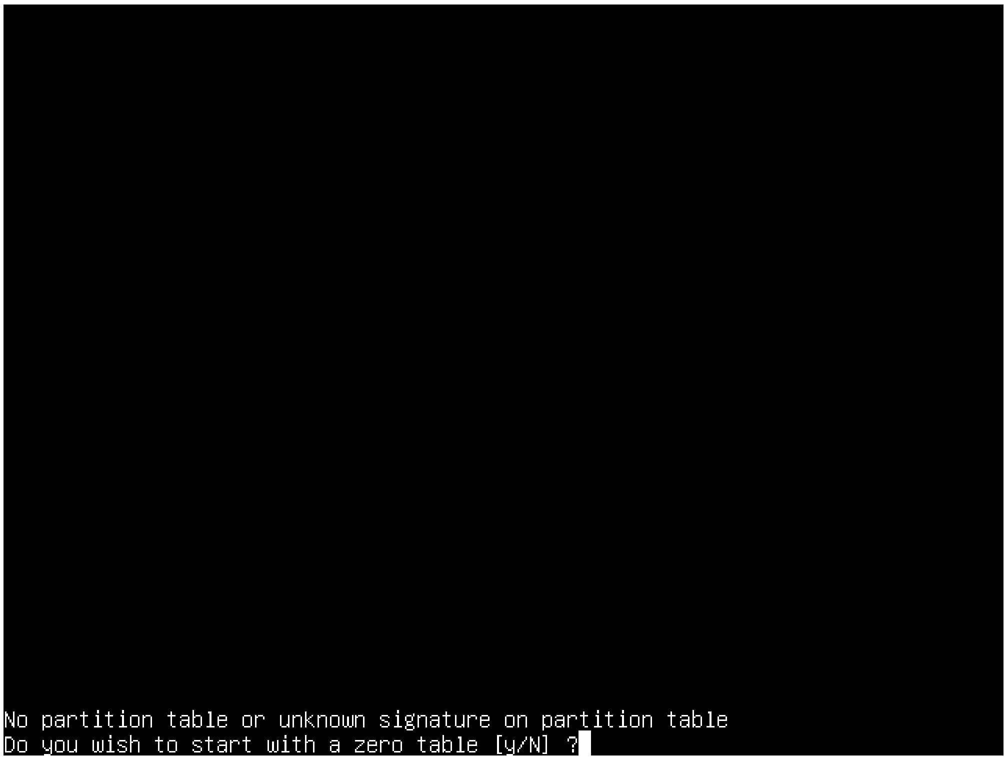
Large disks in older machines



```
LILO Limitations
Lilo, the LInux LOader, when installed in the Master Boot Record,
(MBR) and the alternative `mbr' program both have support for large
disks if you have a modern (newer than early 1994-98) BIOS that
supports LBA and the "Enhanced Disk Drive Support Specification".
Therefore, if you are certain that your BIOS supports the int 0x13
large disk access extensions, you may partition your drive however you
like.

However, if you have an older BIOS, neither Lilo's MBR nor the
alternative `mbr' will be able to load any block of the kernel from a
disk cylinder numbered higher than 1023.  This is often a problem with
disks larger than 528MB, coupled with an older (pre 1995-98, depending
on the manufacturer) BIOS that does not support the int 0x13 large
disk access extensions.  If your BIOS supports LBA or CHS Translation
(aka "Large"), but does not have the large disk extensions, the MBR
will be able to address anything below the translated 1024th cylinder,
usually around 8GB.

One way to solve the problem is to make a small (5-10Mb) partition at
the beginning of the disk, so that the entire partition is below the
1023rd cylinder.  For example, make a 5-10Mb primary partition in the
low cylinder range, marked bootable, another for swap space, one for
"/", and whatever other partitions you like.  Before you install the
operating system and modules, after you have mounted your "/"
```

If you have a large HDD, but an old machine, you may encounter problems booting a Linux system. Please read the notes on this screen carefully. If your machine is from 1998 or later, then you shouldn't be affected by this problem. Press **Enter** to continue.

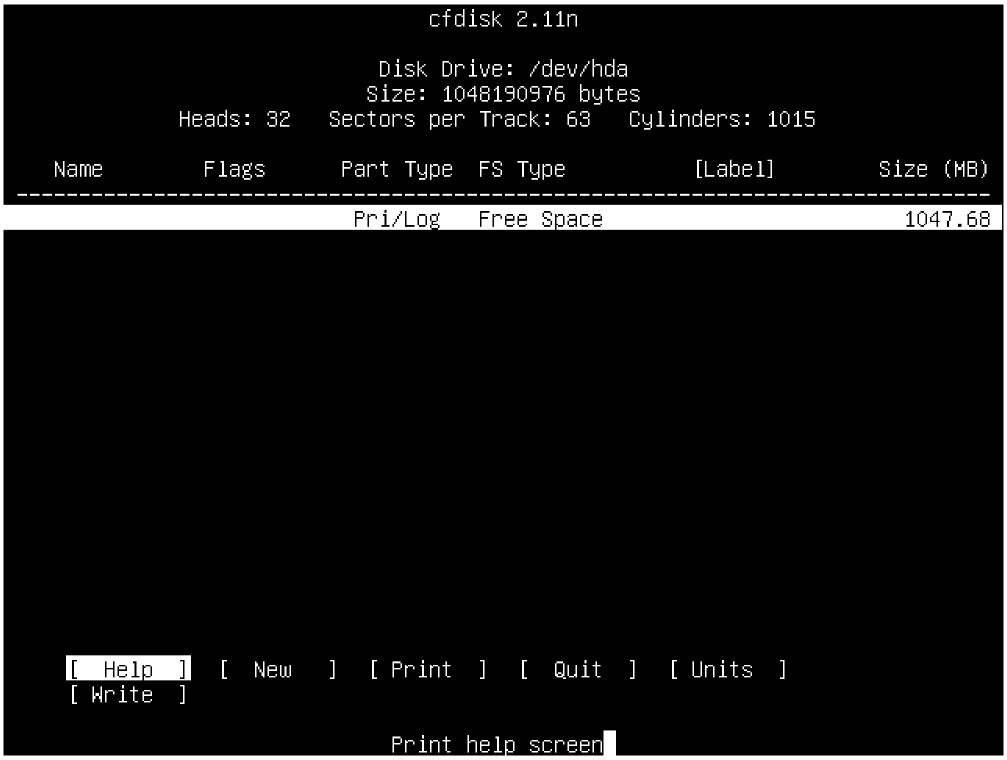
cfdisk



```
No partition table or unknown signature on partition table
Do you wish to start with a zero table [y/N] ?
```

The dbootstrap application has now launched another application called "cfdisk". This is the actual tool that you will use to partition your hard drive. If you are installing onto a new disk, you will be presented with a prompt as above. Simply press "Y" and then **Enter** to continue.

You will then be presented with the cfdisk main screen, which should look like this:



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                            Pri/Log   Free Space                        1047.68



















     [  Help  ]  [  New   ]  [ Print  ]  [  Quit  ]  [ Units  ]
     [ Write  ]

                               Print help screen
```

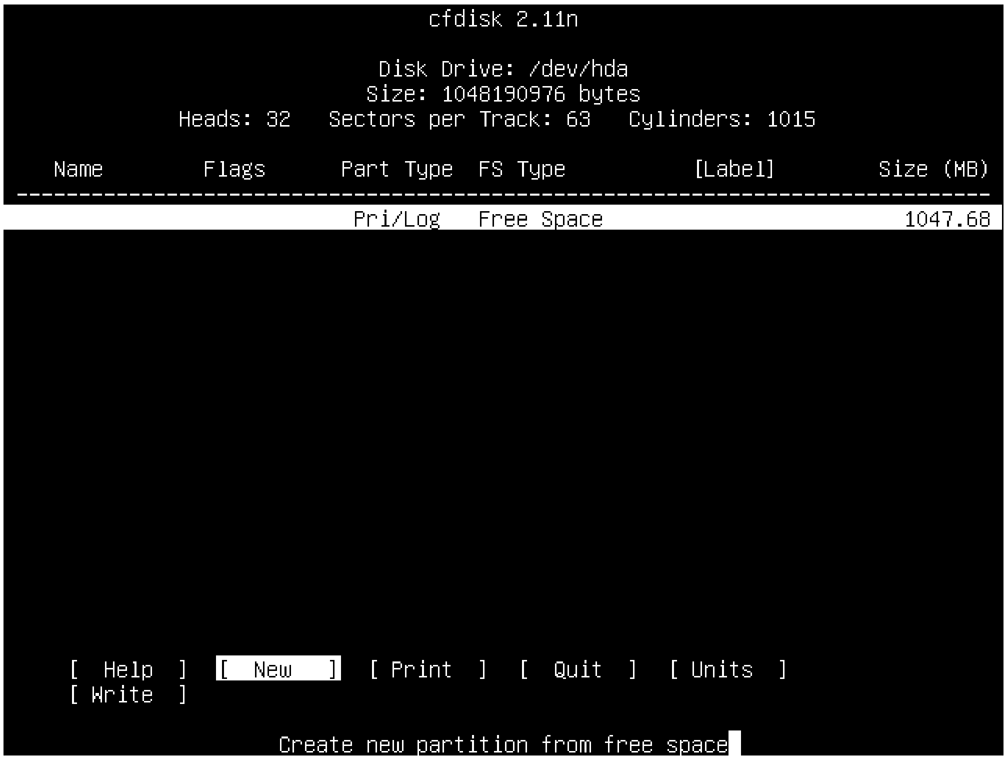
You can use the following keys to navigate around cfdisk:

Table 2.3. Navigation Keys in cfdisk

left/**right** arrow keys	move menu selection highlight
Up/**Down** arrow keys	move partition selection highlight
Enter	select current menu and partition options

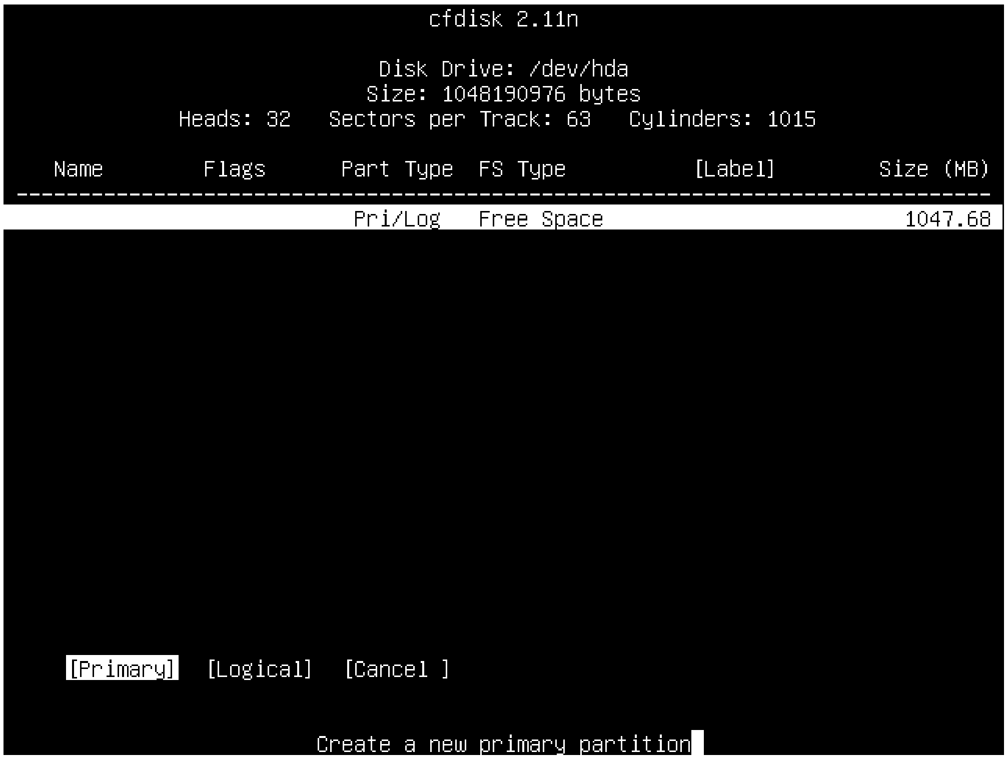
The first thing you should do is press **Enter** and read through the help section to familiarize yourself with its contents.

Once you have read the help section return to the main screen, press the **right** arrow key to highlight the "New" menu option, and then press **Enter**.



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                           Pri/Log   Free Space                         1047.68













     [  Help  ]  [  New   ]  [ Print  ]  [  Quit  ]  [ Units  ]
     [ Write  ]

                      Create new partition from free space
```

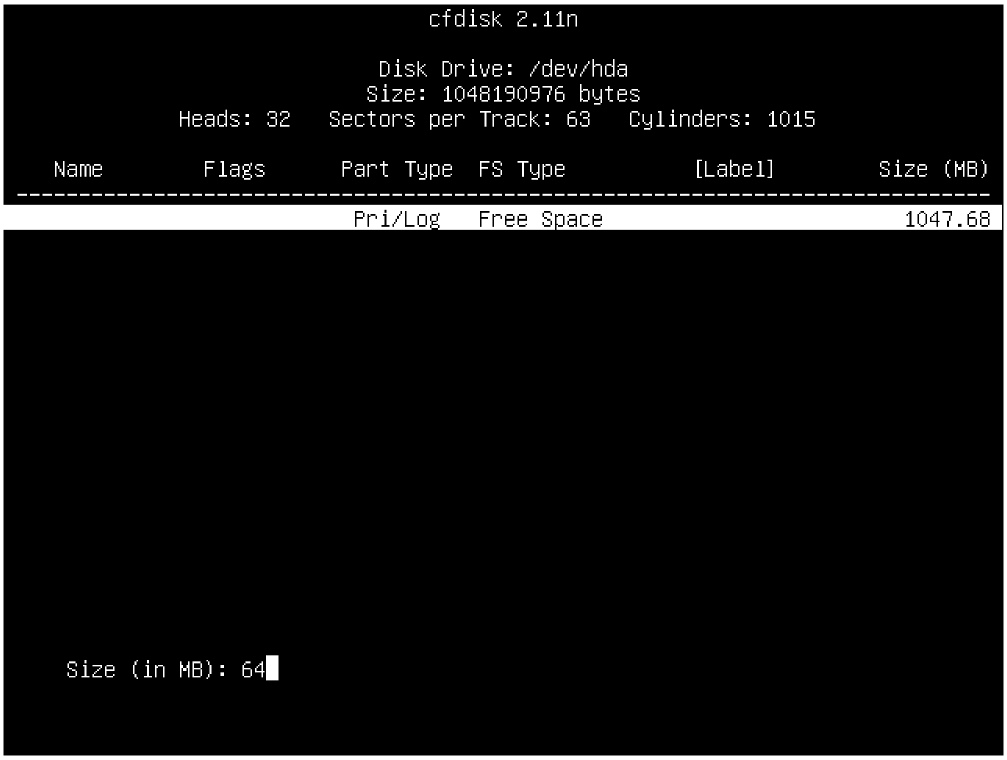
You will then be prompted to chose which type of partition you wish to create; either a primary one or a logical one. Unless you already have four primary partitions on your system, select "primary" and press **Enter**.



```
                                  cfdisk 2.11n

                              Disk Drive: /dev/hda
                             Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                            Pri/Log   Free Space                        1047.68













     [Primary]  [Logical]  [Cancel ]


                         Create a new primary partition
```

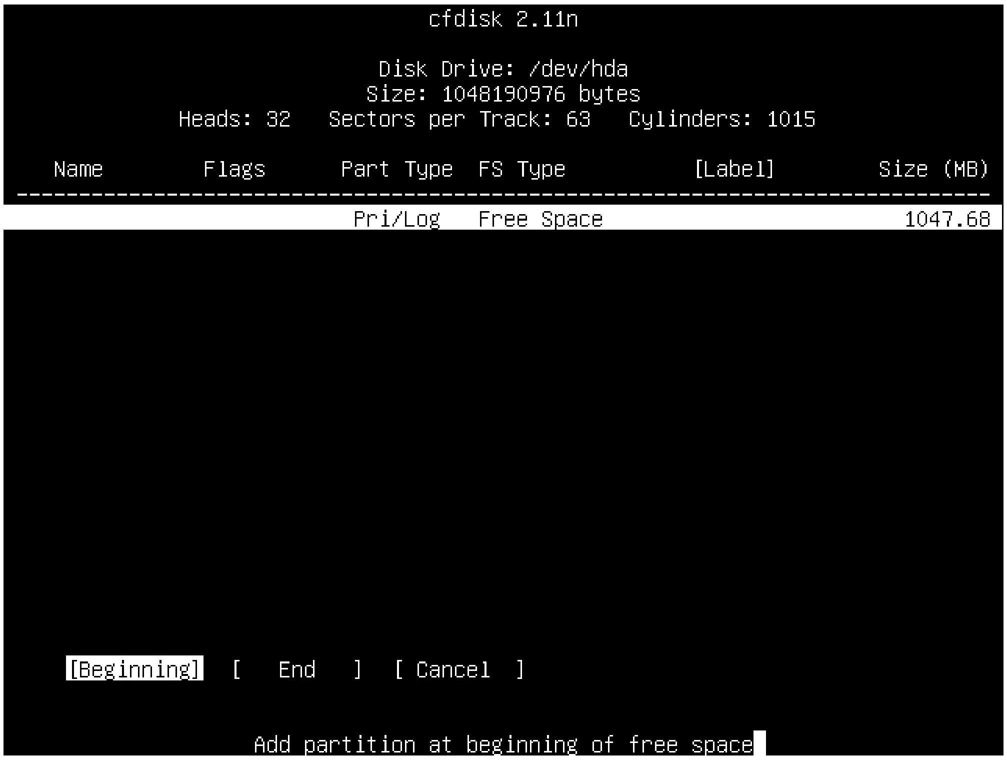
You will then be prompted for the size of the partition, given in MB.

We will create a 64MB sized one, so type "64" and press **Enter**.



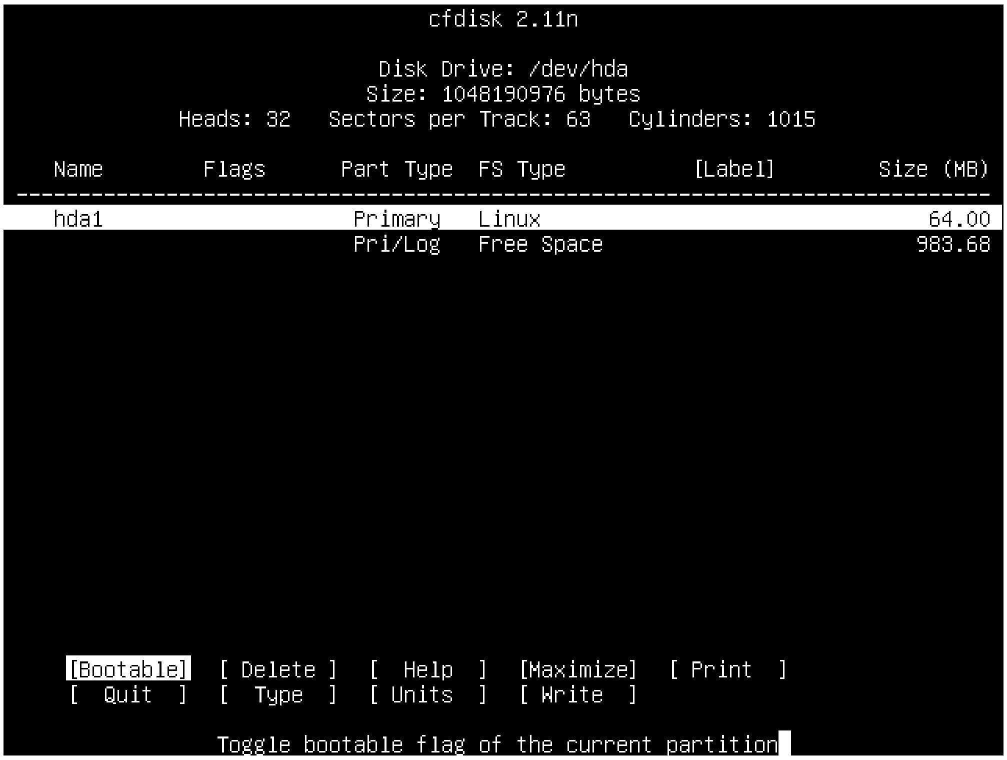
```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
             Heads: 32   Sectors per Track: 63   Cylinders: 1015

   Name        Flags       Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                           Pri/Log    Free Space                       1047.68









    Size (in MB): 64
```

Now you can choose whether you wish to add this partition to the beginning of your free space, or to the end. Select "Beginning" and press **Enter**.



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
           Heads: 32   Sectors per Track: 63   Cylinders: 1015

   Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                            Pri/Log   Free Space                       1047.68










    [Beginning]  [   End   ]  [ Cancel  ]


                  Add partition at beginning of free space
```

You screen should now look something like this:



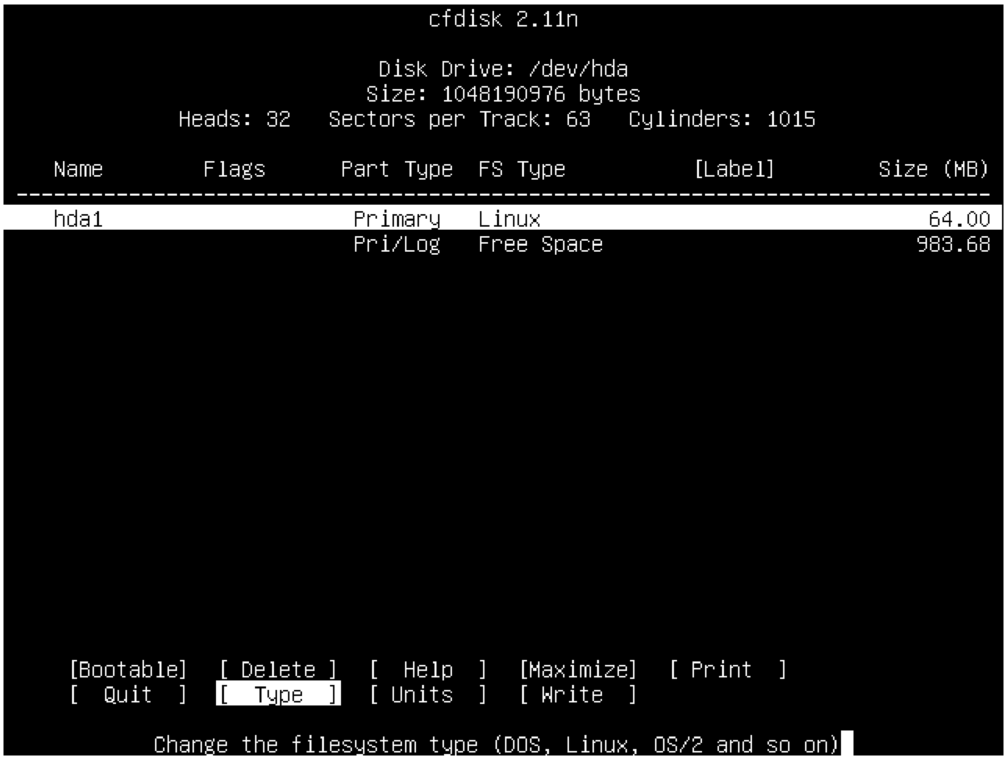
```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux                               64.00
                            Pri/Log   Free Space                         983.68




















     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

                 Toggle bootable flag of the current partition
```

You'll see that the partition you've just created is now displayed. It's called "hda1". As mentioned previously "hda" is your primary master IDE drive; the "1" signifies that it is the first partition.

You'll see that it's also a "Primary" partition, that it's a "Linux" filesystem type and that it's 64MB in size.

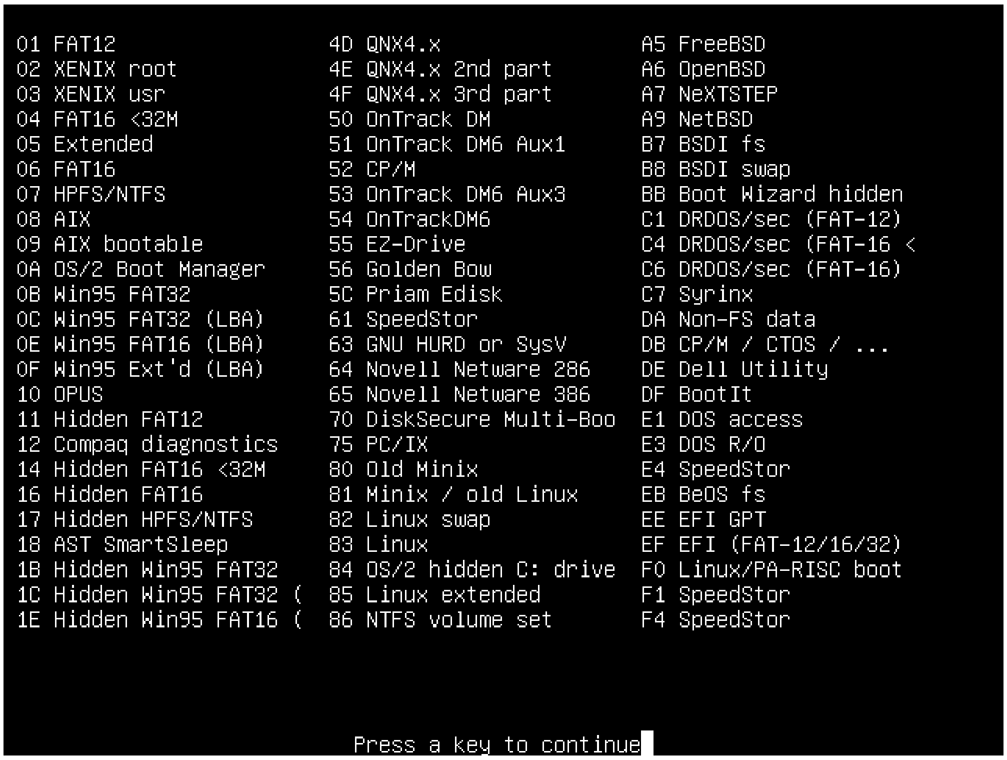
You'll also see that we still have some free space left, 983.68MB worth in the screenshot.

However, we don't want this to be a "Linuxv" partition, we want it to be a "Linux swap" partition. Use the **right arrow** key to move the cursor all the way from "Bootable" over to "Type", and then press **Enter**.

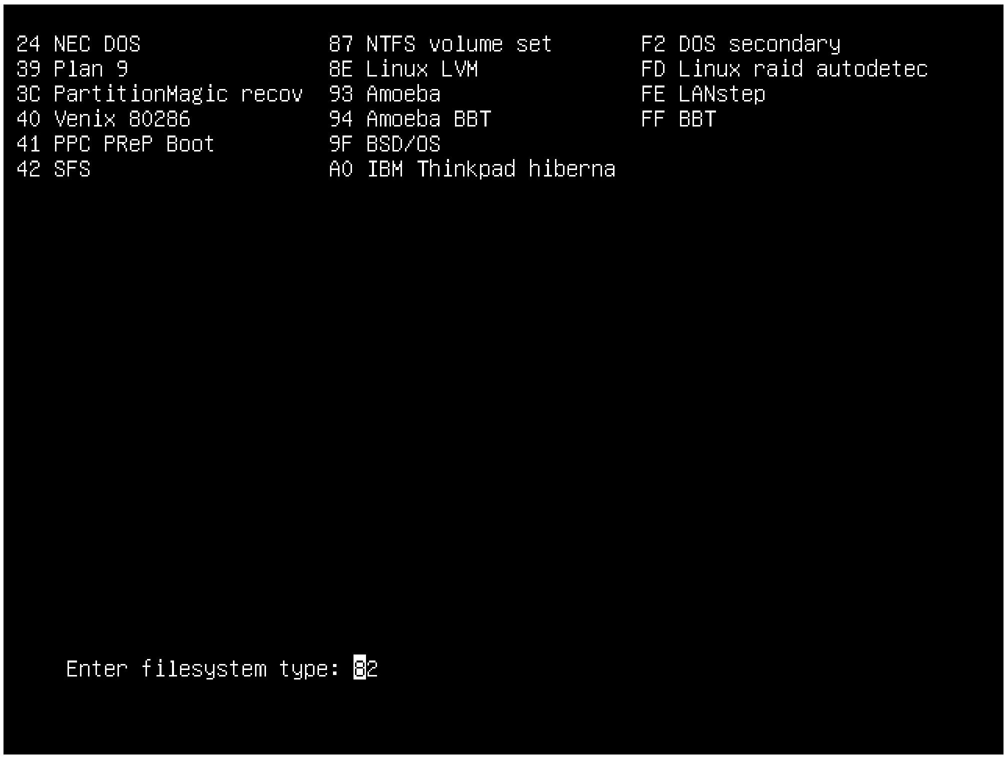


```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux                               64.00
                            Pri/Log   Free Space                         983.68





     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

            Change the filesystem type (DOS, Linux, OS/2 and so on)
```

You'll be presented with two pages worth of partition types, and their associated partition type numbers.



```
01 FAT12                    4D QNX4.x                   A5 FreeBSD
02 XENIX root               4E QNX4.x 2nd part          A6 OpenBSD
03 XENIX usr                4F QNX4.x 3rd part          A7 NeXTSTEP
04 FAT16 <32M               50 OnTrack DM               A9 NetBSD
05 Extended                 51 OnTrack DM6 Aux1         B7 BSDI fs
06 FAT16                    52 CP/M                     B8 BSDI swap
07 HPFS/NTFS                53 OnTrack DM6 Aux3         BB Boot Wizard hidden
08 AIX                      54 OnTrackDM6               C1 DRDOS/sec (FAT-12)
09 AIX bootable             55 EZ-Drive                 C4 DRDOS/sec (FAT-16 <
0A OS/2 Boot Manager        56 Golden Bow               C6 DRDOS/sec (FAT-16)
0B Win95 FAT32              5C Priam Edisk              C7 Syrinx
0C Win95 FAT32 (LBA)        61 SpeedStor                DA Non-FS data
0E Win95 FAT16 (LBA)        63 GNU HURD or SysV         DB CP/M / CTOS / ...
0F Win95 Ext'd (LBA)        64 Novell Netware 286       DE Dell Utility
10 OPUS                     65 Novell Netware 386       DF BootIt
11 Hidden FAT12             70 DiskSecure Multi-Boo     E1 DOS access
12 Compaq diagnostics       75 PC/IX                    E3 DOS R/O
14 Hidden FAT16 <32M        80 Old Minix                E4 SpeedStor
16 Hidden FAT16             81 Minix / old Linux        EB BeOS fs
17 Hidden HPFS/NTFS         82 Linux swap               EE EFI GPT
18 AST SmartSleep           83 Linux                    EF EFI (FAT-12/16/32)
1B Hidden Win95 FAT32       84 OS/2 hidden C: drive     F0 Linux/PA-RISC boot
1C Hidden Win95 FAT32 (     85 Linux extended           F1 SpeedStor
1E Hidden Win95 FAT16 (     86 NTFS volume set          F4 SpeedStor



                           Press a key to continue
```



```
24 NEC DOS                  87 NTFS volume set          F2 DOS secondary
39 Plan 9                   8E Linux LVM                FD Linux raid autodetec
3C PartitionMagic recov     93 Amoeba                   FE LANstep
40 Venix 80286              94 Amoeba BBT               FF BBT
41 PPC PReP Boot            9F BSD/OS
42 SFS                      A0 IBM Thinkpad hiberna




    Enter filesystem type: 82
```

At the end of that, you'll be prompted to enter the number of the partition type that you want.

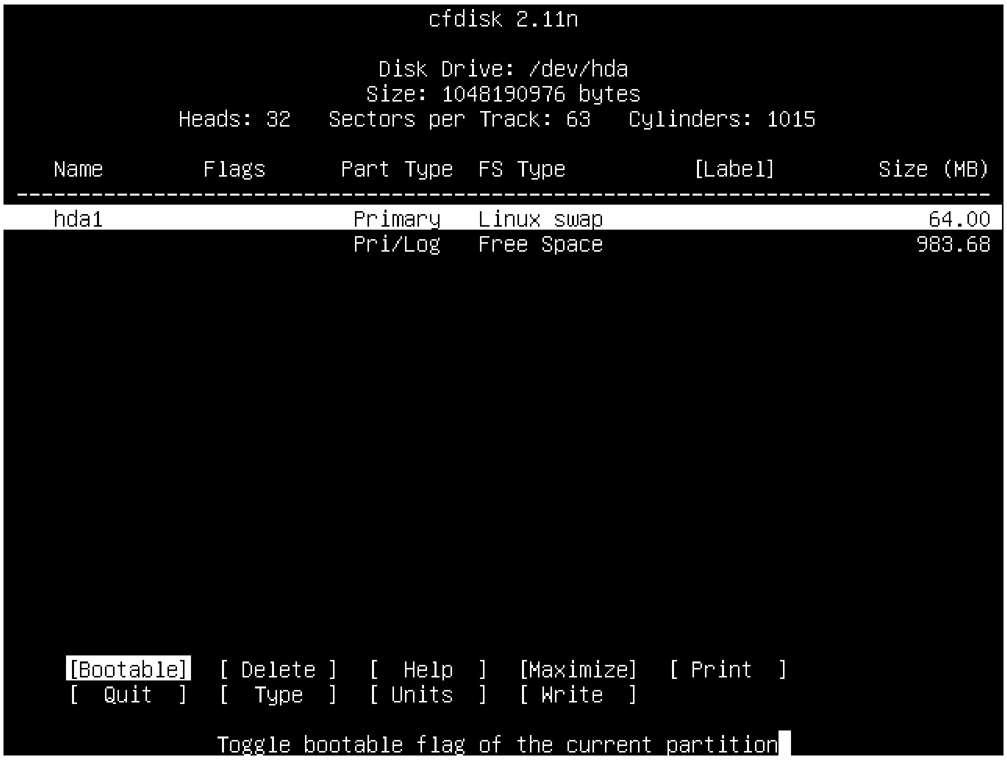
The two numbers that you'll be most interested in remembering are:

```
82 - Linux swap
83 - Linux
```

The rest might be useful if you're having to recover your partition table at some point in time, but they're not necessary for a standard Linux installation.

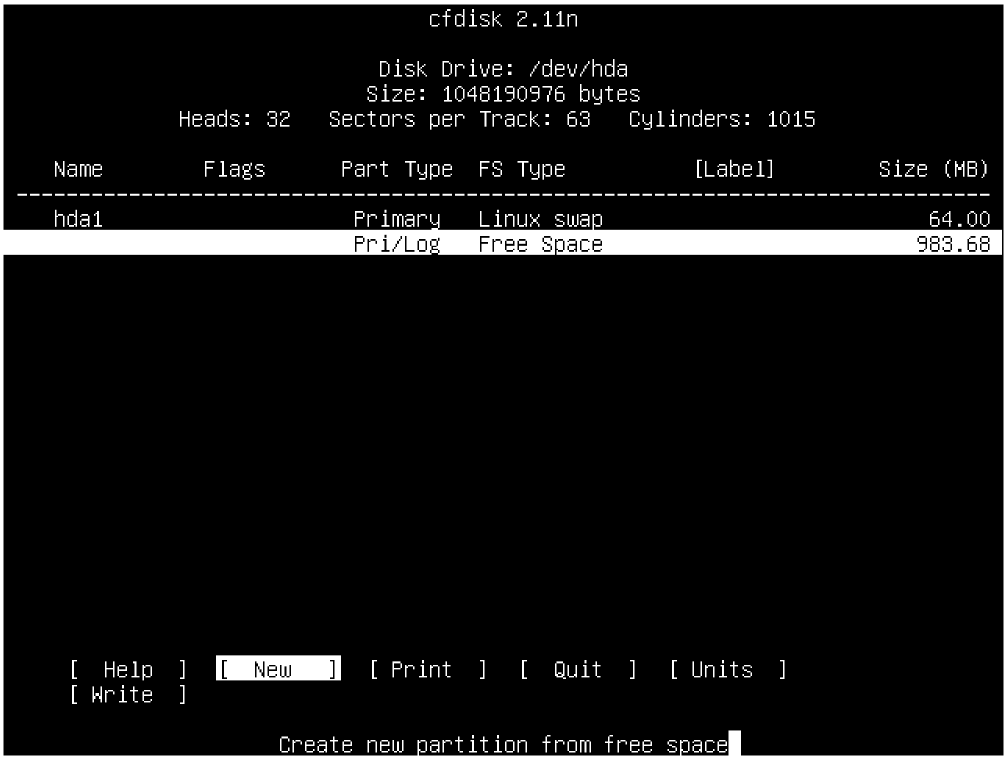
Enter the number "82" (for "Linux swap") and press **Enter**.

You should now see that the main screen has been updated to reflect your choice, and that the FS Type for hda1 is set to "Linux swap".



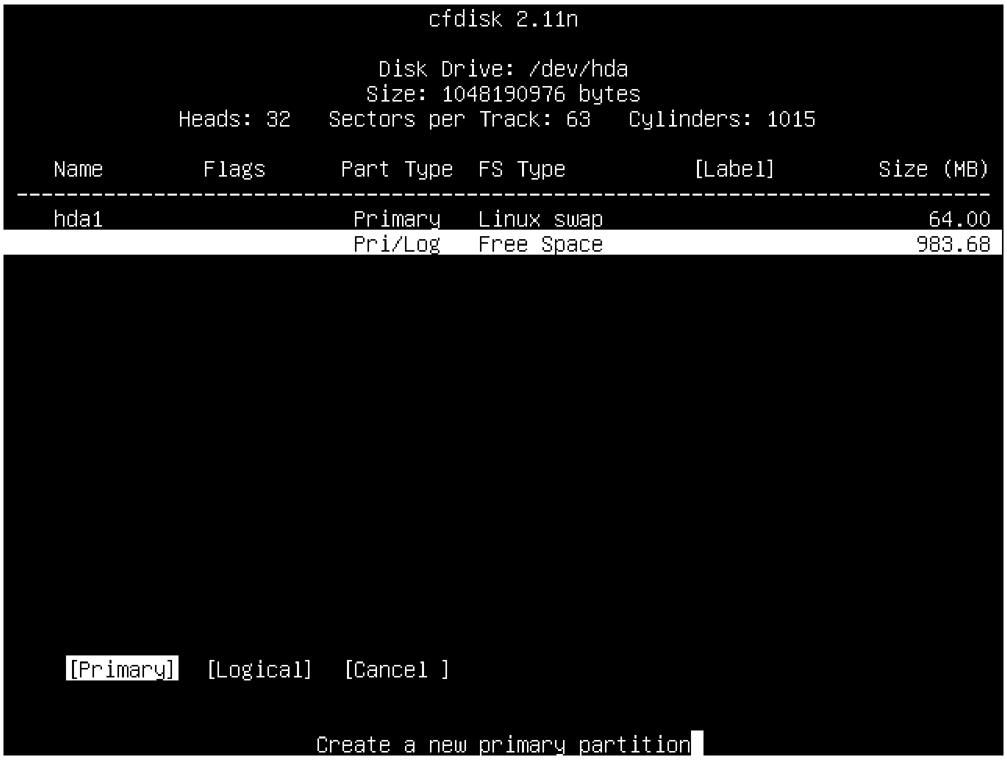
```
                                cfdisk 2.11n

                            Disk Drive: /dev/hda
                           Size: 1048190976 bytes
               Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux swap                          64.00
                            Pri/Log   Free Space                         983.68

     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

                 Toggle bootable flag of the current partition
```

Now we want to create another partition out of the remaining free space. Press the **down arrow** key to move the highlight bar onto the "Free Space" section, and then press the **right arrow** key to select "New".



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux swap                          64.00
                            Pri/Log   Free Space                         983.68









     [  Help  ]  [  New   ]  [ Print  ]  [  Quit  ]  [ Units  ]
     [ Write  ]

                      Create new partition from free space
```

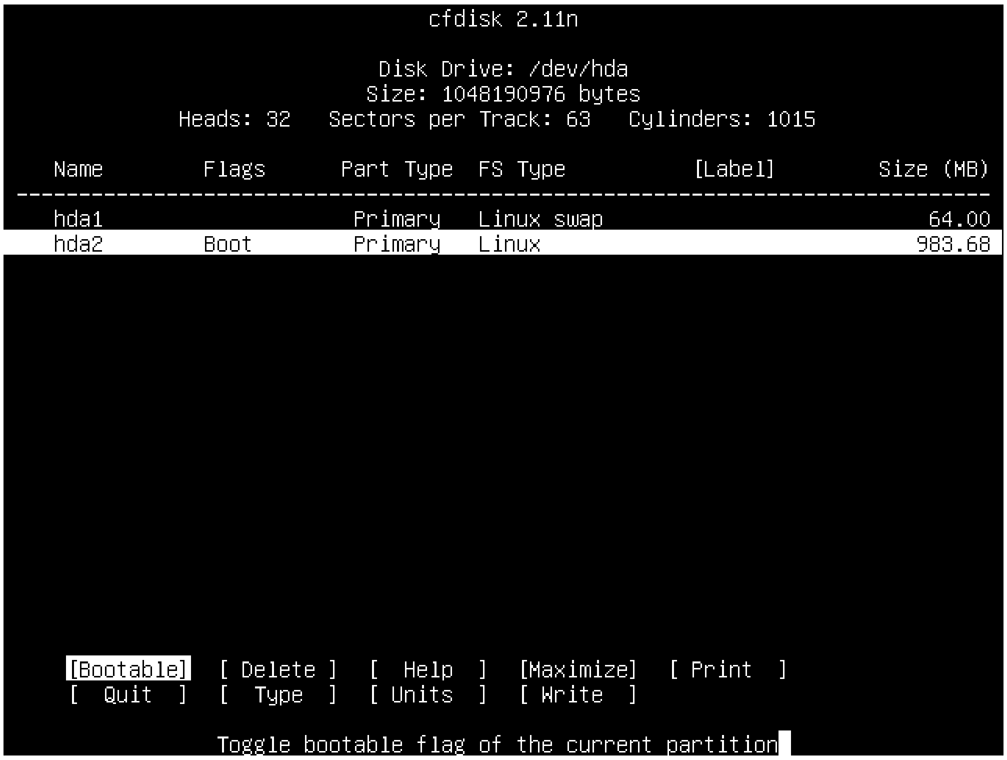
Press **Enter**, and you should be presented with the now familiar menu asking you to select between creating a Primary or a Logical partition. Just press **Enter** again to select Primary.



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

   Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
   hda1                    Primary   Linux swap                          64.00
                           Pri/Log   Free Space                         983.68










     [Primary]  [Logical]  [Cancel ]


                        Create a new primary partition
```

As before, you'll be prompted to select the size of the partition. Just hit **Enter** to select the default, which is to use all the available free space.

Since this is the partition you'll be wanting to boot from, press the **Enter** key again to toggle the partition's "Bootable" flag. This tells your computer than that partition can be booted from. The main screen should now look something like this:

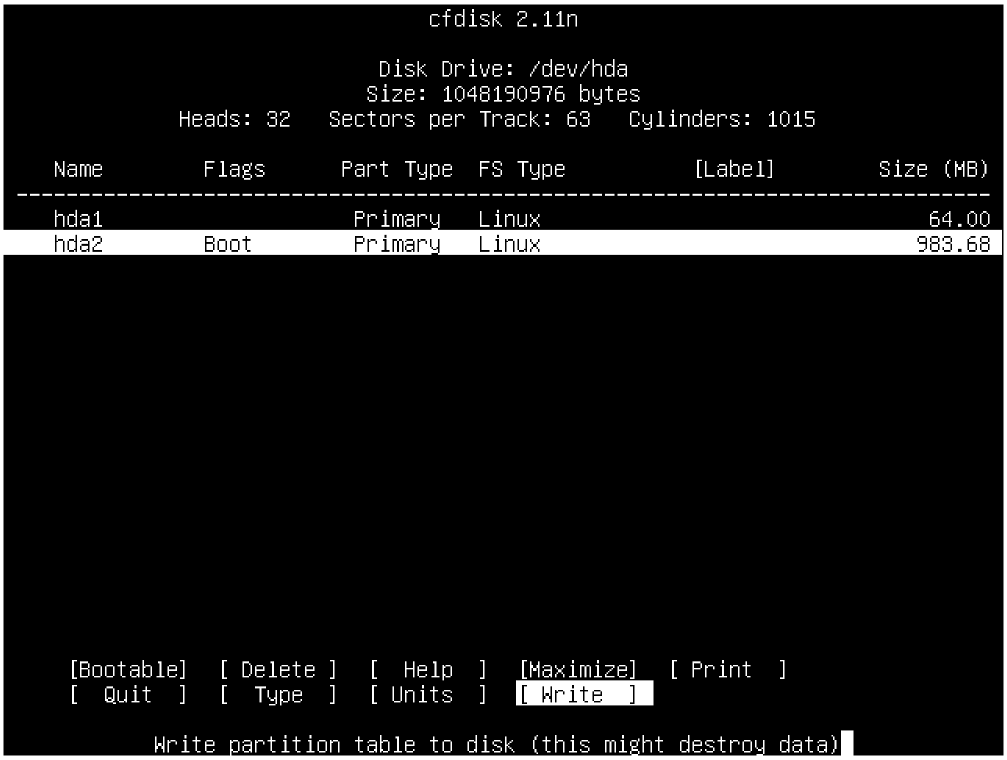


```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux swap                          64.00
    hda2        Boot        Primary   Linux                              983.68

     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

                 Toggle bootable flag of the current partition
```

You can see that you now have two partitions; "hda1" is the previously created Linux swap partition, while "hda2" is your newly created Linux partition. You'll note that the "hda2" partition has its "Boot" flag enabled.

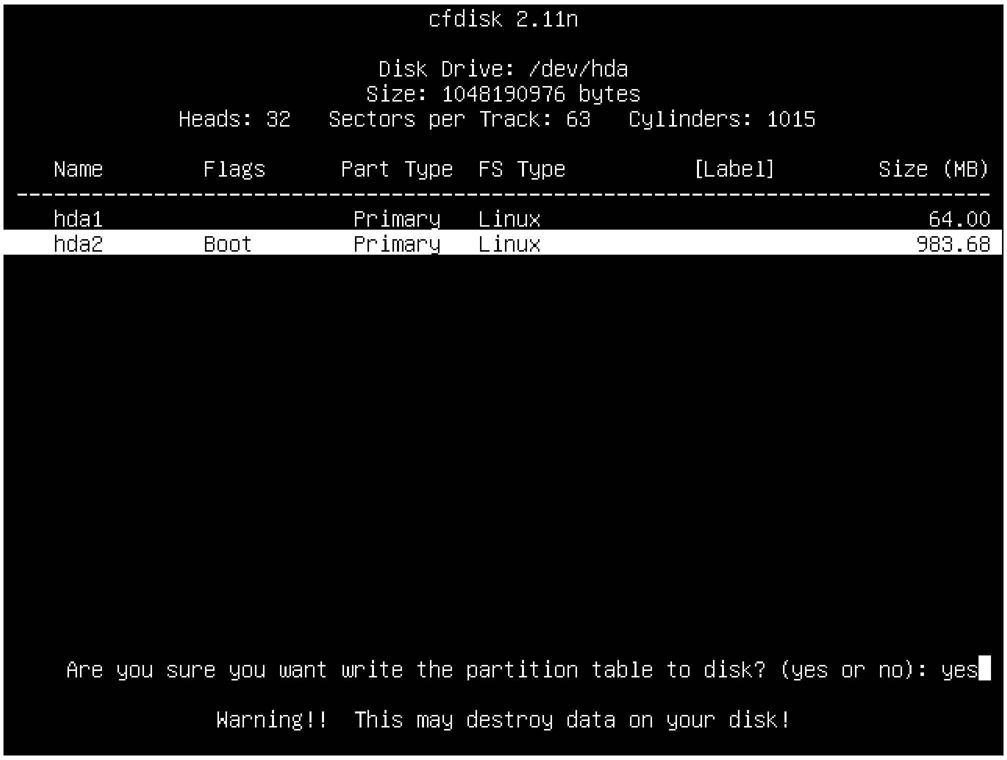
Phew! Now we can finally write this partition table to the hard drive.

Use the **right** arrow key to move the highlight over to the `Write` option, and press **Enter**.



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux                             64.00
    hda2        Boot        Primary   Linux                            983.68









     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

            Write partition table to disk (this might destroy data)
```

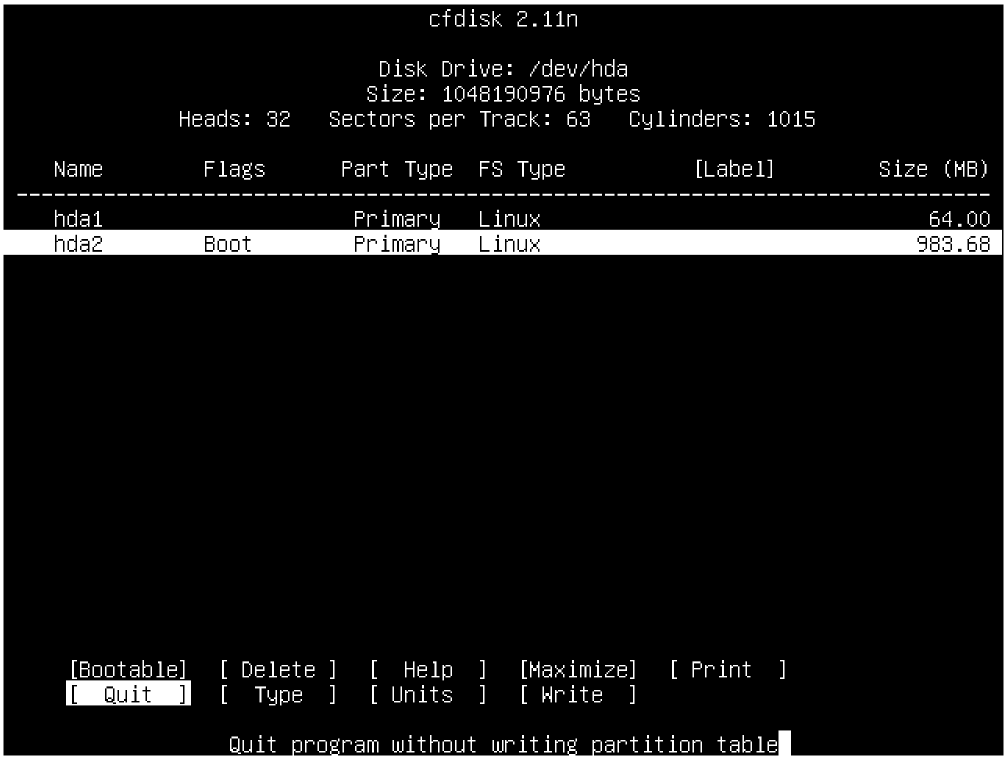
You will be prompted as to whether you do indeed wish to perform this operation:



```
                                  cfdisk 2.11n

                              Disk Drive: /dev/hda
                             Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1                    Primary   Linux                              64.00
    hda2        Boot        Primary   Linux                             983.68










     Are you sure you want write the partition table to disk? (yes or no): yes

                 Warning!!  This may destroy data on your disk!
```

Type out the full word "yes" if you are sure that you are happy with this.

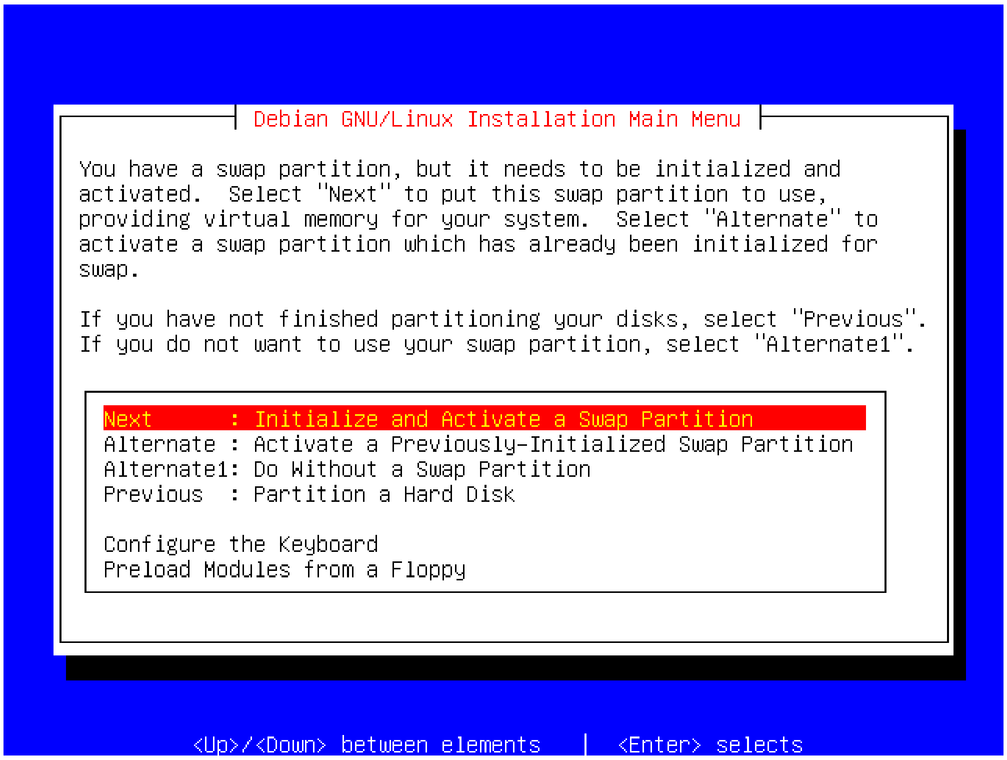
Once the partition table has been written out successfully, you can move the highlight over to `Quit` using the arrow keys, and then press **Enter**.



```
                              cfdisk 2.11n

                          Disk Drive: /dev/hda
                         Size: 1048190976 bytes
              Heads: 32   Sectors per Track: 63   Cylinders: 1015

   Name         Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
   hda1                     Primary   Linux                               64.00
   hda2         Boot        Primary   Linux                              983.68











    [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
    [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

                  Quit program without writing partition table
```

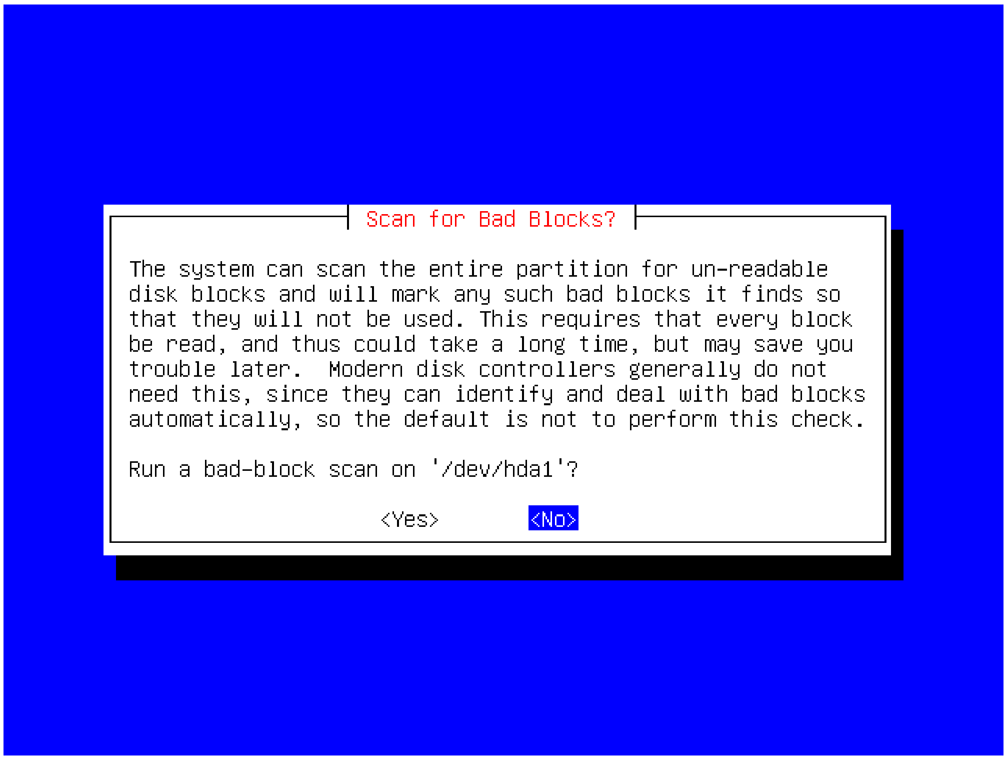
You have now completed the hardest part of the installation and the rest is relatively easy from here.

Initialize and activate swap



```
Debian GNU/Linux Installation Main Menu

You have a swap partition, but it needs to be initialized and
activated.  Select "Next" to put this swap partition to use,
providing virtual memory for your system.  Select "Alternate" to
activate a swap partition which has already been initialized for
swap.

If you have not finished partitioning your disks, select "Previous".
If you do not want to use your swap partition, select "Alternate1".

Next       : Initialize and Activate a Swap Partition
Alternate  : Activate a Previously-Initialized Swap Partition
Alternate1: Do Without a Swap Partition
Previous   : Partition a Hard Disk

Configure the Keyboard
Preload Modules from a Floppy

<Up>/<Down> between elements   |  <Enter> selects
```

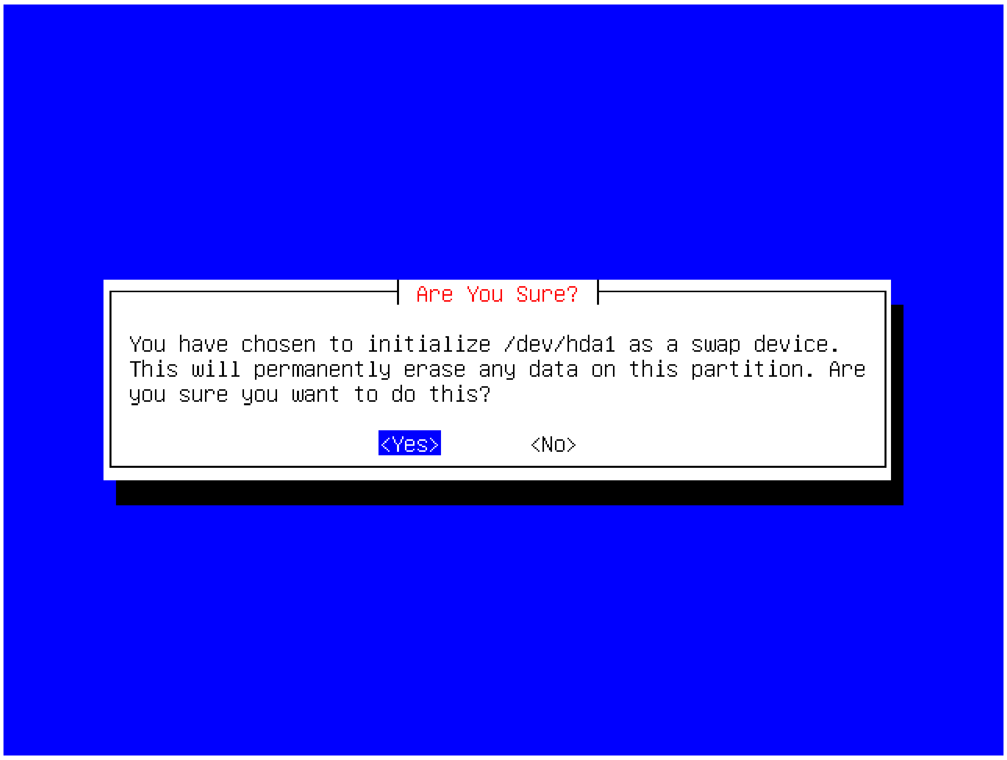
As the display tells you, you've created a swap partition, but you haven't initialized and activated it yet. Press **Enter** to opt to do this now.



```
Scan for Bad Blocks?

The system can scan the entire partition for un-readable
disk blocks and will mark any such bad blocks it finds so
that they will not be used. This requires that every block
be read, and thus could take a long time, but may save you
trouble later.  Modern disk controllers generally do not
need this, since they can identify and deal with bad blocks
automatically, so the default is not to perform this check.

Run a bad-block scan on '/dev/hda1'?

<Yes>          <No>
```

As explained in the write-up on dbootstrap, you can skip this step if you have a relatively modern HDD, as it is able to handle bad blocks on its own.

Select **No** and press **Enter** to continue.



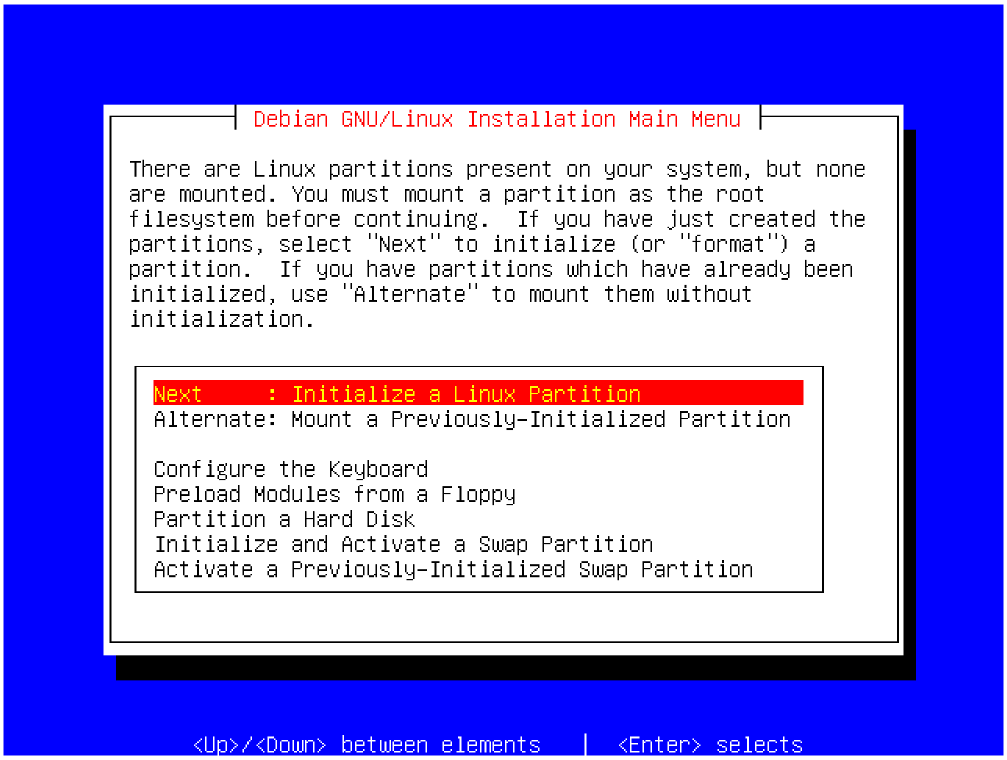
Are You Sure?

```
You have chosen to initialize /dev/hda1 as a swap device.
This will permanently erase any data on this partition. Are
you sure you want to do this?
```

Last chance! Are you sure that this is the correct partition (hda1 in our example)? Once you select **Yes** and press **Enter**, any information stored on that part of the disk will now be overwritten.

Now your system has that disk space available as virtual memory. This is very useful if you're installing on a machine with a small amount of RAM, as dbootstrap can now make use of this resource.

Initialize a Linux partition

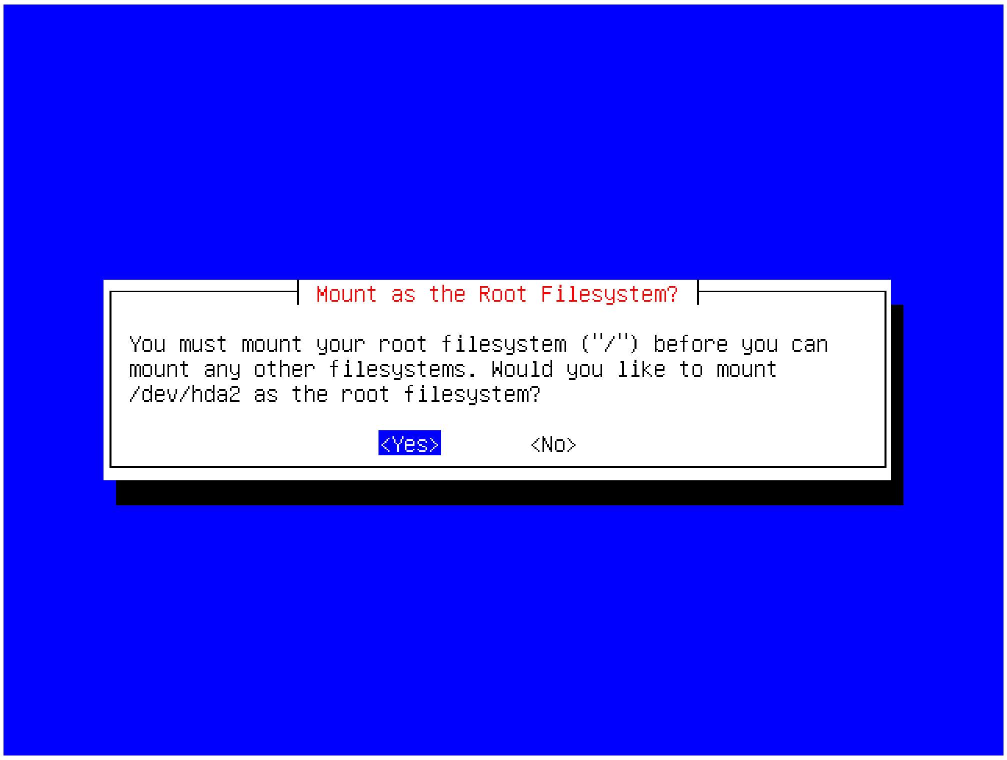


```
Debian GNU/Linux Installation Main Menu

There are Linux partitions present on your system, but none
are mounted. You must mount a partition as the root
filesystem before continuing.  If you have just created the
partitions, select "Next" to initialize (or "format") a
partition.  If you have partitions which have already been
initialized, use "Alternate" to mount them without
initialization.

Next      : Initialize a Linux Partition
Alternate: Mount a Previously-Initialized Partition

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition

<Up>/<Down> between elements   |  <Enter> selects
```

Now we need to Initialize ("format" in DOS parlance) your Linux partition, and then mount it so that we can access it to install Debian onto it.

Press **Enter** to begin this process now.

Depending on the size of your partition, and the speed of your system and drive, this may take a few moments.



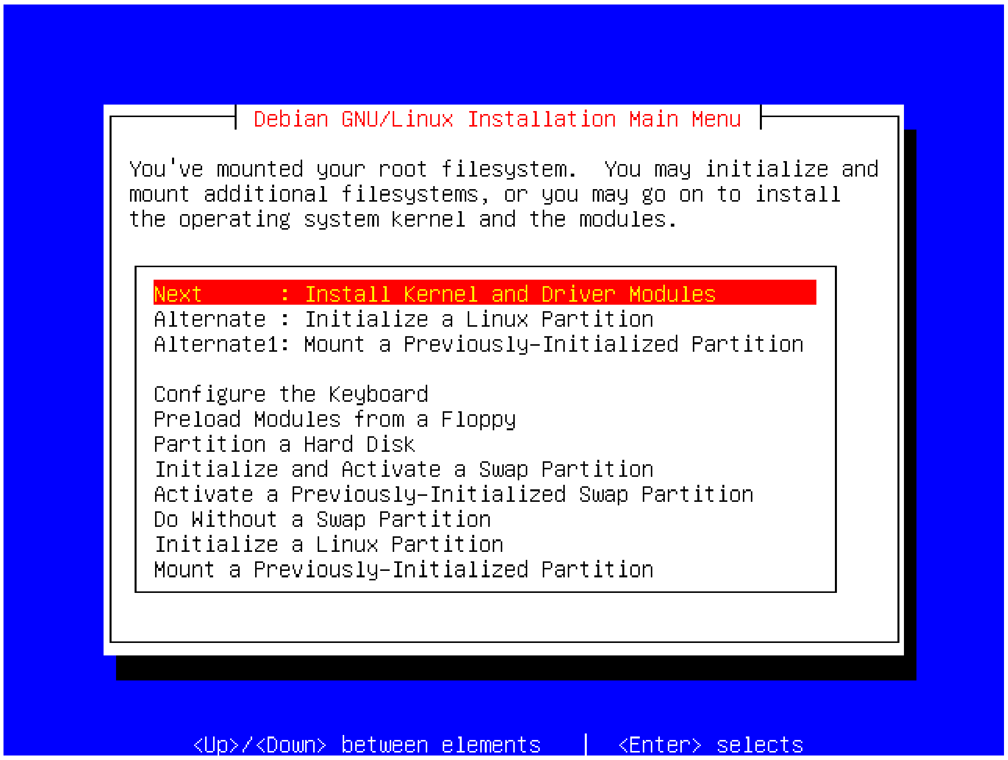
```
Mount as the Root Filesystem?

You must mount your root filesystem ("/") before you can
mount any other filesystems. Would you like to mount
/dev/hda2 as the root filesystem?

<Yes>        <No>
```

The root ("/") is the base of your filesystem; for our example installation, we will only have this single partition. You may wish to make use of a more complicated layout if you have special needs.

For example, if you have a Samba file server sharing user's home directories, you might well want to have the system ("/", **`/usr`**, etc.) on a separate disk to your data (**`/home`**). This will allow you to upgrade your system without interfering with the user data, and, likewise, allow you to expand the amount of data storage space easily, without having to re-install the system from scratch.

However, a single filesystem is usually sufficient for a workstation or similar installation.

Select **Yes** to mount the root filesystem onto our hda2 device.



```
Debian GNU/Linux Installation Main Menu

You've mounted your root filesystem.  You may initialize and
mount additional filesystems, or you may go on to install
the operating system kernel and the modules.

Next       : Install Kernel and Driver Modules
Alternate  : Initialize a Linux Partition
Alternate1: Mount a Previously-Initialized Partition

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition
Do Without a Swap Partition
Initialize a Linux Partition
Mount a Previously-Initialized Partition

<Up>/<Down> between elements   |  <Enter> selects
```

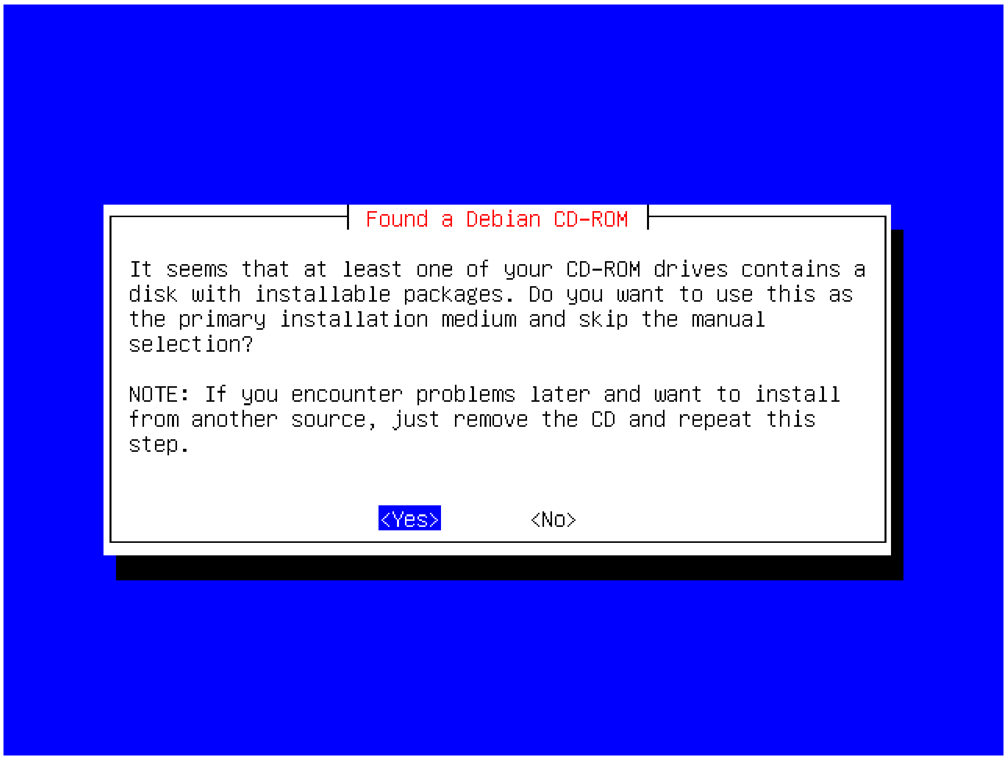
Now that you have mounted your root filesystem, you may choose to mount additional filesystems, or continue on and install the Linux kernel and driver modules.

For our example, as we only have one partition, we chose to do the later.

You might want to mount additional filesystems to install additional software, or if you are attempting to use the installation program to rescue or upgrade your system.

Press **Enter**.

Found a Debian CD-ROM



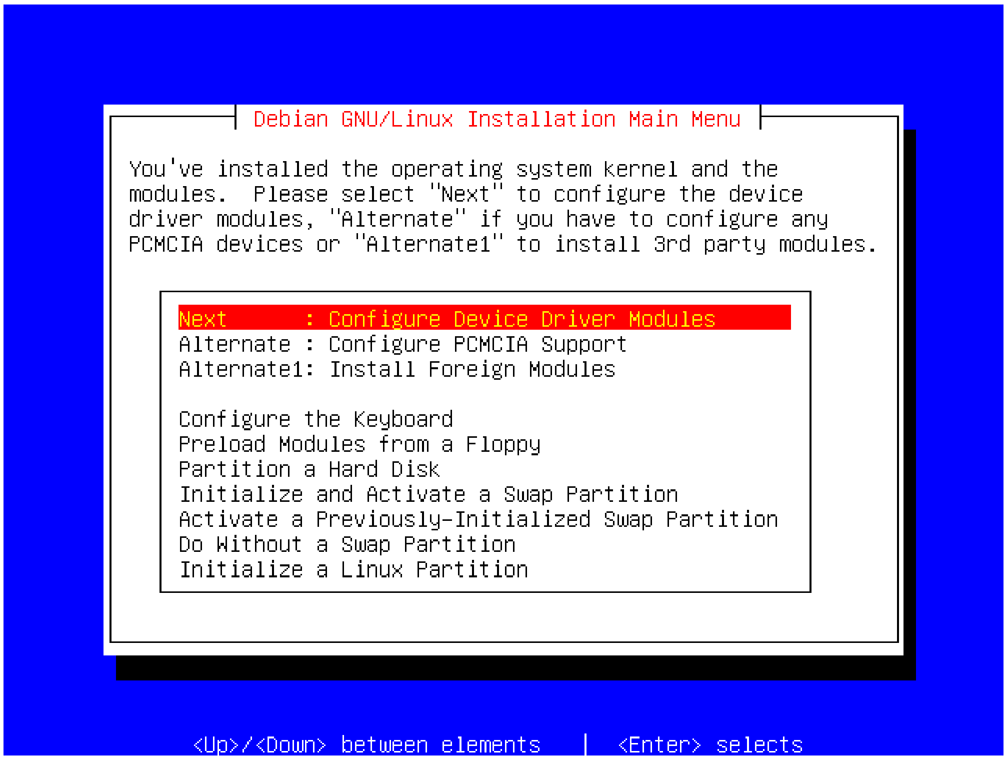
```
Found a Debian CD-ROM

It seems that at least one of your CD-ROM drives contains a
disk with installable packages. Do you want to use this as
the primary installation medium and skip the manual
selection?

NOTE: If you encounter problems later and want to install
from another source, just remove the CD and repeat this
step.

                  <Yes>        <No>
```

The installation program has detected your Debian CD-ROM and is asking whether or not you wish to install the packages from it.

Alternatives would be to install packages from another source, such as over the network or from another hard disk in the system.

We want to use the CD though, so just select **Yes** and press **Enter**.

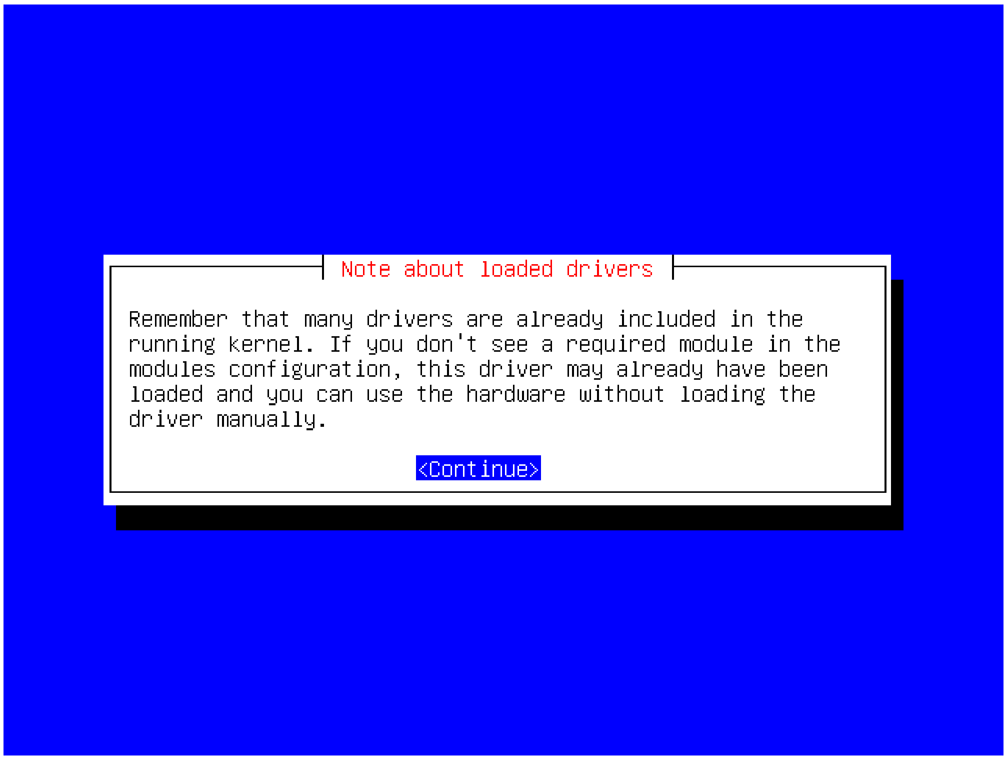
Configure device driver modules



```
Debian GNU/Linux Installation Main Menu

You've installed the operating system kernel and the
modules.  Please select "Next" to configure the device
driver modules, "Alternate" if you have to configure any
PCMCIA devices or "Alternate1" to install 3rd party modules.

Next        : Configure Device Driver Modules
Alternate   : Configure PCMCIA Support
Alternate1: Install Foreign Modules

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition
Do Without a Swap Partition
Initialize a Linux Partition

<Up>/<Down> between elements   |   <Enter> selects
```

This section lets you configure which Linux kernel modules, or device drivers, you wish to use. This is useful if you have a network card which is not automatically detected, or, if you're installing on a laptop, if you have any ACACIA devices which you wish to use.

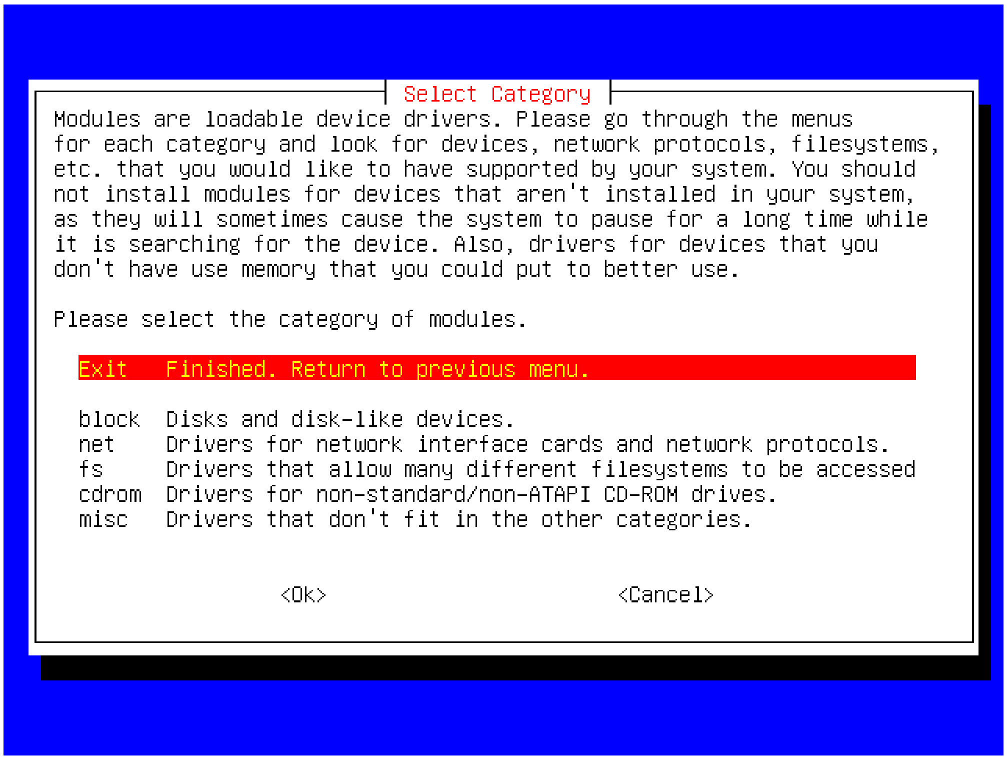
Select **Next** and press **Enter**.



```
Remember that many drivers are already included in the
running kernel. If you don't see a required module in the
modules configuration, this driver may already have been
loaded and you can use the hardware without loading the
driver manually.
```

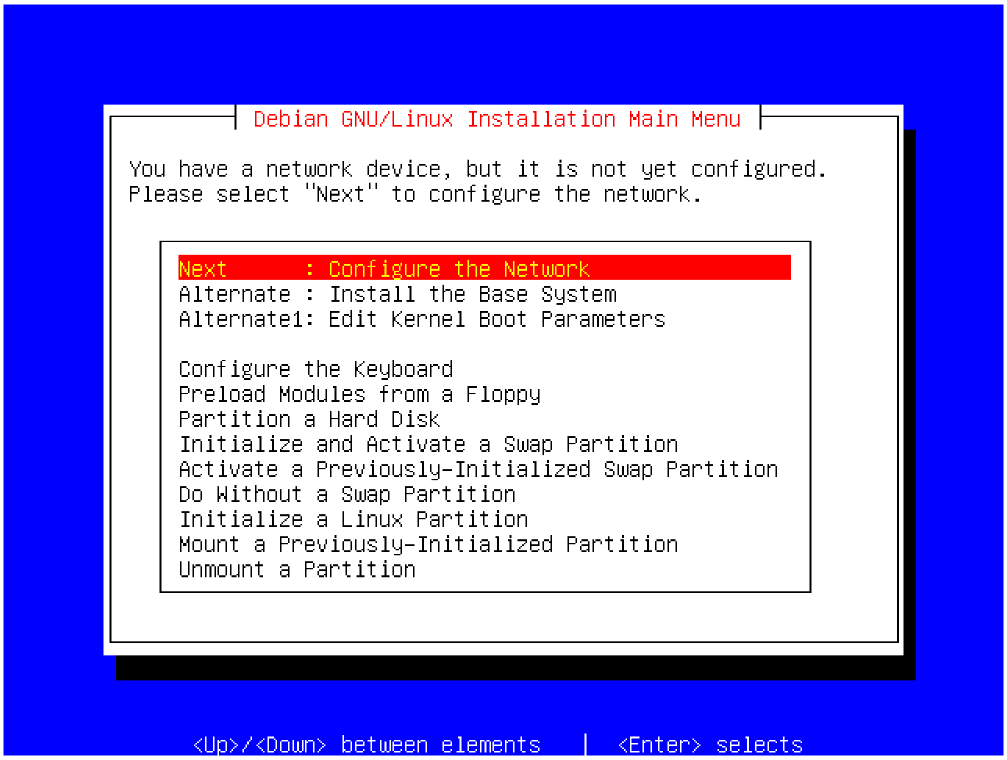
If you have a supported PCI network card, support for it will already exist in the Linux kernel, so you won't have to select a kernel module for it. However, if you have one of the older ISA/EISA network cards, (e.g. an NE2000 compatible one), then you may well have to select a driver for it.

One handy way of doing this is seeing what settings the hardware is detected as having by other operating systems.



```
Select Category
Modules are loadable device drivers. Please go through the menus
for each category and look for devices, network protocols, filesystems,
etc. that you would like to have supported by your system. You should
not install modules for devices that aren't installed in your system,
as they will sometimes cause the system to pause for a long time while
it is searching for the device. Also, drivers for devices that you
don't have use memory that you could put to better use.

Please select the category of modules.

  Exit    Finished. Return to previous menu.

  block   Disks and disk-like devices.
  net     Drivers for network interface cards and network protocols.
  fs      Drivers that allow many different filesystems to be accessed
  cdrom   Drivers for non-standard/non-ATAPI CD-ROM drives.
  misc    Drivers that don't fit in the other categories.


                   <Ok>                       <Cancel>
```

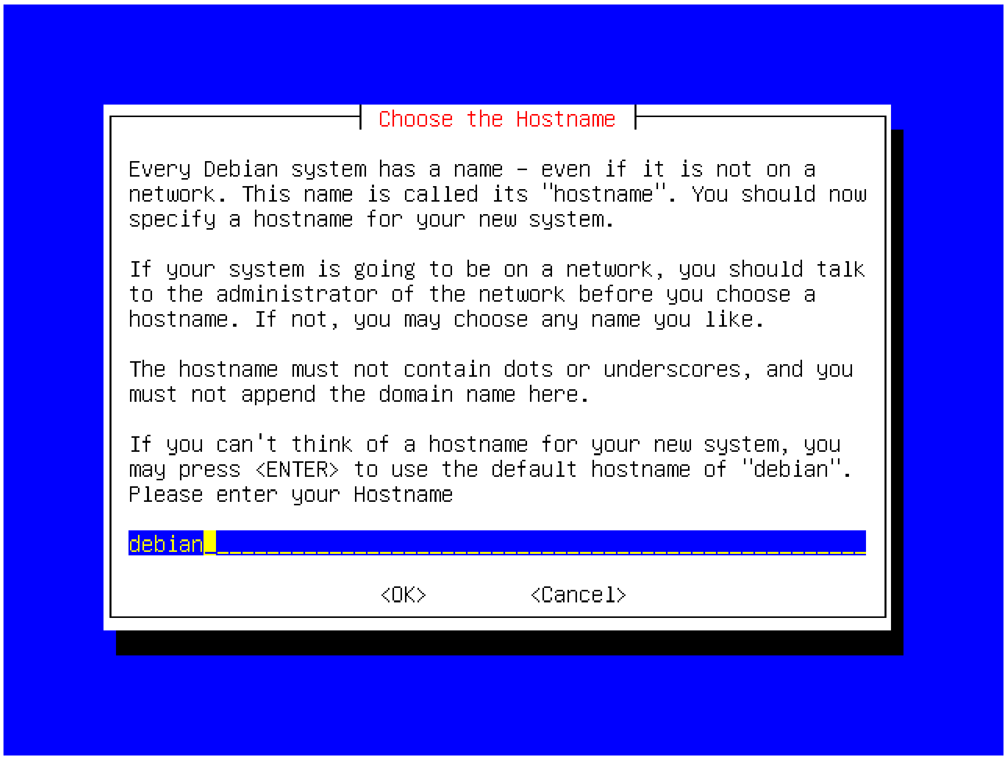
Configure the network



```
Debian GNU/Linux Installation Main Menu

You have a network device, but it is not yet configured.
Please select "Next" to configure the network.

Next        : Configure the Network
Alternate   : Install the Base System
Alternate1: Edit Kernel Boot Parameters

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition
Do Without a Swap Partition
Initialize a Linux Partition
Mount a Previously-Initialized Partition
Unmount a Partition

<Up>/<Down> between elements   |  <Enter> selects
```

If Linux has detected a network card, or if you have selected a network card module, you will be presented with the option of configuring it.

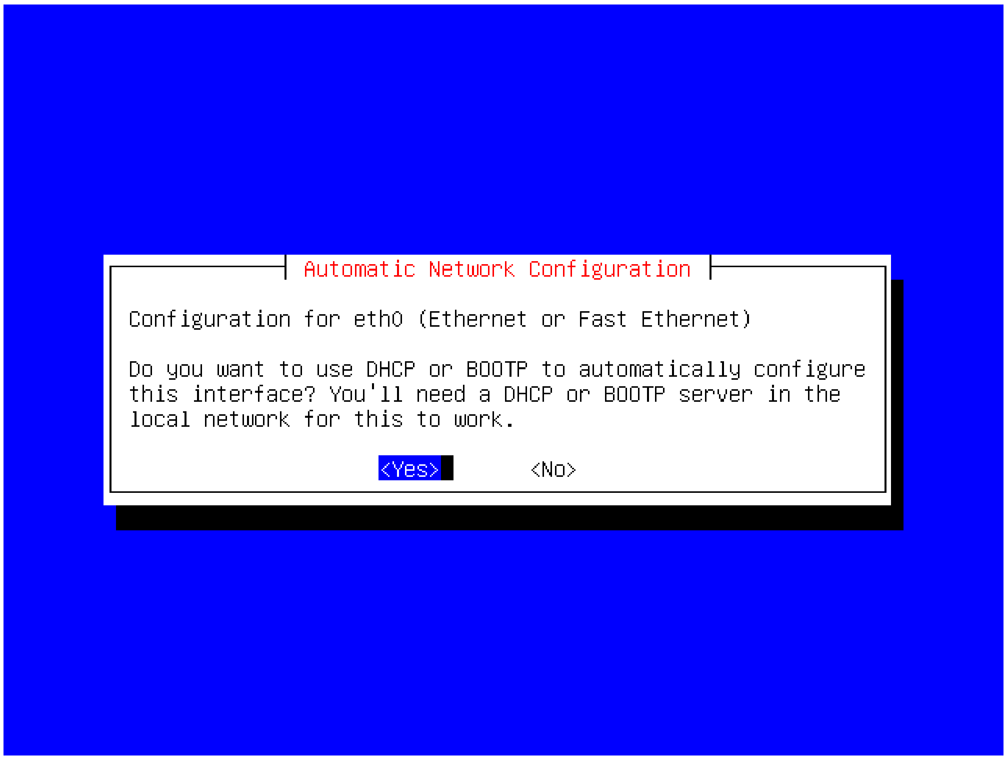
If you don't have a network card, or if you do but don't wish to configure it, you can skip ahead to the next section, otherwise press **Enter** to begin the configuration.



```
Choose the Hostname

Every Debian system has a name - even if it is not on a
network. This name is called its "hostname". You should now
specify a hostname for your new system.

If your system is going to be on a network, you should talk
to the administrator of the network before you choose a
hostname. If not, you may choose any name you like.

The hostname must not contain dots or underscores, and you
must not append the domain name here.

If you can't think of a hostname for your new system, you
may press <ENTER> to use the default hostname of "debian".
Please enter your Hostname

debian_

<OK>          <Cancel>
```

You'll be prompted for a name for your machine; you can call it whatever you like, as long as the name conforms to the rules mentioned on the screen. The default name is "debian", so we'll just stick with that.

Press **Enter** to continue.

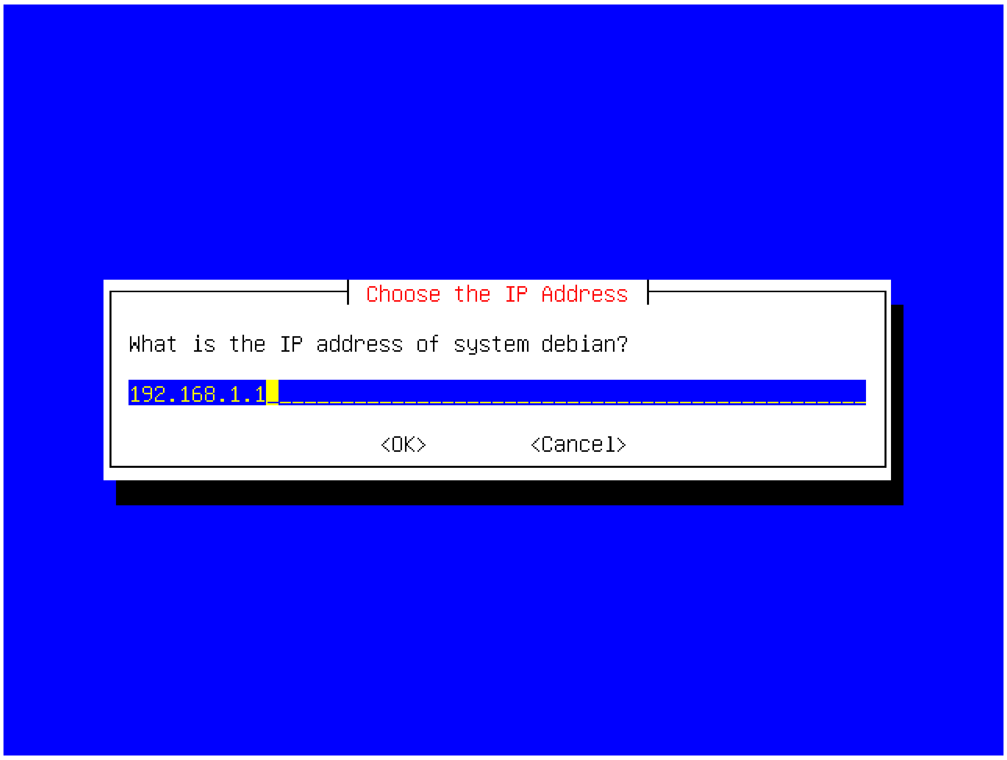


```
Automatic Network Configuration

Configuration for eth0 (Ethernet or Fast Ethernet)

Do you want to use DHCP or BOOTP to automatically configure
this interface? You'll need a DHCP or BOOTP server in the
local network for this to work.
```

Here you'll have to know a little bit about your network. If you're putting your machine on a live network, and you know that your network has a DHCP server, then you can tell Debian to behave as a DHCP client. This is useful as it will automatically configure your network settings for you.

However, if your machine isn't on a network, or your network doesn't support DHCP, then you'll have to do it manually. For this example, we'll do it manually.

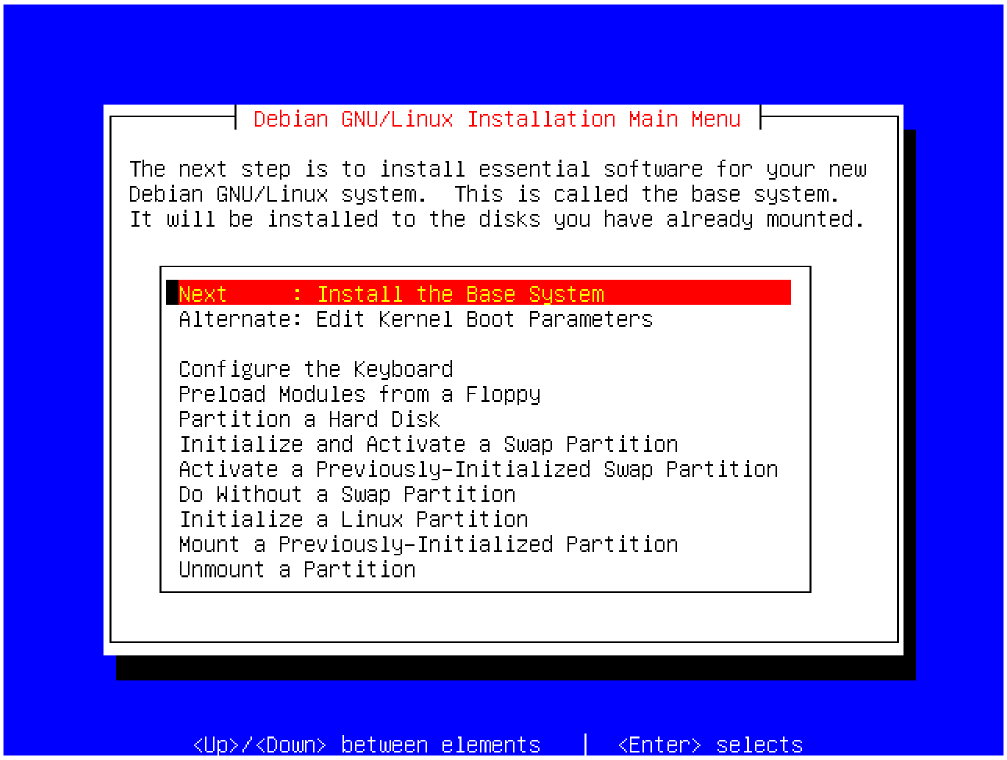
Use the arrow keys to select **No**, and then press **Enter**.



```
Choose the IP Address

What is the IP address of system debian?

192.168.1.1

<OK>        <Cancel>
```

You'll need to provide an IP address here; the default is "192.168.1.1", which is a private IP address. We'll use this one for our example, but you may need to specify a real one depending on your own network configuration.

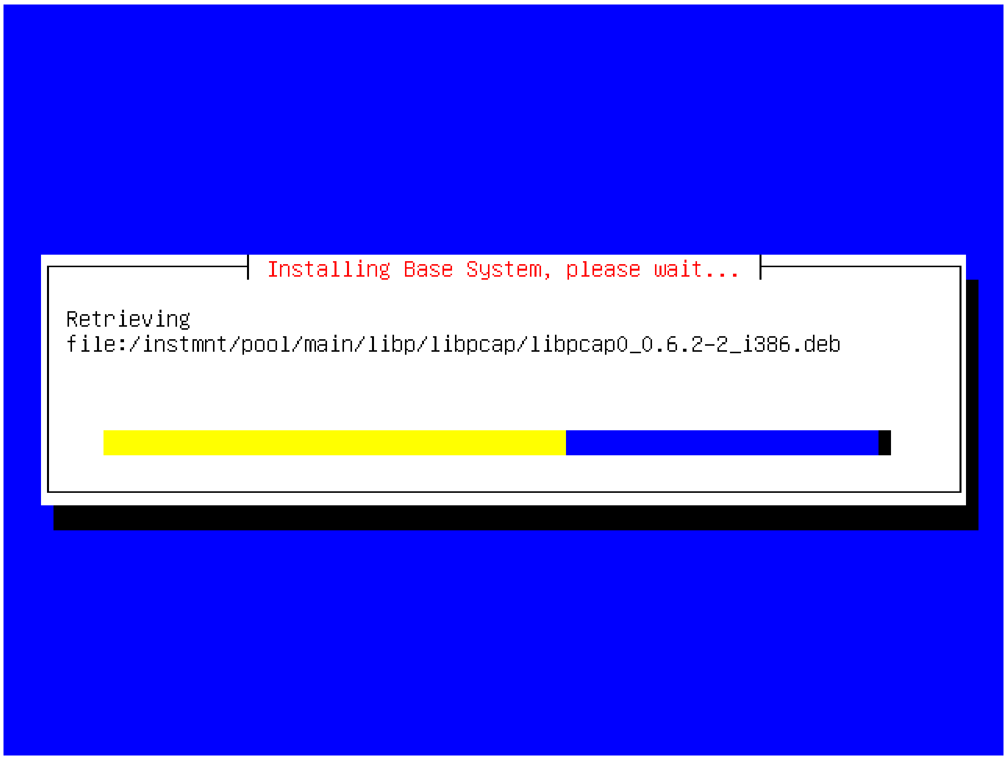
Install the base system



```
Debian GNU/Linux Installation Main Menu

The next step is to install essential software for your new
Debian GNU/Linux system.  This is called the base system.
It will be installed to the disks you have already mounted.

Next      : Install the Base System
Alternate: Edit Kernel Boot Parameters

Configure the Keyboard
Preload Modules from a Floppy
Partition a Hard Disk
Initialize and Activate a Swap Partition
Activate a Previously-Initialized Swap Partition
Do Without a Swap Partition
Initialize a Linux Partition
Mount a Previously-Initialized Partition
Unmount a Partition

<Up>/<Down> between elements   |  <Enter> selects
```

This step will install the "base system", or, put another way, just enough to get your system up and functioning.

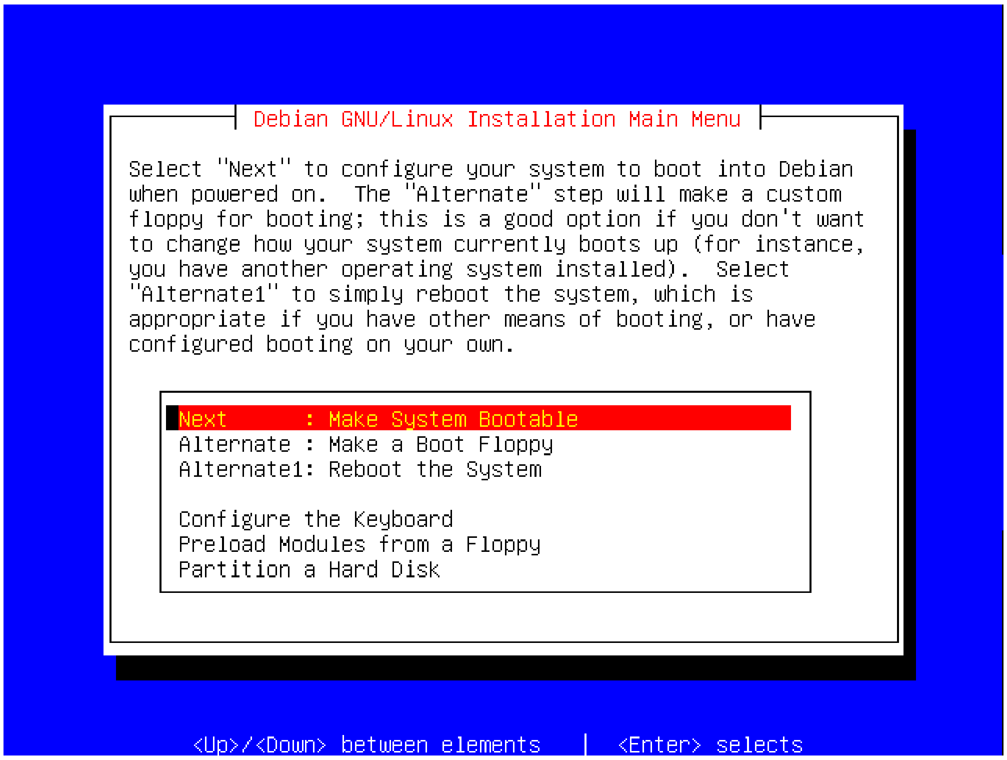
Press **Enter** to continue.



```
Retrieving
file:/instmnt/pool/main/libp/libpcap/libpcap0_0.6.2-2_i386.deb
```

This step will take awhile, as it will have to copy a large number of files off the CD-ROM, extract them and install them on your Linux partition.

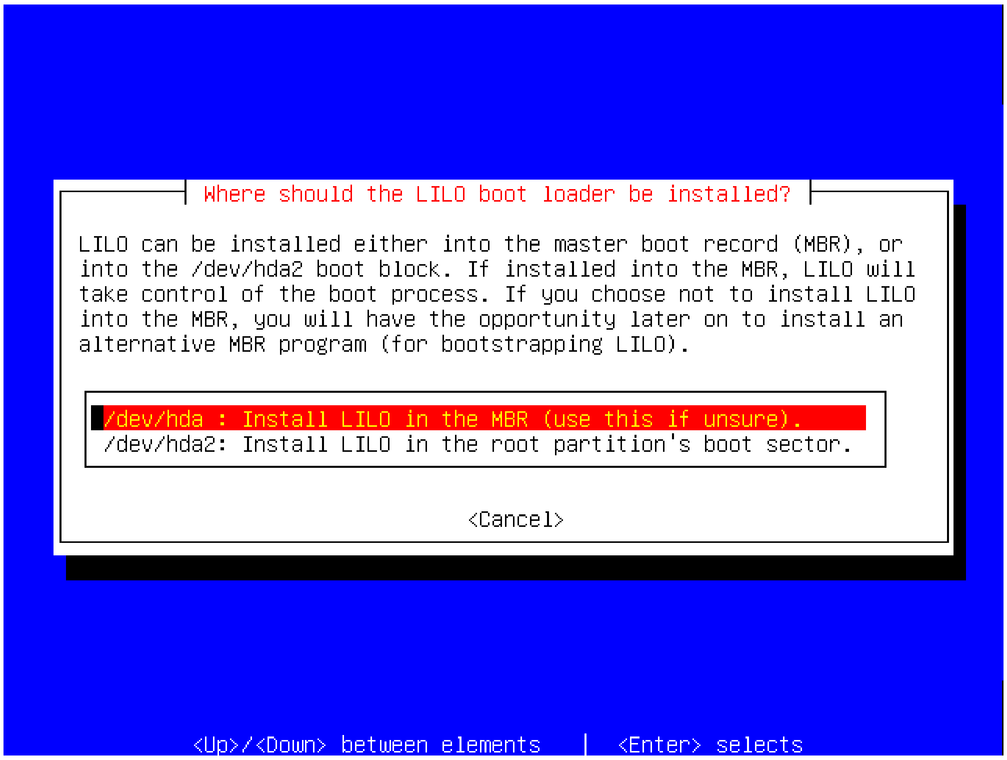
Make system bootable



```
Debian GNU/Linux Installation Main Menu

Select "Next" to configure your system to boot into Debian
when powered on.  The "Alternate" step will make a custom
floppy for booting; this is a good option if you don't want
to change how your system currently boots up (for instance,
you have another operating system installed).  Select
"Alternate1" to simply reboot the system, which is
appropriate if you have other means of booting, or have
configured booting on your own.

    Next       : Make System Bootable
    Alternate  : Make a Boot Floppy
    Alternate1: Reboot the System

    Configure the Keyboard
    Preload Modules from a Floppy
    Partition a Hard Disk

<Up>/<Down> between elements   |   <Enter> selects
```

Once the installation has finished, you'll be asked whether you wish to make the system bootable.

Press **Enter** to continue with this step.



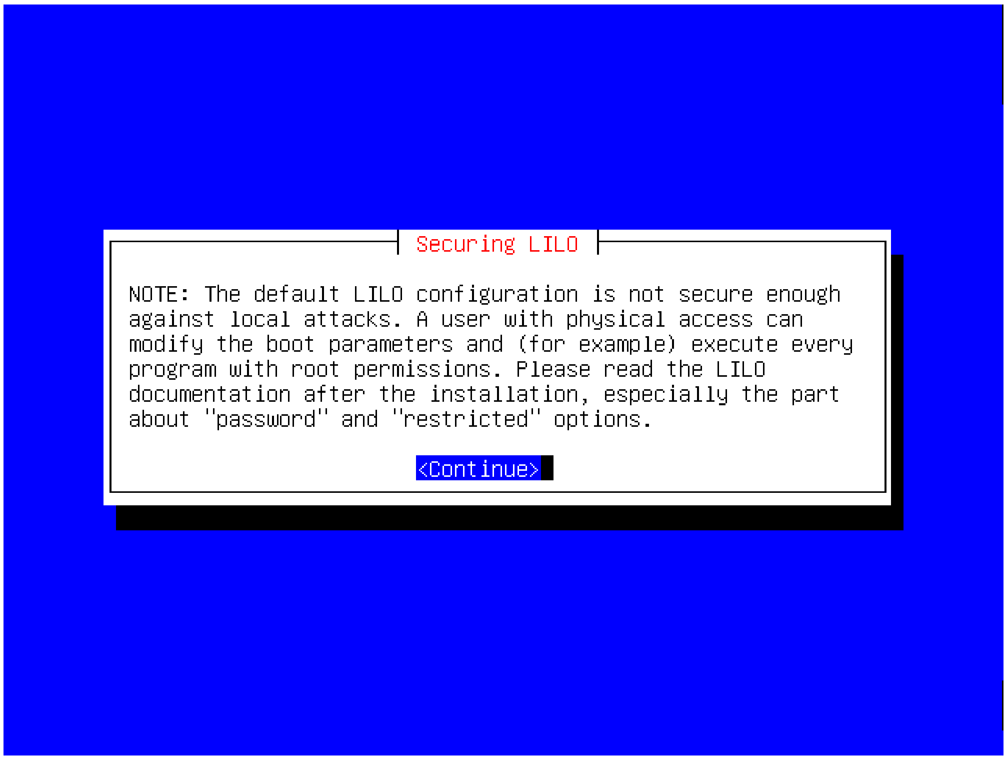
```
Where should the LILO boot loader be installed?

LILO can be installed either into the master boot record (MBR), or
into the /dev/hda2 boot block. If installed into the MBR, LILO will
take control of the boot process. If you choose not to install LILO
into the MBR, you will have the opportunity later on to install an
alternative MBR program (for bootstrapping LILO).

/dev/hda : Install LILO in the MBR (use this if unsure).
/dev/hda2: Install LILO in the root partition's boot sector.

<Cancel>

<Up>/<Down> between elements    |   <Enter> selects
```

For our example installation, we'll install the LILO boot loader in the MBR (Master Boot Record) of the hard drive. If you were installing onto a disk which already had a boot loader, you might opt to instead install LILO in the root partition (hda2, remember?) instead.

note

The space on your hard disk can be partitioned into a maximum of four (4) "primary partitions". Sometimes, your configuration requires that you have more than 4 partitions available. To address this problem, you can convert one of the 4 primary partitions into an "extended partition", within which you can create "logical partitions", which allows you to have more than 4 partitions on a single disk. Each partition can hold a different operating system, or your operating system may be spread over several partitions, each acting as a different drive letter (Windows) or mount point (Linux).

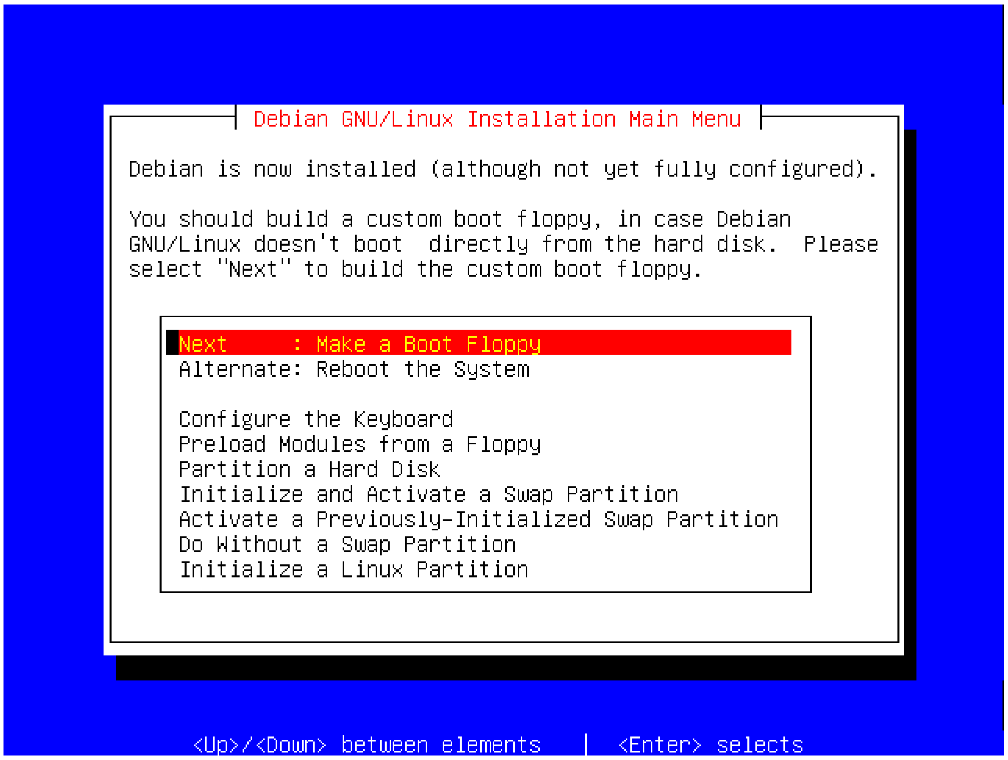
Press **Enter** to install LILO in the MBR.



```
NOTE: The default LILO configuration is not secure enough
against local attacks. A user with physical access can
modify the boot parameters and (for example) execute every
program with root permissions. Please read the LILO
documentation after the installation, especially the part
about "password" and "restricted" options.
```

If you're interested in keeping your machine secure from physical access, you should read this notice carefully.

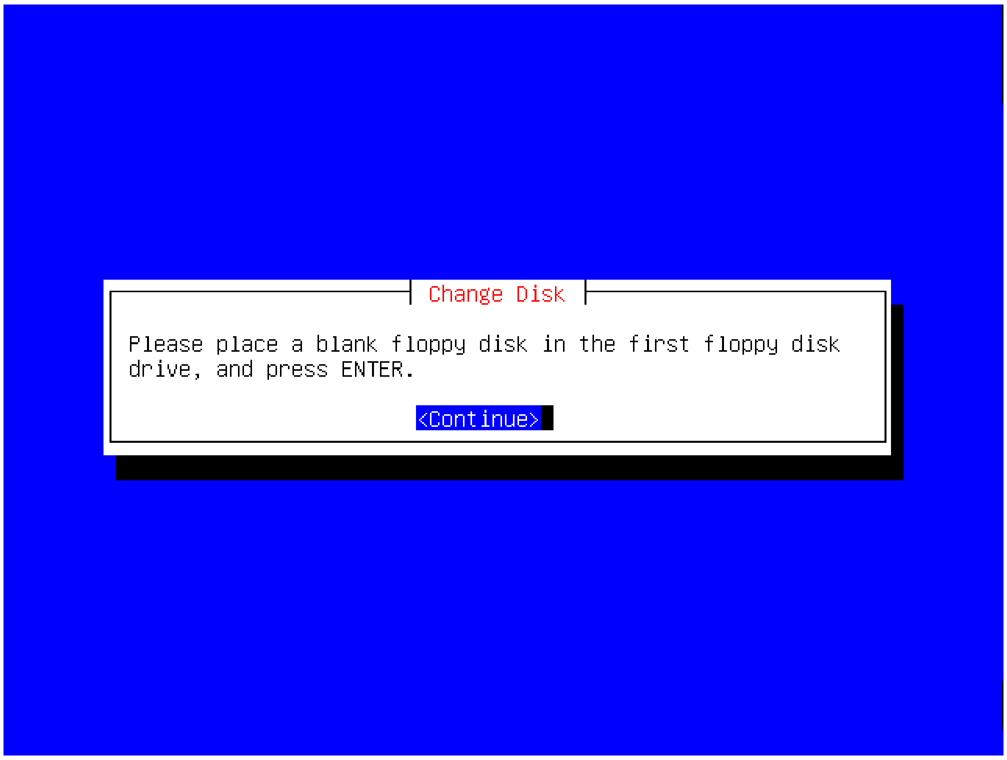
Make a boot floppy



```
Debian GNU/Linux Installation Main Menu

Debian is now installed (although not yet fully configured).

You should build a custom boot floppy, in case Debian
GNU/Linux doesn't boot  directly from the hard disk.  Please
select "Next" to build the custom boot floppy.

    Next      : Make a Boot Floppy
    Alternate: Reboot the System

    Configure the Keyboard
    Preload Modules from a Floppy
    Partition a Hard Disk
    Initialize and Activate a Swap Partition
    Activate a Previously-Initialized Swap Partition
    Do Without a Swap Partition
    Initialize a Linux Partition

<Up>/<Down> between elements   |  <Enter> selects
```

It's recommended that you make a boot floppy, it is a very useful troubleshooting tool; you could use the floppy disk to boot your Debian system if it has problems booting from the hard drive as an example.

Press **Enter**.



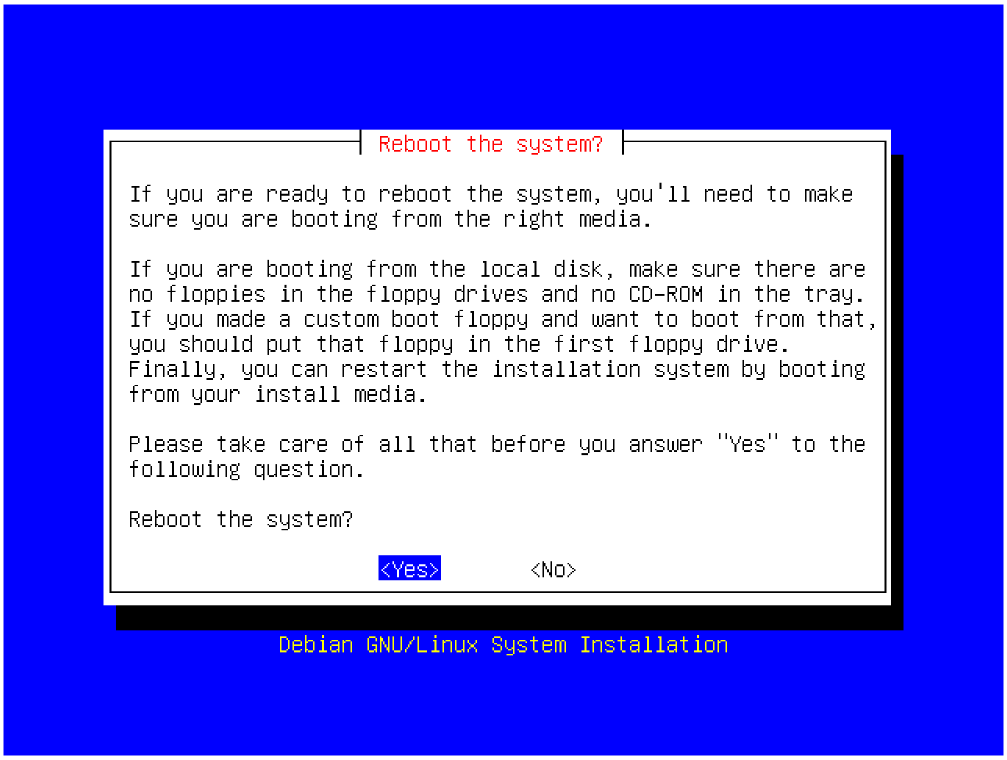
```
Change Disk

Please place a blank floppy disk in the first floppy disk
drive, and press ENTER.
```

Now insert a blank floppy disk, and hit **Enter** again.

Once you've made a boot floppy, you're ready to reboot the system and see if it works. Make sure that you remove the floppy that you have made as well as the installation CD.

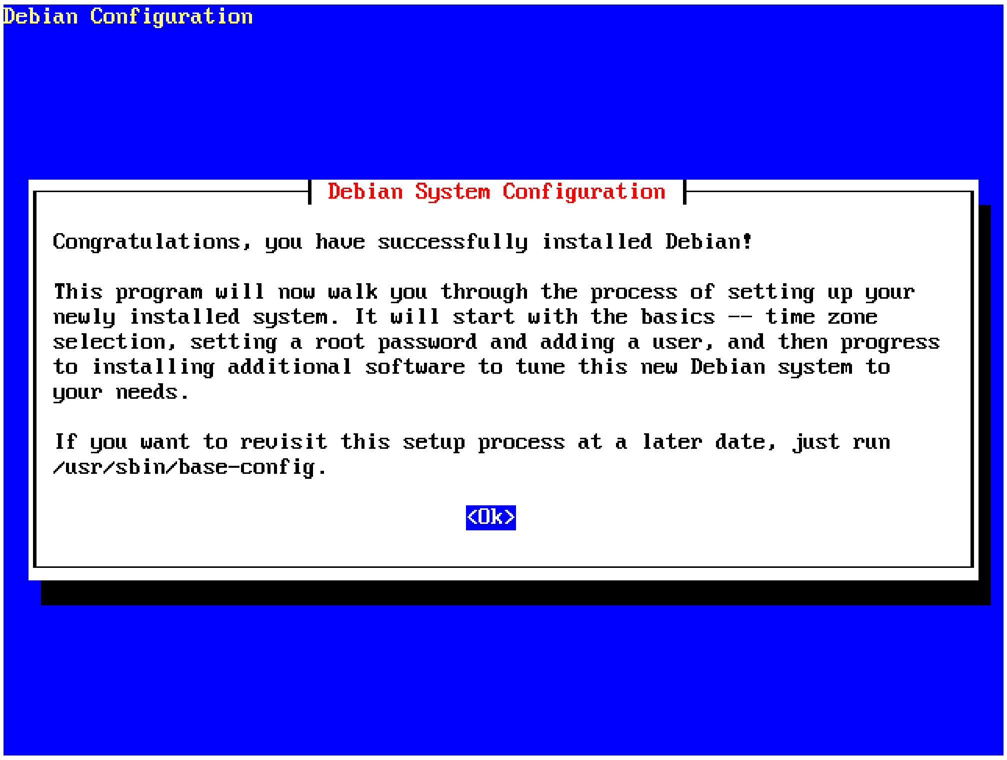
Once that's all done, press **Enter** to reboot.

Reboot



```
Reboot the system?

If you are ready to reboot the system, you'll need to make
sure you are booting from the right media.

If you are booting from the local disk, make sure there are
no floppies in the floppy drives and no CD-ROM in the tray.
If you made a custom boot floppy and want to boot from that,
you should put that floppy in the first floppy drive.
Finally, you can restart the installation system by booting
from your install media.

Please take care of all that before you answer "Yes" to the
following question.

Reboot the system?
```

Your system should then reboot, and you will hopefully be presented with a screen which looks like this:



```
Debian Configuration

                  Debian System Configuration

 Congratulations, you have successfully installed Debian!

 This program will now walk you through the process of setting up your
 newly installed system. It will start with the basics -- time zone
 selection, setting a root password and adding a user, and then progress
 to installing additional software to tune this new Debian system to
 your needs.

 If you want to revisit this setup process at a later date, just run
 /usr/sbin/base-config.

                                <Ok>
```

Congratulations!

Troubleshooting - Whoops, it didn't work, now what?

If you aren't presented with the "Congratulations" screen when you reboot, then something unfortunately must have gone wrong. The first thing I would suggest is attempting to boot off the boot floppy that you made earlier.

Some systems have problems with the LILO boot loader on the hard disk.

Post-installation configuration

Once you've successfully booted your system for the first time, Debian will ask you several questions to configure the packages that you have installed. You can revisit this process at any time by issuing the **/usr/sbin/base-config** command as root.

Time Zone

You will be asked whether you wish to set your hardware clock to GMT. As noted on the screen, if you are only running one operating system on your machine, then this setting is usually fine. However, if you are using a multi-boot system, you may wish to leave the hardware clock set at local time.

You will then be asked to select your time zone.

You can use the "arrow keys" and the **Enter** key to make your selections.

MD5 passwords

You will be prompted about whether or not to enable MD5 passwords. If this is going to be a stand-alone system, then MD5 passwords should work fine. If you are going to be integrating this system into an already existing network of Unix machines, you may wish to use the older DES encryption method. Again, make your selection with the "arrow keys" and then the **Enter** key.

The differences between MD5 and DES password encryption is covered in the Fundamentals section.

Shadow passwords

You should always enable shadow passwords unless you have a very good reason for doing otherwise. Not using shadow passwords leaves your system very vulnerable to local attacks.

Root password

You need to set a password for the "root" or administrator account. Read the instructions on the screen carefully and chose a password.

You'll be prompted to enter the password twice to verify that you didn't make any typos.

Choose a really good password for the root user? Remember that a person having access to the root account would have full permissions on the system.

Create a normal user account

It's usually always a good idea to create a normal user account. For our example, we'll create one called "student", but you can choose whatever your preferred username is.

Again, choose a good password for your normal user account.

ACACIA

If you're not installing on a laptop, Debian will detect this and ask if you wish to remove the ACACIA packages. This is safe to do, unless you know you'll be wanting ACACIA support for something.

PPP dial-up configuration

If you have a modem connected to your machine, and want to enable your Debian system to use it to connect to the Internet, then select **Yes**.

If you don't have a modem, or don't wish to connect to the Internet via PPP (perhaps you have a network card), then you can select **No**.

If you selected **Yes**, you'll be taken through the PPP Configuration Utility. You'll need to know your dial-up number, username and password. Your ISP should be able to provide you with these.

Configuring APT

APT is part of the Debian package management system.

You need to tell APT where it can obtain Debian packages from. If you have a permanent Internet connection, then it's usually a good idea to select a local Debian mirror to obtain packages from. If you don't have an Internet connection, then you can tell Debian to obtain its packages from CD-ROM instead.

APT is clever enough to be able to handle multiple CDs, so you could load all 7 official CD images into its database if necessary.

Since we've just installed off CD-ROM, this is probably the easiest way to continue, so insert the CD and then just select "cdrom" and press **Enter**.

APT will then prompt you for further CDs. If you have them, then feel free to load them in. Otherwise, select **No** and press **Enter**.

APT will then prompt you to select any additional package sources.

Select **No** and press **Enter**.

You'll also be asked if you wish to track security updates from the official Debian security site. If you have a connection to the Internet, this option is definitely recommended.

You can get back to this section after you've finished your installation by running the **apt-setup** command.

tasksel and dselect

You'll be asked if you wish to use "tasksel" (task select), which will allow you to tailor your system to perform specific tasks by installing the correct packages.

Select **Yes** and press **Enter**.

You can use the **Up**/**Down** arrow keys to scroll through the list of "tasks" in tasksel, and you can use the **left**/**right** arrow keys to select either the selection screen, the **Finish**, **Task info** or **Help** buttons.

While the selection screen is active, you can use either the **Spacebar** or **Enter** key to toggle which options you wish to install.

For a first time desktop installation, it is recommended that you select the "X window system" and "desktop environment" options.

Once you've toggle what you want, you can select the **Finish** button and hit **Enter**.

You'll now be prompted about whether you wish to use **dselect**. This is similar tool to "tasksel", but allows your more control over what software packages you're going to install. You can safely skip this step for now though.

Select **No** and press **Enter**.

You will then be presented with a long list of packages that are now going to be installed, as well as the summary on the disk space that will be required to perform this operation. You can select `Y` and **Enter** to continue, or `N` and **Enter** if you wish to abort the installation.

APT will then proceed to load the packages that it requires from the various sources that you'd told it about previously. If you told it to load packages off the CD-ROM, it will prompt you to make sure that the relevant CD is inserted, and to then press **Enter**.

Do this now.

binutils

You may receive an error message about the "binutils" package and a "Kernel link failure". You can just press **Enter** and safely ignore the message, as it is only pertinent if you are performing an upgrade of your system, rather than an installation.

less

You will be prompted to configure the **less** command; you can also safely just press **Enter** here.

locale

You will be prompted to chose which locale(s) you wish to support.

statd and tcpwrappers

You will be notified that the statd daemon uses tcpwrappers. Again, just hit **Enter**.

OpenSSH

You will be prompted about whether to allow only SSH protocol 2.

Again, just hit **Enter** to select **Yes**.

You will also be notified about OpenSSH's privilege separation, and be given the option of installing ssh-keysign with the SUID bit set.

You can hit **Enter** through these two options.

You will also be prompted about whether you wish to run the OpenSSH server. If you intend to remotely access this machine, then you should enable the server. Otherwise, you can leave this off.

psfontmgr and paper size

Enable this if you have a PostScript printer. Most modern printers are.

You will also be prompted for your default paper size.

GDM

You'll be asked to select a default X Windows display manager. You can just select the default.

Mozilla

You should enable TrueType fonts for Mozilla. It looks prettier.

You can leave the "tsp wrapper" set to "none".

Xserver

You should opt to have your X server wrapper managed by "debconf".

You should opt to have your X server configuration file managed by "debconf".

You'll need to select which X server you wish to use, based on your video card. Select "vesa" if you're unsure. Don't worry, you can always come back here and try later.

You can opt to use the kernel framebuffer device interface.

For a beginner, it's easiest to use the `Simple` option to configure your X Windows System.

Select your monitor size, resolution and color depth.

You should now see quite a few lines of text scrolling up your display. These are the individual packages being unpacked and installed on your system.

Once this has been done, the newly installed packages will be configured. Some of this configuration will require your input.

ispell

You'll be asked which dictionaries you wish to use, select either 1 or 2, and then press **Enter**.

You'll be asked if you wish to erase and previously downloaded .deb files. You can just press **Enter** here.

exim

The default MTA[2] that comes with Debian is Exim. You'll be prompted to let Debian know what sort of mail system configuration your environment uses.

You can safely choose "4) Local delivery only".

Provide your normal user account as the address which should receive postmaster-mail.

You should, finally, be presented with a "Have fun!" screen.

Well done, your Debian system is up and running and ready to use!

[2]Mail Transfer Agent

Logging In

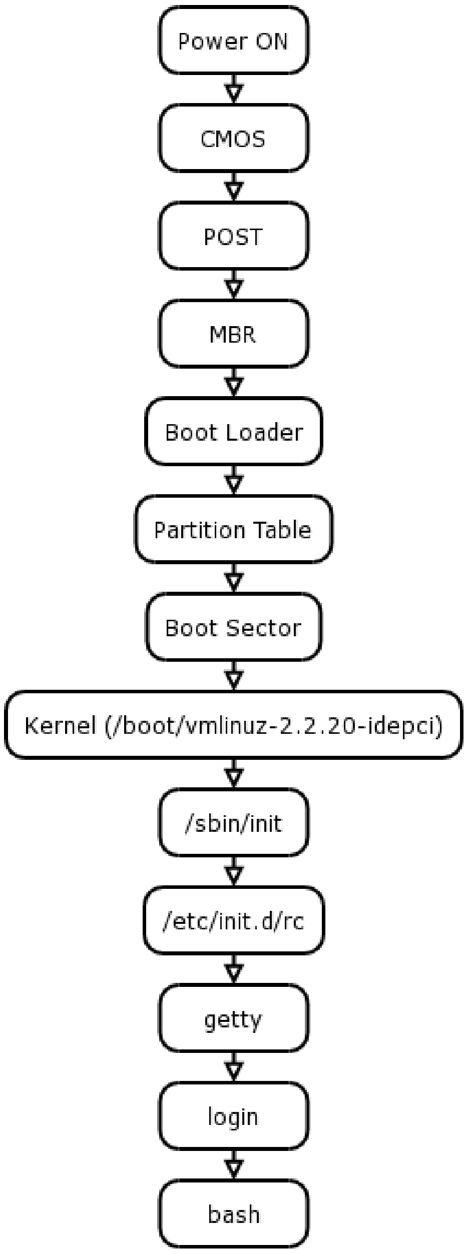
Now you should be ready to finally log in! You should make your initial login with the normal user account that you created earlier.

Once you've logged in under your normal user account, you can then use the "su" (substitute user) command to become the super user, also called the "root user", if you are going to be performing any administration tasks.

```
Linux baloo.zoo.org.za 2.2.20-idepci #1 Sat Apr 20 12:45:19 EST 2002 i68

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

student@debian:~$ su
Password:
debian:/home/student#
```

Bootup

Figure 2.1. The Bootup sequence



This chapter will describe the steps that your machine takes from the time that it is turned on up until your operating system is fully loaded.

This only covers the sequence of events on x86 (Intel 80x86 or compatible). If you are using a Sparc or Alpha architecture machine, the process will be similar, but not identical.

Switch on BIOS and CMOS

When you start your machine, it goes through a process known as "bootstrapping", or "booting".

Simply put, your computer doesn't know what to do when you turn it on, so it has to go and fetch some instructions from somewhere. It will initially start the BIOS[3] off your machines CMOS[4] chip.

As the name implies, the BIOS can handle very simple read and write procedures on your machine, usually the system's hard drive.

Before the BIOS does anything else, it will initiate what is known as POST[5]. These are a series of very simple tests to check that the hardware connected to the system (such as RAM, hard drives, video) are functional.

If you system fails POST, it will emit a series of beeps. You can usually look up the meaning of these beeps in your motherboard's manual.

Master Boot Record

Once POST has been passed, the CMOS will then examine the first sector of your hard disk for the Master Boot Record (MBR), which will contain a boot loader. If it cannot find the MBR, or cannot find a boot loader, the CMOS will halt with an error.

Remember that the space on your hard disk can be partitioned into a maximum of four (4) "primary partitions". Sometimes, your configuration requires that you have more than 4 partitions available. To address this problem, you can convert one of the 4 primary partitions into an "extended partition", within which you can create "logical partitions", which allows you to have more than 4 partitions on a single disk.

Each partition can hold a different operating system, or your operating system may be spread over several partitions, each acting as a different drive letter (Windows) or mount point (Linux).

Boot Loader (LILO)

Once the boot loader has been loaded off the MBR, it will take over control of the machine.

[3]Basic Input/Output System
[4]Complementary Metal Oxide Semiconductor
[5]Power-On Self-Test

The boot loader may have been configured to boot a single operating system, or it may provide the user with a choice of operating systems to choose from. Such functionality is available from the *Li*nux Boot *Lo*ader, LILO.

Once the boot loader knows which partition you want to boot, it will examine the first sector of that partition, also known as the boot sector, for the boot program. If you're booting Linux, this will also be LILO.

Once the boot program has been loaded, it will be able to examine the Linux filesystem and be able to load the Linux kernel into memory and execute it. It will also be able to pass instructions on to the Linux kernel.

The Linux kernel will then proceed to probe the hardware in the machine based on the setting the kernel was compiled with. These will include the device drivers, which enable to kernel to read from the hard disk.

"init" and Run Levels

The default instruction will be for the kernel to load up the "init" process.

The init process is known as the parent of all processes and is generally stored in the **`/sbin`** directory.

The kernel will tell "init" what "runlevel" to start the system at. Runlevel 2 is the default runlevel for Debian.

The runlevels, and what happens at each level, are configured in the **`/etc/inittab`** file.

Table 2.4. Runlevels

Runlevel	action
0	halt the system
1	single user or maintenance mode
2-5	multi-user mode
6	reboot

The init process will then call a shell script called "rc"[6], with the runlevel as its parameter, which will then start up various system services, based on the runlevel that init was given.

The "rc" script will examine the following directory[7]:

[6]The name "rc" is short for "run commands", and originates from a script creation utility called "runcom", which was present in Cray's predecessor to both Multics and Unix.

`/etc/rc#.d/`

Where the "#" is replaced by the current runlevel. For a machine, which is booting normally, this would be runlevel 2, so rc would be examining the contents of:

`/etc/rc2.d/`

In that directory are a collection of scripts.

What is important is their names and the first letter of the name must be a capital "K" (short for kill, used when shutting down the system) or a capital "S" (short for start, used when booting up the system).

Anything else is ignored.

```
debian:~# ls /etc/rc2.d/
S10sysklogd  S19nfs-common  S20lpd
S20ssh  S89cron S99rmnologin
S11klogd     S20exim        S20makedev
S20xfs  S99gdm S99xdm
S14ppp       S20inetd       S20nfs-kernel-server
S89atd  S99kdm
```

The "rc" script will call all "K" scripts with the "stop" parameter, this instructing those processes to stop. It will then call all "S" scripts with the "start" parameter, instructing those processes to start up.

The second and third characters of the script names are usually numbers. Since "rc" executes the scripts in order, you can use this number to determine the order in which processes are started and stopped.

This is useful for doing things like making sure that the network is up and configured before starting your web server, etc.

One of the rc2.d scripts will examine a file called "fstab" (short for file system table) in your **`/etc`** directory. This file lists all the filesystems that you want your system to mount at boot time.

Mounting Filesystems

The script will then examine this file, and mount the filesystems if appropriate.

```
debian:~# cat /etc/fstab
# /etc/fstab: static file system information.
```

[7] The ".d" in the directory name indicates that, historically, this used to be a single file, but was then later split into several smaller files, all of which are now present in this directory.

```
#
# <file system> <mount point>   <type>  <options>  <dump>  <pass>
/dev/hdb2       /               ext2    errors=remount-ro       0       1
/dev/hdb1       none            swap    sw                      0       0
proc            /proc           proc    defaults                0       0
/dev/fd0        /floppy         auto    user,noauto             0       0
/dev/cdrom      /cdrom          iso9660 ro,user,noauto          0       0
```

- The "file system" column contains the device, or special name in the case of **`proc`**, for the file system.
- The "mount point" specifies where that file system will be mounted.
- The "type" column indicates what type of file system it is. The standard type under Debian Linux is "ext2".
- The "options" column specifies any special options that are to be used for this filesystem.
    - ro = read only
    - noauto = don't mount at boot time
- The "dump" number indicates the level the drive should be dumped, or backed up at. A level 0 means a full backup, while higher numbers mean incremental backups.
- The "pass" number indicates the order that the file systems should be checked with "fsck" when the system boots. The root ("/") file system should always be checked first, and so should have a pass number of 1. A pass number of "0" means that the file system won't be checked with "fsck"

Obviously, some of the files may exist in more than one **`rc#.d`** directory. So, in order to ease administration (imagine having to make a change in a file common to all run levels), and to save space, the actual scripts live in the **`/etc/init.d/`** directory, and are then symlinked into their relevant **`/etc/rc#.d`** directories.

```
debian:~# ls -la /etc/rc2.d
total 8
drwxr-xr-x    2 root  root  4096 Mar 12 08:46 .
drwxr-xr-x   59 root  root  4096 Mar 12 09:08 ..
lrwxrwxrwx    1 root  root  18 Mar 12  2004
        S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx    1 root  root  15 Mar 12  2004
```

```
        S11klogd -> ../init.d/klogd
lrwxrwxrwx    1 root  root  13 Mar 12  2004
        S14ppp -> ../init.d/ppp
lrwxrwxrwx    1 root  root  20 Mar 12 08:44
        S19nfs-common -> ../init.d/nfs-common
lrwxrwxrwx    1 root  root  14 Mar 12  2004
        S20exim -> ../init.d/exim
lrwxrwxrwx    1 root  root  15 Mar 12  2004
        S20inetd -> ../init.d/inetd
lrwxrwxrwx    1 root  root  13 Mar 12 08:44
        S20lpd -> ../init.d/lpd
lrwxrwxrwx    1 root  root  17 Mar 12  2004
        S20makedev -> ../init.d/makedev
lrwxrwxrwx    1 root  root  27 Mar 12 08:44
        S20nfs-kernel-server -> ../init.d/nfs-kernel-server
lrwxrwxrwx    1 root  root  13 Mar 12 08:44
        S20ssh -> ../init.d/ssh
lrwxrwxrwx    1 root  root  13 Mar 12 08:46
        S20xfs -> ../init.d/xfs
lrwxrwxrwx    1 root  root  13 Mar 12  2004
        S89atd -> ../init.d/atd
lrwxrwxrwx    1 root  root  14 Mar 12  2004
        S89cron -> ../init.d/cron
lrwxrwxrwx    1 root  root  13 Mar 12 08:45
        S99gdm -> ../init.d/gdm
lrwxrwxrwx    1 root  root  13 Mar 12 08:46
        S99kdm -> ../init.d/kdm
lrwxrwxrwx    1 root  root  19 Mar 12  2004
        S99rmnologin -> ../init.d/rmnologin
lrwxrwxrwx    1 root  root  13 Mar 12 08:46
        S99xdm -> ../init.d/xdm
```

Once the system has reached the state required by the contents of the rc#.d directory, the user will be presented with a login screen.

Getty, issue and motd

The text-based login screen is an application called "getty", which displays the contents of **`/etc/issue`**, which is a file that usually holds the name of the operating system, as well as the terminal name.

```
Debian GNU/Linux 3.0 baloo.zoo.org.za
baloo.zoo.org.za login: root
```

One you provide your username, the getty process then spawns another process called "login".

This asks you for your password, and then checks your answer against the contents of **`/etc/passwd`**, or equivalent authentication mechanism.

If the match succeeds, the login program displays the contents of the **/etc/motd** file, and then replaces itself (using the exec( ) system call) with a copy of your shell, running under your user id.

```
Password:
Last login: Wed Mar 24 10:02:19 2004 on tty2
Linux baloo.zoo.org.za 2.2.20-idepci #1 Sat Apr 20 12:45:19 EST 2002 i686
unknown

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Your shell will then process the **/etc/profile** file, and, if your shell is bash, the **/etc/bashrc and $HOME/.bash_login** files.

```
debian:~# pstree
init-+-atd
#### |-bash---pstree
#### |-cron
#### |-5*[getty]
#### |-inetd
#### |-keventd
#### |-kflushd
#### |-klogd
#### |-kswapd
#### |-kupdate
#### |-lpd
#### |-sshd---sshd---bash---top
#### `-syslogd
```

The Message of the Day (**/etc/motd**) file is useful for system announcements that you want your users to see when they log in. This could be something like your company policy on the use of the computing facilities, to a "Joke of the Day".

Shutting down:

You can use the "shutdown" command to gracefully shut the system down.

```
SYNTAX:
shutdown [ -r | -h ] time [ message ]
```

The "-r" flag tells the system that you wish to reboot once everything's been shut down. The "-h" flag instead tells the system that you wish to simply halt it. Once a system is halted, you will have to power it off and then on again manually.

If you don't specify either of these flags, the system will be brought down into maintenance, or single user mode.

The "time" parameter specifies when the shutdown is to take place. This can be specified either in HH:MM format (i.e., 01:00 means shut down at 1AM), or in "+minutes", where minutes is the number of minutes from the time the command is issued until the shutdown process begins. You can also specify "now" to shut down the system immediately.

The "message" options specifies an optional message which will be broadcast to all shell users currently logged onto the system.

Examples:

Shutdown the system now, for a reboot:

```
debian:~# shutdown -r now

Broadcast message from root (pts/0) (Sat Mar 20 03:34:37 2004):

The system is going down for reboot NOW!
```

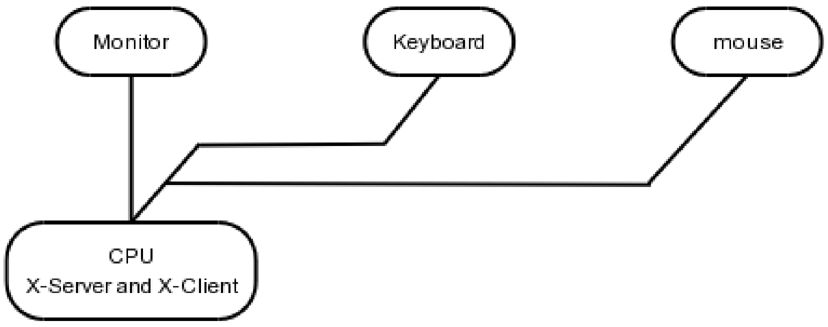
You can use the "shutdown" and "reboot" commands to gracefully shutdown the

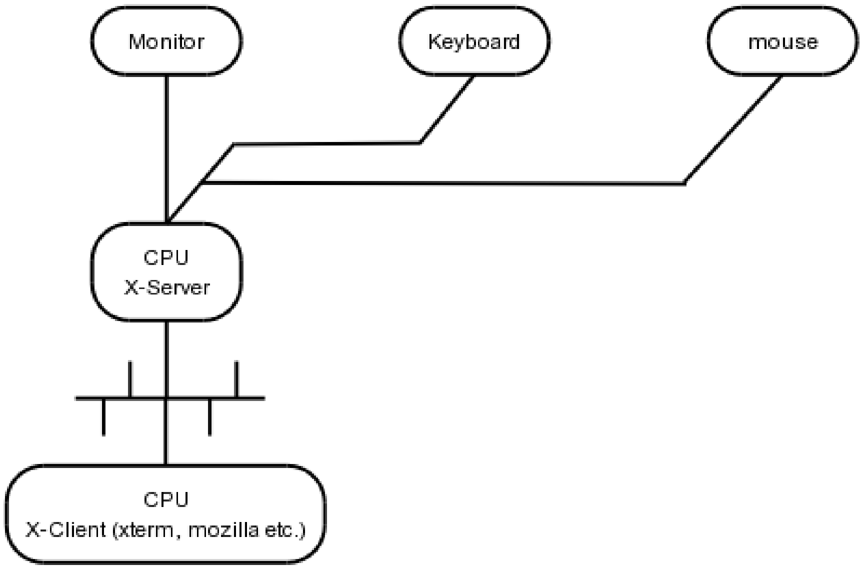
The X11 system

The standard graphical user interface (GUI) that comes with Debian is called XFree86. This is a free implementation of the X Windows System (X11R6), written for the x86 architecture.

The X Windows System uses the client/server architecture. The server runs on the machine which has the video display hardware, while the clients run can run on the same machine, or on a remote system.

Figure 2.2. XWindow client/server model





This means that your X Windows server must know how to handle your video display hardware.

Your X Windows clients are your graphical applications; examples include "xterm", "mozilla" and "openoffice".

In addition, you usually need a system to manage to each of the "windows" that your applications generate. This system is called your "window manager". It is responsible for putting borders around your application windows, and allowing you to perform actions such altering the size (know as "geometry" in X11 parlance) and location of the window on the screen.

Some of the popular window managers include "fluxbox", "fvwm" and "afterstep".

In addition to window managers, you also get "integrated desktop environments".

These are a collection of applications, usually including a menu system, a window manager and a graphical file manager that supports drag'drop between the various desktop elements.

Examples of popular desktop environments include KDE, Gnome and XFCE.

You can find a good summary of available window managers and desktop environments here [http://xwinman.org/]

There are two ways of allowing access to your X Windows system.

You can make use of the **startx** command from a local console, after you've logged in.

Alternatively, you can configure an X Windows Display Manager. The default display manager that comes with XFree86 is called plain "XDM".

There is a KDE specific one called "KDM" and a Gnome specific one called "GDM"; the KDE and GNOME Display Managers, respectively.

All three of these applications provide a graphical login screen, and can be accessed either via the local console, or via the network by means of the XDMCP protocol.

You can use the following command under Debian to configure XFree86 (as root):

```
dpkg-reconfigure xserver-xfree86
```

This will take you through the same configuration procedure as when you first installed it earlier (see the section called “Configuring X Windows” [82]).

Changing your bootloader from LILO to GRUB

The default Debian installation uses the Linux Boot Loader (LILO), and it performs perfectly for simple boot configurations. However, a lot of people find the GRand Unified Bootloader (GRUB) easier to use and more powerful and thus better suited for their needs.

Switching from LILO to GRUB on Debian is a relatively easy process.

First, you'll need to install the GRUB package:

```
debian:~# apt-get install grub
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  grub
0 packages upgraded, 1 newly installed, 0 to remove and 2  not upgraded.
```

```
Need to get 0B/247kB of archives. After unpacking 504Kb will be used.
Media Change: Please insert the disc labeled &apos;Debian GNU/Linux 3.0 r2 _Wo
- Official i386 Binary-1 (20031201)&apos; in the drive &apos;/cdrom/&apos;
and press enter

Selecting previously deselected package grub.
(Reading database ... 9857 files and directories currently installed.)
Unpacking grub (from .../g/grub/grub_0.91-2_i386.deb) ...
Setting up grub (0.91-2) ...
```

Once that's done, you'll need to install the boot loader code on the device that you boot from.

In our example system, that's the second IDE drive on the system ("/dev/hdb"), so the command you would issue would be:

```
debian:~# grub-install /dev/hdb
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install&apos;.

(fd0)   /dev/fd0
(hd0)   /dev/hda
(hd1)   /dev/hdb
```

Once that's done, you'll need to run the **update-grub** command, which is Debian specific, and which will generate a GRUB menu configuration file for you, called **`/boot/grub/menu.lst`**.

```
debian:~# update-grub
Searching for GRUB installation directory ... found: /boot/grub .
Testing for an existing GRUB menu.list file...

Could not find /boot/grub/menu.lst file. Would you like one generated
for you? (y/N) y
Updating /boot/grub/menu.lst ... done


Please note that configuration parameters for GRUB are stored in
/boot/grub/menu.lst . You must edit this file in order to set the
options which GRUB passes to the kernel, as well as the drive which
GRUB looks in to for the kernel.

Everything on the line after &quot;kopt=&quot; is passed to the kernel
as parameters, and &quot;groot=&quot; must be set to the partition
(in GRUB terms, such as &quot;(hd0,0)&quot;) which GRUB will load the
kernel from.
```

```
After you have edited /boot/grub/menu.lst , please re-run
&apos;update-grub&apos;.
```

As the instructions tell you, you will need to edit the "kopt" and "groot" parameters in that file. You should use "vi" to do this:

```
debian:~# vi /boot/grub/menu.lst
```

```
(hd0,0) refers to the first hard disk's first partition
(hd1,0) refers to the second hard disk's first partition
(hd1,1) refers to the second hard disk's second partition
etc.
```

Once you've edited the **`menu.lst`** file to your satisfaction, you will need to re-run the **update-grub** command:

```
debian:~# update-grub
Searching for GRUB installation directory ... found: /boot/grub .
Testing for an existing GRUB menu.list file... found: /boot/grub/menu.ls
Updating /boot/grub/menu.lst ... done
```

Now reboot, and see if it has worked!

```
debian:~# shutdown -r now
```

Debian Package Management

Unlike the RPM package management system which RedHat Linux uses, Debian's packages files have a .deb extension.

Debian uses three different, but related, sets of tools to manage its packages: dpkg, dselect and apt.

dkpg

The **dpkg** command is the most analogous to RPM's "rpm" command; it allows you to add and remove packages, and to query currently installed packages, check their integrity and dependencies.

List installed packages:

```
dpkg --list
```

```
debian:~# dpkg --list
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad
||/ Name           Version        Description
+++-==========================================
ii  aalib1         1.4p5-13       ascii art library
ii  abiword        1.0.2+cvs.2002 Dummy package providing abiword
ii  abiword-common 1.0.2+cvs.2002 WYSIWYG word processor
ii  abiword-gtk    1.0.2+cvs.2002 WYSIWYG word processor based on GTK
ii  adduser        3.47           Add and remove users and groups
ii  apt            0.5.4          Advanced front-end for dpkg
ii  apt-utils      0.5.4          APT utility programs
ii  ark            2.2.2-9        An archiver for KDE
ii  asclock-themes 2.0.12-5       Theme files for ASclock, a clock applet
ii  at             3.1.8-11       Delayed job execution and batch processing
ii  base-config    1.33.18        Debian base configuration package
ii  base-files     3.0.2          Debian base system miscellaneous files
ii  base-passwd    3.4.1          Debian Base System Password/Group Files
[ ... ]
ii  xlibs          4.1.0-16woody1 X Window System client libraries
ii  xnest          4.1.0-16woody1 nested X server
ii  xprt           4.1.0-16woody1 X print server
ii  xserver-common 4.1.0-16woody1 files and utilities common to all X servers
ii  xserver-xfree8 4.1.0-16woody1 the XFree86 X server
ii  xspecs         4.1.0-16woody1 X protocol, extension, and library technical
ii  xterm          4.1.0-16woody1 X terminal emulator
ii  xutils         4.1.0-16woody1 X Window System utility programs
ii  xvfb           4.1.0-16woody1 virtual framebuffer X server
ii  zlib1g         1.1.4-1.0woody compression library - runtime
```

Information about a .deb file:

dpkg --info package.deb

Install a .deb file:

dpkg --install package.deb

Example Installation

We'll install the "bsdgames" package.

```
debian:~# mount /cdrom
```

```
debian:~# dpkg --info /cdrom/pool/main/b/bsdgames/bsdgames_2.13-7_i386.de
 new debian package, version 2.0.
 size 791782 bytes: control archive= 6220 bytes.
     841 bytes,    20 lines      control
   10117 bytes,   158 lines      md5sums
    1791 bytes,    56 lines   *  postinst             #!/bin/sh
     969 bytes,    32 lines   *  postrm               #!/bin/sh
    1580 bytes,    55 lines   *  preinst              #!/bin/sh
     198 bytes,     7 lines   *  prerm                #!/bin/sh
 Package: bsdgames
 Version: 2.13-7
 Section: games
 Priority: optional
 Architecture: i386
 Depends: libc6 (>= 2.2.4-4), libncurses5 (>= 5.2.20020112a-1),
 wenglish | wordlist
 Suggests: wenglish
 Conflicts: bsdgames-nonfree (<< 2.5), suidmanager (<< 0.50)
 Replaces: bsdgames-nonfree (<< 2.5)
 Installed-Size: 2064
 Maintainer: Joey Hess <joeyh@debian.org>
 Description: collection of text games from BSD systems
  This is a collection of some of the text-based games and amusements tha
  been enjoyed for decades on unix systems.
  .
  Includes these programs: adventure, arithmetic, atc, backgammon, battl
  bcd, boggle, caesar, canfield, countmail, cribbage, fish, gomoku, hang
  hunt, mille, monop, morse, number, pig, phantasia, pom, ppt, primes, q
  random, rain, robots, sail, snake, tetris, trek, wargames, worm, worms
  wump, wtf
debian:~# dpkg --install /cdrom/pool/main/b/bsdgames/bsdgames_2.13-7_i38
Selecting previously deselected package bsdgames.
(Reading database ... 36337 files and directories currently installed.)
Unpacking bsdgames (from .../bsdgames_2.13-7_i386.deb) ...
Setting up bsdgames (2.13-7) ...

debian:~# _
```

List files installed by a specific package:

```
dpkg --listfiles package
```

Let's list all the files which we've just installed that were part of the "bsdgames" package.

```
debian:~# dpkg --listfiles bsdgames
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/bsdgames
/usr/share/doc/bsdgames/BUGS.atc
/usr/share/doc/bsdgames/ChangeLog.gz
```

```
/usr/share/doc/bsdgames/README.linux.hunt
/usr/share/doc/bsdgames/README.phantasia
/usr/share/doc/bsdgames/README.boggle
/usr/share/doc/bsdgames/README.linux.boggle
/usr/share/doc/bsdgames/TODO.gz
/usr/share/doc/bsdgames/ChangeLog.0
/usr/share/doc/bsdgames/trek.me.gz
[ ... ]
/var/games
/var/games/bsdgames
/var/games/bsdgames/phantasia
/var/games/bsdgames/sail
/usr/share/doc/bsdgames/NEWS.gz
/usr/share/man/man6/teachgammon.6.gz
/usr/share/man/man6/rot13.6.gz
/usr/share/man/man6/morse.6.gz
/usr/share/man/man6/ppt.
```

Find out which package a specific file belongs to:

dpkg --search filename

```
debian:~# dpkg --search /usr/games/wtf
bsdgames: /usr/games/wtf
```

This is useful if you're going to be upgrading a package and want to preserve its existing configuration information.

Uninstall an installed package:

dpkg --remove package

Uninstall an installed package _and_ its associated configuration files:

dpkg --purge package

```
debian:~# dpkg --purge bsdgames
(Reading database ... 36513 files and directories currently installed.)
Removing bsdgames ...
dpkg - warning: while removing bsdgames, directory
`/var/games/bsdgames/phantasia&apos; not empty so not removed.
dpkg - warning: while removing bsdgames, directory
`/var/games/bsdgames' not empty so not removed.
dpkg - warning: while removing bsdgames, directory
`/var/games' not empty so not removed.
Purging configuration files for bsdgames ...
```

```

```

You'll notice the warnings about "/var/games" not being empty, but if you check now, you'll see that the directory has indeed been deleted:

```
debian:~# ls /var/games
ls: /var/games: No such file or directory
```

This is because the directory was removed as part of the "purging configuration files" step.

You should consult the dpkg(8) manual page for further information about this command.

dselect

The **dselect** command is a frontend to the **dpkg** command and APT system, and lets you access all its functions through a text based menu system.

apt

The APT (Advanced Package Tool) system is probably one of the reasons that Debian has gained such popularity as a Linux distribution.

Unlike dpkg and dselect, APT is able to handle dependencies and makes upgrading the software on your system very easy.

APT maintains a listing of packages and their locations.

You can use the **apt-cdrom** command to add a CD to the current listing. So, for example, you could insert the second Debian CD, and add the packages on it to the list of the ones available:

```
debian:~# apt-cdrom add
Using CD-ROM mount point /cdrom/
Unmounting CD-ROM
Please insert a Disc in the drive and press enter
Mounting CD-ROM
Identifying.. [580650a473d808bc27074b25dff224f7-2]
Scanning Disc for index files..  Found 4 package indexes and 0 source
indexes.
This Disc is called:
 &apos;Debian GNU/Linux 3.0 r2 _Woody_ - Official i386 Binary-1 (2003120
Reading Package Indexes... Done
Wrote 1167 records.
```

```
Writing new source list
Source List entries for this Disc are:
deb cdrom:[Debian GNU/Linux 3.0 r2 _Woody_ - Official i386 Binary-1 (20031201)
/ unstable contrib main non-US/contrib non-US/main
Repeat this process for the rest of the CDs in your set.
debian:~#
```

You can likewise use the **apt-ftp** command to add a remote FTP archive to your list of package sources.

To actually download and/or install packages, you can use the **apt-get** command, which takes the following parameters as commands:

Table 2.5. apt-get commands

Update	Retrieve new lists of packages
Upgrade	Perform an upgrade
Install	Install new packages
Remove	Remove packages
Check	Verify that there are no broken dependencies

So, to install the "bsdgames" package, we would run:

```

debian:~# apt-get install bsdgames
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  bsdgames
0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/792kB of archives. After unpacking 2114Kb will be used.
Media Change: Please insert the disc labeled &apos;Debian GNU/Linux 3.0
r2 _Woody_ - Official i386 Binary-1 (20031201)&apos; in the drive
&apos;/cdrom/&apos; and press enter

Selecting previously deselected package bsdgames.
(Reading database ... 36340 files and directories currently installed.)
Unpacking bsdgames (from .../bsdgames_2.13-7_i386.deb) ...
Setting up bsdgames (2.13-7) ...

debian:~# _
```

As you can see, APT is clever enough to know which CD the package is on, and will prompt you for the correct media when you ask it to install something.

You can also use the **apt-setup** command to configure the list of sources for packages.

RPM Package Management:

RPM, a cyclic acronym for RPM Package Management, forms the base of the RedHat Linux package management system, as well as several other distributions. Although it is not used for the base package management system in Debian Linux, you can still make use of it if required.

RPM packages are given a ".rpm" extension, but otherwise follow a naming scheme very similar to .deb packages. Although the package files are in a different format, the same basic operations are common to the different package management systems:

List installed packages:

rpm -qa

Information about a .rpm file:

rpm -qi package.rpm

Install a .rpm file:

rpm -U package.rpm

List files installed by a specific package:

rpm -ql package

Find out which package a specific file belongs to:

rpm -qf filename

Uninstall an installed package:

rpm -e package

Upgrading your Linux kernel in Debian

Debian makes it very easy to upgrade your kernel by using kernel packages.

We're going to attempt something more challenging though, and try to build a kernel of our own.

We are currently running Linux 2.2.20:

```
debian:~# uname -a
Linux debian 2.2.20-idepci #1 Sat Apr 20 12:45:19 EST 2002 i686 unknown
```

We're going to upgrade to Linux 2.4.18, so we need to install the following packages:

- ncurses-dev
- kernel-package
- kernel-source-2.4.18
- fakeroot

```
debian:~# apt-get install kernel-package kernel-source-2.4.18 ncurses-dev
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  kernel-package kernel-source-2.4.18
0 packages upgraded, 2 newly installed, 0 to remove and 0  not upgraded.
Need to get 0B/24.1MB of archives. After unpacking 24.9MB will be used.
Media Change: Please insert the disc labeled &apos;Debian GNU/Linux
3.0 r2 _Woody_ - Official i386 Binary-1 (20031201)&apos; in the drive
&apos;/cdrom/&apos; and press enter

Selecting previously deselected package kernel-package.
(Reading database ... 36515 files and directories currently installed.)
Unpacking kernel-package (from .../kernel-package_7.107_all.deb) ...
Selecting previously deselected package kernel-source-2.4.18.
Unpacking kernel-source-2.4.18
(from .../kernel-source-2.4.18_2.4.18-13_all.deb) ...
Selecting previously deselected package libncurses5-dev.
Unpacking libncurses5-dev
(from .../libncurses5-dev_5.2.20020112a-7_i386.deb) ...

Setting up kernel-package (7.107) ...

Setting up kernel-source-2.4.18 (2.4.18-13) ...

Setting up libncurses5-dev (5.2.20020112a-7) ...

debian:~# _
```

Now change into your staging area (**`/usr/local/src`** is a good place), and extract the kernel source into that directory:

```
debian:~# cd /usr/local/src/
debian:/usr/local/src# tar -xjf /usr/src/kernel-source-2.4.18.tar.bz2
debian:/usr/local/src# cd kernel-source-2.4.18/
debian:/usr/local/src/kernel-source-2.4.18# _
```

Now you can run **make xmenuconfig** if you want to use the X configuration interface; alternatively you can run **make menuconfig** to use the text based interface. The **make** command looks for a **`Makefile`** (often called Makefile, or makefile), which you can think of as a recipe in order to be able to build something. You can optionally tell make what it is that you want to build, and make will inspect the relevant section in the Makefile to ascertain:

- the ingredients, or dependencies - what is required to make what is requested and
- the instructions - how to take the dependencies and turn them into the requested target

```
debian:/usr/local/src/kernel-source-2.4.18# make menuconfig
```

Peruse the menu, and make your selections. Once that's done, Exit and Save your kernel configuration file.

```
Saving your kernel configuration...

*** End of Linux kernel configuration.
*** Check the top-level Makefile for additional configuration.
*** Next, you must run &apos;make dep&apos;.

debian:/usr/local/src/kernel-source-2.4.18# _
```

Clean the source tree and reset the kernel-package parameters:

```
debian:/usr/local/src/kernel-source-2.4.18# make-kpkg clean
```

Compile the kernel:

```
debian:/usr/local/src/kernel-source-2.4.18# fakeroot make-kpkg /
/--revision=tsf.1.0 kernel_image
```

This command will build a kernel package which we can then install later. You can make the revision anything you want, although it's probably a good idea to give it a meaningful name so that you know what it is.

This can take a long time; probably best to go and take a break and come back and see if it's finished later.

```
[ ... ]
make[2]: Leaving directory `/usr/local/src/kernel-source-2.4.18&apos;
rm -f stamp-build
touch stamp-image
make[1]: Leaving directory `/usr/local/src/kernel-source-2.4.18&apos;
debian:/usr/local/src/kernel-source-2.4.18# _
```

Once it's finished, you'll be able to simply install the new package, and have your kernel upgraded:

```
debian:/usr/local/src/kernel-source-2.4.18# dpkg --install/
/  ../kernel-image-2.4.18_tsf.1.0_i386.deb
```

The package installation procedure will offer to make a boot floppy with your new kernel, as the original installation did with the original kernel. Again, it's suggested that you make one.

The package installation will also install a new MBR for you, if it is needed.

Now reboot, and see if your new kernel works!

Configuring X Windows

As mentioned previously, there are two methods for starting XFree86:

startx

You can use the **startx** command to start up the X11 server and client system from the console. X Windows sessions started via this method will consult the **`.xinitrc`** file in the user's home directory for commands to execute when starting their session.

xdm

You can use the X Display Manager, or equivalent, to provide a GUI login screen on the console. Using the XDMCP (XDM control protocol), it is also possible to make this login screen available to remote systems over the network.

X Windows sessions started via this method will consult the **`.xsession`** file in the user's home directory for commands to execute when logging in.

Installing and running KDE:

KDE is a very complex arrangement of different packages. However, you can use apt to easily install them all:

```
debian:~# apt-get install kde*
```

Once apt has installed the packages, you can use the following command to start up your X session with the KDE desktop environment:

```
student@debian:~$ startkde
```

Debian will also default to now starting up the KDM graphical login interface at startup time.

Installing and running Gnome:

Like KDE, Gnome is also composed of several different packages, and can be easily installed via apt:

```
debian:~# apt-get install gnome*
```

As with KDE, the GDM graphical login interface will be enabled at bootup. If you already have installed KDE, you will be prompted to select with interface you wish to use.

networked installations:

Although not covered in detail in this course, it is possible to install Debian directly from the network, using only a small bootable image. You can find more information on this installation method here: http://www.debian.org/CD/netinst/ [http://www.debian.org/CD/netinstsystem resources]

It is also possible to automate your Debian Linux installations. This might be useful in a Linux lab or Linux cluster installation. The current method of doing this is known as FAI (Fully Automatic Installation), and you can obtain more information about it here:http://www.informatik.uni-koeln.de/fai/

You should also consult the Linux Documentation Projects notes on performing network installations: http://www.tldp.org/HOWTO/Network-Install-HOWTO.html

Chapter 3. File Types and File Systems

Let's review the different file types

Regular files

A simply definition of a regular file would be that it is a one dimensional assortment of bytes that are stored on a disk or other mass storage devices.

1. A program that uses a file needs to know the structure of the file and needs to interpret the file contents. This is because no structure is imposed on the contents of a file by the operating system itself. This is a very powerful feature as it means that you could work with any file that you need to work on e.g. a DOS file.

2. Files are presented to the application as a stream of bytes and then an EOF condition.

3. However the EOF condition is not typed in, it is merely that the stream of bytes is as long as the file size is and then it is at the end. In other words it is actually a sort of offset that will happen when a program attempts to access an offset larger than the file size itself.

4. There are many different types of regular files, text, binary, executable etc. When using the **file** command to establish a file type the command accesses the magic database. If you get a chance have a look at the magic file and see how many different types of files Linux could support.

5. A regular file is referenced by an inode number (see the section called "Inodes" [88]).

Directory files

A simple definition of a directory is that it is a file that provides a mapping mechanism between the names of files and the files (datablocks) themselves.

1. A directory is also called a file. Its purpose is really to make sure that there is a good structure maintained on the system - sort of like a good filing system.

2. The directory only holds inode numbers and filenames. Yet this is also vitally important as this is the only place where a filename is referenced by its inode.

3. If you delete a file from a directory the entry in the list is zeroed and this is then called a shadow inode. The inode is then freed up.

Device files

A device file refers to a device driver and these are important to the kernel. The device drivers are written in C and compiled to make an object file and then placed as part of the kernel. Created a device file using the mknod command.

1. The files in **`/dev`** are used to ensure that we can access hardware such as the printer, cdrom, network etc.

2. If you look at the way Linux uses a device driver, it handles many of the functions that we could compare to the way DOS uses the BIOS. However the differences are often the reason why a piece of hardware that would work with a DOS related system will not work with a Unix or Linux related system. Linux will either see or not see a non-standard piece of hardware.

3. Here we can read and write directly to the device, so the user issues a system call to a device, the kernel performs a successful open on that device, if busy the read/write routine cannot operate, if not busy then reads or writes directly to that device.

4. There are different types of device files:

   - Character device files - writes to and from the device a character at a time. Indicated by a "c" in the first field. Very little preliminary processing required by the kernel so the request is passed directly to the device.

   - A block device only receives a request once block buffering has taken place in the kernel. Indicated by a "b" in the first field. A filesystem is an example of a block buffering device. Block devices generally have an associated character device - for example if you format a diskette you would use the character device file name, if backing up to that diskette you would use the block device name where the blocking is handled in the kernel buffer cache.

Interprocess communication Devices

It is possible in Linux to set up device files that allow communication between processes. Some of these are pipes, semaphores and shared memory devices and we have already seen the effectiveness of using un-named pipes.

Theoretically these are special files and they are handled in a similar way to the device files however they are not true device files. They are created when needed and then removed.

The scope of these definitions moves into development and therefore we are only going to discuss the theory of such, it is important to have an understanding of these features as the system administrator.

Pipes

We are thus far used pipes as un-named pipes when doing shell scripting, however there are also named pipes available for us to use.

In a named pipe situation the processes are being run by the same process group and by the same user. e.g.

```
ls  -li  |  more
```

An un-named pipe would be created as a node using the mknod command and here you could use it when running processes within different process groups. The un-named pipe has to be specified to both processes separately. e.g. When the output from one process becomes the input to another process running in the background.

Semaphore

The shortest definition would be that it is a data structure used by several processes to control and synchronize the operations on one resource.

e.g. If more than one user is accessing a record in a database it is likely that a semaphore would either gain exclusive access for each process in turn, or lock the record being updated.

Shared Memory

This type of file is indicated by an "m" in the first field. A piece of user-memory is allocated as a work-space, where it is possible for a process to read the data at the same time as another writes it.

Major and Minor Device Numbers

Each device is referenced by numbers, which are read when the kernel needs to use a device and subsequent device driver.

These numbers are called the major and minor device numbers.

The major device number refers to the device driver that should be used to access the device and it is a unique number, whereas the minor device number points to the device itself.

If you do a long listing in the **`/dev`** directory you will see that they are represented in the 6th column of the long listing report.

Inodes

As a file is created a unique number is ascribed to that file and this is called an inode number. The inode holds specific information about the file such as:

- The permission mode assigned to that file (at creation time this would have been assigned with the system umask)
- The number of links in place for the file
- The file owners UID number
- The group GID number
- The file size represented in bytes
- The address of the datablocks (or major and minor device numbers)
- The time the file was last modified
- The time that file was last accessed (useful for housekeeping for if a file has not been accessed for 3 years maybe we could remove it)
- The time any part of the inode was changed

When an inode resides on the disk it is called a disk inode, however when a file is opened, the kernel puts the inode onto a generic inode table and the inode is called a generic inode. Inodes are stored on the generic table with a link to a hash queue and this is stored in table that is reference by the kernel each time it opens a file to make sure that the file is not already open.

A detailed look at the filesystem in

Debian

A typical Linux system has, among others, the following directories:

- /

  This is the root directory. This is where the whole tree starts.

- **`/bin`**

  This directory contains executable programs which are needed in single user mode and to bring the system up or repair it.

- **`/boot`**

  Contains static files for the boot loader. This directory only holds the files which are needed during the boot process.

- **`/dev`**

  Special or device files, which refer to physical devices.

- **`/etc`**

  Contains configuration files which are local to the machine. Some larger software packages, like X11, can have their own subdirectories below **`/etc`**. Site-wide configuration files may be placed here or in **`/usr/etc`**. Nevertheless, programs should always look for these files in **`/etc`** and you may have links for these files to **`/usr/etc`**.

- /etc/skel

  When a new user account is created, files from this directory are usually copied into the user's home directory.

- /etc/X11

  Configuration files for the X11 window system.

- /home

  On machines with home directories for users, these are usually beneath this directory, directly or not. The structure of this directory depends on local administration decisions.

- /lib

This directory should hold those shared libraries that are necessary to boot the system and to run the commands in the root filesystem.

- /mnt

  is a mount point for temporarily mounted filesystems

- /proc

  This is a mount point for the proc filesystem, which provides information about running processes and the kernel.

- **`/sbin`**

  Like **`/bin`**, this directory holds commands needed to boot the system, but which are usually not executed by normal users.

- /tmp

  This directory contains temporary files which may be deleted with no notice, such as by a regular job or at system boot up.

- /usr

  This directory is usually mounted from a separate partition. It should hold only sharable, read-only data, so that it can be mounted by various machines running Linux.

- /usr/X11R6

  The X-Window system.

- /usr/bin

  This is the primary directory for executable pro grams. Most programs executed by normal users which are not needed for booting or for repairing the system and which are not installed locally should be placed in this directory.

- /usr/bin/X11

  is the traditional place to look for X11 executable's; on Linux, it usually is a symbolic link to /usr/X11R6/bin.

- /usr/dict

  This directory holds files containing word lists for spell checkers.

- /usr/doc

  You may find documentation about the installed software packages in this directory.

- **`/usr/etc`**

  Site-wide configuration files to be shared between several machines may be stored in this directory. However, commands should always reference those files using the **`/etc`** directory. Links from files in **`/etc`** should point to the appropriate files in **`/usr/etc`**.

- **`/usr/include`**

  Include files for the C compiler.

- **`/usr/include/X11`**

  Include files for the C compiler and the X-Windows system. This is usually a symbolic link to /usr/X11R6/include/X11.

- **`/usr/include/asm`**

  Include files which declare some assembler functions. This used to be a symbolic link to /usr/src/linux/include/asm.

- **`/usr/include/linux`**

  This contains information which may change from system release to system release and used to be a symbolic link to /usr/src/linux/include/linux to get at operating system specific information.

  (Note that one should have include files there that work correctly with the current libc and in user space. However, Linux kernel source is not designed to be used with user programs and does not know anything about the libc you are using. It is very likely that things will break if you let **`/usr/include/asm`** and **`/usr/include/linux`** point at a random kernel tree. Debian systems don't do this and use headers from a known good kernel version, provided in the libc*-dev package.)

- Include files to use with the GNU C++ compiler.

- **`/usr/lib`**

  Object libraries, including dynamic libraries, plus some executable's which usually are not invoked directly. More complicated programs may have whole subdirectories there.

- **`/usr/lib/X11`**

  The usual place for data files associated with X programs, and configuration files for the X system itself. On Linux, it usually is a symbolic link to **`/usr/X11R6/lib/X11`**

- **`/usr/lib/gcc-lib`**

  contains executable's and include files for the GNU C compiler.

- **`/usr/lib/groff`**

  Files for the GNU groff document formatting system.

- **`/usr/local`**

  This is where programs which are local to the site typically go.

- **`/usr/local/bin`**

  Binaries for programs local to the site go there.

- **`/usr/local/doc`**

  Local documentation

- **`/usr/local/etc`**

  Configuration files associated with locally installed programs go there.

- **`/usr/local/lib`**

  Files associated with locally installed programs go there.

- **`/usr/local/info`**

  Info pages associated with locally installed pro grams go there.

- **`/usr/local/man`**

  Manpages associated with locally installed programs go there.

- **`/usr/local/sbin`**

  Locally installed programs for system administration.

- **`/usr/local/src`**

Source code for locally installed software.

- **`/usr/man`**

  Manpages traditionally go in there, into their sub directories.

- **`/usr/sbin`**

  This directory contains program binaries for system administration which are not essential for the boot process, for mounting /usr, or for system repair.

- **`/usr/share`**

  This directory contains subdirectories with specific application data, that can be shared among different architectures of the same OS. Often one finds stuff here that used to live in /usr/doc or **`/usr/lib`** or /usr/man.

- **`/usr/share/man`**

  Manpages go in there, into their subdirectories.

- **`/usr/src`**

  Source files for different parts of the system, included with some packages for reference purposes. Don't work here with your own projects, as files below /usr should be read-only except when installing software

- **`/usr/src/linux`**

  This has always been the traditional place where kernel sources were unpacked. This was important on systems that **`/usr/include/linux`** was a symlink here. You should probably use another directory for building the kernel now.

- **`/usr/tmp`**

  Obsolete. This should be a link to /var/tmp. This link is present only for compatibility reasons and shouldn't be used.

- **`/var`**

  This directory contains files which may change in size, such as spool and log files.

- **`/var/adm`**

  This directory is superseded by **`/var/log`** and should be a symbolic link to **`/var/log`**.

- **`/var/backups`**

  This directory is used to save backup copies of important system files.

- **`/var/lock`**

  Lock files are placed in this directory. The naming convention for device lock files is LCK..<device> where <device> is the device's name in the filesystem. The format used is that of HDU UUCP lock files, i.e. lock files contain a PID as a 10-byte ASCII decimal number, followed by a newline character.

- **`/var/log`**

  Miscellaneous log files.

- **`/var/preserve`**

  This is where vi saves edit sessions so they can be restored later.

- **`/var/run`**

  Run-time variable files, like files holding process identifiers (PIDs) and logged user information (utmp). Files in this directory are usually cleared when the system boots.

- **`/var/spool`**

  Spooled (or queued) files for various programs.

- **`/var/spool/at`**

  Spooled jobs for at(1).

- **`/var/spool/cron`**

  Spooled jobs for cron

- **`/var/spool/lpd`**

  Spooled files for printing.

- **`/var/spool/mail`**

  Users' mailboxes.

- **`/var/tmp`**

  Like /tmp, this directory holds temporary files stored for an unspecified duration.

You can find a more comprehensive description in the hier(7) man page. This layout conforms with the Linux filesystem standard, Release 1.2, and reflects the layout for Debian, RedHat and SuSE Linux.

What is a filesystem?

A filesystem is a logical division within a physical partition.

As discussed during our installation chapter, we know that we have installed MS Windows and Linux on the same hard drive, however they are loaded in two separate disk partitions (physical divisions).

There can only be one of the physical partitions active at any one time, this will be the partition that you boot up with by default when the machine is switched on.

Within the Linux division we can have further divisions or filesystems based on the following decision criteria:

1. Size of hard drive
2. Security issues
3. Swap
4. User requirements

Types of filesystems

The Linux kernel supports a myriad of filesystem types, the most popular of which are: ext2, ext3, Reiserfs and XFS.

Currently, ext2 is the only one supported by Debian out of the box.

The ext3 filesystem is the next generation of the Extended Filesystem, and allows for larger filesystems and files, as well as improving performance.

The Reiserfs and XFS filesystems allow for improved performance when using large filesystems, and are suited to specific applications where data concurrency is important, such as filesystems which contain high volume databases.

The layout of a filesystem

1. Boot block

2. Superblock - contains some general information about the filesystem such as:

   Filesystem size.

   The condition of the filesystem e.g. Would be marked as an error if the machine not shut down correctly and this would start the process of "fsck" on bootup.

   A list of some of the free inodes and blocks available

3. Inode Table- which contains the number of inodes allocated to that filesystem. This is elastic in Linux and will expand if you require more inodes than those allocated. When you create a new file the kernel will allocate an inode for that file from the free list held in the superblock.

4. Data Blocks - containing the file data, a block is fixed in size. The kernel can only access the data blocks via the information held in the inode of that file.

Using Filesystems

mount and unmount

The **mount** command, and it's cousin **umount** can be used to mount and unmount filesystems on your Linux system.

An extract from the mount(1) man page:

> All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree. Conversely, the umount(8) command will detach it again.
>
> —man mount(1)

Issuing the **mount** command without any parameters will display the current mounted filesystems, and their status, on your system:

So when you created the filesystems at installation time a relevant device was created in the `/dev` directory. The devices that were created each have an address that points to a series of datablocks on your hard disk.

As an example let us say that when you were installing the operating system you specified a third filesystem called "database" that you wanted to create as well as the root filesystem and the swap filesystem. A device called database (**`/dev/database`**) would be created at that time and a mount point would be created as well called say **`/database`**.

Now when you do a listing under root you will see an empty directory called "database". When you attach or mount the filesystem to the empty directory now called a mount-point, then you are able to access the series of datablocks reserved on the hard disk for that filesystem as defined by **`/dev/database`** at installation time.

Once mounted the filesystem is now accessible through the mount-point called **`/database`**.

```
debian:~# mount
/dev/hda2 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
```

Here you can see that there are currently three file systems mounted.

One is your root ("/") file system, which is mounted off "hda2", which you should remember is the second partition on your primary master IDE drive. You can also see that the filesystem is mounted as "read-write", and is of type "ext2" (short for Linux second extended file system)

There are then two special filesystems, **`proc`**, and "devpts". The "devpts" filesystem is a simple, virtual filesystem which allows for easy control over the terminal devices and their ownership. We will explain the **`proc`** filesystem within the next section.

You can mount additional filesystems, as long as they been defined in your **`/etc/fstab`** file, by using the mount command:

In the following example we are going to mount **`/dev/cdrom`** on a mount-point called **`/cdrom:`**

```
debian:~# mount
/dev/hdb2 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
debian:~# mount /cdrom
debian:~# mount
/dev/hdb2 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/cdrom on /cdrom type iso9660 (ro,noexec,nosuid,nodev)
debian:~#
```

In the example above, we've successfully mounted a CD-ROM onto the /cdrom mount point. You can now change directory into the /cdrom directory and you will have access to the information held on the CDROM itself. Read write access will depend on the capabilities of the CDROM and the permissions of your username.

To unmount it again once we've finished, we can use the **unmount** command:

```
debian:~# umount /cdrom
debian:~# mount
/dev/hdb2 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
debian:~#
```

If you wish to mount a filesystem which isn't configured in your **`/etc/fstab`** file, then you can use the following syntax:

```
SYNTAX:
mount -t <type> -o <options> <device> <mount point>
```

```
debian:~# mount -t ext2 -o rw /dev/hdb3 /mnt
debian:~# mount
/dev/hdb2 on / type ext2 (rw,errors=remount-ro)
/dev/hdb3 on /mnt type ext2 (rw)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
debian:~# umount /mnt
```

As you can see, it takes fields very similar to what you find in **`/etc/fstab`**. As a point of interest, the currently mounted filesystem information is kept in a file called **`/etc/mtab`**.

```
debian:~# cat /etc/mtab
/dev/hdb2 / ext2 rw,errors=remount-ro 0 0
proc /proc proc rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
```

Make sure that whether you mount a diskette, cdrom or hard drive filesystem that you un-mount it prior to removing it from the system or prior to shutting down the system.

/proc

The /proc filesystem is a special interface to the Linux kernel. It allows you to access various Linux kernel structures as if they were normal files.

Let's take a look:

```
debian:~# cd /proc
debian:/proc# ls -F
./     1886/  2035/  2110/  6/           ksyms
../    1887/  2068/  2146/  7/           loadavg
1/     1888/  2069/  2309/  8/           locks
1080/  1908/  2070/  2325/  899/         lvm/
11/    1940/  2071/  2388/  902/         mdstat
1116/  1941/  2072/  2425/  961/         meminfo
1254/  1959/  2073/  2426/  acpi/        misc
1283/  1962/  2074/  2427/  asound/      mm
1292/  1965/  2075/  2428/  buddyinfo    modules
1300/  1968/  2076/  2439/  bus/         mounts@
1322/  1984/  2077/  3/     cmdline      mtrr
1323/  1986/  2079/  3291/  config.gz    net/
1435/  1987/  2082/  3342/  cpufreq      partitions
1663/  1989/  2083/  3348/  cpuinfo      pci
1668/  1990/  2084/  3349/  crypto/      pisch
1719/  1992/  2087/  3350/  devices      pisch_version
1752/  1994/  2095/  3351/  dma          scsi/
1759/  1997/  2096/  3352/  driver/      self@
1804/  2/     2098/  3517/  execdomains  slabinfo
1806/  2007/  2100/  3562/  fb           splash
1814/  2009/  2101/  3662/  filesystems  stat
1815/  2013/  2102/  3663/  fs/          swaps
1816/  2015/  2103/  3751/  ide/         sys/
1817/  2017/  2104/  3753/  interrupts   sysrq-trigger
1818/  2018/  2105/  3754/  iomem        sysvipc/
1822/  2019/  2106/  3763/  ioports      tty/
1883/  2021/  2107/  386/   irq/         uptime
1884/  2028/  2108/  4/     kcore        version
1885/  2030/  2109/  5/     kmsg         video/
```

The numbers refer to process IDs, and within those directories you can find out information about that currently running process. The **`self`** directory is a symlink to the process that checks it's current PID.

Some useful files in proc are:

/proc/cpuinfo

This file displays information about the system's CPU(s):

```
debian:/proc# cat cpuinfo
processor          : 0
vendor_id          : AuthenticAMD
cpu family         : 6
model               : 3
model name    : AMD Duron(tm) Processor
stepping          : 1
cpu MHz         : 656.476
cache size      : 64 KB
fdiv_bug        : no
hlt_bug         : no
sep_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr
    : mca cmov pat pse36 psn mmxext \
                        mmx fxsr 3dnowext 3dnow
bogomips   : 1307.44

debian:/proc#
```

/proc/uptime

This displays the amount of time that the system has been up, given in seconds.

```
debian:/proc# cat uptime
43736.51 43685.48
```

/proc/loadavg

This displays the system's load average, the same as the **top** and "uptime" commands that were covered earlier.

```
debian:/proc# cat loadavg
0.00 0.00 0.00 1/22 3448
debian:/proc# _
```

file system monitoring

One of the things that almost all operating systems do not handle very well is running out of disk space. Some applications will fail relatively gracefully, but probably won't be able to save their most recent changes. Other applications, usually the more complicated ones, will only be able to partially save, thus resulting in corrupted file data, which means you're going to have to end up restoring from backups!

You will find the following commands useful for keeping an eye on your filesystems:

df

The df (disk free) command shows your the amount of free disk space on the currently mounted filesystems:

```
debian:~# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hdb2              9550788     79056   8986572   1% /
```

You can use the "-h" parameter to tell **df** to display the values in "human readable" form:

```
debian:~# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hdb2             9.8G   81M  9.2G   1% /
```

Specifying the "-i" switch will cause **df** to display the free inode count, instead of the default free block count:

```
debian:~# df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hdb2            1214400   12043 1202357    1% /
```

du

The du (disk used) command can be used to determine how much space has been used by directories. This is useful when you're doing some cleaning up and looking for areas that are consuming a lot of space:

```
debian:~# du -sh /usr
53M     /usr
```

The "-s" switch gives us a summary, rather than a listing of each sub directory, and the "-h" switch again gives us the values in "human readable" form, rather than in block sizes.

You can use a combination of these commands to track down disk usage on your filesystems.

A good system administrator should keep an eye on the output of **df** and note when filesystems are reaching their limit, so that a new disk can be arranged before the system runs out of space!

It's often useful to look for the largest files and directories on a certain drive, as you may be wishing to free up some space quickly. You can use the "du" command together with a few others to achieve this:

```
debian:~# du -k /usr | sort -n | tail -10
37660   /usr/X11R6
39864   /usr/bin
52704   /usr/share/doc
76588   /usr/local/src/kernel-source-2.4.18/drivers
122336  /usr/lib
154308  /usr/local/src/kernel-source-2.4.18
155456  /usr/local/src
155496  /usr/local
192672  /usr/share
583824  /usr
```

The "-k" switch tells "du" to display the size of everything in kilobytes; it's important for the unit of measurement to be the same so that the "sort" command can order things properly.

The "-n" switch tells "sort" to treat the numbers as values, rather than words.

The "tail -10" command specifies that only the last 10 lines of output should be displayed.

This little one liner can now easily give you the top 10 largest files and directories in a specified path!

Advanced file permissions

We've dealt previously with "chmod", and with the "r", "w" and "x" permission bits. However, there are a few more, which we will deal with now.

suid and sgid bits

The "set user id" (suid) and "set group id" (sgid) on execution bits are denoted by an "s" in the column that would normally have an "x" (execute) character.

For example, the "passwd" command has its suid bit set:

```
debian:~# ls -l /usr/bin/passwd
-rwsr-xr-x    1 root     root        24680 Apr  7  2002 /usr/bin/passwd
```

These bits have the effect of causing the command to be run as the user (suid) and/or as the group (sgid) that owns the file. In the case of "passwd", it means that even if you execute the "passwd" command as a normal user, the kernel will elevate the command's privileges to that of the root user.

In this specific case, it is so that the passwd command can access the /etc/shadow password file, so that it can check and change passwords.

Problems with suid and sgid binaries can very often lead to a system compromise, so it is very important to make sure that you know which files on your system have these bits set, and to make sure that you keep that particular package up to date with security fixes, or perhaps even uninstall it if the package isn't necessary.

You can use the "find" command to locate all the suid and sgid binaries on your system:

```
debian:~# find / -xdev -perm +4000 -ls
17591   24 -rwsr-xr-x   1 root     root        21112 Sep
 23  2003 /bin/su
 12789   32 -rwsr-xr-x   1 root     root        31188 Sep
 23  2003 /bin/ping
  9736   36 -rwsr-xr-x   1 root     audio       36142 Sep
  23  2003 /bin/eject
 40687   76 -rwsr-xr-x   1 root     root        77488 Sep
 23  2003 /bin/mount
 12790   28 -rwsr-xr-x   1 root     root        26996 Sep
 23  2003 /bin/ping6
 40688   40 -rwsr-xr-x   1 root     root        40788 Sep
 23  2003 /bin/umount
 64765   60 -rwsr-xr-x   1 root     root        61396 Sep
 24  2003 /opt/kde3/bin/fileshareset
 52172    8 -rwsr-xr-x   1 root     root         6206 Sep
 24  2003 /opt/kde3/bin/artswrapper
 64769   16 -rwsr-xr-x   1 root     shadow      14134 Sep
 24  2003 /opt/kde3/bin/kcheckpass
 54971    8 -rwsr-xr-x   1 root     root         7946 Sep
 24  2003 /opt/kde3/bin/kpac_dhcp_helper
198953   44 -rwsr-xr-x   1 root     video       43996 Sep
 24  2003 /opt/gnome/sbin/zapping_setup_fb
 64521   36 -rwsr-xr-x   1 root     root        35196 Sep
 23  2003 /usr/bin/at
 43724    8 -rwsr-xr-x   2 root     root         4572 Sep
 24  2003 /usr/bin/man
 49943   16 -rwsr-xr-x   1 root     root        15488 Sep
 23  2003 /usr/bin/rcp
 49946   12 -rwsr-xr-x   1 root     root         8228 Sep
 23  2003 /usr/bin/rsh
 69488  228 -rwsr-xr-x   1 root     root       229480 Sep
 23  2003 /usr/bin/ssh
144705   32 -rwsr-xr-x   1 root     trusted     30623 Sep
23  2003 /usr/bin/bing
 58745   76 -rwsr-xr-x   3 root     shadow      77204 Sep
 24  2003 /usr/bin/chfn
 58745   76 -rwsr-xr-x   3 root     shadow      77204 Sep
 24  2003 /usr/bin/chsh
 17782   92 -rwsr-xr-x   1 root     root        90508 Sep
 23  2003 /usr/bin/sudo
 49653   16 -rwsr-xr-x   1 lp       sys         15206 Sep
 23  2003 /usr/bin/lppasswd
202972   16 -rwsr-xr-x   1 root     trusted     14880 Sep
23  2003 /usr/bin/vboxbeep
 71038   24 -rwsr-xr-x   1 root     root        22812 Sep
 23  2003 /usr/bin/crontab
 58662   60 -rwsr-xr-x   1 root     shadow      60944 Sep
 24  2003 /usr/bin/chage
 43724    8 -rwsr-xr-x   2 root     root         4572 Sep
 24  2003 /usr/bin/mandb
```

sticky bit

The sticky bit is denoted by a "t" in instead of the group execute permission ("x"). An example is the /tmp directory:

```
debian:~# ls -ld /tmp
drwxrwxrwt    2 root     root         4096 Mar 15 04:35 /tmp
```

The sticky bit only has an affect on directories, not regular files. It turns a directory into an append-only directory. This means that files in this directory can only be removed or renamed by the owner of the file. Thus, it is very useful for directories, which are publically writable, but where you want to protect the user's files from other users.

Again, you can use the "find" command to show you all the sticky-bit directories on your system:

```
debian:~# find / -xdev -perm +1000 -ls
  97153    4 drwxrwxrwt   2 root     root         4096 Mar 12 04:35 /tmp
1117249    4 drwxrwxrwt   2 root     root         4096 Feb  8  2002 /var/
1182017    4 drwxrwxrwt   3 root     root         4096 Mar 11 14:22 /var/
```

file attributes on the ext2 file system

In addition to the standard Unix file permission scheme, the ext2 file system has the ability to set the following attributes on both files and directories:

no access time (A)

When a file or directory with this attribute is accessed for reading or writing, its last access time (atime) will not be updated. This may be useful on files which are accessed very often, but which you're not interested in recording the atime for. An example might be the system man pages.

append only (a)

Files with this attribute set can only be appended to. For directories, this means that you can only add files to it; you cannot delete or rename files.

no dump (d)

This prevents the dump(8) backup utility from backing up a file or directory marked with this attributed.

immutable (i)

This prevents the file from being modified in any way. Only the root user can set or clear this attribute.

secure deletion (s)

When a file or directory with this attribute is deleted, the space they occupied on the disk is overwritten with zeros before the filesystem entries are removed.

synchronous mode (S)

Modifications to files or directories with this bit set are written to disk immediately, without doing any buffering.

You can use the following commands to examine and set ext2 attributes:

lsattr

The "lsattr" command displays the attributes.

chattr

The "chattr" command can set and remove attributes on files.

```
debian:~# ls -l test.txt
-rw-r--r--    1 root     root            9 Mar 15 03:17 test.txt
debian:~# lsattr test.txt
-------------- test.txt
debian:~# chattr +i test.txt
debian:~# lsattr test.txt
---i---------- test.txt
debian:~# rm test.txt
rm: remove write-protected file 'test.txt&apos;? y
rm: cannot unlink 'test.txt&apos;: Operation not permitted
debian:~# chattr -i test.txt
debian:~# rm test.txt
```

As you can see, even the root user cannot delete an "immutable" file until the attribute has been removed.

Chapter 4. Backups

Introduction

If you've worked with computers for any length of time, you should already be aware of the importance of keeping good, up-to-date backups. The definition of a good backup is one that you are able to restore from!

Backups guard against hardware failures (such as the hard disk drive in your server breaking), accidental file deletion and even physical theft (such as someone pinching your laptop).

Backup media

Linux supports a wide variety of backup media.

Magnetic Tape

The canonical backup media is magnetic tape. There are a wide variety of tape drives available, and Linux supports the majority of them.

Magnetic tape usually has a high storage capacity, but is slow to access relative to the speed of hard drives, and the data stored on it can only be accessed sequentially. This means that if you have something backed up onto tape, you can only restore it by going through the whole tape, from the beginning, until you reach the end of what you want to restore.

This means that tapes are generally only good for archival purposes, or for restoring entire system, rather than restoring individual files.

CD-ROM

Another popular backup media are write-able CD-ROMs. This allows you to backup a small amount of data, but you have random read access to it, which means that it's quicker and easier to restore from.

With the advent of write-able DVDs, this allows you to store even larger amounts of data on a single disc.

Hard Disk

As the price of hard drives has dropped dramatically, it is becoming popular to simply keep your backups on a large, but perhaps slow and relatively cheap, hard drive, usually in a separate machine. Again, this is handy because you will have random read access to your data, and won't have to fiddle with inserting and removing magnetic tapes or CDs.

Which form of Media to use?

Which option you decide to go with will depend on a number of factors:

1. the amount of data you want
2. the desired time to perform a restore
3. your budget

Securing your backup

No matter which media you decide to go with, it is important to remember that the media holding your backed up data should be stored as far away as possible from the system that you wish to keep backed up!

Imagine that you'd been regularly backing up your server, only to lose all your data as well as the backups because you had the magnetic tapes sitting on top of the server chassis when lightning struck your computer room!

For this reason, companies with critical data often make two backups. One is kept on site, for quick and easy restores, while another copy is taken off-site, to guard against site-wide disasters.

Backup media that is kept on-site should be kept in a fireproof safe, to guard against fire and water damage.

Backup concepts

Depending on your choice of backup software, you will have several types of backup paradigms at your disposal:

1. full - as the name implies, this is a backup of everything
2. differential - this is a backup of everything since the last full backup
3. incremental - this is a backup of everything since the last _incremental_ backup

As an example:

You would do a full backup every Sunday morning. Then, from Monday to Friday you would do an incremental backup.

This would mean that if you wanted to do a restore of Thursday's data, you would have to have to restore from the following backups:

- the full backup from Sunday
- all four incremental backups (Mon, Tue, Wed, Thurs)

That could easily be 5 tapes, which you have to shuffle in order to restore something, which you may find to be too time consuming for your needs.

Your other alternative is then:

Do the full backup on Sunday morning, and then do a differential backup each day after that.

Then to do a restore of Thursday's data you would only require:

- the full backup from Sunday
- and the one differential from Thursday.

Obviously, keeping differential backups requires more storage space on your backup media, so you'll need to decide which method works best for your situation.

Some backup utilities allow for an even finer grained set of levels (0-9), where each level only does a backup of the data that's changed since the last backup of the next lowest level.

In other words; level 0 is a full backup, while continually doing a level 1 backup after that would be the equivalent of a differential backup.

Doing a backup at level 0, 1, 2, 3, 4 and 5 would be the same as doing an initial full backup, and then following that by incremental backups through the week.

Backup software

mt

The **mt** command can be used to control the magnetic tape device attached to your Debian machine.

```
SYNTAX:
mt <command>
```

A list of valid commands are:

Table 4.1. mt commands

eof, weof	Write count EOF marks at current position.
fsf	Forward space count files. The tape is positioned on the first block of the next file.
bsf	Backward space count files. The tape is positioned on the first block of the next file.
fsr	Forward space count records.
bsr	Backward space count records.
bsfm	Backward space count file marks. The tape is positioned on the beginning-of-the-tape side of the file mark
fsfm	Forward space count file marks. The tape is positioned on the beginning-of-the-tape side of the file mark.
rewind	Rewind the tape.
offline, rewoffl	Rewind the tape and, if applicable, unload the tape.
status	Print status information about the tape unit.
erase	Erase the tape.

When using a tape stream to make backups onto, you will need to set EOF marks (using the **weof** command) between each backup session on the tape, so that you can easily seek between them. You can then use the **fsf** and **bsf** commands to seek through the tape to the various positions for each backup. In order to make the tape drive eject the current tape, you should issue the **offline** command.

tar

The "tape archiver", or **tar** command is one that you should be familiar with already. This can form the staple part of your backup strategy. This can be used to users to make backups of their own files, or by the system administrator to make backups of entire filesystems.

```
SYNTAX:
tar [c | x | t ] [z ] [ v ] [ -f filename ] [ files ... ]

  c  create an archive
  x  extract from an archive
  t   test an archive

  z   compress/uncompress the archive using gzip

  v   be verbose

  -f  filename  -- this specifies which archive or device to use.
```

The "files ..." is a matching path for the files and/or directories that you wish to archive or extract.

Examples:

To archive your **`/usr`** directory into a file called **`/backups/usr.tar.gz`**, and compress it, you would use:

```
debian:/# tar czf /backups/usr.tar.gz usr
debian:/# _
```

To verify that the archive you created:

```
debian:/# tar tzf /backups/usr.tar.gz
usr/
usr/bin/
usr/bin/whiptail
[ ... ]
```

If you add the verbose ("v") switch, you'll even be given a long directory listing:

```
debian:/# tar tzvf /backups/usr.tar.gz
drwxr-xr-x root/root         0 2004-03-12 15:40:28 usr/
drwxr-xr-x root/root         0 2004-03-25 23:18:34 usr/bin/
-rwxr-xr-x root/root     15936 2002-03-31 03:00:42 usr/bin/whiptail
[ ... ]
```

If we wanted to extract the files into a new directory, we could do something like this:

```
debian:/# mkdir newusr
debian:/# cd newusr
debian:/newusr# tar xzf /backups/usr.tar.gz
```

cpio

The "CoPy In/Out", or **cpio** command is very similar to **tar** in functionality.

Unfortunately, **cpio** is not as intuitive to use as **tar**, but it can handle backing up and restoring device files (those in **`/dev`**), which **tar** cannot.

For an example, we will back up our **`/dev`** directory:

```
debian:~# find /dev -print | cpio -o > backup.cpio
debian:~# ls backup.cpio
backup.cpio
```

We pass cpio a list of files that we want backed up, from the find command, via standard input. The "-o" flag tells cpio that we wish to create a backup archive. We then redirect the output from cpio to a file called "backup.cpio". We could also have directed this output to a tape device if we wanted the backup to go directly to tape.

In order to restore files from a cpio archive, we use the "-i" flags. Let's test restoring our **`/dev/null`** device back to where it should be after we delete it:

```
debian:~# ls /dev/null
/dev/null
debian:~# rm /dev/null
debian:~# ls /dev/null
ls: /dev/null: No such file or directory
debian:~# cpio -i /dev/null < backup.cpio
414 blocks
debian:~# ls /dev/null
```

```
/dev/null
debian:~# cpio -i /dev/null < backup.cpio
cpio: /dev/null not created: newer or same age version exists
414 blocks
debian:~# _
```

You'll notice that cpio won't let us overwrite an already existing file if it is newer or the same age as the one in the archive.

The "-i" flags takes a pattern as a parameter; the pattern should match the file(s) that you wish to have restored. In the example above, we only wanted the **`/dev/null`** file restored

dump and restore

The dump and restore commands, which originated from 4.4BSD, will do a dump (backup) and restore of an ext2 filesystem, respectively. Dump understands the concept of "dump levels", as discussed previously. However, this mode only works on complete filesystems.

Debian doesn't come with these commands by default, so you'll have to install the dump package if you wish to use them:

```
debian:~# apt-get install dump
```

```
SYNTAX: dump [-0 ... -9 ] [-u] -f <target> <filesystem>
```

The numeric options specify the dump level, the "target" is the destination file or device where you wish to send the backup to, and the "filesystem" is the one you wish to back up. The "-u" options tells dump to record the date and time of the dump, so that you can check later when the last backup was done. You can use the "-W" switch to check this later.

To do a full (level 0) backup of our root ("/") partition onto tape (**`/dev/st0`**), we would issue a command similar to:

```
dump -0u -f /dev/st0 /
```

And then to check and see when the last time a dump was done:

```
dump -W
```

The package comes with a lot of useful examples and documentation, which you should definitely take a look through. Remember, you can ask dpkg to give you a list of all the files associated with a specific package:

```
dpkg --listfiles dump
```

dd

The **dd** command can be useful for transferring backups to and from tape and other media.

For example, to send a .tar.gz backup that we'd made earlier to a tape drive, we could issue a command similar to:

```
dd if=backup.tar.gz of=/dev/nrsa0
```

Open source backup solutions

If the thought of rolling your own backup system using a combination of tar, cpio, shell scripting and cron jobs does not appeal, or if you require a more advanced or robust system, there are some open source projects which you should investigate:

AMANDA -- http://www.amanda.org/

The "Automated Network Disk Archiver" was developed at the University of Maryland, and is one of the most widely deployed free backup utilities.

AMANDA is composed of two parts; a backup client which runs on a machine that has your tape storage system attached to it, and backup servers, which run on the machines which you wish to have backed up.

The client then connects to the servers and requests the data which is to be backed up, and stores it on the tape drives attached to the backup machine.

Roll your own

Most small installations, and even some of the larger ones, have their own home grown collection of backup scripts to handle automating the task of making backups.

We'll go through the steps briefly of what the script would need to do, and then give an example of what the script might look like.

We want to backup the two partitions (/ and `/home`) on our server. We have a tape drive in the machine, and a single tape can quite easily hold a full backup for the entire system.

Performing a backup

Procedure 4.1. Steps to backup

1. rewind the tape
2. backup / partition using tar
3. set EOF marker
4. backup /home partition using tar
5. set EOF marker
6. rewind and eject tape

Thus, the commands we would need to issue would be:

Procedure 4.2. Commands to backup

1. mt rewind
2. tar cz /
3. mt eof
4. tar cz /home
5. mt eof

6. mt rewoffl

This could all be easily placed inside a shell script, like this one:

```
debian:~# cat backup.sh
#!/bin/sh
echo &quot;Starting backup...&quot;
mt rewind
tar cz /
mt eof
tar cz /home
mt eof
mt rewoffl
echo &quot;Backup complete!&quot;
```

You could easily schedule this script to be run automatically every evening, using the **cron** command.

Now we need to verify the backup, so that we know that we can actually restore from it.

Verifying Backups

Procedure 4.3. Commands to verify

1. mt rewind
2. tar tz
3. mt fsf
4. tar tz
5. mt rewoffl

This will rewind the tape to the beginning, and then stream the first section of the tape through tar, with the "t" (test) switch, which will verify that tar can read the stream. Once that's done, then we use the **mt** command to position the tape at the start of the next stream (which is our backup of the /home partition), and then we perform another "test". Once we're done, we rewind and eject the tape, and can now store it away and be pretty sure that we can restore from it.

Remember to label your tapes!

Restore from backup

OK, user "Joe" has managed to delete some critical files from his home directory, and he wants them restored. We manage to get the latest tape backup from out of the fire proof safe, and pop it into the tape drive.

Procedure 4.4. Commands to restore

1. Now we need to rewind it: **mt rewind**
2. OK, now we know that the stuff we want is in the second stream (the **`/home`** directory), so we can skip ahead to that part of the tape immediately: **mt fsf**
3. Once we're there, we can then instruct tar to restore just a specific path of files: **tar xf home/joe/important.doc**
4. This will cause tar to seek through the tape stream and only restore the matching files: **mt rewoffl**

Once we're finished - rewind the tape and eject it, and put it away safely.

Chapter 5. User Information and Administration

Introduction

You'll remember from the Fundamentals course that the user information is stored in the system **`/etc/passwd`** and **`/etc/shadow`** files, and that additionally, group membership information is stored in the /etc/group file.

While it is possible to edit these files by hand, it is not recommended.

There exist several command line tools, which can be used to manage these files instead:

useradd

```
SYNTAX:
useradd [switches] <username>
```

This is a powerful command, which lets you easily create new users on the system, with a range of options.

The most common ones are:

Table 5.1. useradd options

-c comment	The new user's GECOS information, normally just their full name.
-d homedir	The user's home directory. Defaults to "/home/${username}".
-e expire date	The date on which this account expires (YYYY-MM-DD format).
-g initial group	Group name or number of the user's primary group. Defaults to "1".
-G grp1,grp2,...	A list of additional groups, which the user will be made a member of.
-m	This option will cause the user's home directory to be created if it does not

	already exist. Any files and directories which exist in /etc/skel will also be copied into the user's home directory.

The last parameter should be the desired login name for the user.

You can consult man page for the other options, and for further details.

One additional switch worth mentioning is "-D", which controls the defaults for **useradd**.

Specifying the "-D" switch on its own will simply display the default settings, while specifying -D in conjunction with other switches will change the defaults to those values.

```
debian:~# useradd -D
GROUP=100

INACTIVE=-1HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
debian:~# useradd -D -s /bin/sh
debian:~# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
```

As you can see, this changes the default shell for created users from "bash" to "sh".

Let's change it back though, and create a test user:

```
debian:~# useradd -D -s /bin/bash
debian:~# useradd -c &quot;Joe Blogs&quot; -m jblogs
debian:~# finger jblogs
Login: jblogs                          Name: Joe Blogs
Directory: /home/jblogs                Shell: /bin/bash
Never logged in.
No mail.
No Plan.
debian:~# ls -la /home/jblogs
total 24
drwxr-xr-x    2 jblogs   users        4096 Mar 12 05:58 .
drwxrwsr-x    4 root     staff        4096 Mar 12 05:58 ..
-rw-r--r--    1 jblogs   users         266 Mar 12 05:58 .alias
```

```
-rw-r--r--    1 jblogs   users          509 Mar 12 05:58 .bash_profile
-rw-r--r--    1 jblogs   users         1093 Mar 12 05:58 .bashrc
-rw-r--r--    1 jblogs   users          375 Mar 12 05:58 .cshrc
```

You can see that the user's home directory has been populated with the files from "/etc/skel". Default configuration files (such as **`.bashrc`**, **`.bash_profile`**) should be kept here, so that new users that you create will be given them automatically. Remember, though, that you can set your system wide defaults in the **`/etc/profile`** file.

groupadd

```
SYNTAX:
groupadd <groupname>
```

This command simply creates additional groups.

```
debian:~# groupadd testing
debian:~# tail -1 /etc/group
testing:x:1001:
```

groupmod

```
SYNTAX:
groupmod -n <newname> <oldname>
```

This command renames a group, from oldname to newname.

```
debian:~# tail -1 /etc/group
testing:x:1001:
debian:~# groupmod -n jblogs testing
debian:~# tail -1 /etc/group
jblogs:x:1001:
```

groups

```
SYNTAX:
groups [username]
```

This simple command displays what groups a user is a member of. It takes the username of user as a parameter. If no username is given, it defaults to the current user.

```
debian:~# groups
root
debian:~# groups jblogs
jblogs : users
```

usermod

```
SYNTAX:
usermod [flags] <username>
```

This command alters account information for users, which already exist. It takes most of the same switches as the **useradd** command.

So, if we wanted to make user "jblogs"'s primary group also be "jblogs" (the group we renamed above), then we could do something like this:

```
debian:~# groups jblogs
jblogs : users
debian:~# usermod -g jblogs jblogs
debian:~# groups jblogs
jblogs : jblogs
```

The **usermod** command also allows the system administrator to disable and re-enable accounts. It's usually a good idea to disable accounts which you know aren't being used, or if you suspect malicious activity from that user.

You can disable an account by using the "-L" (lock) switch:

```
debian:~# usermod -L student
```

Now when the user "student" attempts to log in, they will be prevented from doing so.

You can then re-enable the account, using the "-U" (unlock) switch:

```
debian:~# usermod -U student
```

This will allow "student" to log in again as normal, with the same password as his account had previously.

userdel

```
SYNTAX:
userdel [-r] <username>
```

This command removes a user from the password database. If the "-r" flag is specified, the users home directory and files are also removed.

```
debian:~# ls -la ~jblogs
total 24
drwxr-xr-x    2 jblogs   users        4096 Mar 12 05:58 .
drwxrwsr-x    4 root     staff        4096 Mar 12 07:57 ..
-rw-r--r--    1 jblogs   users         266 Mar 12 05:58 .alias
-rw-r--r--    1 jblogs   users         509 Mar 12 05:58 .bash_profile
-rw-r--r--    1 jblogs   users        1093 Mar 12 05:58 .bashrc
-rw-r--r--    1 jblogs   users         375 Mar 12 05:58 .cshrc
debian:~# userdel -r jblogs
debian:~# ls -la ~jblogs
ls: ~jblogs: No such file or directory
```

groupdel

```
SYNTAX:
groupdel <group>
```

This command removes a group from the group database.

adduser, addgroup, deluser and delgroup

The **adduser**, **addgroup**, **deluser** and **delgroup** commands are more user- friendly front-ends to the commands explained earlier. They will prompt you interactively instead of requiring command line switches.

```
debian:~# adduser
Enter a username to add: jane
Adding user jane...
Adding new group jane (1001).
Adding new user jane (1001) with group jane.
Creating home directory /home/jane.
Copying files from /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for jane
Enter the new value, or press return for the default
        Full Name []: Jane Doe
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [y/n] y
debian:~# ls -la ~jane
total 24
drwxr-xr-x    2 jane     jane         4096 Mar 12 08:02 .
drwxrwsr-x    4 root     staff        4096 Mar 12 08:02 ..
-rw-r--r--    1 jane     jane          266 Mar 12 08:02 .alias
-rw-r--r--    1 jane     jane          509 Mar 12 08:02 .bash_profile
-rw-r--r--    1 jane     jane         1093 Mar 12 08:02 .bashrc
-rw-r--r--    1 jane     jane          375 Mar 12 08:02 .cshrc
```

You'll notice that, by default, the adduser command creates a group with the same name as the username, and makes this group the primary group for that user. This is called a user private group (UPG).

chfn and chsh

While all the commands described above are only available to the root user, the **chfn** and **chsh** commands are available to normal users too.

The chfn command allows the user to alter their GECOS information, while the chsh command allows the user to alter their shell.

```
jane@debian:~$ chsh
Password:
Changing the login shell for jane
Enter the new value, or press return for the default
Login Shell [/bin/bash]: /bin/sh
```

Debian and disk quotas:

Disk quotas are very useful in large multi-user environments where you have a large number of users with access to the filesystem, whether because they log into the system or use their file space remotely, through either Samba or NFS.

Debian supports BSD-style disk quotas on the ext2 filesystem. This allows you to limit the amount of disk space that your users consume.

You've already learnt that an ext2 filesystem has two finite resources; blocks and inodes. You can limit the quantity of both of these that each user is allowed to use. You can specify a "hard limit", which is the user is not allowed to exceed - the will receive an error message if they try.

You can also optionally specify a "soft limit", which the user _is_ allowed to exceed, but only for a limited amount of time. This amount of time is known as the "grace period" and is usually a week (7 days).

Additionally, you can impose quotas on individual users, as well as groups.

For example:

You have three users: Joe, bob and jane

Joe has his own project funding, which has covered half of the cost of the hard drive for the system.

Bob and Jane work together, but for another project, which has paid for the other half of the hard drive. We'll call their project "project-x".

You can now fairly divide up the disk, giving half to Joe, and the other half to both Bob and Jane, who will share that 50%.

You can do this by allocating a *user* quota to user "Joe", and a*group* quota to "project-x".

If "Joe" exceeds his quota, he will have to clean up some of his old files (or buy more disk space). However, if "bob" or "jane" exceed their group quota, they will have to decide amongst themselves how best to reduce their disk usage.

initial setup

You'll need to install the quota package for Debian:

```
debian:~# apt-get install quota
Reading Package Lists... Done
Building Dependency Tree... Done
```

```
(Reading database ... 6077 files and directories currently installed.)
Unpacking quota (from quota_3.04-1_i386.deb) ...
Setting up quota (3.04-1) ...
```

You will be asked whether or not you wish to send e-mail out to users who have exceeded their quota. The default is **No**. If your users are more likely to check their e-mail in case of problems, then you can opt to specify "yes".

Your kernel also needs to have quota support compiled in.

Now you need to tell the system which filesystems should be controlled by the quota system. To do this, we need to edit the **`/etc/fstab`** file and add the "usrquota" and/or "grpquota" options to the relevant lines.

Before:

```
debian:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>                       <dump>  <pass>
/dev/hdb2       /               ext2    errors=remount-ro               0       1
/dev/hdb1       none            swap    sw                              0       0
proc            /proc           proc    defaults                        0       0
/dev/fd0        /floppy         auto    user,noauto                     0       0
/dev/cdrom      /cdrom          iso9660 ro,user,noauto                  0       0
```

After:

`/etc/fstab`

```
debian:~# vi /etc/fstab
debian:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>   <dump> <pass>
/dev/hdb2       /               ext2    errors=remount-ro,usrquota,grpquota
/dev/hdb1       none            swap    sw
proc            /proc           proc    defaults
/dev/fd0        /floppy         auto    user,noauto
/dev/cdrom      /cdrom          iso9660 ro,user,noauto
```

Now you need to initialize the quota files. These files live in the root directory of the filesystem which has quotas enabled, and are named **`quota.user`** and **`quota.group`** for user and group quota databases, respectively. These files contain both the user and group quota information, as well as the filesystem usage information.

To do this, you need to run the **quotacheck** command. However, because **quotacheck** has to determine the disk usage by inspecting the file structure on the disk, there can be no write activity on the disk at the time. Otherwise, this may corrupt your quota database. For this reason, it is suggested that **quotacheck** be run in single-user mode:

```
debian:~# shutdown now

[ ... ]

Give root password for maintenance
(or type Control-D for normal startup):
```

Enter the root password, and then issue the follow command:

```
debian:~# quotacheck -F vfsold -c -a -v
quotacheck: Scanning /dev/hdb2 [/] done
quotacheck: Checked 791 directories and 15539 files
debian:~# _
```

The "-F vfsold" flag instructs the system to use the old VFS quota (version 1) system. The newer (version 2) quota system doesn't work in the standard Debian installation.

The "-c" flag specifies that the quota database files should be created, and the "-a" flag specifies that the command should be run on all filesystems which have quotas enabled. If you only wanted to perform the operation on a specific filesystem, you could have used the following:

```
debian:~# quotacheck -F vfsold -c /home
```

By default, **quotacheck** only performs this operation for "user" quotas. You will need to run the same command again, this time with the "-g" flag, if you also wish to enable "group" quotas:

```
debian:~# quotacheck -F vfsold -c -a -g
debian:~# _
```

You should now have both the quota database files created and present in the root directory of the filesystem for which you have quotas turned on:

```
debian:~# ls /*quota*
/quota.group  /quota.user
```

Once you're finished, you can type "exit" or press **Ctrl-d** to return to multi-user mode.

repquota

The **repquota** command is a utility for reporting quota summary information.

```
SYNTAX:
repquota [ -u | -g ] [ -a | filesystem ]
```

As with **quotacheck**, the **repquota** command only displays user quotas by default, but you can specify that you wish to see group quotas by using the "-g" switch. You also need to specify the filesystem for which you wish to see a report, or you can specify the "-a" switch to see a report for all the filesystems for which quotas are enabled.

```
debian:~# repquota -a
*** Report for user quotas on device /dev/hdb2
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   80088       0       0          12025     0     0
daemon    --       8       0       0              3     0     0
man       --     172       0       0             17     0     0
lp        --      12       0       0              3     0     0
mail      --      56       0       0             14     0     0
news      --       4       0       0              1     0     0
student   --      12       0       0              3     0     0
```

The "block limits" refer to the data blocks (the default is 1 block = 1k). The "file limits" refer to the number of files, or inodes, that have been consumed. The "grace" field corresponds to the number of grace days remaining before the user is locked out of their account. The user has until that time to reduce their quota to below the "soft" level.

edquota

You can use the **edquota** command to edit user and group quotas.

```
SYNTAX:
edquota [ -u | -g ] <username or groupname>
```

For example, to edit the quota for "student":

```
debian:~# edquota student
```

The command will open up an instance of your preferred editor, with the following information:

```
Disk quotas for user student (uid 1000):
  Filesystem    blocks       soft       hard     inodes     soft      har
  /dev/hdb2         12          0          0          3        0
```

The "blocks" and "inodes" fields cannot be edited, they are there for information purposes only. However, you can edit the soft and hard fields. The first pair refer to blocks, and the second to inodes.

For our example, let's set the "student" quota so that the user has 8MB of disk space available, and can use an extra 2MB temporarily if required.

```
Disk quotas for user student (uid 1000):
Filesystem     blocks       soft       hard     inodes     soft      hard
/dev/hdb2          12       8192      10240          3        0         0
```

Once you've made the changes, you can save and exit. The **edquota** utility will complain if it doesn't like the values that you've given.

quota

Unlike **repquota** and **edquota**, which only the root user can make use of, the **quota** command is available to all users, and it allows them to query their current quota information.

```
SYNTAX:
quota [ -q ] [ username ]
```

The root user may optionally specify the username of a user in order to check their quota:

```
debian:~# quota student
Disk quotas for user student (uid 1000):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grac
      /dev/hdb2      12    8192   10240               3       0       0
```

If you log in as the "student" account, you should be able to use **quota** to check your disk usage:

```
student@debian:~$ quota
Disk quotas for user student (uid 1000):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grac
      /dev/hdb2      12    8192   10240               3       0       0
```

The optional "-q" switch tells **quota** not to output anything unless the user has exceeded their quota limits:

```
student@debian:~$ quota -q
student@debian:~$ _
```

This is a useful command to place in the system wide /etc/profile file. That way, whenever a user logs in, they will be presented with a summary of the disk usage information, but only if they are over quota.

Let's test and see if our quota system works. We'll use the **dd** command to create a 9MB file as the "student" user:

```
student@debian:~$ dd if=/dev/zero of=9mb-file bs=1024 count=9216
student@debian:~$ du 9mb-file
9232    9mb-file
student@debian:~$ quota
Disk quotas for user student (uid 1000):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grac
      /dev/hdb2    9248*   8192   10240   6days       5       0       0
```

You'll notice now that there is an asterisk ("*") next to the number of blocks that we've used, and that there is a "6day" value underneath the account's "grace period". This shows us that we've exceeded our soft quota, and have 6 days in which to reduce the amount of space that we're using.

If we were to attempt to create another file slightly larger than 1MB, and thus exceed our hard limit, we would receive an "out of disk space" error message.

This page intentionally left blank

Chapter 6. Process Accounting

Introduction

Over 10 years ago, it cost a lot to keep computing facilities running, with masses amounts of money being required to keep a CPU going for only a few hundred cycles. For this reason, it was important to keep track of who was using the CPU, and how much they were using it.

Universities and companies used to keep track of this information, and then bill their students or customers accordingly.

Although it costs far less to use CPU time now on most standard PC systems, you may still sometimes want to have this functionality; perhaps if you have a large rendering cluster than you're renting out to CG companies.

Installing the accounting package

In order for Debian to support process accounting, you need to install the "acct" Debian package

```
debian:~# apt-get install act
Reading Package Lists... Done
(Reading database ... 6023 files and directories currently installed.)
Unpacking acct (from acct_6.3.5-32_i386.deb) ...
Setting up acct (6.3.5-32) ...
Starting process accounting:
```

This will start up the system accounting process, and beginning logging all system activity to the following file:

`/var/account/pacct`

When you install the Debian package, it will also add entries to the system cron folders (**`/etc/cron.daily`** and **`/etc/cron.monthly`**) in order to keep track of these files, and make sure that they don't grow too large.

You will now be able to use the following commands to query the accounting database:

ac

This commands prints out statistics about users' connection time, in hours. This is the amount of time that the user has been connected to the system, either remotely via SSH or a serial terminal, or while on the console.

```
debian:~# ac
        total       10.44
debian:~# _
```

In the screenshot above, we can see that the root user has been logged in for over 10 hours so far this month.

lastcomm

The **lastcomm** command displays information about the previously executed commands. The most recent entries are given at the top of the list. Also displayed is the total amount of CPU time that each process used.

```
debian:~# lastcomm
lastcomm      root     stderr     0.02 secs Mon Mar  1 02:36
ls                   root     stderr     0.01 secs Mon Mar  1 02:34
ls                   root     stderr     0.00 secs Mon Mar  1 02:34
ls                   root     stderr     0.01 secs Mon Mar  1 02:34
ls                   root     stderr     0.01 secs Mon Mar  1 02:34
ls                   root     stderr     0.01 secs Mon Mar  1 02:34
sa                  root     stderr     0.00 secs Mon Mar  1 02:34
sa                  root     stderr     0.01 secs Mon Mar  1 02:34
sa                  root     stderr     0.00 secs Mon Mar  1 02:33
acct               root     stderr     0.03 secs Mon Mar  1 02:33
accton    S     root     stderr     0.00 secs Mon Mar  1 02:33
```

Here we can see that the root user executed the **sa** command three times, and the **ls** command 5 times, followed finally by the **lastcomm** command which is displaying the current listing.

You'll notice that there's a field after the command, but before the username, which can contain the following symbols:

- S - command executed by super-user
- F - command executed after a fork but without a following exec
- D - command terminated with the generation of a core file
- X - command was terminated with the signal SIGTERM

sa

The **sa** command displays a summary of the accounting database to date. A sample output may appear as follows:

```
debian:~# sa
     195    2083.07re       0.03cp         0avio       418k
      29     512.10re       0.01cp         0avio       507k   sh
       7     184.77re       0.01cp         0avio       401k   less
       4       0.63re       0.01cp         0avio       619k   troff
       6       0.33re       0.00cp         0avio       462k   dpkg
      14     528.33re       0.00cp         0avio       400k   pager
       8      17.97re       0.00cp         0avio       372k   lastcomm
      12       0.18re       0.00cp         0avio       373k   ls
      25     151.47re       0.00cp         0avio       402k   gzip
      14     529.15re       0.00cp         0avio       420k   man
       4      29.98re       0.00cp         0avio       465k   grotty
       8      67.00re       0.00cp         0avio       448k   ***other*
      12       0.12re       0.00cp         0avio       336k   ac
      12       0.07re       0.00cp         0avio       312k   expr
       4      30.22re       0.00cp         0avio       508k   nroff
       4       0.15re       0.00cp         0avio       542k   dhclient-s
       4       0.25re       0.00cp         0avio       445k   tbl
       5       0.03re       0.00cp         0avio       317k   sa
       4       0.07re       0.00cp         0avio       317k   zsoelim
       4      30.02re       0.00cp         0avio       441k   groff
       4       0.03re       0.00cp         0avio       314k   locale
       4       0.10re       0.00cp         0avio       508k   nroff*
       4       0.00re       0.00cp         0avio       310k   uname
       3       0.10re       0.00cp         0avio       508k   sh*
```

The first field contains the total number of times that the specific command has been executed on the system. The "re" field shows the amount of "real time" that the command consumed, cumulatively, and in seconds. The "cp" column shows the number of seconds actually spent on the CPU. The "avio" column is the average number of I/O operations per execution. Finally, the "k" field shows the average amount of CPU core usage, in 1k units. Finally, the last column shows the command itself. The asterisk (*) denotes that the command fork()ed, but didn't call exec().

last

```
SYNTAX:
last [ -count] [ username ] [ -f file ]
```

Although the **last** command is not part of the "acct" package, it is still worth mentioning here.

[[exercise: determine which package the **last** command is part of]]

The **last** command displays the login and logout times for all users who've connected to the system. This data is stored in the **`/var/log/wtmp`** file.

The "-count" switch limits the number of entries to display. The "username" parameter specifies a particular user whose records should be examined; the default is to display the records for all users.

The "-f file" switch allows you to specify a different file to extract this information from, instead of the default **`/var/log/wtmp`** file.

The **wtmp** file also logs reboot and shutdown commands. You can use the word "reboot" instead of a username to just display these entries.

In addition to the **wtmp** file, you can also configure your system to use a **btmp** file. This file maintains a record of all the failed login attempts.

You can use the **lastb** command to view this file. The **lastb** command is the same as **last**, except it has the default file to display as **`/var/log/btmp`** instead of **wtmp**.

You will need to create the file initially for the system to start using it. You can do this with the **touch** command:

```
debian:~# lastb
lastb: /var/log/btmp: No such file or directory
Perhaps this file was removed by the operator to prevent logging lastb info.
debian:~# touch /var/log/btmp
debian:~# lastb

btmp begins Sat Mar 27 05:46:05 2004
```

Chapter 7. Managing Processes

Introduction

What is a process?

As we've mentioned previously, the Linux kernel enables us to have a multi-tasking operating system. This means that more than task can be running at the same time; while they are running, each of these tasks, or programs, are referred to as a process.

Each process has a unique number assigned to it; this number is known as the process ID, or PID. Each process also has a user (uid) and group (gid) assigned to it, much like files on the filesystem.

The process table and the nice command

The kernel maintains a list of all the current processes in a "process table"; you can use the **ps** command to view the contents of this table.

Each process can also be assigned a priority, or "niceness" level; a value which ranges from -20 to 19. A priority of "-20" means that the process will be given access to the CPU more often, whereas a priority of "19" means that the process will only be given CPU time when the system is idle.

You can use the **nice** and **renice** commands to specify and alter these values for specific processes.

Process creation

From your shell prompt (bash), you will usually instruct the system to run an application for you; for example, "vi". This will then cause your "bash" process to "fork" off a new process. The initial process is referred to as the "parent process", and the process which it forked as the "child process".

The process table contains the parent PID (PPID), and uses this to track which processes spawned which other ones.

System Processes

As well as the standard user processes that you would expect to find running, such as your shell and perhaps your editor, there are also several system processes that you would expect to find running. Examples of these include the cron daemon (crond), which handles job scheduling, and the system log daemon (syslogd), which handles the logging of system messages.

Scheduling Command execution with batch (cron) jobs

There are two methods of scheduling jobs on a Unix system. One is called **at**, which is used for once-off batch jobs. The other is called **cron**, which is used for regularly run tasks.

The **at** jobs are serviced by the "at daemon (atd)".

at

```
SYNTAX:
at [-f script] TIME
```

This command is used to schedule batch jobs.

You can either give it a script to run with the "-f" parameter, or you can specify it after you've typed the command.

The "TIME" parameter can be in the form of HH:MM, or "now + n minutes". There are several other complicated methods of specifying the time, which you should look up in the man page for at(1).

```
debian:~# at now + 5 minutes
warning: commands will be executed using /bin/sh
at> echo hello!
at> <EOT>
job 1 at 2004-03-12 13:27
```

We have now scheduled a job to run in 5 minutes time; that job will simply display (echo) the string "hello!" to stdout.

To tell **at** that you're finished typing commands to be executed, press **Ctrl-d**, that will display the <EOT> marker that you can see above.

atq

```
SYNTAX:
atq
```

This command displays the current batch jobs that are queued:

```
debian:~# atq
1       2004-03-12 13:27 a root
debian:~#
```

This is the job that we queued earlier.

The first number is the "job id", followed by the date and time that the job will be executed, followed by the user who the job belongs to.

atrm

```
SYNTAX:
atrm <job id>
```

This command simply removes jobs from the queue.

```
debian:~# atrm 1
debian:~# atq
debian:~#
```

We've now removed our scheduled job from the queue, so it won't run.

Let's add another one, and see what happens when it is executed:

```
debian:~# at now + 1 minute
warning: commands will be executed using /bin/sh
at> touch /tmp/at.job.finished
at> <EOT>
job 3 at 2004-03-12 13:27
debian:~# atq
3       2004-03-12 13:27 a root
debian:~# date
Fri Mar 12 13:26:57 SAST 2004
debian:~# date
Fri Mar 12 13:27:04 SAST 2004
debian:~# atq
```

```
debian:~# ls -l /tmp/at.job.finished
-rw-r--r--    1 root     root            0 Mar 12 13:27 /tmp/at.job.finished
```

As you can see, we scheduled a job to execute one minute from now, and then waited for a minute to pass. You'll notice how it was removed from the queue once it was executed.

cron

The **cron** jobs are serviced by the "cron daemon (crond)".

crontab

```
SYNTAX:
crontab [ -u user ] { -l | -r | -e }
crontab [ -u user ] filename
```

You can use the **crontab** command to edit, display and delete existing cron tables.

The "-u" switch lets the root user specify another user's crontab to perform the operation on.

Table 7.1. crontab options

l	lists current crontab
r	removes current crontab
e	edits current crontab

If a filename is specified instead, that file is made the new crontab.

The syntax for a crontab is as follows:

```
# minute hour day month weekday command

Example:

# minute  hour  day  month  weekday  command
       0     1    *      *         * backup.sh
```

This cron job will execute the **`backup.sh`** script, at 01:00 every day of the year.

A more complicated example:

```
# minute  hour  day  month  weekday  command
       5     2    *       * 5 backup-fri.sh
```

This cron job will execute the **`backup-fri.sh`** script, at 02:05 every Friday.

Weekdays are as follows:

```
01 - Monday
02 - Tuesday
etc.
07 - Sunday
```

There is also a "system crontab", which differs slightly from the user crontabs explained above. You can find the system crontab in a file called **`/etc/crontab`**.

You can edit this file with vi, you must not use the **crontab** command to edit it.

You'll also notice that this file has an additional field, which specifies the username under which the job should run.

```
debian:~# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab&apos;
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other crontabs d

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
25 6    * * *   root    test -e /usr/sbin/anacron ||
                run-parts --report /etc/cron.daily
47 6    * * 7   root    test -e /usr/sbin/anacron ||
                run-parts --report /etc/cron.weekly
52 6    1 * *   root    test -e /usr/sbin/anacron ||
                run-parts --report /etc/cron.monthly
#
```

Some of the daily system-wide jobs that run are:

1. logrotate - this checks to see that the files in **`/var/log`** don't grow too large.
2. find - this builds the locate database, used by the **?locate?** command.
3. man-db - this builds the "whatis" database, used by the **whatis** command.
4. standard - this makes a backup of critical system files from the **`/etc`** directory, namely, your passwd,shadow and group files - that backups are given a .bak extension.

Monitoring system resources

The follow commands are vital for monitoring system resources:

ps and kill

The **ps** command displays the process table.

```
SYNTAX:
ps [auxwww]

 a -- select all with a tty except session leaders
 u -- select by effective user ID - shows username associated with each process
 x -- select processes without controlling ttys (daemon or background processes
 w -- wide format
```

```
debian:~# ps
  PID TTY          TIME CMD
 1013 pts/0    00:00:00 bash
 1218 pts/0    00:00:00 ps
```

```
debian:~# ps auxwww
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.4  0.1  1276  496 ?        S    13:46   0:05 init
root         2  0.0  0.0     0    0 ?        SW   13:46   0:00 [kflushd]
root         3  0.0  0.0     0    0 ?        SW   13:46   0:00 [kupdate]
root         4  0.0  0.0     0    0 ?        SW   13:46   0:00 [kswapd]
root         5  0.0  0.0     0    0 ?        SW   13:46   0:00 [keventd]
root       140  0.0  0.2  1344  596 ?        S    13:46   0:00 /sbin/syslogd
root       143  0.0  0.3  1652  836 ?        S    13:46   0:00 /sbin/klogd
root       151  0.0  0.1  1292  508 ?        S    13:46   0:00 /usr/sbin/inetd
```

```
daemon      180  0.0  0.2  1388  584 ?        S    13:46   0:00 /usr/sbin
root        183  0.0  0.2  1652  684 ?        S    13:46   0:00 /usr/sbin
root        682  0.0  0.4  2208 1256 tty1     S    13:48   0:00 -bash
root       1007  0.0  0.4  2784 1208 ?        S    13:51   0:00 /usr/sbin
root       1011  0.0  0.6  5720 1780 ?        S    13:52   0:00 /usr/sbin
root       1013  0.0  0.4  2208 1236 pts/0    S    13:52   0:00 -bash
root       1220  0.0  0.4  2944 1096 pts/0    R    14:06   0:00 ps auxwww
```

The USER column is the user to whom that particular process belongs; the PID is that processes unique Process ID. You can use this PID to send signals to a process using the **kill** command.

For example, you can signal the "sshd" process (PID = 1007) to quit, by sending it the terminate (TERM) signal:

```
debian:~# ps auxwww | grep 1007
root       1007  0.0  0.4  2784 1208 ?        S    13:51   0:00 /usr/sbin
debian:~# kill -SIGTERM 1007
debian:~# ps auxwww | grep 1007
```

The "TERM" signal is the default that the **kill** command sends, so you can leave the signal parameter out usually.

If a process refuses to exit gracefully when you send it a KILL signal; e.g. "kill -SIGKILL <pid>".

top

The **top** command will display a running process table of the top CPU processes:

```
 14:15:34 up 29 min,  2 users,  load average: 0.00, 0.00, 0.00
20 processes: 19 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:   1.4% user,   0.9% system,   0.0% nice,  97.7% idle
Mem:    257664K total,    45104K used,   212560K free,    13748K buffers
Swap:    64224K total,        0K used,    64224K free,    21336K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
    1 root       0   0   496  496   428 S     0.0  0.1   0:05 init
    2 root       0   0     0    0     0 SW    0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW    0.0  0.0   0:00 kupdate
    4 root       0   0     0    0     0 SW    0.0  0.0   0:00 kswapd
[ ... ]
```

nice and renice

```
SYNTAX:
nice -<niceness> <command>
```

Example:

To run the **sleep** command with a niceness of "-10":

```
debian:~# nice --10 sleep 50
```

If you then run the **top** command in a different terminal, you should see that the sleep's NI column has been altered:

```
  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 2708 root      -9 -10   508  508   428 S <   0.0  0.1   0:00 sleep
```

You can use the **renice** command to alter the niceness level of an already running process.

```
SYNTAX:
renice <niceness> [ -p <pid> ] [ -u <user> ]
```

The "-p pid" parameter specifies the PID of a specific process, and the "-u user" parameter specifies a specific user, all of whose currently running processes will have their niceness value changed.

Example:

To renice all of user "student"'s processes to a value of "-10":

```
debian:~# renice -10 -u student
```

vmstat

The **vmstat** command gives you statistics on the virtual memory system.

```
SYNTAX:
vmstat [delay [count]]
```

```
debian:~# vmstat 1 5
   procs                     memory     swap          io     system
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us
 0  0  0      0 212636  13748  21348   0   0     4     4  156    18   1
 0  0  0      0 212636  13748  21348   0   0     0     0  104    12   0
 0  0  0      0 212636  13748  21348   0   0     0     0  104     8   0
 0  0  0      0 212636  13748  21348   0   0     0     0  104    10   0
 0  0  0      0 212636  13748  21348   0   0     0     0  104     8   0
debian:~#
```

Field descriptions:

Table 7.2. procs

r	processes waiting for run time
b	processes in uninterpretable sleep
w	processes swapped out but otherwise runnable

Table 7.3. memory

swpd	virtual memory used (Kb)
free	idle memory (Kb)
buff	memory used as buffers (Kb)

Table 7.4. swap

si	memory swapped in from disk (kB/s)
so	memory swapped out to disk (kB/s)

Table 7.5. io

bi	blocks sent to a block device (blocks/s)
bo	blocks received from a block device (blocks/s)

Table 7.6. system

in	interrupts per second (including the clock)
cs	context switches per second

Table 7.7. cpu

us	user time as a percentage of total CPU time
sy	system time as a percentage of total CPU time
id	idle time as a percentage of total CPU time

system monitoring tools:

It is often useful to be able to keep a historical record of system activity and resource usage. This is useful to spot possible problems before they occur (such as running out of disk space), as well as for future capacity planning.

Usually, these tools are built by using system commands (such as **vmstat**, **ps** and **df**), coupled together with rrdtool or mrtg, which store the data and generate graphs.

rrdtool: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

mrtg: http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

Some of the more complex monitoring systems that have been built using these tools include the following:

Cacti: http://www.raxnet.net/products/cacti/

Zabbix:http://www.zabbix.com/

All of the above tools are open source and free to use.

ulimit:

You may use the bash shell built-in command "ulimit" to limit the system resources that your processes are allowed to consume.

The following is an excerpt from the bash man page:

```
SYNTAX:
  ulimit [-SHacdflmnpstuv [limit]]

Provides control over the resources available to the shell and to
processes started by it, on systems that allow such control.

The -H and -S options specify that the hard or soft limit is set for
the given resource.  A hard limit cannot be increased once it is set;
a soft limit may be increased up to the value of the hard limit.  If
neither -H nor -S is specified, both the soft and hard limits are set.

The value of limit can be a number in the unit specified for the
resource or one of the special values hard, soft, or unlimited, which
stand for the current hard limit, the current soft limit, and no
limit, respectively.  If limit is omitted, the current value of the
soft limit of the resource is printed, unless the -H option is given.
When more than one resource is specified, the limit name and unit are
printed before the value.

Other options are interpreted as follows:

  -a     All current limits are reported
  -c     The maximum size of core files created
  -d     The maximum size of a process's data segment
  -f     The maximum size of files created by the shell
  -l     The maximum size that may be locked into memory
  -m     The maximum resident set size
  -n     The maximum number of open file descriptors
          (most systems do not allow this value to be set)
  -p     The pipe size in 512-byte blocks (this may not be set)
  -s     The maximum stack size
  -t     The maximum amount of cpu time in seconds
  -u     The maximum number of processes available to a single user
  -v     The maximum amount of virtual memory available to the shell

If limit is given, it is the new value of the specified resource (the
-a option is display only).  If no option is given, then -f
is assumed.  Values are in 1024-byte increments, except for -t, which
is in seconds, -p, which is in units of 512-byte blocks, and -n and
-u, which are unscaled values.  The return status is 0 unless an
invalid option or argument is supplied, or an error occurs while
setting a new limit.
```

On a Debian system, the default ulimit settings should appear as follows:

```
debian:~# ulimit -a
core file size        (blocks, -c) 0
data seg size         (kbytes, -d) unlimited
file size             (blocks, -f) unlimited
max locked memory     (kbytes, -l) unlimited
max memory size       (kbytes, -m) unlimited
open files                    (-n) 1024
pipe size          (512 bytes, -p) 8
stack size            (kbytes, -s) 8192
cpu time             (seconds, -t) unlimited
max user processes            (-u) 256
virtual memory        (kbytes, -v) unlimited
```

The common use of this command is to prevent long running processes, such as web servers (e.g., Apache) and CGI scripts from leaking memory and consuming all available system resources. Using the ulimit command to reduce the locked memory and memory size options before starting up your web server would mitigate this problem.

Another common use is on shell servers where users may not "tidy up" after themselves; you can then set the cpu time limit in /etc/profile, thus having the system automatically terminate long running processes.

Another example specifically relates to core files. You'll notice that the default core file size is set to 0. This means that when an application crashes (called a "segmentation fault"), it does not leave behind a core file (a file containing the memory contents of the application at the time that it crashed). This core file can prove invaluable for debugging, but can obviously be quite large as it will be the same size as the amount of memory the application was consuming at the time! Hence the default "0" value.

However, to enable core dumps, you can specify the "-c" switch:

```
debian:~# ulimit -c
0
debian:~# ulimit -c 1024
debian:~# ulimit -c
1024
```

Any further applications launched from this shell, which crash, will now generate core dump files up to 1024 blocks in size.

The core files are normally named "core" or sometimes **`processname.core`**, and will be written to the current working directory of the specific application that crashed.

Working with log files

On a Linux system, you should find all the system log files are in the **`/var/log`** directory.

The first place you should look if you were experiencing problems with a running system is the system "messages" logfile.

You can use the tail command to see the last few entries:

```
$ tail /var/log/messages
```

It's sometimes useful to keep the log scrolling in a window as entries are added, and you can use tail's -f (follow) flag to achieve this:

```
$ tail -f /var/log/messages
```

Other files of interest in /var/log:

1. **`auth.log`** -- log messages relating to system authentication
2. **`daemon.log`** -- log message relating to running daemons on the system
3. **`debug`** -- debug level messages
4. **`syslog`** -- system level log messages
5. **`kern.log`** -- kernel messages

The process which writes to these logfiles is called "syslogd", and its behavior is configured by **`/etc/syslog.conf`**.

Each log message has a facility (kern, mail, news, daemon) and a severity (debug, info, warn, err, crit). The syslog.conf file uses these to determine where to send the messages.

As machines continue to run over time, these files can obviously become quite large. Rather than the system administrator having to manually trim them, there is a utility called "logrotate".

This utility can be configured to rotate (backup and compress) old log files and make way for new ones. It can also be configured to only store a certain amount of log files, making sure that you keep your disk space free.

The files which controls this behavior is **`/etc/logrotate.conf`**. See the logrotate(8) man page for details on the syntax of this file.

Chapter 8. Hardware Installation

Adding a new drive to a Linux system:

While the installation process takes you through all the steps necessary in order to prepare a drive to install your system onto, you may find that you wish to add an additional drive at a later stage.

We will cover the steps you need to follow in order to achieve this:

Does the kernel detect the drive?

Once you've installed the physical drive media in your machine, and booted up into Linux, you will need to check and see if the system has detected your drive.

You should consult the output of the **dmesg** command to verify this:

```
debian:~# dmesg
[ ... ]
hda: ST360014A, ATA DISK drive
hdb: WDC WD100EB-00BHF0, ATA DISK drive
hdc: 24X10, ATAPI CDROM drive
[ ... ]
Partition check:
 hda: hda1 hda2! < hda5 hda6 hda7 >
 hdb: hdb1 hdb2
[ ... ]
debian:~# _
```

Here, we can see that there are two drives in the system (hda and hdb), and that the first drive has a primary partition (hda1) and an extended partition (hda2), which is further split up into 3 logical partitions (hda5, hda6 and hda7).

The second drive (hdb) only has two primary partitions (hdb1 and hdb2).

Partition the new drive

You must then use the **cfdisk** command to partition the new drive.

You will need to run the command with the device name of the new drive as a parameter.

Example, to modify the partition table on the second hard disk:

```
debian:~# cfdisk /dev/hdb
```

Partition the disk to your liking, as discussed in the original installation section.

Build Linux filesystem on any relevant partitions

If you created any Linux partitions in the previous step, you will now need to create a filesystem on them. You will use the **mksf** command to achieve this. Again, you specify the device that you wish to create the filesystem on as a parameter.

If we wanted to create a Linux filesystem on the second partition of the second drive, we would use the following command:

```
debian:~# mkfs /dev/hdb2
```

Decide on a mountpoint and create it

Now you'll need to decide where you want to mount the new filesystem, and create the mountpoint. This is very simply, as a mountpoint is simply an empty directory.

For our example, we'll decide that we want to mount the new filesystem under **`/data`**:

```
debian:~# mkdir /data
```

Add the new filesystem to `/etc/fstab`

The last step will be to add the newly created filesystem to your fstab file, so that the filesystem will be mounted at boot time.

fstab before:

```
debian:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>                 <dump>  <pass>
/dev/hda2       /               ext2    errors=remount-ro         0       1
```

```
/dev/hda1       none            swap    sw                      0       0
proc            /proc           proc    defaults                0       0
/dev/fd0        /floppy         auto    user,noauto             0       0
/dev/cdrom      /cdrom          iso9660 ro,user,noauto          0       0
```

fstab after:

```
debian:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>               <dump>  <pa
/dev/hda2       /               ext2    errors=remount-ro       0       1
/dev/hda1       none            swap    sw                      0       0
proc            /proc           proc    defaults                0       0
/dev/fd0        /floppy         auto    user,noauto             0       0
/dev/cdrom      /cdrom          iso9660 ro,user,noauto          0       0
/dev/hdb2       /data           ext2    defaults                0       1
```

The line that we added specifies that /dev/hdb2 should be mounted on the /data mountpoint, is of a ext2 type filesystem.

Now you can issue the mount command to mount this filesystem now:

```
debian:~# mount /data
```

Print configuration

Printing on Debian can be handled by one of several pieces of software; the original BSD lpr (line printer) system, a rewrite of this system, called lprNG, and also by CUPS (Common Unix Print System).

BSD Print Spooler (lpr)

In order to use the BSD Print Spooler, you will first need to install it as follows:

```
debian:~# apt-get install lpr
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  lpr
0 packages upgraded, 1 newly installed, 0 to remove and 0  not
upgraded.
```

```
Need to get 0B/93.1kB of archives. After unpacking 237Kb will be used.
Selecting previously deselected package lpr.
(Reading database ... 6049 files and directories currently installed.)
Unpacking lpr (from .../lpr_2000.05.07-4.2_i386.deb) ...
Setting up lpr (2000.05.07-4.2) ...
Starting printer spooler: lpd.

debian:~# _
```

/etc/printcap

Once installed, you will need to modify the /etc/printcap file to suit your tastes:

```
debian:~# cat /etc/printcap

# /etc/printcap: printer capability database. See printcap(5).
# You can use the filter entries df, tf, cf, gf etc. for
# your own filters. See /etc/filter.ps, /etc/filter.pcl and
# the printcap(5) manual page for further details.

lp|Generic dot-matrix printer entry:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:

# rlp|Remote printer entry:\
#         :lp=:\
#         :rm=remotehost:\
#         :rp=remoteprinter:\
#         :sd=/var/spool/lpd/remote:\
#         :mx#0:\
#         :sh:
```

The default installation has a local printer, attached to your first parallel port (/dev/lp0).

Explanation of the legend used

The legend for the two-character entries in the /etc/printcap file are as follows:

Name	Type	Default	Description

lp	str	/dev/lp	local printer device, or port@host for remote
sd	str	/var/spool/lpd	spool directory
af	str	NULL	name of accounting file
lf	str	/dev/console	error logging file name
pl	num	66	page length (in lines)
pw	num	132	page width (in characters)
pc	num	200	price per foot or page in hundredths of cents
mx	num	1000	max file size (in BUFSIZ blocks); 0=unlimited
sh	bool	false	suppress printing of burst page header

There are several more obscure options which are also available, and you can find them in the printcap(5) man page.

Explaining the Generic entry

The default entry is for a printer called "lp", and which can also be referred to as Generic dot-matrix printer entry.

An extract:

```
lp|Generic dot-matrix printer entry:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:
```

You can use the pipe ("|") symbol to separate as many different names for the same printer as you want.

The print jobs themselves are spooled into a directory called **`/var/spool/lpd/lp`**, and are then shipped off to the "/dev/lp0" device.

You can use the following commands to operate the print spooler:

lpr

```
SYNTAX:
lpr [ -P printername ] <filename>
```

The **lpr** command is used to submit jobs to the printer. You can optionally specify the printer to use by way of the "-P" switch. Otherwise, you will print to the default printer, **lp**.

Sending the contents of **`test.txt`** to the printer:

```
debian:~# lpr test.txt
debian:~# _
```

lpq

```
SYNTAX:
lpq [ -P printername ]
```

The "lpq" command queries what's inside the printers queue.

```
debian:~# lpq
lp is ready and printing
Rank   Owner      Job  Files                                       Total Size
active root       1    test.txt                                    9 bytes
```

Again, you can optionally specify the printer queue to query using the "-P" switch otherwise it defaults to the **lp** printer.

Here we can see that our previously submitted job is currently being processed.

Once it has finished printing, it will disappear from the queue.

lprm

```
SYNTAX:
lprm [ -P printername] jobnumber
```

The **lprm** command can be used to remove jobs from the queue. This is useful if there's been a paper jam, or if you suddenly realize you've sent your job to the wrong printer, or similar.

You can find out your print job's id number but inspecting the output of "lpq", as we did above.

```
debian:~# lpq
lp is ready and printing
Rank   Owner      Job  Files                                          Total Size
active root       1    test.txt                                       9 bytes
debian:~# lprm 1
dfA001debian dequeued
cfA001debian dequeued
debian:~# lpq
no entries
```

CUPS - Common Unix Printing System

For more information, visit this [http://www.cups.org/] page

The Common Unix Printing System, CUPS, aims to provide a portable printing solution for Unix based systems, based on the Internet Printing Protocol (IPP). CUPS provides System V and Berkeley command-line user interfaces, as well as command line and web-based configuration interface.

CUPS allows for easy integration of printing resources for both open source and proprietary operating systems in a homogeneous network.

The CUPS server (cupsd) should be run on the machine which is physically connected to your printing devices (in the case of a parallel, serial or USB printer), and should be the machine where you wish to spool jobs while they wait to be printed.

The CUPS client tools should, obviously, be installed on any Debian machines from which you wish to print to the remote printer(s).

installation

Debian already has CUPS packages available, so you can simply use apt to install them:

```
# apt-get install cupsys-client cupsys
```

This will install both the client and server portions of the CUPS system. This should be done on the machine which is to be the CUPS server.

lpinfo

```
SYNTAX:
lpinfo -v
```

```
# lpinfo -v
network socket
network http
network ipp
network lpd
direct parallel:/dev/lp0
serial serial:/dev/ttyS1?baud=115200
serial serial:/dev/ttyS2?baud=115200
network smb
```

The "lpinfo -v" command lists the supported and available devices on the system. The first word is the type of device; direct, file, network or serial.

The "direct" and "serial" devices are printers that are physically connected to the machine, while the rest are all available over the network.

lpadmin

The **lpadmin** command allows you to add, remove and configure printers.

You can also use the web based configuration tool, by connecting to the following URL from the local machine:

```
http://localhost:631/admin
```

Adding a printer:

```
lpadmin -p <printername> -E -v <device> -m <ppd>
```

Where "printername" is the name you wish to give your printer, "device" is the device as we discussed earlier, and "ppd" is the printer driver to use.

You can find a list of PPD drivers on the CUPS site.

Examples:

This adds a local printer, connected on the parallel port, and gives it the name "deskjet":

```
lpadmin -p deskjet -E -v parallel:/dev/lp0 -m deskjet.ppd
```

This adds a remote, network printer, on 192.168.1.22, and gives it the name "laserjet":

```
lpadmin -p laserjet -E -v socket://192.168.1.22 -m laserjet.ppd
```

Removing a printer:

```
lpadmin -x <printername>
```

You can simply remove a printer by specifying the "-x" switch.

Example:

This removes our previously defined "laserjet" printer:

```
lpadmin -x laserjet
```

Chapter 9. Configuration of the Kernel

The /proc filesystem:

The **`/proc`** filesystem is a pseudo-filesystem, which acts as an interface to kernel data structures. Most of it is read-only, but some files allow kernel variables to be changed.

You can use the **cat** command to view the contents of most files, and can use a combination of **echo** and a redirect to set the values for files you wish to change.

As an example, we'll change the hostname of the system, at least as far as the kernel is concerned.

The relevant file in the /proc system is **`/proc/sys/kernel/hostname`**.

```
debian:~# cat /proc/sys/kernel/hostname
debian
```

As we can see, the current value is "debian"; this is the name that we have given our machine.

We can also verify this with the **hostname** command:

```
debian:~# hostname
debian
```

To change the hostname to "something":

```
debian:~# echo something > /proc/sys/kernel/hostname
debian:~# hostname
something
debian:~# cat /proc/sys/kernel/hostname
something
```

Obviously, this change has only been made in the currently running kernel, and will be lost when the system is next booted.

excercise:

where would you set this value in order for it to become permanent?

Appendix A. Troubleshooting

Log files

If you encounter problems with a running system, always remember to check the log files in /var/log first. Usually, very valuable information will be logged here, which will hopefully allow you to more easily debug any problems that you might come across.

It's usually a good idea to have at least one window open with a **tail -f /var/log/messages** command running at all times. This means that you'll be able to very quickly spot any problems as they arise.

Common problems:

Lost root password:

1. Reboot the system; pressing **Ctrl-Alt-Del** from the console should achieve this without causing and damage to the running system.

2. At the LILO prompt, type:

   ```
   linux init=/bin/bash
   ```

   This will cause the Linux kernel to boot, and drop you straight into a running copy of the bash shell

3. When the system boots, it leaves the root filesystem in read-only mode. Once the system's booted and you're in bash, type the following command to remount the root filesystem as read-write:

   ```
   mount -o remount,rw -n /
   ```

4. Now you can run **passwd** and simply enter your preferred new root password.

5. Now you can type **exit** and the system will return to multi-user mode, and you should be able to log in as root with your new password!

Troubleshooting LILO:

As we've mentioned previously, Debian makes use of the Linux Loader (LILO) as the boot manager.

As LILO has to fit into a very small space in the first part of the disk, it doesn't come with very descriptive error messages.

If you attempt to boot your LILO-enabled Linux system and are presented with a partial LILO prompt, the following table will hopefully be able to help you track down the cause of the problem:

display status description

```
L 00   internal error
L 01   illegal command
L 02   address mark not found
L 03   write-protected disk
L 04   sector not found
L 06   change line active
L 07   invalid initialization
L 08   DMA overrun
L 09   DMA attempt across 64k boundary
L 0C   invalid media
L 10   CRC error
L 11   ECC correction successful
L 20   controller error
L 40   seek failure
L 80   disk timeout
L BB   BIOS error
LI       the second-stage boot loader loaded, but could not run
LIL     the descriptor table could not be read
LIL?    the second-stage boot loader loaded at an incorrect address
LIL-    LILO found a corrupt descriptor table
LILO    LILO ran successfully
```

The following error codes usually indicate a problem with the physical hardware:

```
L 02, L 0C, L 40
```

You should attempt to reboot the system a few times and see if this error is transient; if not, it may be time to replace that disk!

The other "L ##" error codes usually indicate a problem with the BIOS and its understanding of the device you are attempting to boot from.

A common cause of the "L 01" error message is as follows:

If your hard drive is not the primary master drive in the system (i.e. hda), then LILO may fail to detect this when it looks for the first (i.e., bootable) drive in the system.

Assuming that your drive is the secondary master (hdb), you can add the following entries to your **`/etc/lilo.conf`** file to fix the problem:

```
---
disk = /dev/hdc
bios = 0x80
---
```

Once that's done, rerun the **lilo** command to install the new boot loader. The "0x80" value is the drive that the BIOS considers to be the first drive it detects in the system. Therefore, if you wished to boot from the second drive that the BIOS normally detects during bootup, you would specify "0x81".

If you can't boot from your hard drive because of a LILO error, use the boot floppy that you created during the Debian installation process.

Some other LILO related problems could be caused by:

- not rerunning **lilo** following a kernel change
- installing and booting Linux from a very large (>2GB) partition you should either make the partition smaller, or switch to using GRUB
- installing an additional operating system, which overwrites the master boot record, and thus LILO
- errors in your **`lilo.conf`** file

LILO comes with some intensive documentation, which is available on your Debian system in the following directory:

```
/usr/share/doc/lilo/
```

Appendix B. Additional exercises:

Task One

You look after a Debian server for a small company; apart from you, there are three other people:

- Billy Bob, your boss.
- Jane Doe, the secretary and receptionist.
- Joe Blogs, the sales rep.

1. Create login accounts for these three people, and generate good passwords for them. Make sure that each user has their own UPG.
2. Lock Jane Doe's account.
3. Can you find out where the system makes a note of which accounts are locked? [Hint: read the man page for passwd(5)]
4. Verify that each user has their own UPG; what command do you use to do this?
5. Now create a directory called **`/home/shared`**. Make sure that Billy Bob and Joe Blogs have read and write access to it. Jane Doe, however, should only have read access. (It doesn't matter if everyone on the system has read access to the directory).
6. Pretend that you don't know the passwords for the user accounts, and make use of the **su** command to become the other users, and verify that they have the correct access permissions to the **`/home/shared`** directory.
7. Now create a cron job, which runs at 08:00AM every weekday, which sends an e-mail to Billy Bob, displaying the contents of the **`/home/shared`** directory.
8. Now create an additional cron job, which backs up the contents of the **`/home/shared`** directory into a tar & gzipd file called **`/home/shared-backup.tar.gz`**. The job should run every Friday at 6PM.
9. Which cron table did you edit to create the first cron job? Did you use the same

one for the second cron job, or a different one?

Task Two

1. Use the apt system to install the "zip" package.
2. Can you find out where the bin files for this package have been installed? What command did you use?
3. Now manually delete the bin files that this package installed.
4. Does dpkg still think that the package is installed? Why?
5. Use dpkg to re-install the missing files.
6. Now use dpkg to remove the package completely.

Appendix C. Practical lab

Matthew west has created a practical lab that will test your knowledge of System Administration. This can be downloaded from this link [../images/saprac.tar.gz]

This is how to use the scripts:

1. extract the archive: **tar -xzvf saprac.tar.gz**
2. run the "sabrk" to start the practical: **./nabrk**
3. run the "sachk" to check that you have successfully completed it: **./nachk**

This page intentionally left blank

Index