Dwarf's Guide to Debian GNU/Linux

2001 Dale Scheetz

dedicated
to
Linux users everywhere

CREDITS

First I want to thank Ian Murdock for writing the History section. His perspectives on those early years have helped latecomers like Dwarf understand the founding principles upon which Debian is based.

While Dwarf wrote much of the remainder of the book from his own experience, his fellow developers have always been willing to help resolve his ignorance. Without the fabulous mind-share of the development community, this book could never have been written.

This release includes graphic snapshots of the actual screens presented by the current installation. These shots could not have been taken without the use of VMware^{TM} Workstation software. VMware's support of this free software effort has included the donation of license to use it's MultipleWorlds^{TM} technology without fee. Dwarf is grateful for the quality of the presentation these screen shots bring to this book, and wishes to thank VMware for all their help in making this possible.

Dwarf also wishes to thank Peter S Galbraith for making this book available as the Debian package **dwarfs-debian-guide**, and Anthony Fok for his patches to the layout, and for catching several typos.

Typography

Bold

Bold type face is used to identify words that are name or titles, such as the names of headings on a display, the names of file formats or protocols, and even program names when the usage is pointedly the name and not the use or function of the program.

Italic

Italic type face is used to identify files, directories, and programs that reside on a file system, as well as their use in commands.

The note Style

Note: This style is used to deliver auxiliary information about the subject under discussion. Most often this is information that the reader may or may not understand, and is considers optional reading.

The screen Style

```
This style presents text
that might be displayed
on the screen of your computer.
```

Contents

Chapter 1

Introduction

About the Author

As Dwarf, the author began traveling the Information Super-highway in the early 1990's. Several years later he discovered Linux, when someone gave him a copy of the first edition of **The Linux Journal**. He purchased a CD from one of the adds in that magazine, and after only 3 or 4 tries, was successful installing Slackware on a PC that had previously only run DOS. He was almost immediately disappointed because many of the source packages that he was able to download off the net would not build because some library, or utility, appeared to be missing from the system. Being barely able to manage *make* provided none of the skills needed to resolve the problem. As Dwarf later discovered, the reason for these failures came from the nonstandard locations for libraries and utilities in a Slackware system. With that knowledge it is a pretty simple matter to edit the make file so the libraries can be successfully found. Lack of this knowledge forced Dwarf to try another distribution. Software Landing Systems (SLS) was installed next, and it did a much better job of supplying everything necessary to build packages directly off the net.

In fact, Dwarf used this system for all his network activities until he lost his computer to lightning two years later.

Once a new machine arrived a decision had to be made. Although the SLS system had been more successful than Slackware, the success rate was still down around 3 successes for every 4 attempts. Searching the various CDs in his growing collection, Dwarf discovered Debian. Although the release available on that CD had no PPP support, what was provided was much more successful at building the collected source files than either of the previous distributions! Finding the debian-devel mailing list and upgrading his system to a version of Debian with PPP support got Dwarf working as a developer, and he has been working with Debian ever since.

After that short history of the author, a short history of Debian seems to be in order. The following was provided by Ian Murdock, the creator of Debian:

A Short History of Debian

In late 1993, a college student and computer enthusiast named Ian Murdock was using SLS, an early distribution of Linux. He liked Linux but found himself disappointed that SLS had many problems and that, even worse, new releases of it failed to fix many of them. Convinced that this was mostly due to an overloaded, overworked maintainer, he decided to adapt the model used in the development of Linux itself and use it to create a new distribution with a decidedly different philosophy. He called the new distribution "Debian Linux", and it was to be developed by a distributed group of volunteers. This group was completely open and anyone was welcome to get involved.

Ian posted his intentions to the Usenet in August of 1993 and immediately found outside interest in his idea, including that of the Free Software Foundation, the creators of much of the core software of all Linux-based systems. Ian credits this early interest as being pivotal to the acceptance of Debian into the free software world.

Through the fall and winter of 1993, development of Debian proceeded through several internal releases, culminating in the public release of Debian 0.91 in January of 1994. Debian 0.91 gave the world its first glimpse of the Debian philosophy in action. By this time, a dozen or so people were involved in development, though Ian was still largely packaging and integrating the releases himself.

After the first public release of Debian, attention was turned toward developing the package system called dpkg. A rudimentary dpkg existed in Debian 0.91, but at that time was mostly used for manipulating packages once they were installed, rather than as a general packaging utility. By the summer of 1994, early versions of dpkg were becoming usable, and other people besides Ian began to join in the packaging and integration process by following guidelines that explained how to construct packages that were modular and integrated into the system without causing problems.

By the fall of 1994, an overloaded Ian Murdock, now coordinating the efforts of dozens of people in addition to his own development work, transferred responsibility of the package system to Ian Jackson, who proceeded to make many invaluable enhancements, and shaped it into the current system.

After months of hard work and organization, the Debian Project finally made its first distributed release in March of 1995, Debian 0.93 Release 5. Debian 0.92 had never been released, and Release 1 through Release 4 of Debian 0.93 had been development releases made throughout the fall and winter of 1994.

By this time, the Debian Project, as it had come to be called, had grown to include over sixty people. In the summer of 1995, Ian Murdock transferred responsibility of the base system, the core set of Debian packages, to Bruce Perens, giving Ian time to devote to the management of the growing Project. Work continued throughout the summer and fall, and a final a.out binary format release, Debian 0.93 Release 6, was made in November of 1995 before attention turned to converting the system to the ELF binary format.

Ian Murdock left the Debian Project in March of 1996 to devote more time to his family and to finishing school; Bruce Perens assumed the leadership role, guiding the Project through its first ELF release, Debian 1.1, in June 1996.

The Debian Development Team

From its humble beginnings, with a mere handful of developers, the Debian Development Team has grown to nearly 400 active developers and enough additional help from regular contributors to bring the total list of participants to over 600 people. Most of these people have never physically met each other. Almost all of the communication between developers takes place via e-mail using the various mailing lists that the project supports through donor supplied hardware and bandwidth.

All contributions to the project are completely voluntary, including the many fine people, businesses, and institutions that contribute hardware to operate those lists and disk space to store the archives. What little actual cash is necessary, has in the past been paid out of pocket by the person taking care of the issue. **Software in the Public Interest** now provides services that allow for the project to collect donations for such cash necessities. This volunteer organization, by its very nature, does not lend itself to the hierarchical structures usually found in development organizations. Control from a central location is ineffective, at best, and counterproductive in many cases. The reason this process works without those controls stems from the modular package scheme that was developed so early in the project. This allows an individual developer to take responsibility for a "known" piece of the distribution. The efforts of this diverse group are directed by the bug tracking system, and the set of Policy documents that define the construction of Debian packages, and to some extent by the Debian Project Leader.

The combination of mailing lists and bug reporting system provides the only checks and balances needed to adequately control each individual developer. If developers have any questions about the proper way to deal with package responsibilities, the mailing list provides the access to other developers who will gladly assist them with suggestions and comments. If developers generate packages which are poorly formed, the bug tracking system allows anyone who notices the problem to bring it to their attention. With the recent rapid growth of the development group, this system became inadequate for dealing with the problems brought on by that rapid growth. This need has created multiple small teams within the larger structure, whose specific task has some narrowly focused agenda. Some of the current agendas include: Publicity; Documentation; Quality Assurance; and Testing.

These subgroups operate on the same general principles as the larger group, usually with their own mailing list and a team leader. These teams have a far better chance of coming to closure on the issues that they deal with, than the larger group was ever able to accomplish. Within this loose structure, the driving forces come from the universal desire of all participants to create an exceptional product of the highest possible quality. This helps to quell personal agendas and keep people focused on the general goals. It is this development model, borrowed from the Kernel Development Team and modified to suit the needs of the Debian team, that has allowed Debian to become the powerful distribution that it is today.

What Makes Debian Different?

The major difference between the Debian distribution and other Linux distributions that are currently available is its open development model. This is not, however, the only difference between Debian and the other distributions.

The second, equally important difference, is Debian's strict adherence to the "Free Software" ideal. It is quite impressive, when you think about it, that this distribution is composed of freely redistributable software, complete with source code. Now, most other distributions also supply source code and these same programs, but they will also put packages into their distributions that can not be redistributed under certain conditions, without any concern for the legal problems that they deliver to their customer. Users of the Debian distribution can be assured that what they find in that distribution will have no constraints on the free distribution and modification of that software, leaving them free to build "value added" systems from this Distribution without fear that they will find themselves in court for misuse of someone else's intellectual property.

Debian provides areas within the archives for packages that do not meet these rigid standards, but are desired by the Debian community. Because of its free software status, the Debian packaging system can be used to package software that does not meet its own standards for free distribution.

Packages that are not freely distributable are found in the **non-free** section of the distribution. An additional category, called **contrib** is for those packages that would otherwise be free, except that they depend on some other package that is **non-free**. In this way, Debian provides a wide variety of software outside the distribution, in a way that protects its users from the legal ramifications of the **non-free** nature of that software.

There are substantial technical issues that separate Debian from the other distributions available today. Debian is dedicated to a strict interpretation of the Linux File System Standards, soon to be known as simply the File Hierarchy Standards. It is this strict adherence to these technical guidelines that helps make Debian such a dependably useful system. Foreign packages brought to a Debian system are typically easier to integrate into the system than with other Linux distributions.

As important as this standard is, the real technical superiority provided by Debian is its unique packaging system. This system allows for incremental upgrades of individual packages without the constant danger of ending up with a broken system. The modularity of the packaging system keeps each potential disaster localized within the narrow confines of the offending package. There are still plenty of ways to break the system, but the packaging system goes a long way toward protecting the system from such failures. Most of this protection comes from the dependency checking that is provided by this packaging system. Packages can declare their dependency on other packages and even declare that dependency to encompass a particular version of that other package. The installation software enforces these dependencies in a way that forces the dependencies to be satisfied before installation proceeds. This yields a functional package at the end of the installation, instead of one that will not operate for some strange reason.

Even with Debian's high quality, the complexity of Linux systems provides many ways to confuse the unwary user. Although Debian attempts to create an installation default that will fit the most general needs of the user, there are many areas where knowledge and skill are required to get the most value from this operating system. Debian provides for these needs with mailing lists. The debian-user mailing list is an open subscription list. To subscribe you simply send an e-mail message to:

```
debian-user-REQUEST@lists.debian.org
```

with the subject and message body containing the single word **subscribe**. This will bring you into contact with other users, potentially having had your problem in the past, who can help reduce your confusion. The strength of this list is that developers lurk on the list as well as users, so there is the possibility of getting expert advice from someone who understands your problem.

In addition, the debian-devel mailing list is also open to public subscription. Although its goals are far more technical in nature, much useful information can be gleaned from lurking on this list. As with the user list, a simple e-mail to:

```
debian-devel-REQUEST@lists.debian.org
```

with the **subscribe** statement in the subject line will result in mail from this list being sent to your e-mail address. The volume of these lists is quite high at times, which tends to scare folks away who are new to the Internet, but the quality of response is usually quite high. This is in stark contrast to other distributions, many of which don't even supply any e-mail address for questions. Those which do manage user mailing lists tend to be more closed and less helpful to the average user. Much of the difference comes from Debian's open development model, which welcomes the synergy created by multiple points of view. The helpful nature of these lists is way above average for this type of mechanism and most users find it a happy place to be while learning the ins and outs of the Debian system.

These differences have made Debian a competitive system with the other current leaders in the Linux community. The fact that Debian, without corporate sponsorship, has created a distribution that competes favorably against others with strong financial backing, is a surprising and remarkable achievement.

What is in this book?

The rest of this book is organized into three major chapters:

Chapter 2: Package Management Tools
Chapter 3: Installation
Chapter 4: Basic System Administration

Chapter 2 contains three major sections, covering the three principle tools used in managing packages. Starting with dpkg, the command-line utility that performs the principle package management tasks; the chapter continues with a discussion of dselect, the first user interface to the distribution archives; and finishing up with an introduction to apt-get, the newest package installing utility which provides a smoother handling of dependencies, among other useful features. These discussions are only intended to give the reader an introduction to these tools, with enough examples to explain installation, upgrading, and removing packages with these tools. For a detailed understanding of the advanced features of these tools the reader must wait until they have a Debian system installed. A search for documents on these tools will teach the novice user many useful things.

Chapter 3 provides detailed information about the installation of your own Debian system. The installation process proceeds in two major steps. After information intended to prepare the user and the machine for the upcoming installation, this chapter covers both stages of the installation, as seen from a CDROM installation. Debian provides many different methods of installation, both from local media as well as several network methods. While all of these methods are basically similar, there are minor differences that make it difficult to deliver a general description. Since most editions of this book will come with some kind of Debian CD, this method was chosen so that a straight forward description of the install process could be presented. Helpful alternatives are outlined at the end of this chapter.

Chapter 4 offers an introduction to the basics of administering your new Debian system. Many of the commands described in this chapter are also useful in daily user operations, so this chapter is useful information even when the target machine is a private workstation with no network connections. In addition, examples are provided for adding users, assigning group permissions and other rudimentary tools for administering a multi-user system like Linux.

In addition to these three chapters there are several appendices, covering various details useful to the installation process. Also contained in these appendices are several Debian documents of interest, as well as a copy of the license under which this book is delivered.

Dwarf hopes the reader will find the material in this book to be useful in their discovery of the Debian GNU/Linux operating system.

Luck,

Dwarf

Chapter 2

Package Management Tools

Introduction

A package in Debian is delivered in a single file using the .deb extension to identify it as a binary package. This package file contains all the data and program executable files delivered by the package, as well as all the control information used by the package manager to install the files correctly on your system. While you can install packages without the tools, it is much easier to install Debian packages with the *Debian Package Management Tools*.

This chapter covers the three primary package management tools found in the Debian distribution; dpkg, dselect, and apt-get. The following pages will cover the use of these tools to install, upgrade, or remove individual packages as well as groups of packages. In addition the use of these tools to provide administrative information about the packages installed on the system which will be described. Details about how packages are actually built can be found in the developers documents provided with the Debian system, and are not a part of the material covered in this chapter.

dpkg is a command line interface to the basic package management functions in a Debian system. This is the earliest tool provided for the system, and provides all the direct access to the package manager database as well as the direct manipulation of the packages to install, upgrade, or remove individual packages. In this section the use of *dpkg* to install packages, check on the status of packages, upgrade packages, check which packages are installed on the system correctly and which failed to completely install, and removal of the package will be discussed. Examples of each activity will be provided to make clear how each of these features can be used on your new system.

While *dpkg* is the real workhorse of the package management system, the system maintainer will probably only use it on rare occasions to actually install packages. The first tool to provide features that allowed the installation of multiple packages in one pass was *dselect*. This tool is a full screen interface that allows browsing and choosing from all of the packages available in an archives. This section will cover such tasks as selecting an archival method, updating the available packages database, selecting packages for installation, dependency management, installation, and configuration. While this tool has some limitations, there are access methods for every conceivable archives type, and is often the tool of choice for installing several packages at once.

The major deficit for *dselect* is that it is incapable of installing the packages in a preferred order. Sometimes this leads to packages that will not install on the first pass because some package that it depends upon has not yet been installed. *apt-get* understands the order requirements of the packages it is going to install, as well as automatically dealing with the dependencies on the packages specified. While *apt-get* is primarily a command line interface, it has many advantages when installing multiple packages, and works as well over the net as with a local archives. A complete description of how to set up and use *apt-get* will be presented in the last section of this chapter and will include a discussion of tasks packages as another way to install multiple packages in a useful group.

While you will most likely use *apt-get* for all your package installation and upgrade needs, a basic understanding of the other tools will help you make the most out of the package management system provided in the Debian distribution. It is the goal of this chapter to provide you with this basic understanding of the package management tools.

dpkg

Introduction

The *dpkg* program is the workhorse of the Debian packaging system. When run with root privilege, it installs and removes groups of software files called packages. For developers, it builds packages. For normal users, *dpkg* supplies information on specific packages, the contents of any particular package, or a list of the installed packages on the system. It does all of these package management tasks and more. It is used by *dselect*, another application discussed in this chapter, to perform the actual installation of packages. Many Perl programs and scripts make calls to *dpkg* as part of their operation. A familiarity with *dpkg* provides solutions to many general packaging problems. When all other installation methods fail, *dpkg* is often capable of resolving the problem.

Dealing with the interdependence of packages is one of the major tasks of the *Debian Package Management System.* The dependency checking features of *dpkg* are well advanced when compared to other solutions available in the Linux community. What does this mean? It means that, if you use *dpkg* to attempt the installation of a package, and that package depends for its proper operation on another package that has not yet been installed, *dpkg* will complain and fail to install the new package. While this may, at first, seem to be a problem, it is actually the solution to the problem. It is far more unsatisfactory to be able to install a package that will not work after it has been installed. This dependency mechanism protects the system from such installations. As *dpkg* provides information on the package dependency that has not been satisfied, it is a straightforward task to install the dependent packages. Once the dependencies have been satisfied the package will install without problems, and run as expected afterwards. On some systems, one or more of these “dependent” packages might only be supplied by a “local” version of the software, unknown to the packaging system. There are ways to force *dpkg* to install the depending package even when it does not know that the dependency is satisfied. These options should be used with extreme care, and an understanding of the consequences. All of these interdependency issues are dealt with by an interface to *dpkg* called *dselect*, described in the *dselect* section starting on page 47.

The following discussion will cover all of the options that can be specified when using *dpkg* focusing on how to use them. All of the information contained here is available in "sparse" form by executing *dpkg -h |less*. Each of those options will be described in as much detail as possible in the following section.

Running dpkg

dpkg can be run at two levels. Many of the information features of *dpkg* are available to an unprivileged user account. For tasks like removing and installing packages, root privilege is required. This root privilege may be obtained from tools like *sudo* or *su* if root login is undesirable. Scripts that use *dpkg* to install or remove packages must either be run with root privilege or have the facility to gain root access for the critical operations.

sudo is provided as an alternative to logging in as root to obtain those privileges. Entries in the controlling file (*/etc/sudoers*) determine the commands that a given user can execute with root privilege. In this way, individuals or groups may be given access to areas of the system which are normally off limits.

What's with all this need for root access? Isn't it dangerous to use root for such a simple thing as package installation? Well, look at it from another point of view. Would you wish any user on your system to be able to install a potentially important package on your system? Those, hopefully, few people that are trusted to do the maintenance jobs that require root access are also those folks who should be trusted to install and remove packages.

Options Recognized by dpkg

Installation and Removal Options

-i | --install <package file>

This option is the standard way to install an individual package. The <*package file*> is the file name of the .deb file containing the package and must contain an adequate path to the file. So if you are currently in the directory */usr/debian* and the archives are in */usr/debian/stable*, you can either type:

```
dpkg -i ./stable/binary-i386/admin/cron_3.0pl1-38.deb
or:
dpkg -i /usr/debian/stable/binary-i386/admin/cron_3.0pl1-38.deb
```

with the same results. Either of the above lines will install the cron package on your system from the given location.

This is fine for installing packages one at a time, but an installation of all of the packages in a given directory tree is possible with the use of the **-R** option. This option causes *dpkg* to attempt installation of every .deb file that it finds while doing a recursive search of the declared directory tree.

So, if you wanted to install the entire admin directory as given in the above example the command would look like:

```
dpkg -i -R /usr/debian/stable/binary-i386/admin
```

Because of dependencies between packages this approach will not always (or even very often) be successful. It is very useful if you have a nonstandard archive that contains all the packages you wish to install and all of their dependent packages as well. Constructing such archives using FTP to download only those packages that you need will create archives that *dpkg* can install as a group. If the packages are all found in the directory */usr/local/Debian/archive*, then the command:

```
dpkg -i -R /usr/local/Debian/archive
```

will direct *dpkg* to install all the packages found there. *dpkg* will install the packages in the order it finds them. This leaves open the possibility that dependencies will not be satisfied during the first pass. On large complex collections of packages this can potentially take several passes at the package files.

A more general solution is to break the installation phase into two steps. The first step unpacks the package file, while the second step configures the unpacked files. Just which files are unpacked, and where they are placed after unpacking, is determined by a collaboration between *dpkg* and the package.

--unpack <package file>

Sometimes, as when dependencies get in the way of installation, it is desirable to break the installation into two separate steps. The *--unpack* option performs the first step of the installation. *dpkg* extracts files from the package and places them in their correct locations in the file system.

The configuration of the package, including the installation of configuration files, known as *conffiles*, is postponed until a later time.

Like install, this option will accept the *-R* | *--recursive* option. This allows for unpacking all of the packages in the given directory tree. Following up on the example discussed for *--install*, a large collection of packages gets unpacked by a command like the following:

```
dpkg --unpack /usr/local/Debian/archive
```

--configure <package name>

This option is used in conjunction with the *--unpack* option to complete the installation of a package. This option may involve scripts specific to the package that put the finishing touches on the package, or it may require that *dpkg* manage the installation of files known as *conffiles*.

conffiles are files that have been declared by their package to require special handling during their installation. The *passwd* file, that holds the password information, is just such a file. *dpkg* provides the opportunity to replace the current password file with a new one whenever the package is updated. In the case of *passwd* this is usually not desirable, while with other packages, such as *mimetypes*, it may be perfectly appropriate. If you do replace an old *conffile* with a new one, the old file is backed up with the added extension *.dpkg-old*. If you reject the new *conffile* a copy is provided with the extension *.dpkg-dist*. This allows for recovery from mistakes, but it also provides the material from which a composite *conffile* can be constructed. Thus, with the password example, if major changes happened to the structure of the password file, the new file is accepted and the old data edited into the new file. This produces a composite password file with the new changes combined with the old user information.

This *--configure* option causes these remaining details of configuration to be completed for a given package. To configure all packages that are unpacked but not configured the *-a* | *--pending* option replaces the package name. To complete the installation of packages previously only unpacked, the command looks like:

```
dpkg --configure --pending
```

and will complete the installation of any unpacked packages.

-r | --remove <package name>

This causes the package to be removed but leaves the *conffiles* installed. When a package will be removed, but later replaced, it is sometimes desirable to keep the *conffile*. An extreme example would be the *passwd* file. The password package could be successfully removed without damaging current system operations only if the *passwd* file remains intact after the package has been removed. This option provides that capability and allows for a package's removal and replacement without loss of the *conffile*.

The *-a* | *--pending* option in place of a package name will cause *dpkg* to remove all packages marked for removal in the *status* file. This *status* file looks much like a *Packages* file with the addition of a *Status:* field. *dselect* (discussed in the next section) edits this file based on the choices selected using its interface. It is possible to edit the file */var/lib/dpkg/status* (where this data is kept), but it must be done carefully. As with all of the files in this "database" maintained by *dpkg* and *dselect*, disastrous results can come from improper changes. A reasonable amount of knowledge and understanding should be acquired before changes are attempted in these files.

Use the tools provided to manage the package system. Only the most dire of circumstances will warrant abandoning the tools in favor of modifying the files directly.

--purge <package name>

Unlike *--remove*, the *--purge* option not only removes the package but also removes all the *conffiles* associated with that package. In this way all traces of the package are eradicated from the system. This option also supports the *-a* | *--pending* option in place of the package name.

Available Packages File Management Options

dpkg and *dselect* both use the available packages information found in the file */var/lib/dpkg/available*. This file is constructed from the *Packages* file found in the binary, contrib, and non-free sections of these archives. There are several tools that *dpkg* provides, as options, for creating and managing this file.

--clear-avail

dpkg will clear the available packages file of all information when given this option. There are times when the current file has been corrupted or otherwise broken in some fashion. This can happen for a number of reasons.

The most common reason is that it has gotten updated from a "broken" *Packages* file. In extreme cases, changes to the file format have resulted in files that failed to work with the old version of *dpkg*. This requires that the new version be installed before the file can be fixed. Cleaning out this file allows the upgrade to go forward. Once completed the available file can again be updated with the new *Packages* format and the installation may progress unhindered. Executing the command:

```
dpkg --clear-avail
```

is a way to clean the slate and start over from scratch and will typically be followed by a *dpkg --merge-avail* command.

-A | --avail <package file>

This will add the package information, found in the .deb file pointed to by <package file>, to the available file. So the command:

```
dpkg -A /usr/local/Debian/archive/joe_2.8-7.deb
```

will add an entry to the available file if there is not an entry there already. If the package has an entry, the entry will be updated with the information about this version of the package.

--update-avail | --merge-avail <Packages-file>

When the *--update-avail* option is used, the old information about the packages in available is replaced by the package information contained in the <*Packages-file*>. This file is found in the top directory of a binary distribution, so *Packages* files can be found in *binary-i386/*, *contrib/*, and *non-free/*.

Issuing the command:

```
dpkg --update-avail /home/ftp/stable/binary-i386/Packages
```

will replace the old available file with the contents of the *Packages* file found in */home/ftp/stable/binary-i386*. This option is intended for use when a new release archive is being used.

The *--merge-avail* adds the *Packages* file from another binary tree to the list of available packages. The new information gets combined with the old list, adding information on new packages to the available file. The following commands:

```
dpkg --clear-avail
dpkg --merge-avail /home/ftp/stable/binary-i386/Packages
dpkg --merge-avail /home/ftp/contrib/Packages
dpkg --merge-avail /home/ftp/non-free/Packages
```

will create an available file that reflects all three of these areas of the distribution archives.

Package System Information Options

-s | --status <package name>

This option delivers the status information, contained in the entry for the package named, from the status database. Much useful information about the package can be obtained from this display in addition to the installation status. Among them, the version of the package, its maintainer, and a brief description of the package. For instance, the command:

```
dpkg -s mc
```

will produce the output:

```
Package:  mc
Status:  install ok installed
Priority:  optional
Section:  utils
Maintainer:  Fernando Alegre <alegre@mars.superlink.net>
Version:  3.2.1-1
Depends:  libc5, ncurses3.0, ncurses-base
Suggests:  gpm
Conffiles:
/etc/mc/mc.ext 6e672da6b5961ba9125e11824fcb2fef
/etc/mc/mc.ini d41d8cd98f00b204e9800998ecf8427e
/etc/mc/mc.lib fcdd319ffefa2a9cb4b7a5fa3ccad72d
/etc/mc/mc.menu 4be9e09f0728f993fc468b98db403be0

Description:  Midnight Commander - A feature-rich full-screen file
manager.  Midnight Commander is a feature-rich file manager.  It has
mouse support in a Linux console and in an xterm.  It started as a Norton
Commander clone but now it has new features on it's own.

It does not use libncurses, but it uses the terminfo database.  Support
for XView and TCL/Tk is available, but it is still experimental.  It is
not included in the current binary.
```

--print-avail <package name>

Much the same information is provided by this option as by the *--status* option. While the status information is not presented here, the installed-size and architecture and package size fields are given in this listing.

Issuing the command:

```
dpkg --print-avail mc
```

produces the following output:

```
Package:  mc
Priority:  optional
Section:  utils
Installed-Size:  890
Maintainer:  Fernando Alegre <alegre@debian.org>
Architecture:  i386
Version:  3.5.17-1
Depends:  libc5 (>= 5.4.0-0), libgpm1, ncurses-base
Suggests:  gpm
Size:  371140

Description:  Midnight Commander - A feature-rich full-screen file
manager.  Midnight Commander is a feature-rich file manager.  It has
mouse support in a Linux console and in an xterm.  It started as a Norton
Commander clone but now it has new features on it's own.

It does not use libncurses, but it uses the terminfo database.

Support for XView and TCL/Tk is available, but it is still experimental.
It is not included in the current binary.
```

-L | --listfiles <package name>

All of the files that *dpkg* installed for <*package name*> are listed by this option. This does not, of course, include files created by installation scripts, as they are unknown to *dpkg*. Thus, the command:

```
dpkg -L mc
```

produces the following output:

```
/.
/usr
/usr/bin
/usr/bin/mcserv
/usr/bin/mc
/usr/lib
/usr/lib/mc
/usr/lib/mc/icons
/usr/lib/mc/bin
/usr/lib/mc/bin/create_vcs
/usr/lib/mc/bin/cons.saver
/usr/lib/mc/bin/mcfn_install
/usr/lib/mc/term
/usr/lib/mc/term/ansi.ti
/usr/lib/mc/term/linux.ti
/usr/lib/mc/term/vt100.ti
/usr/lib/mc/term/xterm.ti
/usr/lib/mc/term/xterm.tcap
/usr/lib/mc/term/xterm.ad
/usr/lib/mc/term/README.xterm
/usr/lib/mc/extfs
/usr/lib/mc/extfs/zip
/usr/lib/mc/extfs/zoo
/usr/lib/mc/extfs/lslR
/usr/lib/mc/extfs/a
/usr/lib/mc/extfs/rpm
```

```
/usr/lib/mc/extfs/deb
/usr/lib/mc/extfs/ftplist
/usr/lib/mc/extfs/extfs.ini
/usr/lib/mc/extfs/README
/usr/lib/mc/mc.hint
/usr/lib/mc/mc.hlp
/usr/lib/mc/mc.tcl
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/mc.1
/usr/share/man/man8
/usr/share/man/man8/mcserv.8
/usr/share/doc
/usr/share/doc/copyright
/usr/share/doc/copyright/mc
/usr/share/doc/mc
/usrshare/doc/mc/README
/usr/share/doc/mc/FAQ
/usr/share/doc/mc/NEWS
/etc
/etc/mc
/etc/mc/mc.ext
/etc/mc/mc.menu
/etc/mc/mc.lib
/etc/mc/mc.ini
```

listing every file installed by the **mc** package.

-l | --list [<package name pattern>]

Packages that match <*package name pattern*> are listed by this option, along with their installed status, the version number and the short description.

If no <*package name pattern*> is supplied then the list is a complete list of the installed packages on the system. Following our example with **mc** the following command:

```
dpkg -l mc
```

will produce the following output:

```
Desired=Unknown/Install/Remove/Purge
| 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 mc 3.2.1-1 Midnight Commander - A feature-rich full-scr
```

-S | --search <search file pattern>

The package containing the file described by <*search file pattern*> is returned by this option. Both the package and the location of the file are displayed in the output. To deviate from our current example, with something a little more informative, the following command:

```
dpkg -S whereis
```

creates the output:

```
util-linux:  /usr/bin/whereis
util-linux:  /usr/share/man/man1/whereis.1.gz
```

providing the information that this program is available in the package called **util-linux**. Installing that package will provide the *whereis* program.

-C | --audit

All of the packages that are not completely or correctly configured are listed by this option. Most often the correct solution is to run *dpkg --configure ¡package¿* to complete the installation. Executing the command:

```
dpkg -C
```

will create a listing something like:

```
The following packages have been unpacked but not yet configured.
They must be configured using dpkg --configure or the configure
menu option in dselect for them to work:
svgalib1-bin SVGA display utilities
ncftp A user-friendly and full-featured FTP client.
statserial statserial - displays serial port modem status lines
ftape QIC-117 (Floppy-tape) driver, in source code form.
j1 J is a dialect of APL freely available on a wide variety
tk40-dev The Tk toolkit for TCL and X11 - Development Package.
expect The expect/expectk programs and libraries.
tclX Extended Tcl (TclX).
acct The GNU accounting utilities.

The following packages are only half configured, probably due to problems
configuring them the first time.  The configuration should be retried
using dpkg --configure <package> or the configure menu option in dselect:

xdevel XFree86 3.1.2 developer's toolkit
```

Force Options

Force options are used to override the default behavior of *dpkg*. These options should be used with some forethought, due to the potentially dangerous nature of these commands. Each of these commands forces *dpkg* to perform in a way that is foreign to its normal behavior. This opens possibilities for damage to the system that are otherwise well contained.

The force options have two actions:

--force-<thing> which reduces errors on *<thing>* to warnings
--refuse-<thing> stops execution on encountering errors on *<thing>*

--no-force-<thing> == --refuse-<thing>

thus each of the options in this section can be preceded by either of the above prefixes.

downgrade

Install the selected package even when the current installed package is a newer version than the one to be installed. This option is enabled by default.

configure-any

Configure any other packages, as well as this one, that this package depends upon, which is currently only unpacked and not yet configured.

remove-reinstreq

Even if the package has been marked for required re-installation this force option will allow its removal. It is understood that under some circumstances this will result in files being left behind without dpkg's knowledge.

remove-essential
Removal of essential packages is not recommended because these packages are required for the proper operation of the system and removing them may break the system. In some rare cases, like the name change of a package, or the replacement of functionality by privately built software, it becomes necessary to remove one of these packages. This force option will allow that removal to proceed.

Miscellaneous Options

--print-architecture
Outputs the architecture of the machine for package building purposes.

--print-gnu-build-architecture
Prints the gnu format architecture of the machine.

--print-installation-architecture
Same as *--print-architecture*, but used by installation programs.

--license
Prints license and copyright information

--version
Prints the version of dpkg

--help
Displays a list of the options that dpkg currently supports, with a brief description of the options and their function.

dselect

Introduction

dselect is both a very powerful tool and a very difficult one for the new user to work with. Some folks have problems with the interface because it is unlike anything else on the planet. Many of the keys used are not the "standard" keys used to perform these functions in other software. Others are surprised by some of the "unexpected" things that *dselect* will do for, and in some cases, to them. Yet, there are many satisfied *dselect* users, some who would complain loudly if *dselect* were to change in any way. This suggests that the tool does what it was designed to do. The following discussion is intended to help the beginner figure out how to make *dselect* work for them.

In general, the principal rule when using *dselect* is: Pay very close attention to all the messages that appear on the screen before you press any keys. Even if you have seen the screen dozens of times before, you are cautioned to think twice before you initiate any key strokes. Read the help information that is available every time you use the program, until you are completely familiar with the information contained there. Take notes when odd behavior is noticed.

dselect uses separate scripts to manage the access portion of its functionality. These scripts, called methods, deal with the task of mounting the device containing the archive. As such, they are referred to as, mountable methods. There is a package called *dpkg-mountable* that not only adds another mountable method to *dselect* but also provides a log of all *dpkg* activity during the *dselect* session. This can be invaluable in determining just what went wrong with a particular installation attempt. This method is now a part of *dselect* and may be chosen from the **Access** menu option.

Using dselect

When *dselect* is executed, the first screen is the main menu which contains the following seven options:

```
  0.  [A]ccess Choose the access method to use.
  1.  [U]pdate Update list of available packages, if possible.
  2.  [S]elect Request which packages you want on your system.
  3.  [I]nstall Install and upgrade wanted packages.
  4.  [C]onfig Configure any packages that are unconfigured.
  5.  [R]emove Remove unwanted software.
  6.  [Q]uit Quit dselect.
```

These selections can be chosen by either using $\hat{\text{P}}$ and $\hat{\text{N}}$ ($\hat{}$indicates the control key is pressed with the letter) to move up and down the list (up and down arrows work as well) and pressing enter when the highlight bar is on the option you wish to choose, or by pressing the key for the character found in square brackets at the beginning of the option name.

0. [A]ccess

If no additional methods (like dpkg-mountable) have been installed then selecting *[A]ccess* will provide the six access methods: cdrom, nfs, hard disk, mounted, floppy, and ftp. Each of these "methods" needs slightly different information. The goal is to find the *Packages* file for the various pieces of the distribution.

The top half of the screen, topped with the red title bar, shows the menu items. The list can be traversed with the arrow keys and enter selects the item under the highlight bar.

The bottom half of the screen is bounded by blue highlight bars. The top bar names the access method currently highlighted. In between the bars is descriptive help information about the highlighted selection. The bottom bar indicates the percentage of text viewed, and the key that will change the current text position in the window.

This is a common pattern in *dselect*. The upper portion of the window i s where the activities occur and the bottom half provides additional information to help make the choice.

cdrom:

What you need to know:

- Device driver name for CD-ROM if not yet mounted
- Is the CD-ROM image a “standard” distribution?
- Mount point if already mounted.

The cdrom method will find an already mounted CD-ROM or will ask for the device driver that will mount the correct drive. All such drivers are found in */dev* and thus, the specification of an sbpcd driver would be */dev/sbpcd0* for the SoundBlaster Pro driver, */dev/scd0* for the first SCSI CD-ROM, and possibly */dev/hdb1* for an IDE ATAPI type drive that is the slave on the first IDE controler.

Once the drive is mounted the next problem is finding the distribution on the CD-ROM. The standard archives, as well as the CDs, use the path *debian/dists* for all the releases held in the archives. The release archives, called stable are pointed to by a symbolic link in this directory, so the path that *dselect* is looking for is: *debian/dists/stable*. *dselect* will append */main/binary-i386* to obtain the complete path. If you have a nonstandard CD, look for the directory *main/binary-i386* and supply the path which leads to it.

It is safe to assume that the archive is standard, since *dselect* will notice if it is not and ask again for the location of the top of the archive. See the Notes section on page 53 for more details.

nfs:

During a fresh installation, when you first get introduced to *dselect*, there will only be an Internet connection available if the machine uses an Ethernet card and it is recognized by the installation kernel. Without this kind of configuration a PPP/SLIP connection, typically through a modem, must be established before proceeding with this step. Note: There are several other ways to make a connection to another machine, local area net, or wide area net. Ethernet and PPP/SLIP are the most common. By whatever method, once a connection to the desired host is available, this method can be used.

What you need to know:

- URL to the NFS mount.
- Directory structure found on that mount.

When this option is chosen, the first request is for the domain name of the NFS server. This can be a numeric IP address, or a fully qualified domain name, of the form *something.someplace.net*, which will be used to contact the server. The installation software will mount the NFS volume found at the declared IP address.

Once mounted, the location of the “top level” of the distribution is required. *dselect* is expecting the path to a directory called stable that contains *main/binary-i386*. Some archives don’t support this path structure. Under these circumstances it becomes necessary to specify each binary distribution individually, and the correct answer is “none”. See the Notes section on page 53 for more details.

harddisk

Use this access method only when the hard disk device containing the archive is not yet mounted.

What you need to know:

- Device name for the hard disk partition containing the archive.
- Path to the various archives.

dselect will ask for the device name of the disk partition containing the archive. Once it has successfully mounted this partition it will request the path to the “top level” of the distribution. If you have a standard distribution this should be the path to the sub-tree, *main/binary-i386*, otherwise the answer is “none”. See the Notes section on page 53 for more details.

mounted

When the archive of the distribution is already mounted, this option only needs to know where the “top level” of the distribution is. This method does not care if the device containing the archive is a hard disk, a CD-ROM, or an NFS mount on another machine, so this method can be used to make a nonstandard CD-ROM or NFS mount “look” like a standard distribution. This can be done by constructing a subdirectory tree that will act as the “top level” of the distribution. Create a mount point (subdirectory) in the “top level” directory and mount the CD-ROM or NFS to that mount point.

Then create symbolic links for “stable”, “contrib”, and if available, “non-free”. Make these links point to their respective places in the mounted archive. As an example: If the mount point is */mnt/dist/debian* and the path on the CD-ROM to *binary-i386* is */bo/binary-i386*, then the link would be made by the following commands:

```
cd /mnt/dist
ln -s debian/bo main
```

with similar constructions for the **contrib** and **non-free** sections.

Once the directory structures are in place then just give */mnt/dist* as the “top level” of the distribution. If this seems too complicated then simply answer “none” to the first prompt and then give the explicit path to each of the parts of the distribution being installed. (See the Notes section on page 53 for details.)

floppy:

This is not a recommended installation method. Floppies are notorious for their failure rates. If there is a machine with an archive and Linux, and floppies are going to be made on that machine for one that only has floppy drives, it would be far more successful to bring the machines close enough together to install a null modem cable between them and establish a SLIP connection between the two machines and proceed with an FTP installation.

If this is impossible. . . , then the problem becomes one where you must have far too much idle time on your hands. CD-ROM drives are cheap. Buy one. If you wish to install the complete binary distribution, it rings in at just under 400 MB. Even on 1.44 MB floppies that’s still around 280 floppies.

For a limited number of packages this method has a reasonable chance of success. The best approach for this is to place all of the .deb files for the packages to be installed into one directory and build a *Packages* file for them. See Building Packages Files on page **??** for details.

ftp:

What you need to know:

- The domain name for the FTP site that is to be used.
- The path structure of the archive on that site.

Selecting this method will result in a prompt for the domain name of the ftp site which will be used for the installation. As with the other access methods, the path to the archive must be known and supplied for the proper continuation of the installation.

Notes

Each of the above access methods will at some point ask for the location of the "top level" of the distribution. If the archive is in the "standard" form, pointing to this "top level" will be all that is necessary. *dselect* will find a *Packages* file for the binary-i386 portion of the distribution as well as contrib, and non-free. The path to the local distribution will be asked for separately, as well as any of the above that were not found in the "top level".

If the distribution is in a nonstandard layout, *dselect* will notice this and complain that what was identified was not, in fact, the "top level". In this case simply answer "none" and each section will be prompted for separately.

Note: Switching VCs (see below) will allow an investigation of the distribution archive so that the path to *binary-i386* can be found as well as *contrib* and *non-free*.

Note: The Virtual Console (**VC**) is a method for sharing the console monitor with more than one login process. Debian configures six Virtual Consoles. The left ALT key pressed along with a function key (F1 thru F6) will place the corresponding session on the monitor. It's like having 6 separate terminals!

In addition to the three "standard" distribution collections, *dselect* offers the option of giving the location of the "local-distribution". This local distribution is for use at the direction of the local system administrator. Packages that have been created as .deb package files that are not available from the Debian archives are candidates for placement in the local archive. Although alternate packages that do exist in the archive can go here, the *Packages* file for the local archive must be "merged" by hand using *dpkg --merge-avail* before *dselect* will recognize them for installation. This is because *dselect* is unwilling to downgrade a package, so those packages must be the only ones in the available file for proper installation. If the intent is to operate in this fashion then the update of available will need to be done by hand and the "Updated Available file" option should not be executed.

1. [U]pdate

Choosing this option will update the available packages file from the *Packages* files available from the various distribution sections you identified in the "Access" section. If any custom construction of the available packages file has been done, then this option should not be executed.

2. [S]elect

This option delivers the main power of *dselect* as well as providing the greatest challenge. Here is where a thorough knowledge of the various keys used by *dselect* will provide major benefit.

Novice use of [S]elect

For those installing Debian for the first time with no experience with *dselect*, the choice of **[S]elect** should be considered an educational experience where no "real" actions will be taken. This should simply be considered to be a "look around" at what is available so a familiarity can be established with this complex and powerful section of *dselect*. No selections need be made here, since all "standard" packages are marked for installation at this time. This is a good starting point when configuring a new system.

On first entry to the **[S]elect** screen it should be noticed that the top of the screen is the normal selection area with the bottom area of the screen for information display.

Things to look at:

- Note the operation of the o key.
- Note the effect of the v key.
- Use the search key (/) to find a package in the list.
- Note the effect of the i and I keys here.
- Add a package to the install list.
- Dealing with depends screens.

On first entry into the select window a complex list of sections and packages is visible in the top section of the screen. The arrow keys will move the highlight bar up and down this list/menu. The order of the packages on this list can be changed using the 'o' key. Press the key several times slowly and the display changes the order in which packages are displayed from the starting point, ordered by priority, to ordered by section, and finally to an alphabetical listing of all available packages.

Further examination of the display will reveal, on the left edge of the display a cryptic set of symbols like *_* or some other form of arcana. Pressing the v key will expand these symbols into their "English" meanings, each symbol described by one word on the display line. This results in the loss of the available and installed versions as well as the short description that was found on the line before pressing v. Press v again and return the display to the terse mode.

Next, let's see if we can find a particular package among this information rich list. Midnight Commander (mc) is a nice package to have, but is not included in the "standard" packages, so it will not be currently selected for installation. Use the / key and enter **mc** when asked what to search for. Like any string search the package found will not necessarily be the one you are looking for. There are several other packages that start with mc and, unless the list is ordered in simple alphabetic order, these other packages are as likely to be found first as the desired one. To search for the next possibility, simply press the back slash (\) and the next match will be found. Several repetitions of this action will eventually find the package in question.

Once the **mc** package has been found, the bottom section of the screen will display the "long" description for this package. Pressing the 'i' key will provide several other informative screens. The first gives the installation status of the package. The next two give various control file displays which provide additional information about the priority, section, dependencies and package maintainer. Some of this information is helpful when tracing down various errors. The 'I' key removes this section of the screen all together, providing more space on the screen for listing packages. Pressing it again consumes most of the screen in the information section leaving only a few lines for the selection list. Thus this key cycles the display between these three different configurations.

At this point the short tour is over and if only the "standard" installation is desired the return key will exit the selection phase and the install can be started.

For those with a more adventurous spirit, this discussion will continue with a description of how to select an additional package for installation. This can be returned to after the standard distribution is complete, or be dealt with by using *dpkg* "by hand".

The plus key "+" is used to select a previously unselected package for installation. Note that the minus key "-" is the opposite of the plus key and designates the package for removal. This is not the same as "don't install" and should only be used when a package is really supposed to be removed. With packages for which no action is desired, such as "don't upgrade" the proper key is "H" which puts the package on Hold (no action to be taken with regard to this package).

With the highlight bar over the **mc** package file line in the display, press the '+' key. The "Dependency Conflict Resolution Help Screen" will pop up as a consequence of this action. This means that there is a dependency or a conflict that must be resolved before the chosen package can be installed. In this case it is a dependence on **libgpm1** that must be satisfied. Pressing the space bar drops the help screen and shows the dependencies necessary for **mc**. *dselect* is typically very smart about doing what is necessary to resolve the dependencies or conflicts. Only rarely will it set things up contrary to the wishes of the package administrator. The same keys that work in the main selection screen can be used here to force the desired actions by *dselect*. If *dselect* still doesn't agree, then the 'Q' key will force acceptance of the configured actions.

In the case of Midnight Commander though, *dselect* makes the correct adjustments and adds **libgpm1** to the list of packages for installation. Thus simply press "enter" to exit the resolution screen. Press enter again to exit the selection screen, or, if you are feeling really brave, look for more packages you might like to install. *dselect* takes the time to work through the complete list of available packages when doing the install, reporting which packages are being skipped and which are already installed. It is a good idea to do package selection once, so as not to have to wade through the list over and over during each successive installation phase.

Several packages that are not found in the standard section, but might be valuable are: **joe** and **vim**. Other packages of interest depend on the specific needs of the system being configured and are left up to the person doing the installation to decide.

3. [I]nstall

This option will use *dpkg* to install the packages chosen in the **[S]elect** section. Often one package or another will fail to install because its dependent packages have not yet been configured. This step should be run over again until all the packages have installed.

4. [C]onfig

Use this option to configure any packages that were not configured in the installation phase. It is often useful to select this option between each unsuccessful intall attempt, as it will potentially configure packages needed by the next install step, thus speeding up the overall install process.

5. [R]emove

This option will remove any packages marked for removal or purge in the **[S]elect** phase.

6. [Q]uit

When you are finished with *dselect* this option will exit the program.

This section has covered the basics of package installation using *dselect*. If the help screens are read every time dselect is used, a slowly growing understanding of the additional features of *dselect* will be the result. Try something new each time you use it and you will eventually move from being a novice user of *dselect* to having a well-rounded knowledge of its many powerful features. Just remember to take it slow and be sure that you understand the consequences of your actions before you take them. With care and patience *dselect* can become one of the formidable weapons in your arsenal of *Debian Package Management Tools.*

Upgrading with dselect

Once you have installed a system and have all the necessary packages installed, maintaining such a system with *dselect* is relatively easy. *dselect* (actually *dpkg*) keeps a database detailing the state of the system. What packages are installed, their versions, and additional necessary information are contained in the directory */var/lib/dpkg/info* while the status of installed packages are kept in */var/lib/dpkg/status*. The file */var/lib/dpkg/available* is updated by *dselect* whenever a new archive is to be installed. This is not automatic, but happens when the **[U]pdate** item is chosen from the main *dselect* menu.

From the information in these areas, *dselect* can do an upgrade with little interaction. There are several preparatory tasks that should be done first, to insure *dselect*'s success. There were several incremental upgrades to *dpkg* during the life of the 1.2 distribution. At one point upgrading may have "broken" the available file. The transition to actually using a new feature did not go as smoothly as expected. To get past this problem it will be necessary that the available file get removed. This is done by the command:

```
dpkg --clear-avail
```

After executing this command it will be possible to install the following packages using dpkg:

- The new version of ld.so
- The new version of libc5
- The new version of dpkg

These are critical packages and, although *dselect* can often "do the right thing" when upgrading an installed system, it is advisable to install these by hand. In any case it is often necessary to upgrade dpkg by hand. This package has the newest version of *dselect* as well as *dpkg* and often packages in the new release will need features that are only available in the newest version of *dpkg*.

Upgrades from 2.0 to 2.1, 2.2, or 2.3 should have none of these problems.

Once this preliminary work has been done, *dselect* can be run in normal fashion to perform a complete system upgrade. If there are packages that should not be upgraded for one reason or another, they can be placed on hold in the [**S**]**elect** screens. To do this, search out the package using the '/' and press 'H'. This will put the package on hold and *dselect* will do nothing with respect to this package. Additional packages of interest can be added at this time as well, so the system can be enhanced by new packages that are of interest.

After configuring these details in [**S**]**elect**, the upgrade will proceed as expected.

apt-get

Introduction

When the Debian distribution was still somewhat small, *dselect* and *dpkg* were adequate tools for package management. As the collection of packages has grown in size and complexity these tools are no longer enough to provide a smooth and easy installation.

The Apt project is expected to resolve many of these difficulties by providing a new suite of tools to augment *dpkg* and provide a better installation tool than is provided by *dselect*. While a full-screen interface is a part of the design, only the command-line interface *apt-get* and an apt method for *dselect* are currently in a useable form. These tools are fully integrated into the current installation process, and have done a lot to improve the installation.

In addition to installing packages, *apt-get* is an excellent tool for upgrading an existing system. Once you have the tool configured properly, you can upgrade your entire system over the Internet with a single command.

The following pages describe how to set up this package manager as well as just how to use it to install, upgrade and remove packages.

Selecting Archives

With *dselect* the Accept menu option is used to declare the location of the archives. *apt-get* obtains this information from the configuration file */etc/apt/sources.list*. When the system installation has been completed from CDROM this file will contain lines like:

```
  deb cdrom:[<disk label>]/ unstable contrib main
```

for each of the CDs that were registered during the installation process. While this may be adequate for all you future needs, *apt-get* can also access archives over the Internet. While your CDs will never change, the FTP archives are always being updated. Even the stable release will be updated with packages that need security fixes, and other serious bugs.

Depending upon how much risk you are willing to take, you can even follow the latest "unstable" packages using *apt-get*, providing the latest packages being produced by the Debian developers. However, immediately following the release of a new "stable" version of Debian, the unstable release tends to be very unstable and is not suitable for the average user. If you choose to take this risk, be advised that this could result in the installation of a broken package that is critical to your system function, making your system unusable.

In addition to the stable and unstable labels, each release has a code name that does not change as the release progresses from unstable thru frozen to the stable release. Dwarf recommends that you track stable, by tracking the named release. For the 2.2 release of Debian this name is potato. The 2.3 release is named woody. Tracking potato, instead of unstable, keeps your system stable, even after the new release and gives you the choice of when to upgrade to the new release.

In order to set up *apt-get* to use FTP to retrieve packages from an archive on the net, first comment out the cdrom: lines. (Comments are formed by placing a # as the first character in the line.) Then add the following line:

```
deb ftp://ftp.debian.org/debian potato main contrib
```

The *ftp://ftp.debian.org/debian* term names the archives site on the Internet. There are many Debian mirrors available throughout the world. Each mirror will report alternative sites when you log in using an FTP client. Choose one that is "close" to you on the Internet.

The *potato* term identifies the release to use within the archives, and the remaining terms indicate which sections of the archives *apt-get* will search for packages.

For those people who reside inside the United States, several important packages are not available from US servers because of cryptographic laws there. These packages can be legally obtained by US citizens from servers outside the US, so if you want access to these packages you will need to add another line specifying that server and the section you wish to access.

Updating Packages Files

When the system is installed using CDROMs, the *Packages* file is read from each CD inserted during the installation process. These files are used to update the apt database so that *apt-get* will be able to find the packages it is asked to install. For the 2.2 release this amounted to 4 binary CDs when the complete archives is included, providing over 4000 packages.

Under these circumstances you will not need to update the *apt-get* database again until you are ready to use a new release set of CDs. Then you will use the command *apt-get update*, and insert the new CDs when requested. This will bring the database up-to-date with the packages on the new CDs.

If you use an FTP archives of Debian to maintain your new system in the future, you only need to do an update whenever the archives changes. For the stable release this is quite rare, although it does happen, making it useful to keep track of the latest update release for stable. Running update when the release changes will keep your system up-to-date with the archives.

If *apt-get*'s **sources.list** file references the unstable distribution, you will probably need to run update each time you wish to upgrade your system. Since the unstable distribution changes often, and without notice, this is the only way to keep in sync with the archives.

Installing Packages

Installing packages with *apt-get* is very simple. You only need to know the name of the package. The version numbers and the location in the archives are managed by *apt-get* without any additional information. So, to install Midnight Commander (mc) simply enter the command:

```
apt-get install mc
```

and *apt-get* will do the rest. In this case, the dependence on a library is discovered and added to the list of packages to install. Once *apt-get* has determined all the packages that need to be installed, it will ask permission to proceed with the installation. Once a 'Y' has been entered the installation process continues to completion.

Many packages can be installed at one time simply by including their names in the installation list on the command line. So, **joe** (an editor), **zgv** (a graphics viewer), and **slrn** (a news reader), could all be installed at once using the command:

```
apt-get install joe zgv slrn
```

and *apt-get* will include any dependencies in the list of packages to install. Dwarf says this is the way to install two packages that depend upon each other and must be installed "at the same time".

Upgrading Packages

Upgrading a package, or a set of packages, is just as simple as installing them when you use *apt-get*. Once you are ready to upgrade to the new version of Midnight Commander, simply enter the command:

```
apt-get upgrade mc
```

and the package will be upgraded.

If you want to upgrade the entire system, leave off any package names:

```
apt-get upgrade
```

will upgrade all of the packages on your system.

This command will normally work just fine, but under circumstances where the dependencies have changed on an upgraded package, this command doesn't always do the right thing. To deal with these dependency problems, *apt-get* provides the **dist-upgrade** command, which uses a more sophisticated dependency resolution technique.

Removing packages

There are many reasons to remove a package from your system, and *apt-get* makes it just as easy to remove a package as it is to install one. To remove Midnight Commander simply enter the command:

```
apt-get remove mc
```

and the package will be removed. Any packages that depend on the package being removed will also be removed to keep the system operational.

Chapter 3

Installation

Introduction

There are several ways to begin the installation of Debian. This section will focus primarily on the CD-ROM method. While using one of the other installation techniques may require different medium the basic process will be the same for each installation method.

If the CD-ROM will not boot in the target machine, take a look at the installation instructions in:

```
debian/dists/stable/main/disks-i386/install.en.txt
```

Here you will find instruction on how to construct boot floppies, or install files on a DOS-bootable partition that can be used to boot the installation process with a DOS command. Once you have an installation kernel booted using one of these alternative techniques, and the drivers are installed for you CD-ROM drive, you can continue the rest of the installation from the CD-ROM.

The Quick Start section that follows is an outline view of the installation process. The next section will cover the preparation information you will need to begin the installation. The remainder of this chapter is delivered in two major sections: the First Stage, or the Base installation; and the Second Stage, complete package installation. When these two stages are accomplished there will be a complete Debian GNU/Linux system installed on your PC.

Quick Start

If the target machine can boot a CD then you will be able to begin the installation by putting the CD in the drive and re-booting the machine. If you don't have that luxury, boot the machine in DOS, either from a "DOS rescue" floppy or from the hard disk. As a last resort you may need to build a boot disk and a drivers disk on another DOS machine to get the installation started. These methods are covered in detail in the *docs/install.en.txt* file previously described. What follows next is a simplified, step by step, procedure that will result in the installation of a Standard Debian GNU/Linux system.

- If the machine will not boot the CD, use one of the alternative methods provided to boot the rescue disk.
- Set color or monochrome at the opening screen, read the intro, and proceed to the Main Menu, then choose Next to configure your keyboard.
- Partition and/or install a swap partition and, at minimum, a root partition. These will be created and mounted at your direction by the installation software when the file system is created. Repeated execution of this step will create whatever other partitions are necessary for the installation.
- Install the Operating System Kernel and Modules from the CD, the hard disk, or floppies, depending on which procedure is being followed.
- After the Modules have been copied into the target partition they can be installed for use during the remainder of the installation.
- Configuring the Network is provided next, so that the base system can be obtained from a remote site if the machine has an Ethernet card installed. This can be a many step configuration process, or, for CD installs to a stand alone machine, simply a matter of choosing the machine's name.

- Installing the Base System can now be done from a variety of sources. The installation software will offer to search the media you have chosen, declaring the correct path to the base2_2.tgz file.

- Configuring the Base System is simply a matter of choosing the appropriate time zone information. The rest of the information necessary for proper Base System configuration has already been collected.

- Now the system is made bootable. Even if some other boot method is to be used in the final system, follow that step with "Make Boot Floppy". Make a boot floppy so you can be certain of booting your new system.

- Reboot the system.

- Give root a password; decide whether to add a new user account to the system; several other questions, including whether you wish *simple* or *advanced* installation: and the rest of the installation begins when the install script runs either *apt-get* or *dselect* depending upon the choice made.

- It is easy to get a standard system using *dselect*. More complex package management is possible with *dselect*, but usually not by the novice adventurer. This piece of software should be your first serious candidate for study once the system installation is complete. Choosing *simple* presents a list of package groups called *tasks* which you can select for installation using *apt-get*. Skip the next *dselect* items if you use the *simple* installation method.

- From the *dselect* main menu, choose the Access option. Here you can choose from the CD-ROM, the Hard Disk, an FTP site, an NFS mount, or, heaven forbid, floppies. Tell *dselect* where to find the *Packages* files and it's back to the main menu.

- First time in, either for a new install or for an upgrade from a new archives, always choose Update from the menu next. This will update the available file from the Packages files in the archives.
- Enter Select: the standard set of packages have already been pre-selected for you, unless you chose a profile. Now go straight to Install. This process is fairly long and will ask some questions near t he end of the installation period. When this completes with no errors it is time to quit dselect and start using your new system.
- At this point a Standard installation has been achieved. However, you may find packages that are not installed but are desirable and available in this release. To install them, restart dselect (as root) and go back to the Select screens. Now begin to learn about the heart of dselect. Pay close attention to the help screens and don't follow your instincts.
- Now the real fun begins!

Before You Begin

There is a bit of preparation that may need to be done before the installation can begin. If you are familiar with creating boot floppies, and partitioning the hard drive you may wish to skip this section and move ahead with the installation steps.

Boot Media

If your PC will boot the installation CD-ROM provided with this book (Disk #1 of the 3 CD set), then you already have all the boot media you will need.

If the CD-ROM will not boot on the target machine, see the instructions on creating boot floppies provided in the **install.en.txt** file located in:

```
debian/dists/stable/main/disks-i386/current/doc/
```

The rescue disks are a 2 floppy set of a separate boot and root disk. In order to get to your CD-ROM you will most likely need the additional 3 floppy set of disks that hold the various drivers needed by the kernel to access the different kinds of CD-ROM devices that may be installed on the target machine. With these floppies installed, the rest of the installation data can be accessed directly from the CD-ROM.

Partitioning the Hard Drive

Outside of choosing which packages, of the many provided by Debian, that you wish to install on your system, the hardest part of any Linux installation is determining how to partition your hard disk drive. The following discussion is an attempt to simplify the problem for the first time user of Debian.

First, what is a partition anyway? Many DOS machines only offer the C: drive as a valid disk device, but many of these machines have D: devices and more! Many times these additional "drives" are actually another physical hard drive connected to the machine, but more often they are not. Many OS installations (and even some Linux distributions) simply consume the physical disk device in one large chunk, called a partition. This single partition gets formatted with the desired file system and then sub-directories and files can be created within that file system.

Using partitions it is possible to divide one physical device into several partitions, and assign each partition a unique device name. In DOS these would be C:, D:, E:, etc... In Linux these are called /dev/hda1, /dev/hda2, etc... for the first physical IDE device on the system and /dev/hdb1, /dev/hdb2, etc... for the second physical IDE device.

The Swap Partition

Linux systems typically use a partition as swap space. This is where unused items taking up space in memory are placed until they are needed again so that the memory they are using can be used for something else. Other operating systems use a file on the file system for this purpose, and while Linux can be set up to do the same thing it is typically set up to use a swap partition because it is faster and isolated from the rest of the file systems.

You can't have too much swap space. Although excessive swap space may amount to wasted resources, it will not cause the system to exhibit thrashing or otherwise cause operational problems. It simply will not get used unless memory usage escalates.

Note: **Thrashing** - When the operating system spends most of its time paging real memory in and out of the swap space it is said to be "thrashing" and no useful work gets done. This can happen for a variety of reasons. On another operating system, larger than "optimum" swap space would quickly drive the system into thrashing. Deciding the amount of RAM that will be swapped out based on the size of swap space causes "all" pages to be swapped out immediately after they are used and the machine spends all its time moving pages in and out of memory and never gets anything done. (Well, at least it seems like it) The more reasoned approach is to base swapping activity on the age of the unused page and the amount of free memory. As the amount of free memory decreases, pages get swapped at an earlier age. This can still be made to thrash if placed under enough load. With many processes allocating and using many buffers, memory can fill up faster than it can be swapped out. The end result is either machine lockup because of exhausted memory or it begins **thrashing** when the swap age gets to zero.

More real memory does not necessarily mean more swap space. The need for swap is much more dictated by the demands of the processes that consume memory. Thus a swap ratio of 3 to 1 which works fine compiling kernels, as in the 24 MB swap to 8 MB RAM in the above example, fails when given a larger task. But the same ratio on a 16 MB machine gives 48 MB of swap space. This will be adequate for a large task but may still fail under a heavier load.

Although large disk drives are dirt cheap, there is not always any dirt available, and you are forced to compromise. Often after parceling out the disk, the left over partition space is not adequate to the swap needs. You are better off to take the space from the root partition and live in cramped disk space than to fall short of swap space. A workstation with 16 MB of memory will be able to do most any relatively complex work, including compilation of a math intensive package, without running out of virtual memory, with only a 48 MB swap partition. More memory in this environment means less swap. Less memory means more swap.

The root partitions

Now that decisions about swap size are as clear as mud, let's see if Linux partition sizes are any easier to determine. The base system takes slightly less than 20 MB of disk space, while the "standard" installation consumes somewhat more than 200 MB of disk space. A "complete" installation may exceed 400 MB, but doesn't need to go on one partition. Here is where the first decision must be made. Is the whole installation going to go on one partition? Well, if there is one physical drive large enough, what reason would there be to create more than one large partition? (Leaving room for a swap partition, of course.) There are, of course, several. Consider two of the most likely.

First, control of the user's consumption of disk space. If */home* is contained on the root partition the user disk consumption can consume all but 5% of the available disk space choking off all disk use except by the root account. This 5% is set aside for root privilege specifically so this event would not kill system activity. Root can then remove the offending files and return the machine to usefulness. Placing */home* on another partition allows that partition to contain only the space you wish to give to user activity, rather than the complete root partition. This means that even when users have exhausted their disk space the rest of the system has plenty to continue servicing other processes needs. The system continues happily along, oblivious of the plight of the poor user. This is, of course, very useful when the same platform is supporting both user activity and serving web pages. Having users plug up the web server is highly undesirable. There are also other solutions to this problem, such as the quota package, found in the admin section but, this is a complete hardware solution requiring no additional software or configuration-management.

Second, control over the contents of */usr*. Many systems do this by making */usr* read-only. The simplest way to accomplish this is to put */usr* on its own partition and mount it read-only. When the system administrator wants to add something to */usr* the partition gets re-mounted read-write, the changes get made and the partition gets re-mounted read-only. This partition could just as easily be on a CD-ROM which would guarantee the read-only nature of the file system. Of course any changes to */usr* would require re-mastering the CD.

Similar arguments can be made for placing */var*, */tmp*, and possibly others on their own partition. Controlling use of these areas by limiting their disk space to a specific partition size is a common way to manage disk resources. Let's see if this can be made clearer with a couple of examples. These examples will all be based on a 2 GB IDE hard disk, but the principles translate to SCSI and other disk devices as well. The first example is very simple, while the second is much more complex.

Example #1: Simple partitioning

Partition	Mount Point	Size (MB)
/dev/hda1	/	1000
/dev/hda2	swap	100

Notice that this example doesn't use all of the available disk. There is still some 900 MB of space left on the disk that can be used to create more partitions at a later time. The swap partition should be adequate to almost any system needs, and the root file system is large enough to accommodate a reasonable installation and still leave room for some user created data files.

Example #2: Complex partitioning

Partition	Mount Point	Size (MB)
/dev/hda1	/boot	16
/dev/hda2	swap	100
/dev/hda3	/	300
/dev/hda5	/home	200
/dev/hda6	/tmp	300
/dev/hda7	/usr	500
/dev/hda8	/usr/local	200

Using a */boot* partition keeps the kernel images stored there within the sector requirements for LILO, and gives one partition that can be booted which can then mount some other partition as root. Under the given situation that is */dev/hda3*, but there is still room on this drive for another partition and potentially another root file system. The kernel for that alternate system would also reside on the */boot* partition and be configured with LILO to use the alternate root file system. */home*, */tmp*, */usr*, and */usr/local* are on separate partitions so that their growth can be controlled.

Another interesting note about **Example #2** is that the numbers of the partitions skips the number 4. IDE partitions come in two flavors: primary, and logical. There can only be 4 primary partitions on any physical IDE device, but many more logical partitions can be created. However, to create logical partitions, at least one of the primary partitions must be turned over for this purpose. Thus, in our example above, the fourth primary partition has been taken over for the logical partitions 5 thru 8. Even if we made partition 3 into a logical partition as well, the logical partition numbers would still begin with 5, leaving the numbers for unused primary partitions completely unassigned.

While **Example #1** will produce a perfectly useful Linux installation, it is often useful for something a bit more complex. Feel free to choose just those components of the complex partitioning example that fit the specific needs for your system.

Networking Information Sources

Networking can range from two computers with a direct connection to each other to the many computers at General Motors linked with each other through a variety of methods. Connecting to other computers is wonderful, but even the setup of a simple PPP connection to an Internet Service Provider can be frustrating. In one chapter, all of the aspects of networking with Debian can't be addressed. There are many books available about networking in addition to information on the Internet. The focus here will be to complete a workstation Ethernet and/or PPP dial-up connection.

Linux provides a variety of networking methods such as Ethernet, SLIP, PPP, Arc-net and Apple-talk. The setup of these will be left as a challenging exercise to the reader. In either case there are a host of other sources for information on this subject. Once a standard Debian system has been installed, there is a complete set of FAQs and HOW-TOs in the directory, */usr/doc*. A good starting point would be the *NET-2-HOWTO.gz* file found in */usr/doc/HOWTO*. This document has a very good bibliography in the beginning, pointing to a wide variety of related HOWTOs that are very useful in understanding the situation.

Many of these documents are available in HTML format on the Internet. For instance the Ethernet-HOWTO is available at:

```
http://sunsite.unc.edu/mdw/HOWTO/Ethernet-HOWTO.html
```

or you can browse the index at:

```
http://sunsite.unc.edu/mdw/HOWTO/HOWTO-INDEX-3.html
```

The details that can be found there may be helpful in making use of the information that will be discussed in the following section.

Preliminary Networking Information

The Debian installation software provides for the configuration of the network as early as possible in the installation process. This creates the potential for performing the remainder of the installation from a network mounted device.

The network installation process is concerned with the construction of three configuration files:

```
/etc/hosts
/etc/resolv.conf
/etc/init.d/network
```

It does so by collecting the following information:

```
Host Name
Domain Name
Full Name
IP address
Netmask
Network Address
Broadcast Address
Gateway Address
Nameserver Addresses
```

The *Host Name* is the name given to the machine itself. The installation software provides the name **debian** as the default value. Any “reasonable” string of characters can be used as a host name, but normally this is intended to be readable and should thus be a legitimate name. On some local area networks these names are determined by the network administrator. In this case the name will need to be supplied by the administrator.

The *Domain Name* is typically the name of the network (i.e. planetlinux.com) that the machine is connected to. This will also be supplied by the network administrator. If the machine connects to the Internet through an Internet Service Provider (ISP) then the domain name is typically the domain of the provider.

The *Full Name* is constructed from the *Host Name* and the *Domain Name*. If the machine is named **fred** and the domain is **somewhere.net** then the full name becomes: **fred.somewhere.net**. This is the name that others will use when attempting to contact this machine with *telnet* or *FTP* or any other remote login client, so this represents the "complete address" of the machine in question. Typically a "name server" is used to convert this "full name" into an IP address for the site. When a site is not accessible with its full name, but is accessible by its IP address, this indicates that the system being used has no DNS facilities available or the IP address hasn't been registered through the Internic. Declaration of these servers is done in */etc/resolv.conf* and will be discussed in detail later in this section.

The *IP Address* that corresponds to the *Full Name* is a string made up out of the numbers between 0 and 255 and the dot (.) character. A typical IP address has the form: 192.64.1.1 and represents the "address" of the target machine. If your ISP can provide "static" IP addresses, you will be able to declare the address that they assign, at installation time. If your connection to the Internet uses dynamic IP addressing, supply the address 0.0.0.0 as a value to fill the entry.

The *Netmask* is a string of bits used to determine whether a given address is on the local sub-net or outside of it. Debian installation assumes a class C network and provides the default value of 255.255.255.0. This value is used to compute several other addresses for the system. If the machine belongs to a broader (or more restricted) network the system administrator will know the correct value to provide for the netmask.

The *Network Address*, the *Broadcast Address*, and the *Gateway Address* are all constructed from the machine *IP Address* and the *Netmask*. In our example above, the Netmask of 255.255.255.0 "masked" (logical AND) against the IP Address, 192.64.1.1, gives the Network Address of 192.64.1.0. To obtain the Broadcast Address, the Netmask is first inverted, to become 0.0.0.255 and a logical OR performed with the Network Address, yielding a Broadcast Address of 192.64.1.255 for our example machine. The address of the machine on the local net that connects to the rest of the Internet is know as the Gateway Address. These machines are known as routers and typically are the first (but sometimes the last) address on the local net. This would imply that our example IP address, 192.64.1.1, is that of a router. In this case the machine is its own gateway, which is fine depending on your setup.

The last set of information involves *Name Servers*. When you can ping an *IP Address* but can't ping the equivalent *Full Name* then there is a name server missing or not configured. If the machine connects to the Internet via an ISP, then they will provide the name server and will be able to declare the correct address. If the machine is connected by Ethernet the local system administrator will provide that information. The Debian installation will accept up to 5 name servers for entry into */etc/resolv.conf* to be used for name lookup.

The */etc/hosts* file from our example above would look like:

```
127.0.0.1 localhost
192.64.1.1 fred.somewhere.net
```

While */etc/resolv.conf* might look like:

```
domain somewhere.net
search somewhere.net
nameserver 192.64.1.1
nameserver 199.44.34.2
```

and, if there is an Ethernet card in the machine */etc/init.d/network* might look like:

```
#!  /bin/sh
# Configure the loopback device
ifconfig lo 127.0.0.1
route add 127.0.0.1

# Configure the Ethernet device
ifconfig eth0 192.64.1.1
route add -net 192.64.1.0
route add -net 0.0.0.0 gw 192.64.1.1
```

If you still have questions about the correct values for the various addresses and masks that will be requested during the installation ask the local network administrator or your Internet Service Provider, depending on which provides the Internet access for the machine.

PPP Configuration Information

If the target machine gains its Internet access using PPP rather than Ethernet, there will be some configuration necessary before access to the Internet can be established. The Debian Base System, comes with the PPP daemon installed, but not configured. Once configured, a PPP connection can be used to complete the installation, when necessary. Beginning with the 2.0 release, Debian has provided a PPP configuration utility, which will be offered at the first reboot after the installation of the base system. You should probably still read over the following information.

What will be described here is a simple connection to a service provider with no special authentication. This is the most common connection method and should work in most cases.

What You Need to Know:

From the Internet Service Provider:

```
The dialup phone number
The Login Name to use for the PPP connection
The account password
The IP address of the ISP's Name Server
```

The *dialup number* is not the same number you use to contact tech support or the front office of your ISP. This is the number that they will provide for connecting to their machine. The ISP will also provide you with a Login Name (possibly of your choosing) and a password to start with. This password should be changed to something else at the first opportunity.

The *Name Server* address will also be provided by the ISP upon request. This address allows the system to convert domain names into IP addresses, so that you will only need to remember something like **ftp.debian.org**, instead of **130.207.7.21**.

Debian now provides a utility called *pppconfig*, which is used during the installation to configure the PPP connection. This program uses the "peer method" of configuring the ppp daemon. This allows the configuration of several PPP connections to satisfy the various ISP or work and school related connections. For each peer you will need the previously discussed information about that connection. pppconfig will let you construct a new peer configuration as well as editing an existing peer. This script constructs a peer based options file linked with a chat script for that peer along with a possible papsecrets file. These files allow the PPP daemon to make a proper connection to the designated peer.

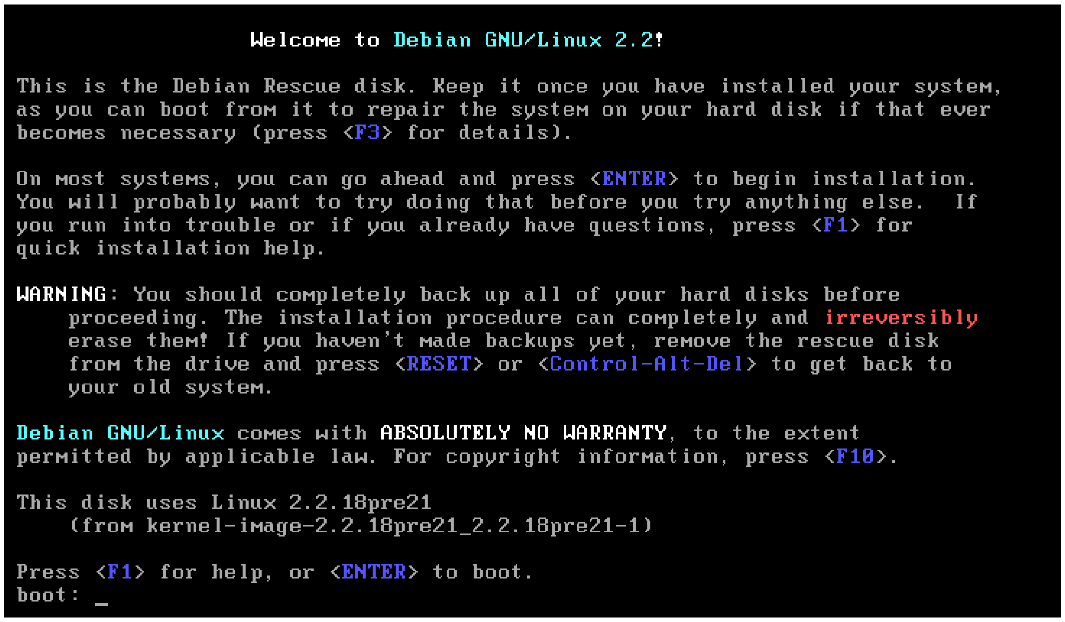
In addition to the configuration script there are two scripts, *pon* and *poff* provided for starting your various peer connections. To start the ppp connection simply enter the command *pon <peer>*, while *poff* will shut down the connection when you are finished with it.

You should now have all the information you need to begin the installation.

Stage One: The Base Installation

Once you have chosen a boot media, *Stage One* of the installation process can begin. In this Stage of the installation the Base System is installed. This is the smallest Debian installation possible that provides all the tools needed to complete the installation of additional packages upon rebooting the system for *Stage Two*.

If you boot from the rescue floppy or from the CD-ROM, the first screen you see will look something like the following:

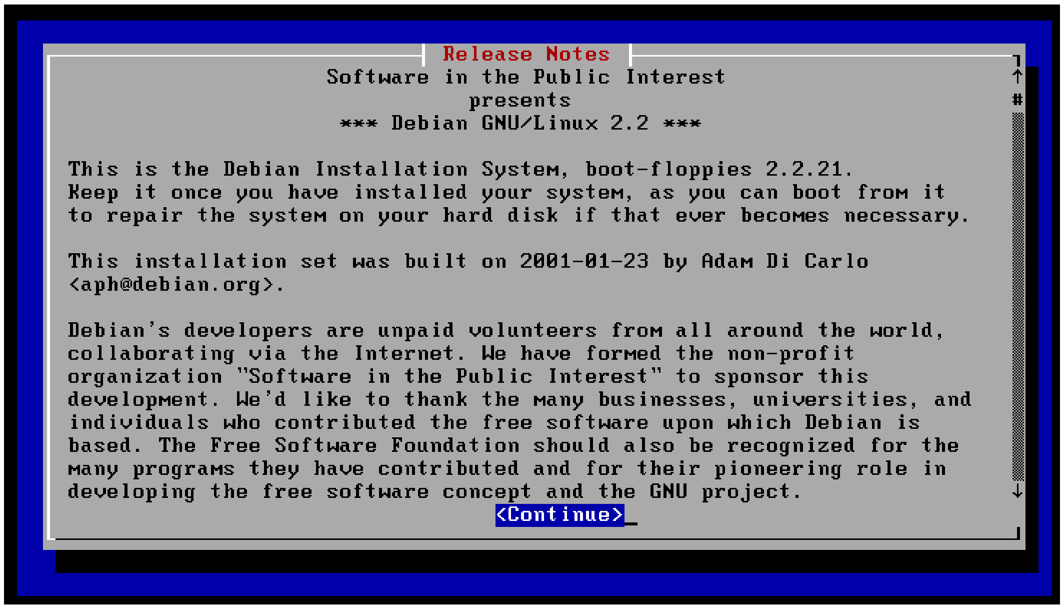


```
                  Welcome to Debian GNU/Linux 2.2!

This is the Debian Rescue disk. Keep it 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).

On most systems, you can go ahead and press <ENTER> to begin installation.
You will probably want to try doing that before you try anything else.  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 rescue disk
    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>.

This disk uses Linux 2.2.18pre21
    (from kernel-image-2.2.18pre21_2.2.18pre21-1)

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

Screen 1: Welcome to Debian 2.2!

Pressing enter will begin the boot process.

If everything goes successfully the following *Release Notes* screen will be presented:



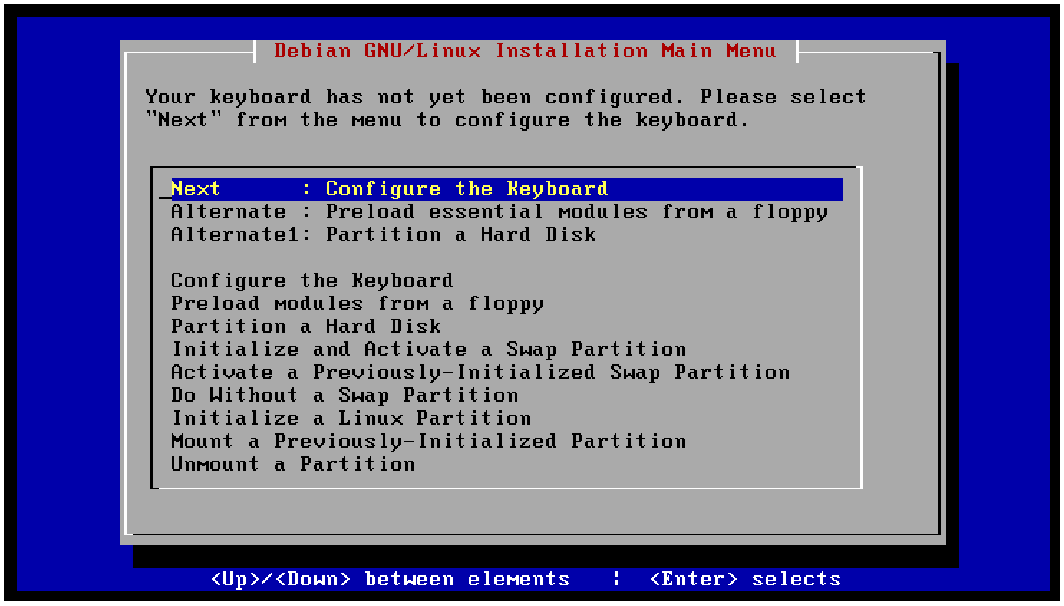
```
                    Software in the Public Interest
                              presents
                     *** Debian GNU/Linux 2.2 ***

This is the Debian Installation System, boot-floppies 2.2.21.
Keep it 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.

This installation set was built on 2001-01-23 by Adam Di Carlo
<aph@debian.org>.

Debian's developers are unpaid volunteers from all around the world,
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.
```

Screen 2: The Release Notes.

Pressing enter here will present the installation main menu screen:

Task 1: Configure the Keyboard

The first time the *Installation Main Menu* is presented you will be asked to *Configure the Keyboard.* A you can see in the screen shot presented below, there is an **Alternate** option suggesting *"Partition the Hard Disk"*. In addition to **Next** and **Alternate** as many of the rest of the menu options are presented as will fit on the screen.



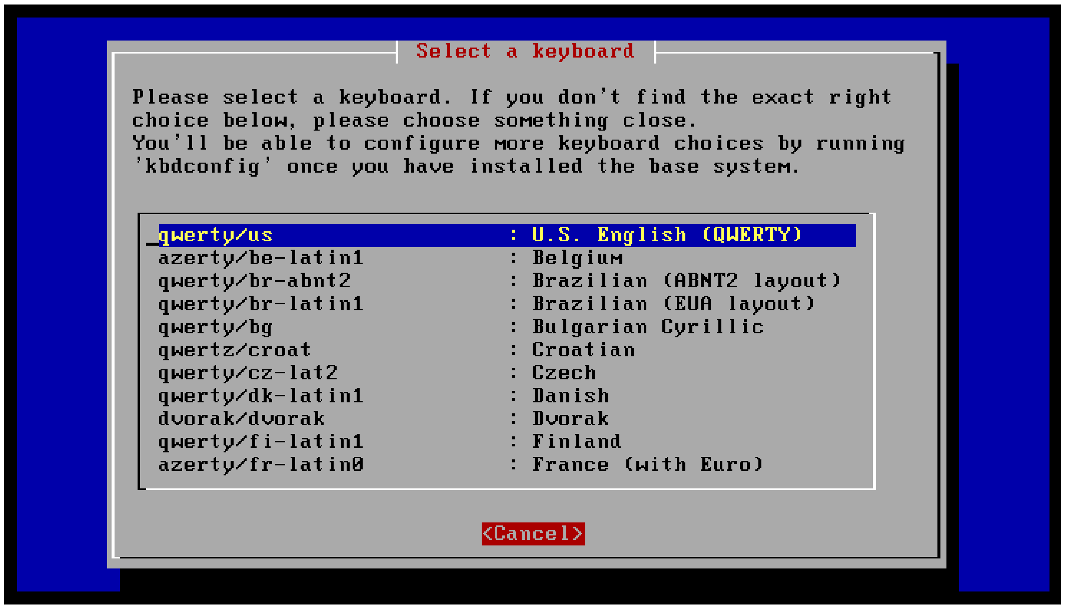
```
Debian GNU/Linux Installation Main Menu

Your keyboard has not yet been configured. Please select
"Next" from the menu to configure the 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
 Unmount a Partition

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

Screen 3: Main Menu; Configure the Keyboard.

The arrow keys can be used to move the cursor down the selection list. Moving beyond the bottom of the list would present the remaining options: Configure the Network, Install the Base System, Make Linux Bootable Directly From Hard Disk, Make a Boot Floppy, Start New System, Reboot the System, View the Partition Table, Execute a Shell, Configure PCMCIA Support, and Restart Installation System.

If you are doing something special, you can execute any of these options by placing the cursor over the desired option and pressing the return key. For a typical installation, the options offered at the top of the menu are most often the desired one.

Every time the script returns to the main menu the installation program will inspect the current state of the machine to determine which step in the installation is supposed to be performed next. It will then present several possible options for the next step in the installation at the top of the menu. The most likely choice is offered first with the highlight bar placed for its selection. Pressing the enter key when the cursor is over the **Next** option in the previous screen will present the following keyboard configuration screen:



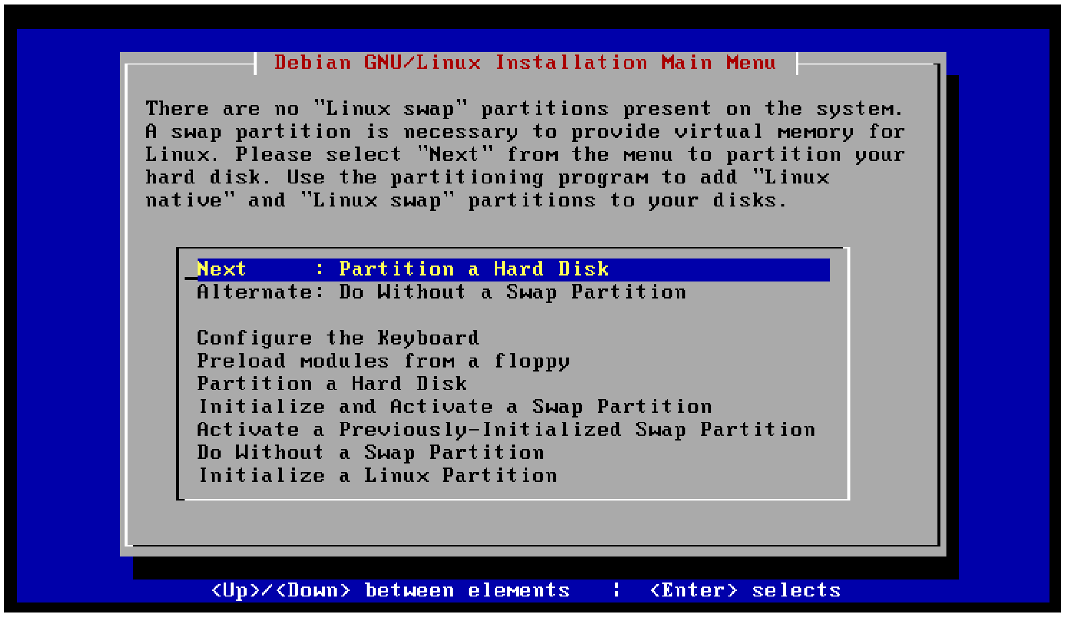
```
Select a keyboard

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
'kbdconfig' once you have installed the base system.

 qwerty/us                    : U.S. English (QWERTY)
 azerty/be-latin1             : Belgium
 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             : Finland
 azerty/fr-latin0             : France (with Euro)

                    <Cancel>
```

Screen 4: Keyboard Configuration Options.

Down arrow to see items below the bottom edge of the window and press enter to select an item.

Task 2: Partition the Hard Disk

Once the keyboard is configured, the installation software looks over the system again. If partitions don't already exist on the installed drives, then the menu will look something like the following:



```
Debian GNU/Linux Installation Main Menu

There are no "Linux swap" partitions present on the system.
A swap partition is necessary to provide virtual memory for
Linux. Please select "Next" from the menu to partition your
hard disk. Use the partitioning program to add "Linux
native" and "Linux swap" partitions to your disks.

    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
    Initialize a Linux Partition

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

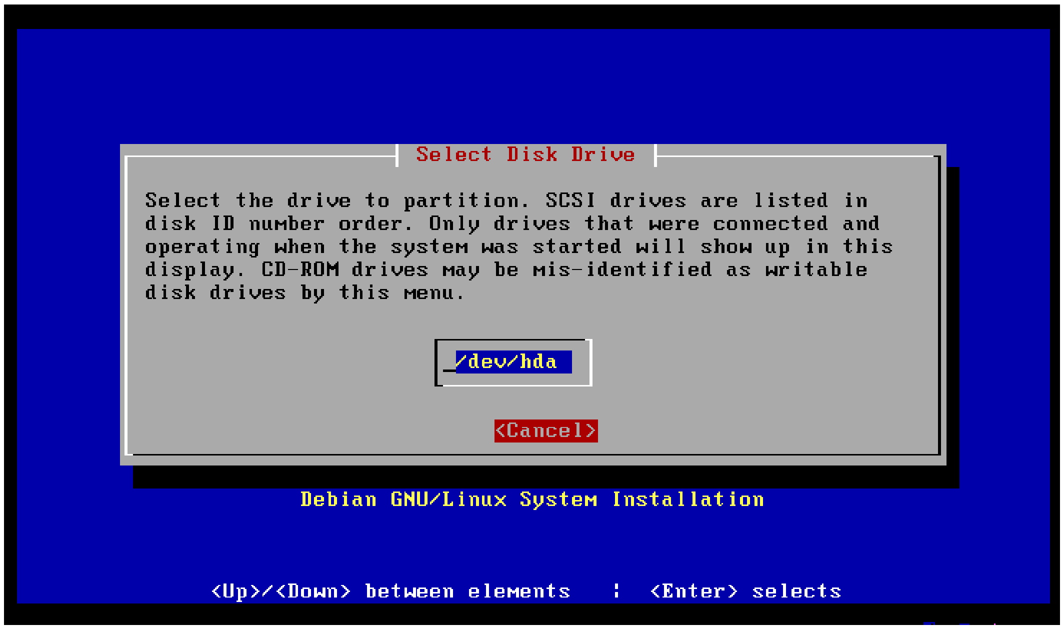
Screen 5: Main Menu: Partition a Hard Disk.

Note: Even if you are going to operate without a swap partition (not highly recommended) you will need a root partition on which to install your system. If partitions have already been built, the menu will look like the one in the next step and you can proceed to **Task 3** now.

The opportunity to select which physical drive will be partitioned is presented when the option to **Partition a Hard Disk** is chosen. *Cfdisk* is used to partition the specified drive.

Use this tool to create at least one Linux partition and, unless circumstances get in the way, a swap partition. With a 24 MB swap partition, and 8 MB of RAM, a 486 will compile a kernel without running out of virtual memory. However, when compiling very large packages, 24 MB is nowhere near enough, even on a Pentium, with 16 MB of memory. However, with 16 MB, a 48 MB swap partition becomes quite adequate. From this it can be seen that deciding on the size of the swap partition isn't really a simple formula. Depending on the tasks the system is expected to perform, the formula will be different for each configuration.

When **Partition a Hard Disk** is selected, the following dialogue box is presented:

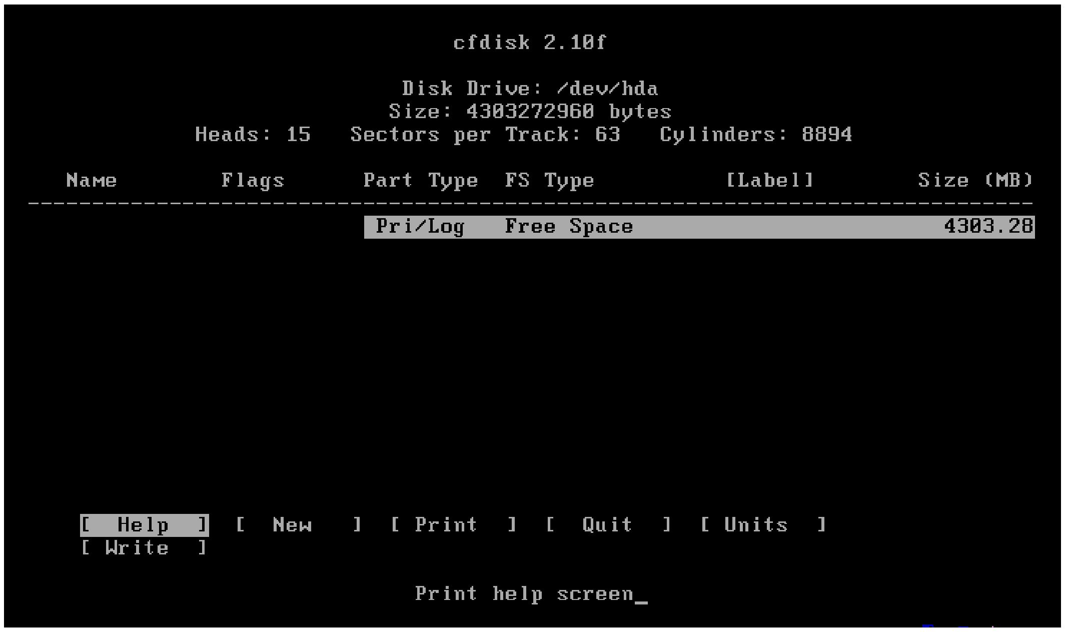


```
Select Disk Drive

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
```

Screen 6: Select Disk Drive.

Note that this machine has only one physical drive installed. In this example the choice is trivial, selecting */dev/hda*.

If this drive has not been partitioned yet, the screen presented will look something like:



```
                              cfdisk 2.10f

                          Disk Drive: /dev/hda
                         Size: 4303272960 bytes
              Heads: 15   Sectors per Track: 63   Cylinders: 8894

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                            Pri/Log   Free Space                       4303.28








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

                              Print help screen_
```

Screen 7: cfdisk; Empty Partition.

Debian Linux uses *cfdisk* which is a smaller, more trimmed down, partition editor than *fdisk*, and provides all the functionality necessary for partitioning a new device from scratch. Everything in the installation system has been chosen for its small size and *cfdisk* is no exception.

With no partitions created the obvious choice is **New**, which will give the options: **Primary**, **Logical**, and **Cancel**

Only 4 primary partitions can be created on any one physical drive. Usually this is sufficient. When it isn't, logical partitions are used to extend the number of partitions. Logical partitions, sometimes known as "extended" partitions live inside of a primary partition. To add logical partitions will thus require the removal of one of the 4 possible primary partitions. With only 3 primary partitions whatever space that is left on the drive can be divided up into as many logical partitions as needed. The limit on logical partitions is only constrained by the space on the disk.

Whether you are constructing a primary, or a logical partition, the total space left on the device will be offered for the new partition. Simply type in a new value to override the displayed size. Pressing enter to accept the value typed will present the options: **Beginning**, **End**, and **Cancel**. The new partition may be placed at either the beginning of the free space, or at the end of it, or its creation can be canceled all together, by choosing the appropriate option.

A few suggestions:

- Create the root partition first, and make it a primary partition.
- Create the swap partition second. It too must be a primary partition.
- Create whatever additional primary or logical partitions are desired.

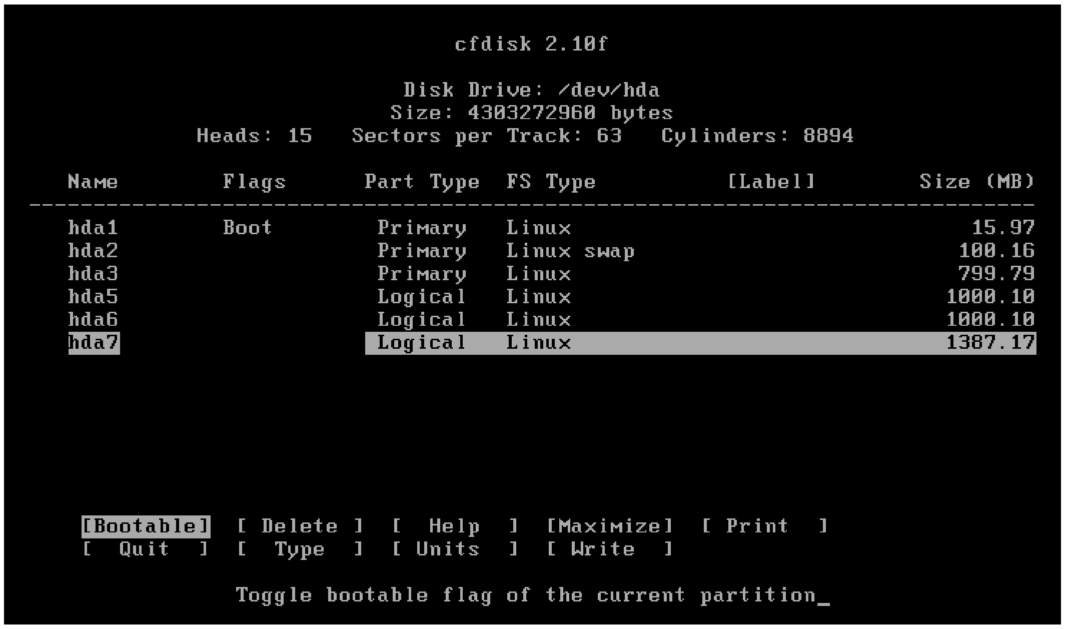
Whenever *cfdisk* creates a partition, it assumes that it will be a Linux partition and creates it so. When creating a swap partition the FS (file system) type is not correct. It should be a Linux Swap FS type. When the highlight bar rests on a partition the options section of the screen looks like:

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

While the up/down arrow keys move from one partition information line to the next, the left/right arrow keys move the highlight bar from one option to the next/previous one. In addition the first letter of the option will activate it as well, so pressing the 't' key will allow the FS type to be changed. This will display a list of all the allowed FS types. Typing the number of that FS type will change the partition to the new type when enter is pressed. The correct FS type for a swap partition is number 82, Linux Swap.

In addition, the first partition should be marked bootable if it is to be the boot partition. This is not necessary for additional drives that get added to the system.

In the current example, we need a partition for */boot*, a swap partition, a partition for the root file system, a partition for */home*, and a partition for */var*, with an additional partition for other possibilities (like an archives). The screen would look like the following when these partitions are completed.



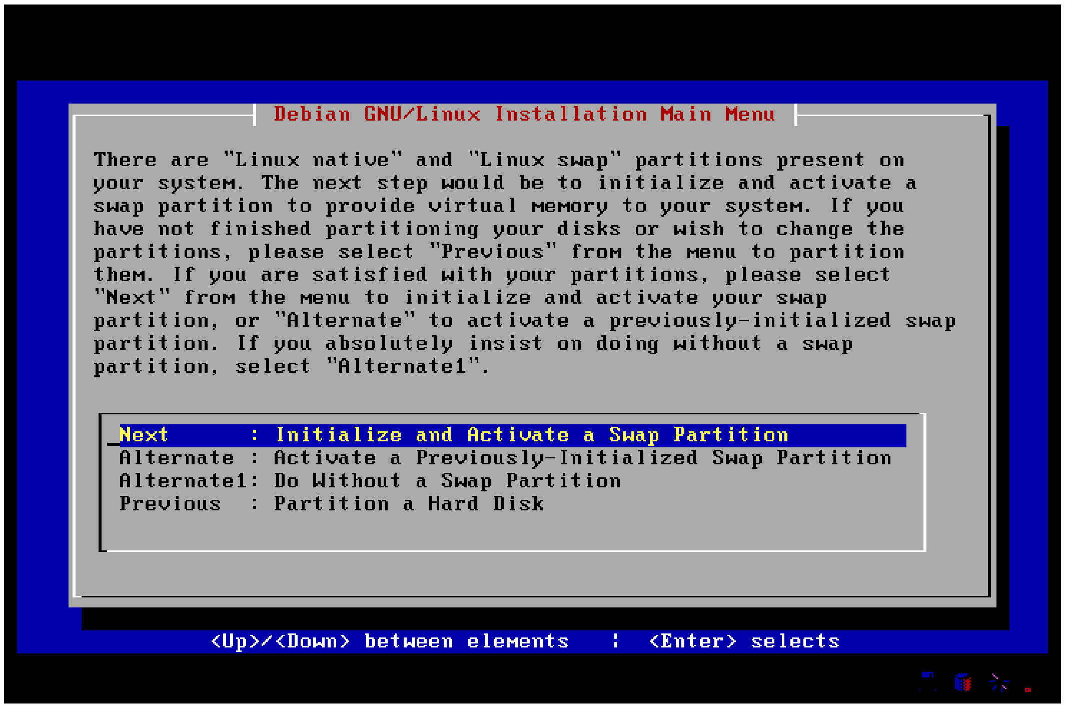
```
                              cfdisk 2.10f

                          Disk Drive: /dev/hda
                         Size: 4303272960 bytes
               Heads: 15   Sectors per Track: 63   Cylinders: 8894

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1        Boot        Primary   Linux                               15.97
    hda2                    Primary   Linux swap                         100.16
    hda3                    Primary   Linux                              799.79
    hda5                    Logical   Linux                             1000.10
    hda6                    Logical   Linux                             1000.10
    hda7                    Logical   Linux                             1387.17






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

                 Toggle bootable flag of the current partition_
```

Screen 8: cfdisk; Partitioning Complete.

Once the partitioning is complete you still need to Write the partition information to the disk. Use the arrow keys to move the cursor to the **write** option and press enter. The program will ask if you are sure you wish to write the new partition information. This particular tool demands that the word 'yes' be typed in full. The program is about to wipe out all data currently on this device so it wants to make very sure this is correct.

Task 3: Swap Partition

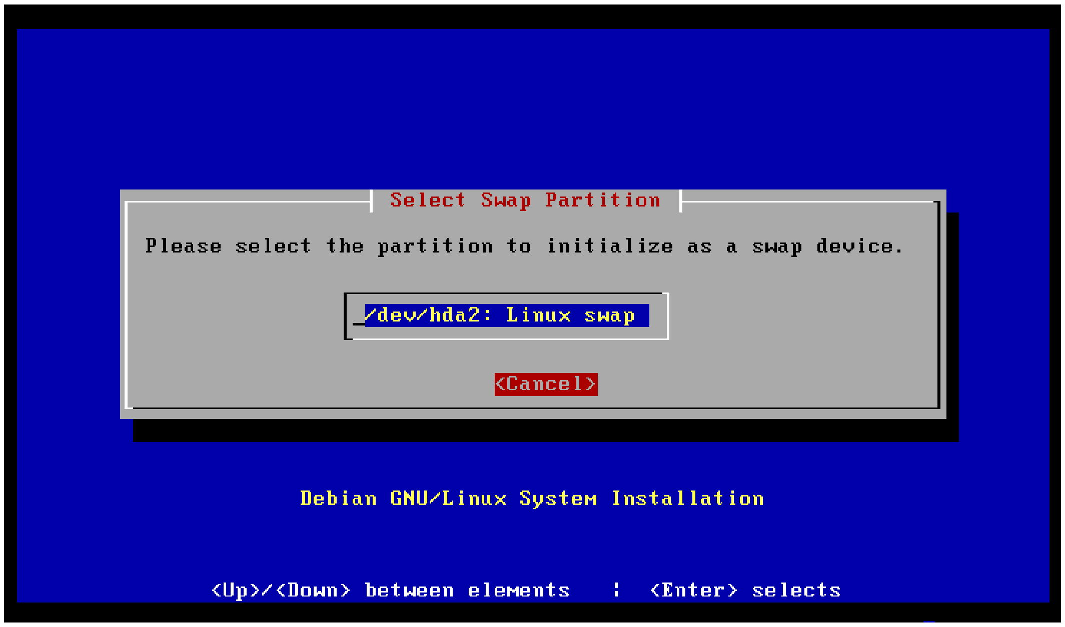
If the drive is pre-partitioned, or if **Task 2** has already been performed, the installation software will now suggest initializing and activating a swap partition. The Main menu will look like:



```
Debian GNU/Linux Installation Main Menu

There are "Linux native" and "Linux swap" partitions present on
your system. The next step would be to initialize and activate a
swap partition to provide virtual memory to your system. If you
have not finished partitioning your disks or wish to change the
partitions, please select "Previous" from the menu to partition
them. If you are satisfied with your partitions, please select
"Next" from the menu to initialize and activate your swap
partition, or "Alternate" to activate a previously-initialized swap
partition. If you absolutely insist on doing without a 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

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

Screen 9: Main Menu: Initialize and Activate a Swap Partition.

As we built a swap partition in the last section the **Next** choice is the correct one to follow. In the above case this will present the following dialogue box:



```
Select Swap Partition

Please select the partition to initialize as a swap device.

/dev/hda2: Linux swap

<Cancel>

Debian GNU/Linux System Installation

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

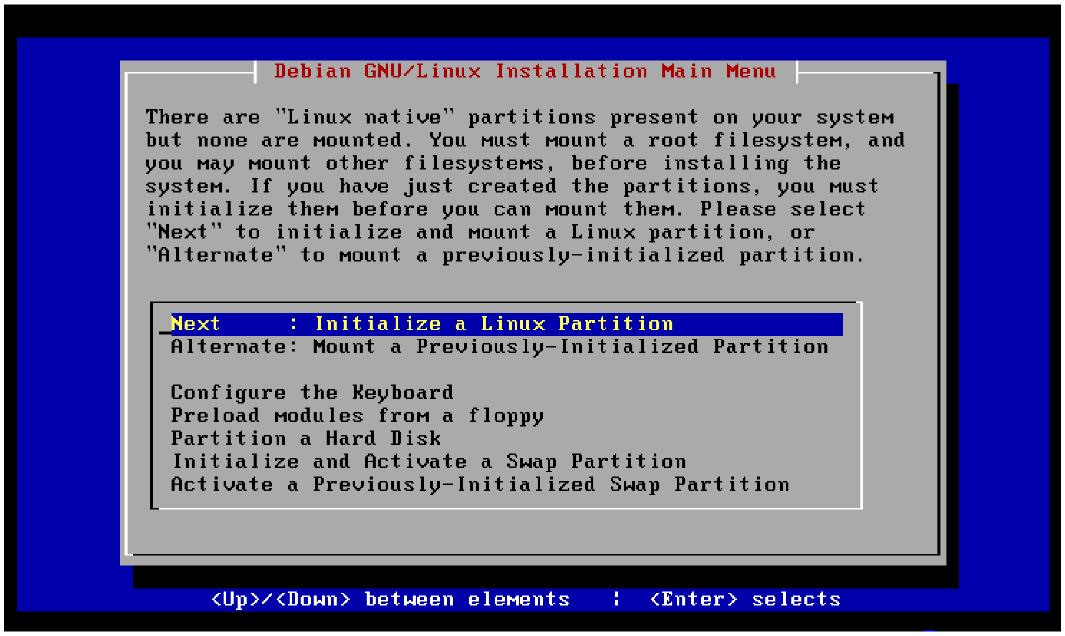
Screen 10: Swap Partition Selection

Notice that only the partition that was marked 'Linux Swap' is presented as an option for a swap device.

Pressing Enter here will present an information box asking if the partition should be checked for bad blocks. Answer yes here and the next question is “Are you sure you want to do this?”. Answer yes here and the swap partition begins to be initialized. When the initialization is complete the swap partition is mounted and the program returns to the main menu, after checking out the system to see how far the installation has progressed.

Task 4: Initializing Linux Partitions

Upon returning from setting up the swap partition the menu will look like:

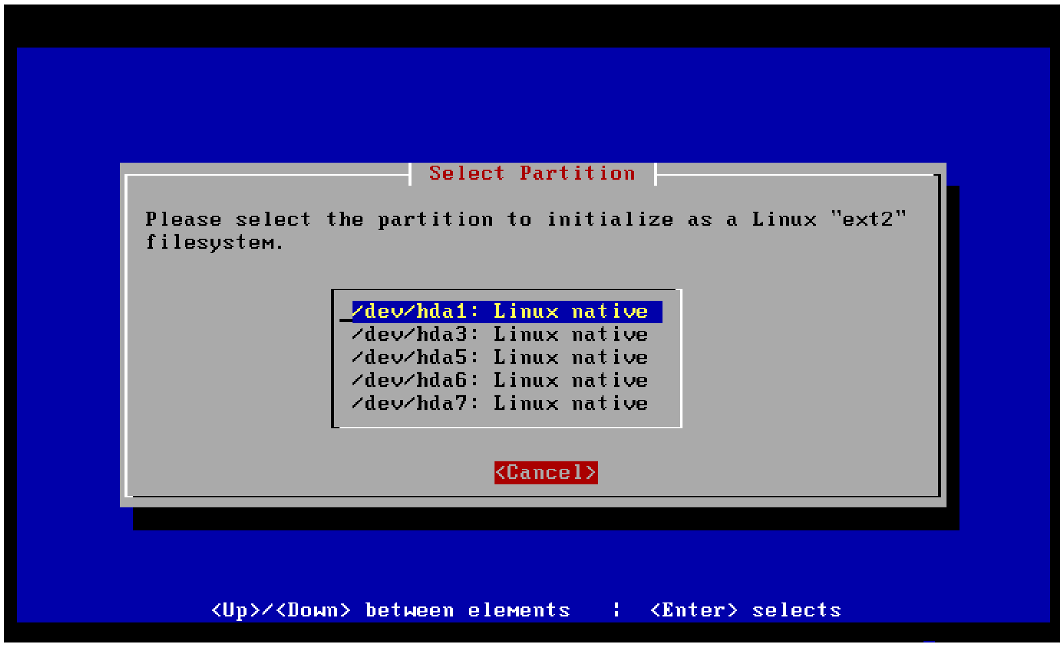


```
Debian GNU/Linux Installation Main Menu

There are "Linux native" partitions present on your system
but none are mounted. You must mount a root filesystem, and
you may mount other filesystems, before installing the
system. If you have just created the partitions, you must
initialize them before you can mount them. Please select
"Next" to initialize and mount a Linux partition, or
"Alternate" to mount a previously-initialized partition.

 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
```

Screen 11: Main Menu; Initialize Partitions.

If the Linux partitions have not already been initialized then **Next** is the correct choice at this menu.

When selected, the following dialogue box will provide the available partitions for initialization:



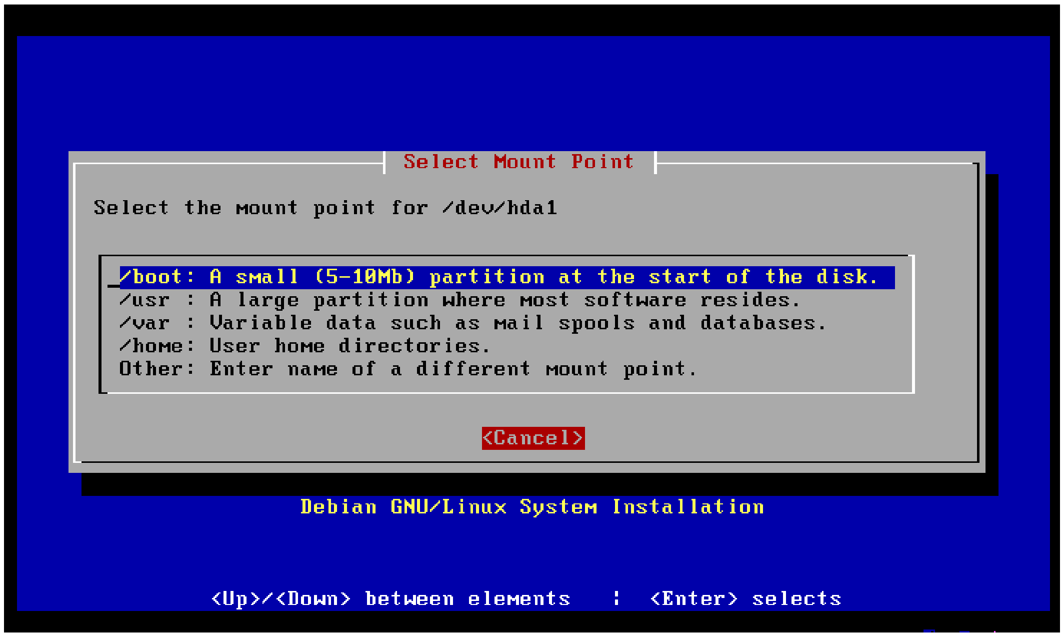
```
Select Partition

Please select the partition to initialize as a Linux "ext2"
filesystem.

    /dev/hda1: Linux native
    /dev/hda3: Linux native
    /dev/hda5: Linux native
    /dev/hda6: Linux native
    /dev/hda7: Linux native

              <Cancel>

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

Screen 12: Choose a Partition.

After choosing a partition to initialize, the next dialog box asks about 2.2 kernel ext2 features and 2.0 compatibility. Answer yes to continue to the “Are You Sure” screen. If you are sure you picked the correct partition to initialize answer yes to this question as well.

When the initialization is complete an inquiry box will ask if this partition should be mounted as the root file system (“/”). The first time through this step should be to create just such a root file system, so the answer should be yes. Additional partitions will be mounted on mount points within this root partition.

We have planned several separate partitions, so the **Initialize a Linux Partition** menu option should be chosen until all partitions have been initialized and mounted. Each time you initialize a partition, after mounting the **root** partition, the following screen will offer various mount points available for the partition just initialized.



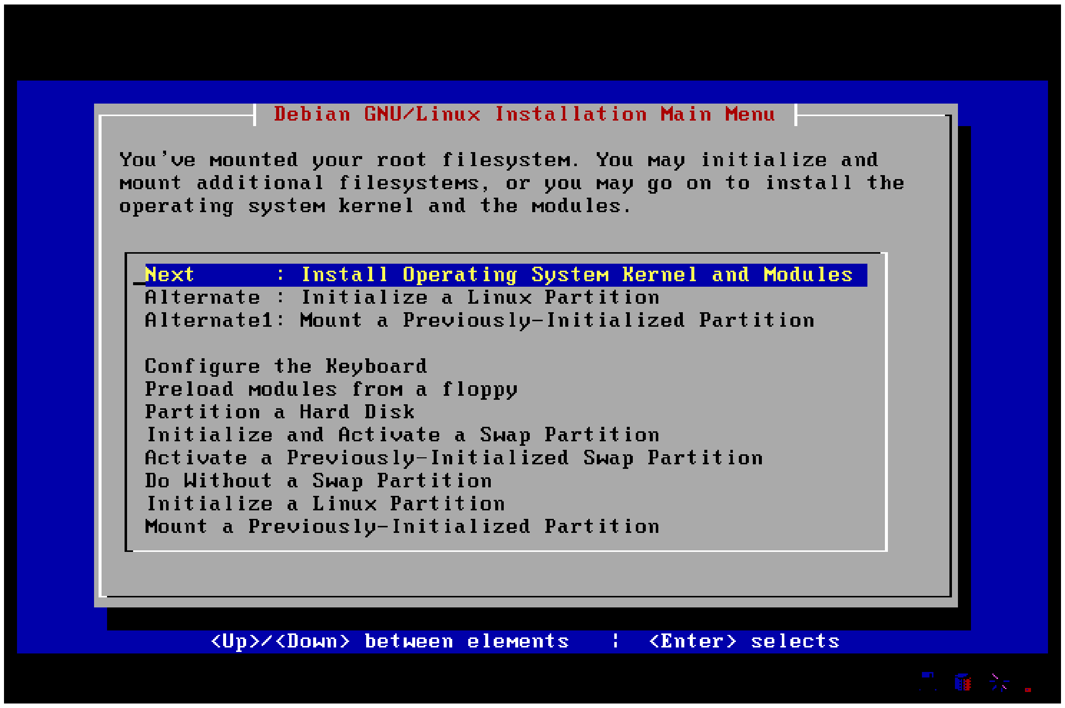
```
Select Mount Point

Select the mount point for /dev/hda1

/boot: A small (5-10Mb) partition at the start of the disk.
/usr : A large partition where most software resides.
/var : Variable data such as mail spools and databases.
/home: User home directories.
Other: Enter name of a different mount point.

<Cancel>

Debian GNU/Linux System Installation

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

Screen 13: Choose a Mount Point.

For the partition on */dev/hda1* you should choose */boot* as the mount point. The other two partitions can be mounted on */home* and *var*.

Task 5: Install Kernel and Modules

After the first initialization step is complete, the main menu will be redrawn like:



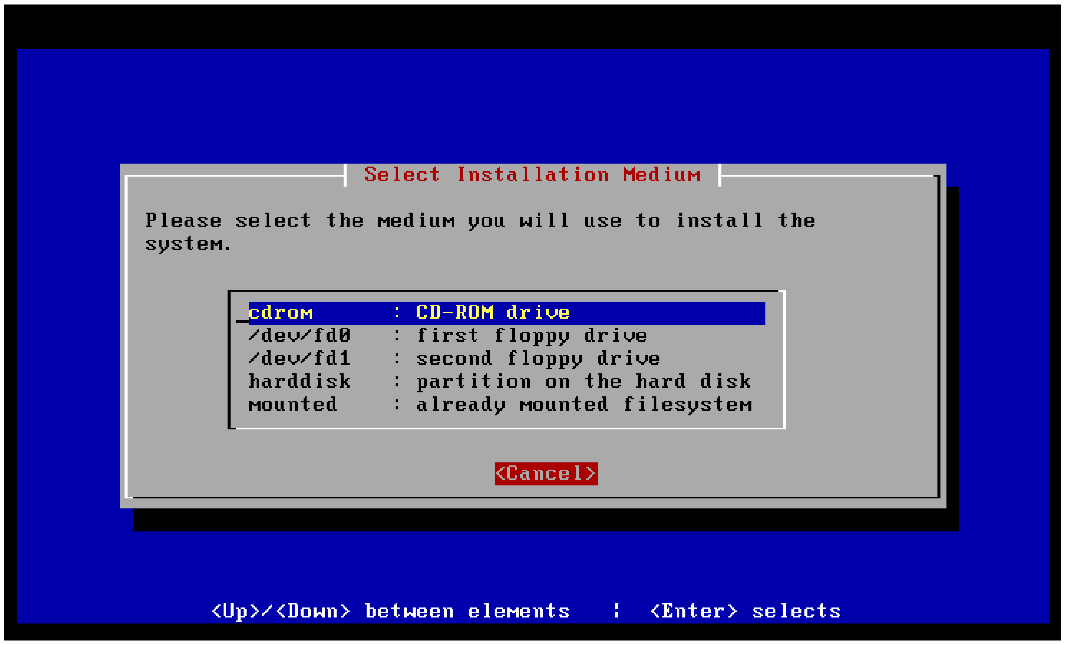
```
Debian GNU/Linux Installation Main Menu

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 Operating System Kernel and 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
```

Screen 14: Main Menu: Install Operating System Kernel and Modules

If there are still additional partitions to initialize and mount, as is the case in our example partition, choose the Alternate selection, **Initialize a Linux Partition**, as many times as is necessary to complete the partition initialization process on all desired partitions. (See the previous step.)

Once all of the desired partitions have been initialized and mounted it is time to choose Next, and install the operating system kernel and any modules that will be needed to complete the installation.

First the installation medium must be chosen from the following dialogue box:



```
Select Installation Medium

Please select the medium you will use to install the
system.

cdrom       : CD-ROM drive
/dev/fd0    : first floppy drive
/dev/fd1    : second floppy drive
harddisk    : partition on the hard disk
mounted     : already mounted filesystem

<Cancel>

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

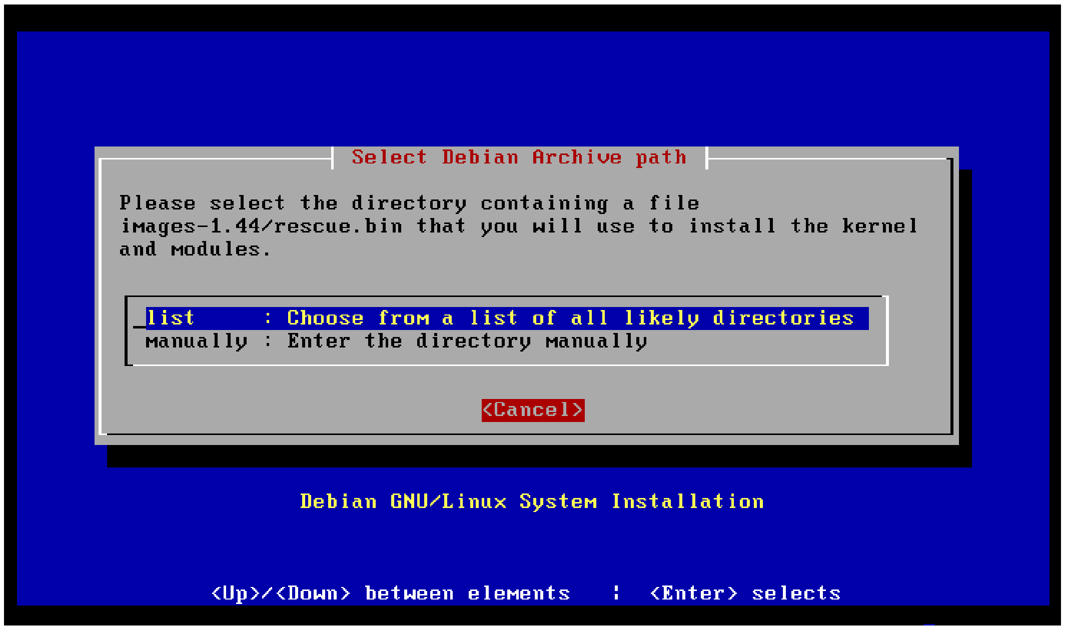
Screen 15: Media Selection

If the CD-ROM drive is either a SCSI device or an IDE/ATAPI style interface then the installation can proceed from the CD-ROM device.

If the CD-ROM device uses a proprietary interface then the drivers are not currently available to the kernel, and you will not be able to access the drive until they are loaded. You will need a rescue floppy and the three floppies required to hold the drivers. See the *install.en.txt* file on the CD-ROM for information on creating these floppies. Once you have created these disks, choose "First Floppy Drive" and follow the directions for inserting disks.

If the CD-ROM drive is a supported SCSI or IDE/ATAPI interface, then simply choose cdrom from the **Select Installation Medium** dialog box.

Upon selecting **cdrom** as the installation medium, the next screen asks that you insert the CD-ROM in the drive and press ENTER to continue and the following screen is presented.

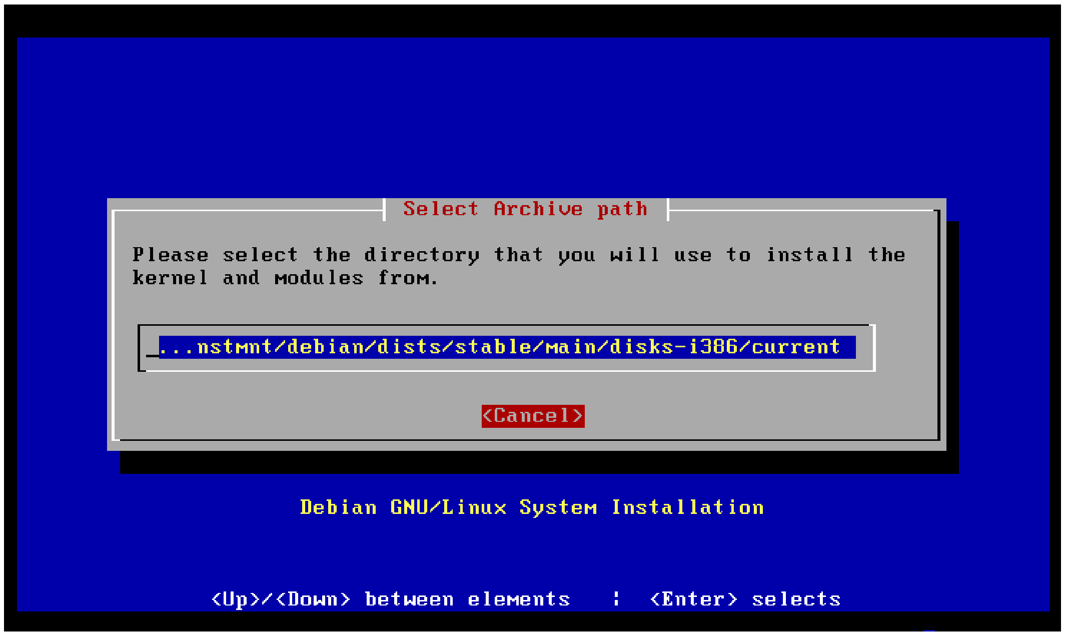


```
                    Select Debian Archive path

  Please select the directory containing a file
  images-1.44/rescue.bin that you will use to install the kernel
  and modules.

    list     : Choose from a list of all likely directories
    manually : Enter the directory manually

                             <Cancel>

              Debian GNU/Linux System Installation

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

Screen 16: Select Archive Path

For the CD install simply press ENTER and move on to the next screen. Only under extreme circumstances are you going to provide a specific path for these files. Letting the installation program find the files is the normal choice.

Pressing ENTER presents a screen that lists every path containing the desired files.



```
Please select the directory that you will use to install the
kernel and modules from.

...nstmnt/debian/dists/stable/main/disks-i386/current
```

<Cancel>
Debian GNU/Linux System Installation
<Up>/<Down> between elements : <Enter> selects

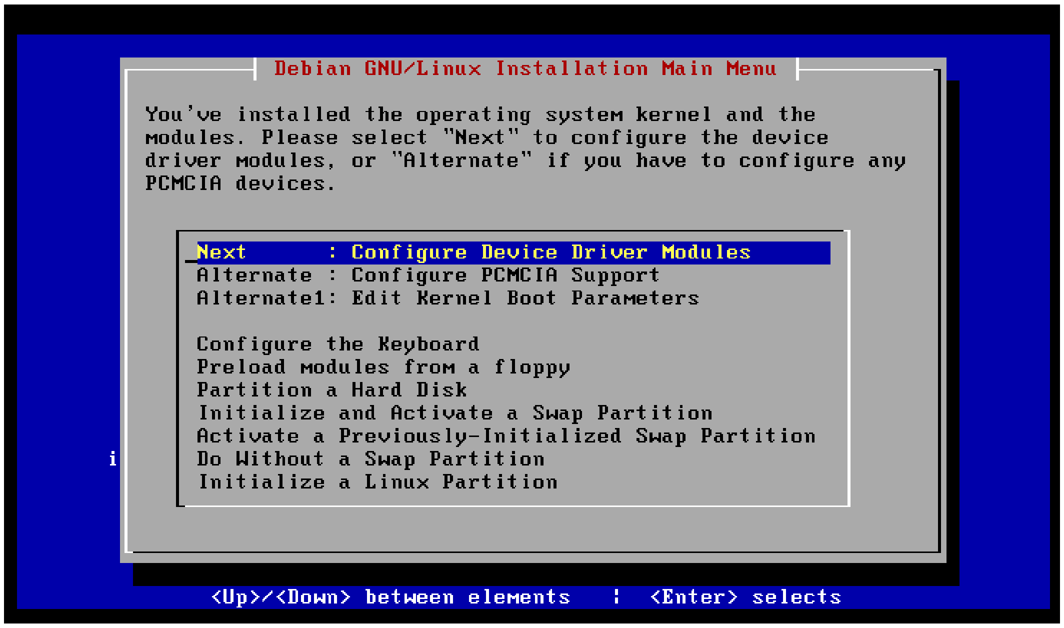
Screen 17: Available Archive Paths

The simplest response to this screen is to press ENTER, just like the previous screen. The path provided has already been checked for the desired contents, and will work just fine.

After pressing enter here, the actual process of installing will be detailed in several dialogue boxes.

Task 6: Configuring Driver Modules

Upon successful completion of the kernel and modules installation, the following main menu is presented:



```
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, or "Alternate" if you have to configure any
PCMCIA devices.

Next       : Configure Device Driver Modules
Alternate  : Configure PCMCIA Support
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

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

Screen 18: Main Menu: Configure Device Driver Modules

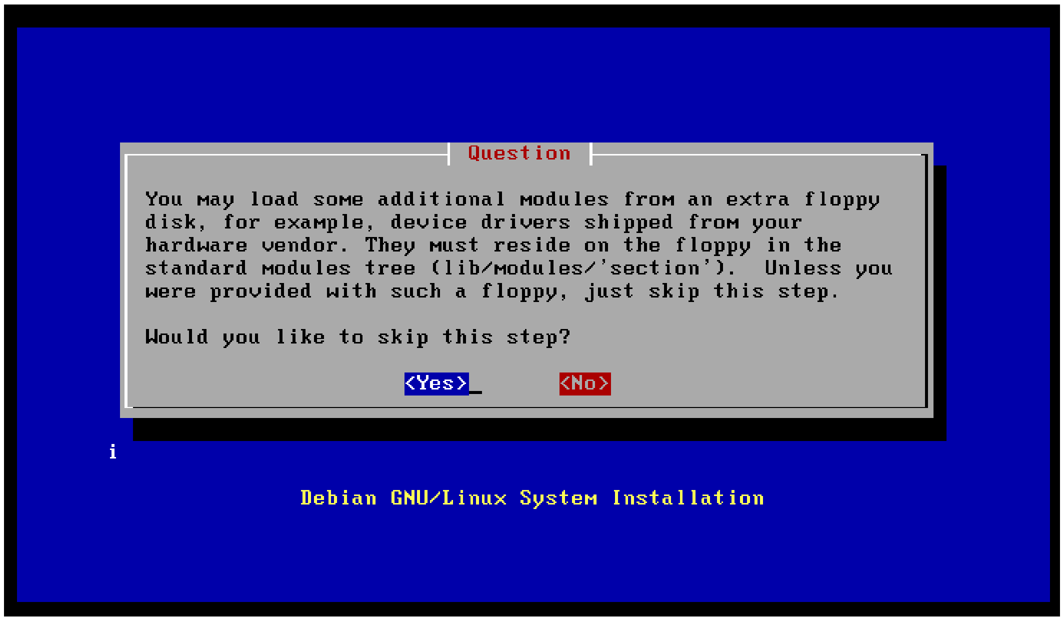
Configuring device drivers can be a crucial step in the installation. If the hardware includes either a modem useful for connecting to an Internet Service Provider, or a CD-ROM drive supported by the current kernel but not already installed in the kernel, here is where those drivers are installed. If the connection is through an Ethernet card, drivers for that card are installed during this phase as well. Many other specialty devices have their drivers provided here, so look over the list carefully to find your particular needs. Serial and parallel drivers are found in the **mis** section.

Even if you do configure all of the necessary modules at this time, it may be desirable to edit */etc/modules* and activate the “auto” line by removing the leading '#' character. This will allow modules to be loaded as needed and earlier in the boot up process than when *kerneld* is normally running. Modules explicitly declared after the **auto** will be loaded at boot-up and never unloaded by *kerneld* until shutdown. This is useful in maintaining start-up configuration on some drivers, most notably a **serial** driver with nonstandard interrupts.

In addition to the option to configure device drivers, there is an alternate selection that allows the PCMCIA support to be configured as well. This is of primary interest to laptop machines with removable devices like modems, Ethernet cards, and even CD-ROM drives and other SCSI devices.

Choosing the PCMCIA configuration option will present the options to configure: a PCMCIA controller, Serial, and CD-ROM devices. Configuration now will make the rest of the installation possible via these devices.

When the modules configuration option is chosen, the following screen will be presented.

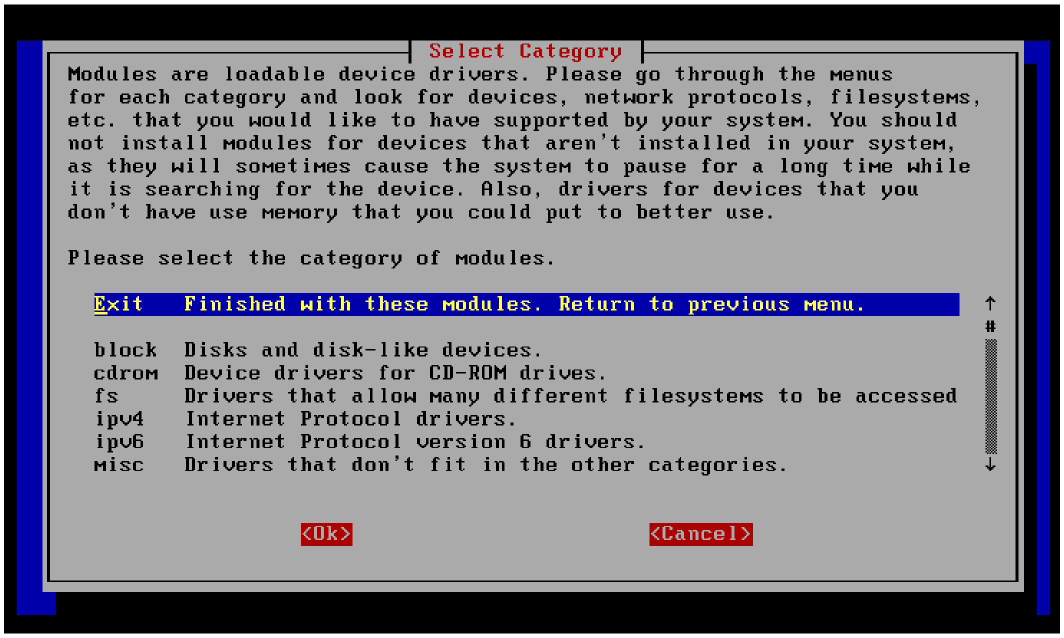


```
                          Question

You may load some additional modules from an extra floppy
disk, for example, device drivers shipped from your
hardware vendor. They must reside on the floppy in the
standard modules tree (lib/modules/'section'). Unless you
were provided with such a floppy, just skip this step.

Would you like to skip this step?

                     <Yes>        <No>
```

Screen 19: Additional Modules

If you have any modules not provided by the distribution, this is your chance add them to the *modules* directory on your new system. These modules must be provided on a floppy disk. Normally you will wish to choose **No** on this screen.

After this screen has been exited, the program *modconf* will display the following screen:



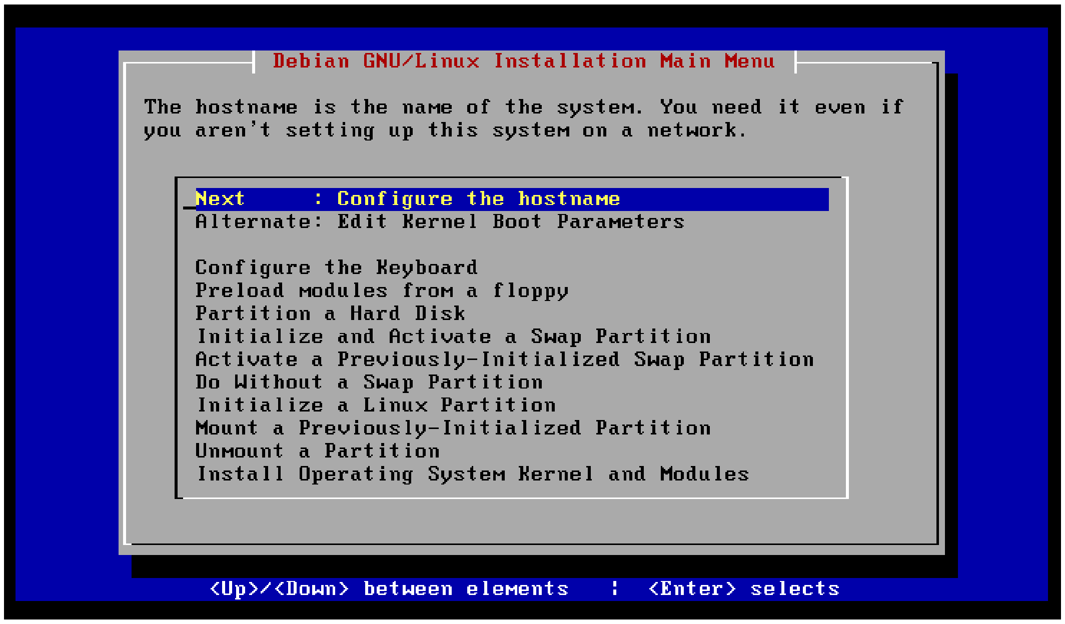
```
Select Category
Modules are loadable device drivers. Please go through the menus
for each category and look for devices, network protocols, filesystems,
etc. that you would like to have supported by your system. You should
not install modules for devices that aren't installed in your system,
as they 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 with these modules. Return to previous menu.

  block   Disks and disk-like devices.
  cdrom   Device drivers for CD-ROM drives.
  fs      Drivers that allow many different filesystems to be accessed
  ipv4    Internet Protocol drivers.
  ipv6    Internet Protocol version 6 drivers.
  misc    Drivers that don't fit in the other categories.

             <Ok>                       <Cancel>
```

Screen 20: modconf Main Menu

Work through the menu selecting and installing those modules that will be useful to the new system, as well as those that may aid in the installation process. Once all necessary modules have been added to the system, choose **Exit** and return to the main menu. If you have had no trouble getting to the CD, i.e. you don't need a driver for your CD-ROM drive, you can postpone the rest of your module configuration until after the installation process.

Task 7: Configure the Network

If the target machine has no Ethernet card, then configuring the network is not going to provide any real functionality. Under those conditions the Main Menu will look like:

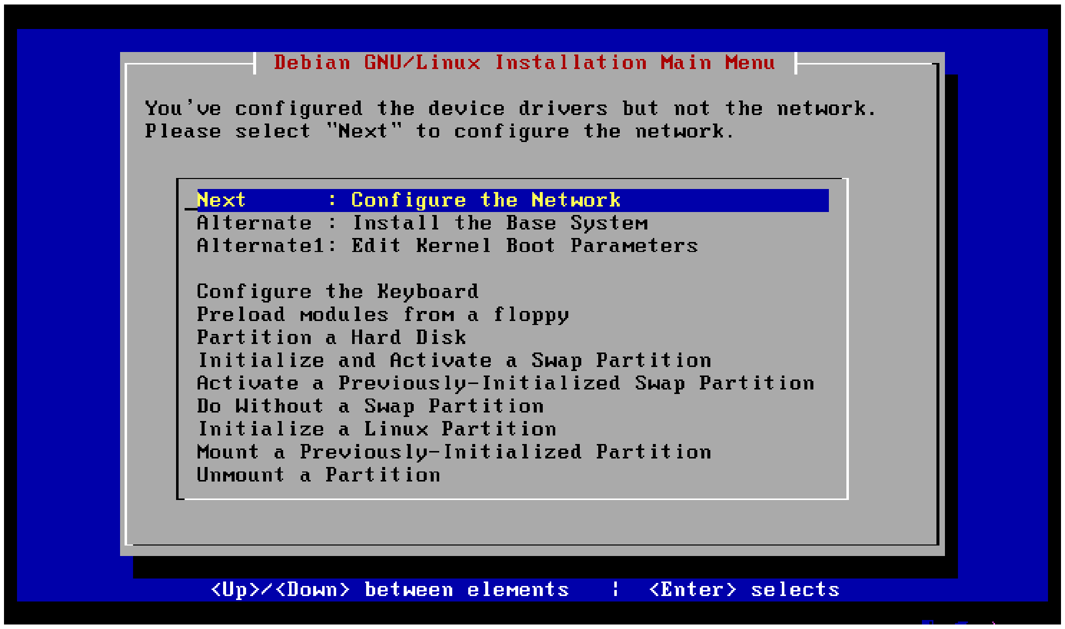


```
Debian GNU/Linux Installation Main Menu

The hostname is the name of the system. You need it even if
you aren't setting up this system on a network.

Next      : Configure the hostname
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
Install Operating System Kernel and Modules

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

Screen 21: Main Menu: Configure the hostname

and the only network configuration action allowed will be creating the new system's **hostname**.

If there is an Ethernet adapter in the machine, and its driver has been loaded in previous steps, the Main Menu will look like:



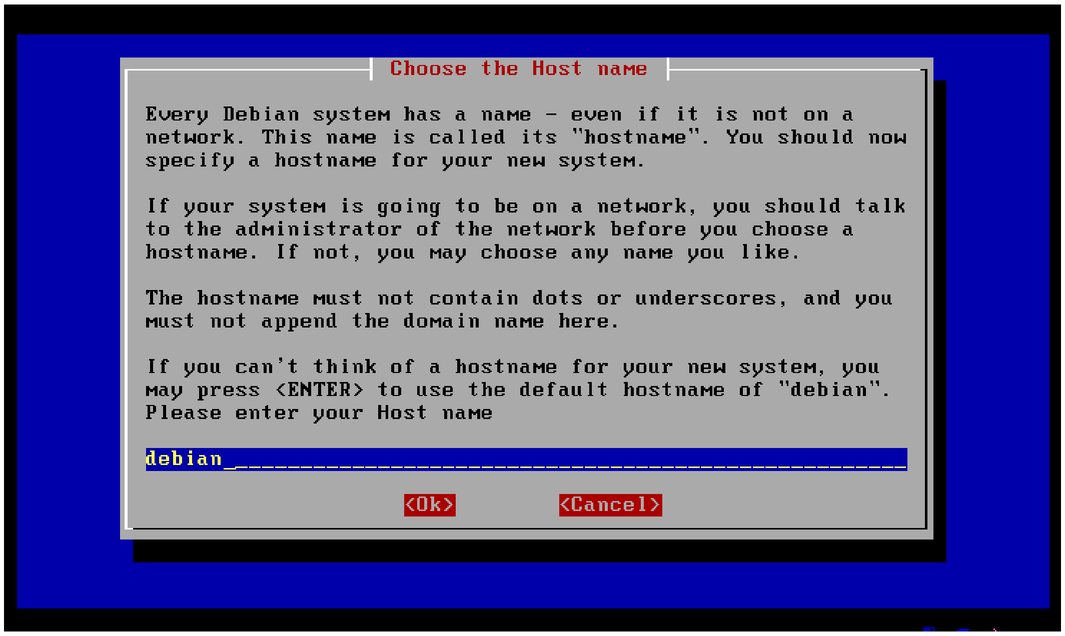
```
Debian GNU/Linux Installation Main Menu

You've configured the device drivers but not the network.
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
```

Screen 22: Main Menu: Configure the Network

The option to **Configure the Network** is presented at this time to provide the possibility of installing the Base System over an Ethernet connection. Even if the base system is to be installed from floppy disk, it is probably a good idea to go through this section anyway, as it creates files that will be of use later if networking is desired.

There will be an option to set up a PPP connection in the **Second Stage** of the installation.

Pressing ENTER at **Configure the Network** will present the following:

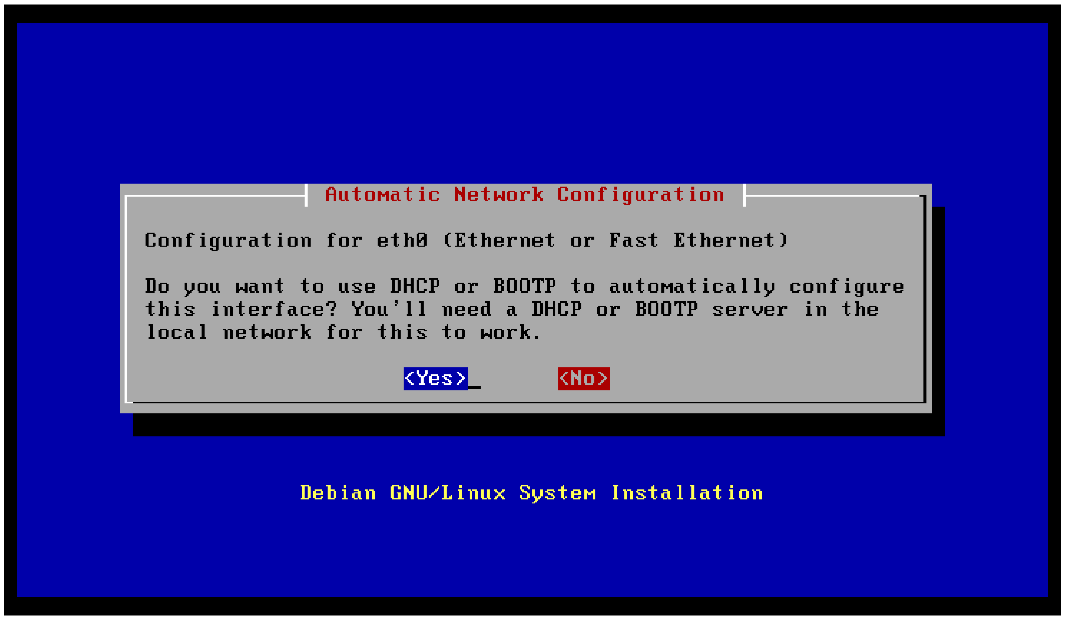


```
Choose the Host name

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 Host name

debian_

<Ok>          <Cancel>
```

Screen 23: Choose a hostname

If you do decide to go ahead with network configuration, here is where you enter the **hostname** for this machine. If you have no special name to give this machine, then press enter and the default, **debian** will be used. For this example installation, the hostname **alice** has been chosen.

Selecting a **hostname** will present the following dialogue box:


Automatic Network Configuration

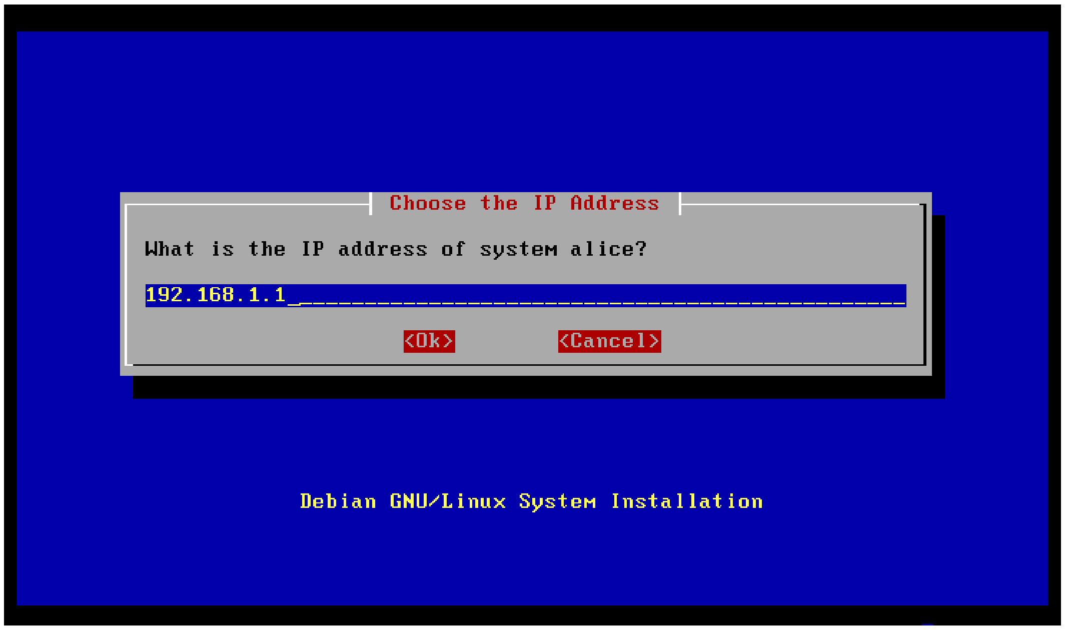
```
Configuration for eth0 (Ethernet or Fast Ethernet)

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.
```

<Yes>
<No>
Debian GNU/Linux System Installation

Screen 24: Automatic Network Configuration

You only want to say **yes** to this option if these DHCP or BOOTP servers are available somewhere on the LAN. Otherwise tab over to the **No** button and press ENTER.

The next screen will ask for an IP address for the target machine.

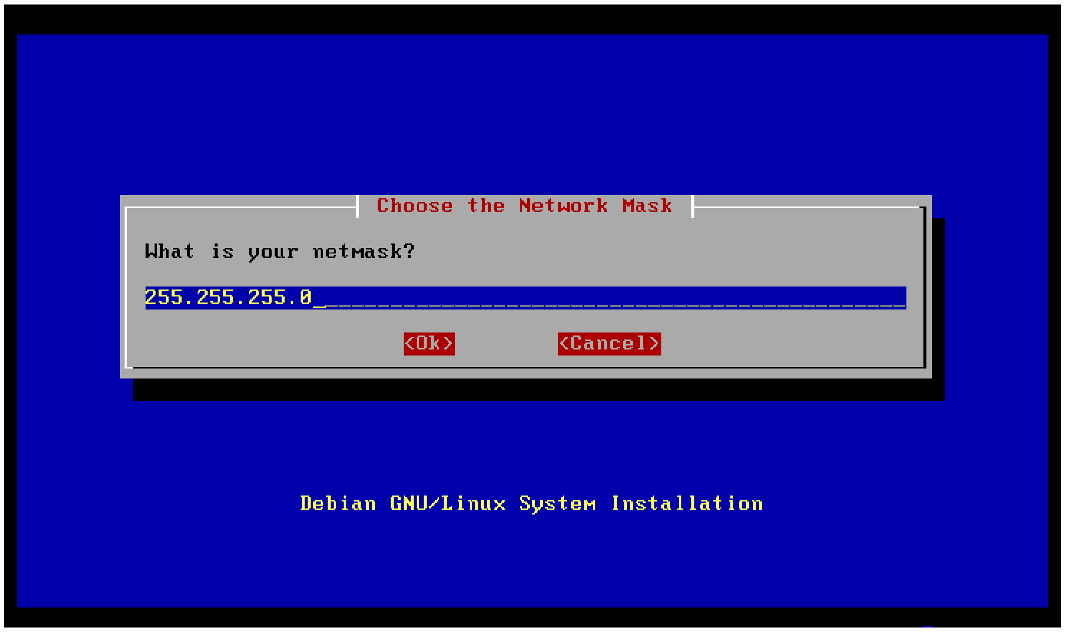


```
Choose the IP Address

What is the IP address of system alice?

192.168.1.1_

<Ok>          <Cancel>

Debian GNU/Linux System Installation
```

Screen 25: Choose an IP Address

You may need to check with your system administrator to obtain the correct address to insert in place of the example default provided on the screen. This is the address that the target machines Ethernet card is expected to respond to.

Next you are presented with a choice of network mask.

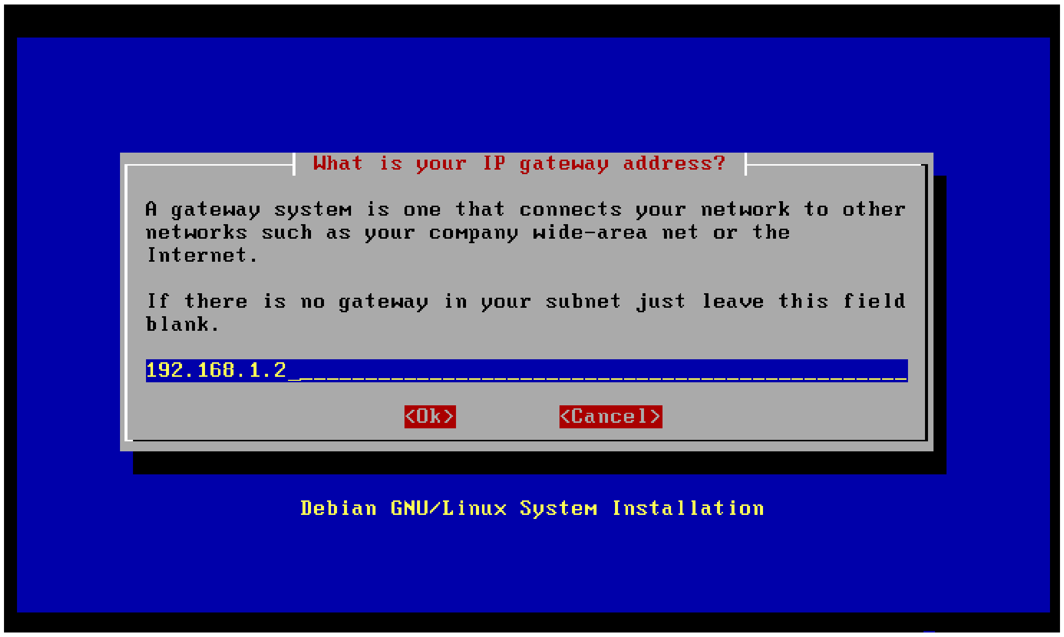


```
Choose the Network Mask

What is your netmask?

255.255.255.0_

<Ok>        <Cancel>

Debian GNU/Linux System Installation
```

Screen 26: Choose a Network Mask

The default value provided is most often correct, so all that is needed here is the press of the return key. If you think this is not correct, check with your system administrator for confirmation.

Next you will be asked for the gateway address for your LAN.

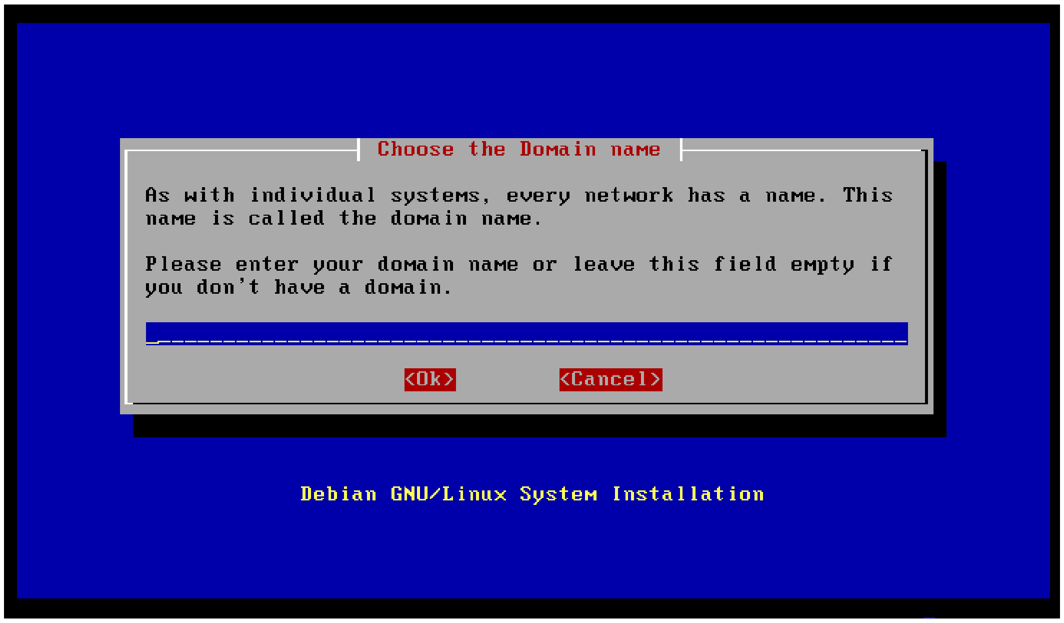


```
What is your IP gateway address?

A gateway system is one that connects your network to other
networks such as your company wide-area net or the
Internet.

If there is no gateway in your subnet just leave this field
blank.

192.168.1.2_

<Ok>          <Cancel>

Debian GNU/Linux System Installation
```

Screen 27: Choose a Gateway Address

If there is no gateway simply clear the line and press ENTER. As before, if you have any questions, ask your system administrator for help.

Now you will be offered the opportunity to provide the **domain name** for your LAN.

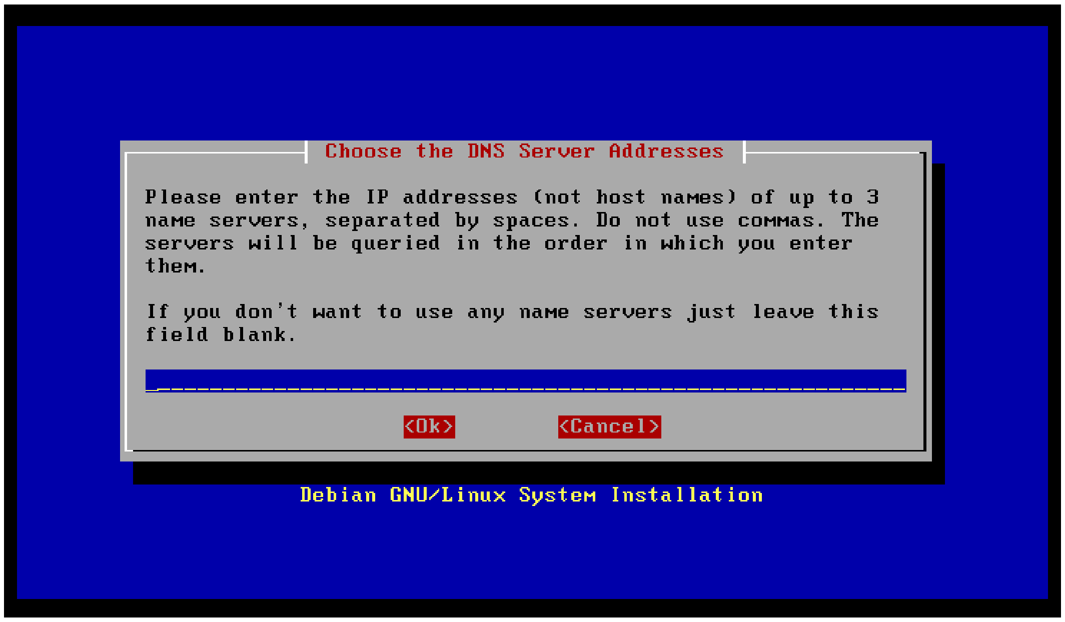


```
Choose the Domain name

As with individual systems, every network has a name. This
name is called the domain name.

Please enter your domain name or leave this field empty if
you don't have a domain.

_________________________________________________________

<Ok>            <Cancel>

Debian GNU/Linux System Installation
```

Screen 28: Choose a Domain Name

This name, appended to your **hostname**, forms the complete name designation for the target machine. Unlike the **hostname**, which can not contain the "." (dot) character, the **domain name** is often broken into several parts using the dot character.

The following screen provides for the entry of up to 3 **Designated Name Servers**.


```
Choose the DNS Server Addresses

Please enter the IP addresses (not host names) of up to 3
name servers, separated by spaces. Do not use commas. The
servers will be queried in the order in which you enter
them.

If you don't want to use any name servers just leave this
field blank.
```

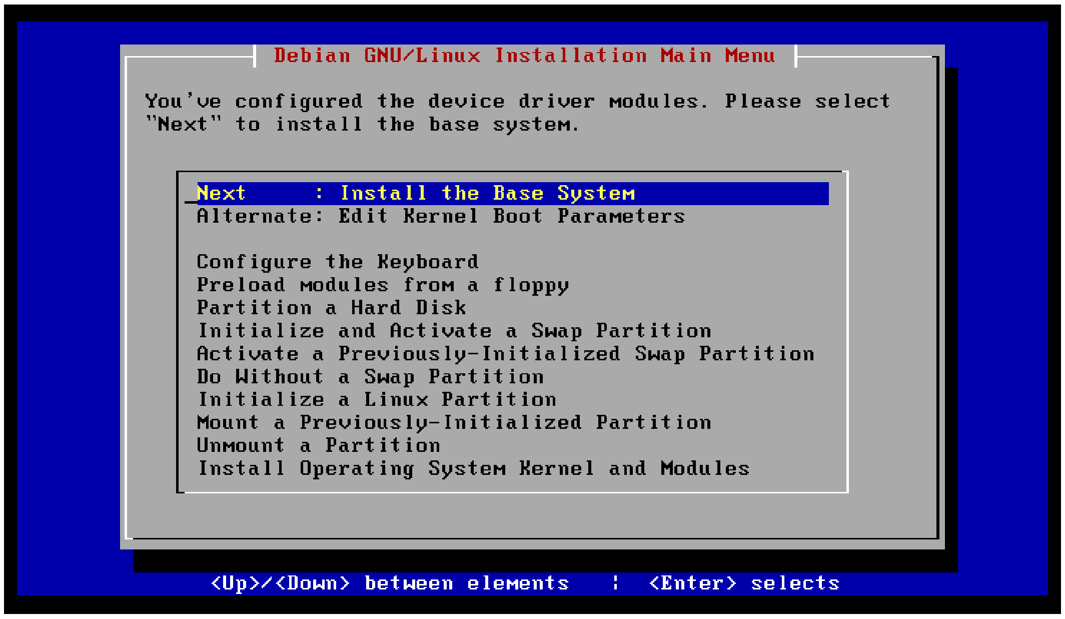
Screen 29: Choose a Designated Name Server

Enter the IP address of the name servers you wish to use, separated by blanks, onto the line provided in the above dialog box.

This completes the network configuration step, although you can always re-select this option from the main menu if you made a mistake and want to change something.

Task 8: Installing the Base System

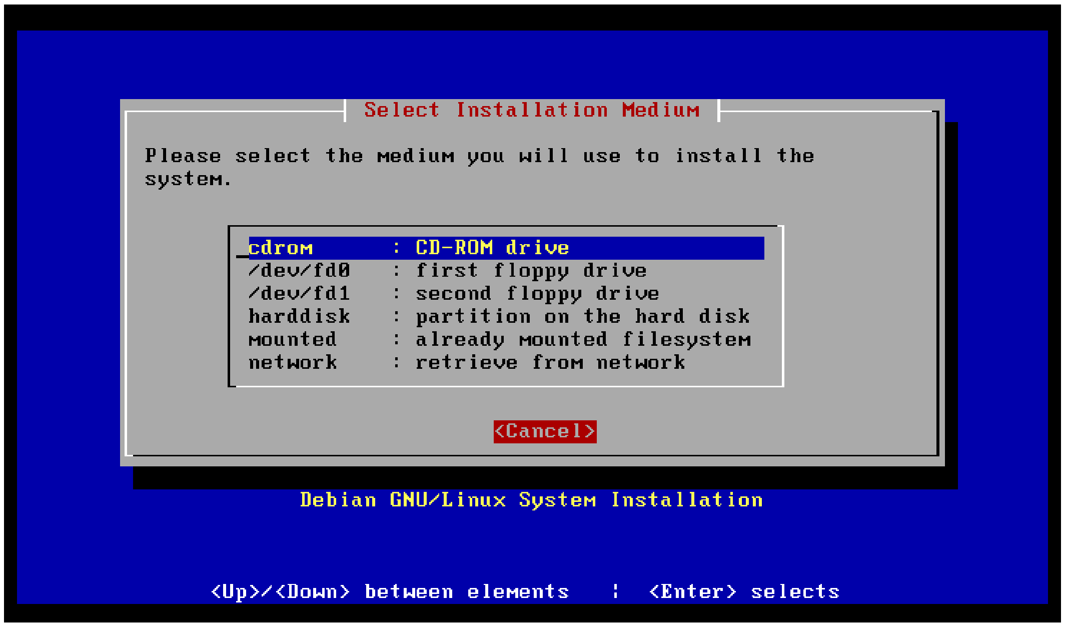
Upon returning from the network configuration the Main Menu looks like:



```
Debian GNU/Linux Installation Main Menu

You've configured the device driver modules. Please select
"Next" to install the base system.

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
Install Operating System Kernel and Modules

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

Screen 30: Main Menu: Install the Base System

Choosing "Next" will present the previously seen selection screen:



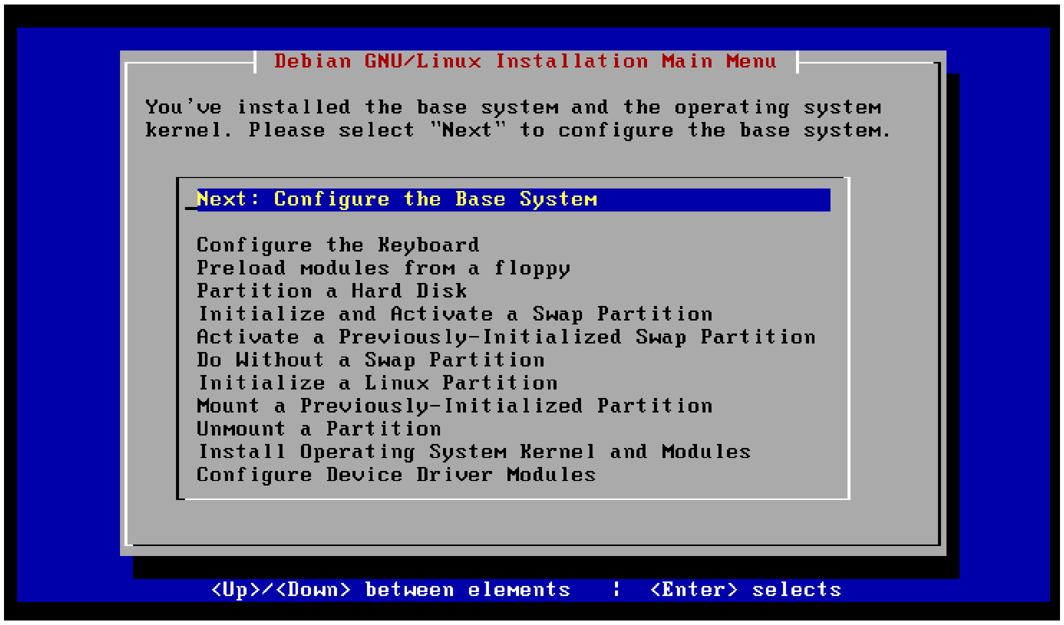
```
Select Installation Medium

Please select the medium you will use to install the
system.

cdrom       : CD-ROM drive
/dev/fd0    : first floppy drive
/dev/fd1    : second floppy drive
harddisk    : partition on the hard disk
mounted     : already mounted filesystem
network     : retrieve from network

<Cancel>

Debian GNU/Linux System Installation

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

Screen 31: Choose Installation Media

Note: When you choose **cdrom** the same two Archive path screens are presented. As with the kernel install, press ENTER at each of these screen.

Task 9: Configuring the Base System

After installing the **Base System**, the Main Menu will look like:

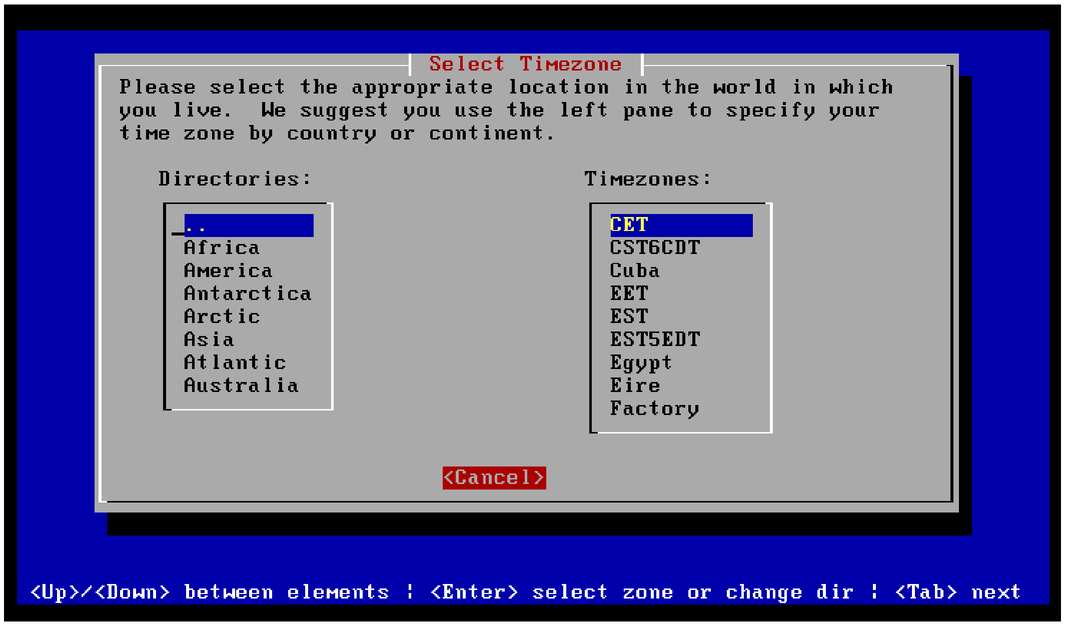


```
Debian GNU/Linux Installation Main Menu

You've installed the base system and the operating system
kernel. Please select "Next" to configure the base system.

  Next: Configure the Base 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
  Mount a Previously-Initialized Partition
  Unmount a Partition
  Install Operating System Kernel and Modules
  Configure Device Driver Modules

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

Screen 32: Main Menu: Configure the Base System

Time zone configuration is the sole task in this step.

Choosing the **Next** option of **Configure the Base System** will display:



```
Select Timezone
Please select the appropriate location in the world in which
you live.  We suggest you use the left pane to specify your
time zone by country or continent.

   Directories:                      Timezones:

   ..                                CET
   Africa                            CST6CDT
   America                           Cuba
   Antarctica                        EET
   Arctic                            EST
   Asia                              EST5EDT
   Atlantic                          Egypt
   Australia                         Eire
                                     Factory

                    <Cancel>

<Up>/<Down> between elements | <Enter> select zone or change dir | <Tab> next
```

Screen 33: Select Time zone

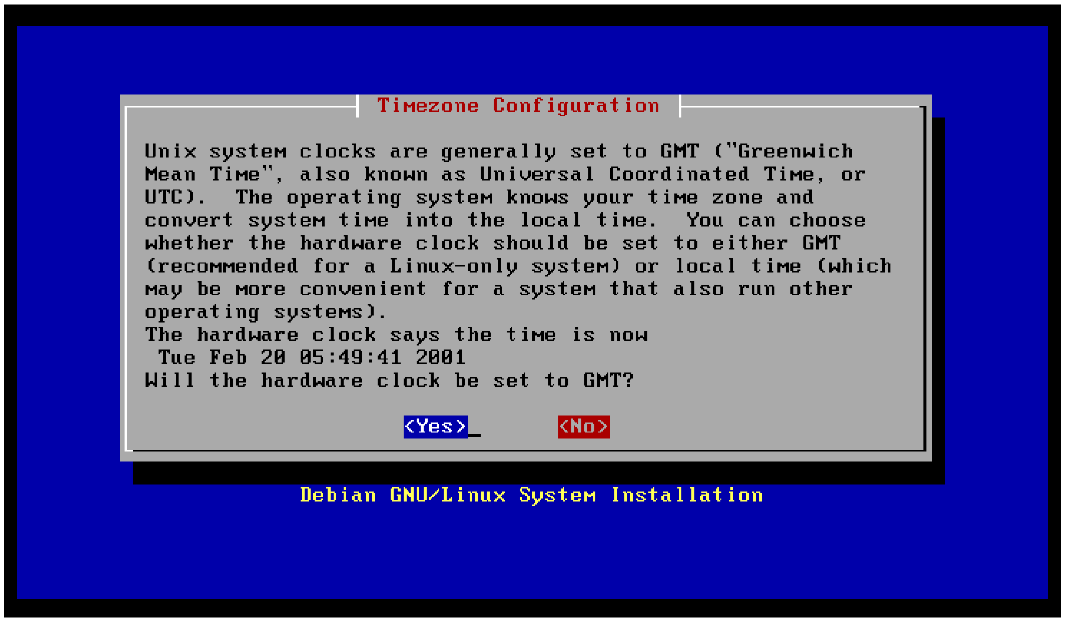
The object is to choose the time zone that represents your version of local time. You may either choose a time zone from the list on the right, or choose a geographical area from the list on the left. This can be very misleading at times.

For example: If you are in the Eastern time zone in the United States, you would be tempted to choose America which leads to a list of cities, instead of the US section which leads to the Eastern time zone.

On the other hand you can choose a reasonable time zone from the list on the right. If you want daylight savings time to take effect at the appropriate time choose EST5EDT, otherwise choose EST to configure Eastern Standard.

Whatever you choose at this time can be changed later by running *tzconfig*.

After selecting an appropriate time zone the following question will be asked:



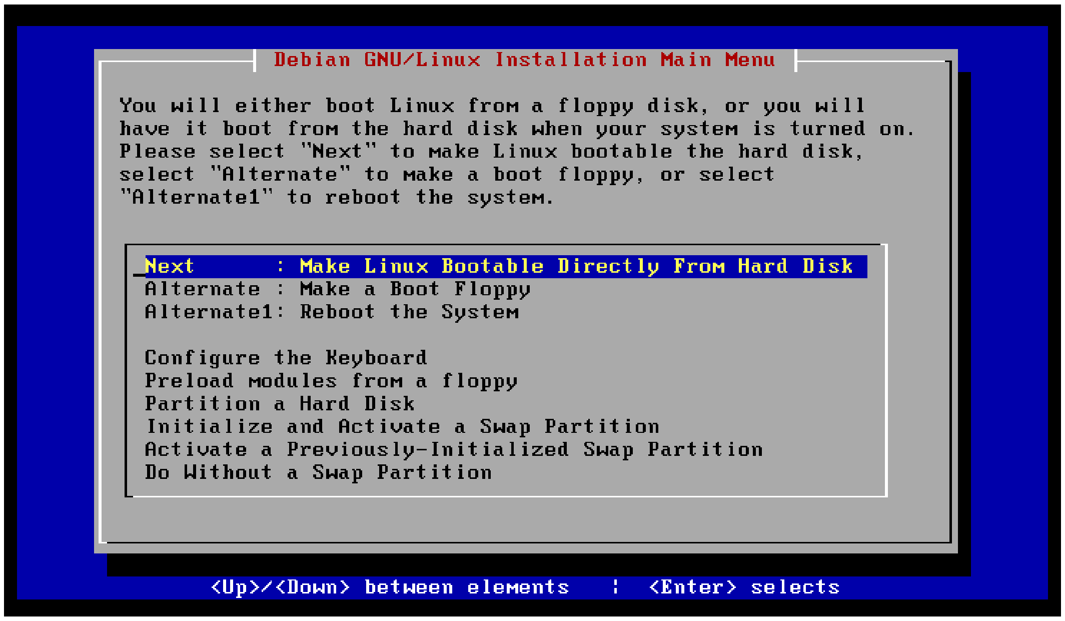
```
Unix system clocks are generally set to GMT ("Greenwich
Mean Time", also known as Universal Coordinated Time, or
UTC).  The operating system knows your time zone and
convert system time into the local time.  You can choose
whether the hardware clock should be set to either GMT
(recommended for a Linux-only system) or local time (which
may be more convenient for a system that also run other
operating systems).
The hardware clock says the time is now
 Tue Feb 20 05:49:41 2001
Will the hardware clock be set to GMT?
```

Screen 34: Select Time zone

This question asks "Is your system clock set to GMT?". Unix system clocks are generally set to Greenwich Mean Time, also known as Universal Time, and use time zone files to display the local time. If this machine is to be set up in this fashion then the answer is yes. Most MS-DOS machines set their clocks to "local time" for convenience. If you wish to maintain this compatibility with a DOS or Windows system then the proper answer is "no".

Task 10: Creating a Bootable System

After configuring the time zone, the Main Menu will look like:

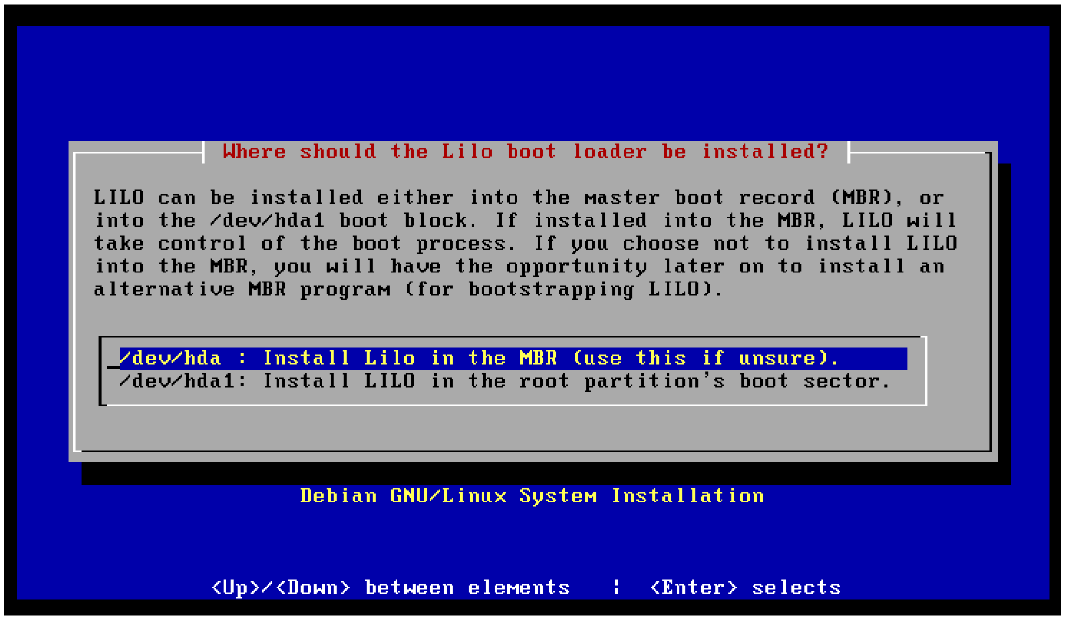


```
Debian GNU/Linux Installation Main Menu

You will either boot Linux from a floppy disk, or you will
have it boot from the hard disk when your system is turned on.
Please select "Next" to make Linux bootable the hard disk,
select "Alternate" to make a boot floppy, or select
"Alternate1" to reboot the system.

  Next       : Make Linux Bootable Directly From Hard Disk
  Alternate  : Make a Boot Floppy
  Alternate1: 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

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

Screen 35: Main Menu: Make Linux Bootable Directly From Hard Disk

Even when you choose to make Linux bootable from the hard disk, you should still probably create a boot floppy afterwards to insure a bootable system. LILO is a reliable boot loader when it is configured properly. Depending on various hardware issues, the initial configuration may not work correctly. For this reason a boot floppy is a valuable tool early in the new system's life.

Choosing **Next** will present the following screen:

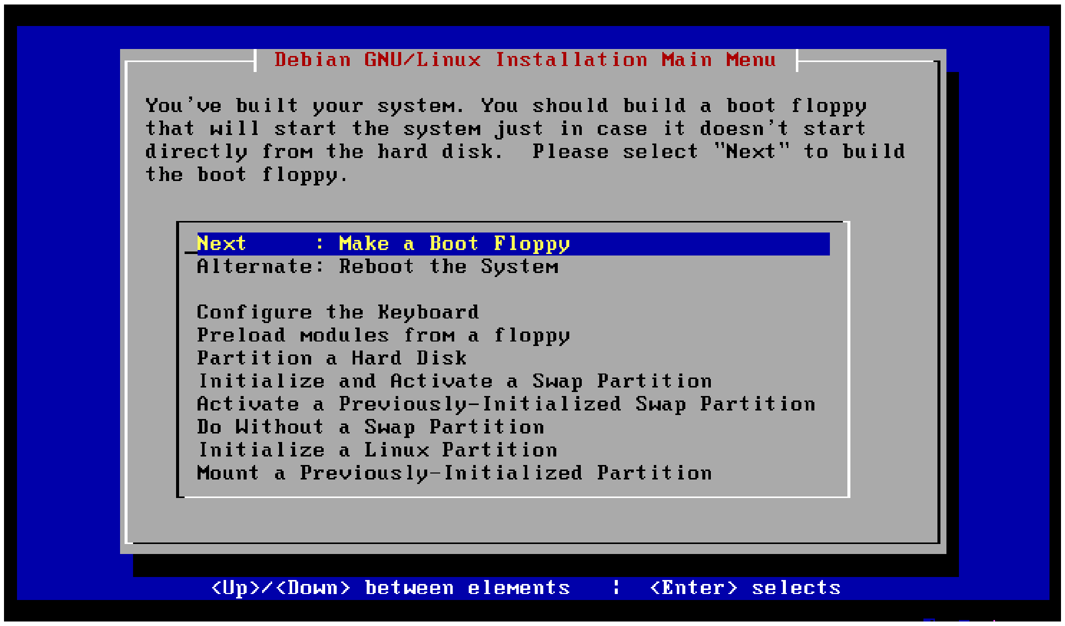


```
Where should the Lilo boot loader be installed?

LILO can be installed either into the master boot record (MBR), or
into the /dev/hda1 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/hda1: Install LILO in the root partition's boot sector.

Debian GNU/Linux System Installation

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

Screen 36: Where do we put LILO?

As this screen suggests, the first option is the most likely. The only reason you might not want LILO on the **Master Boot Record** is when the system is set up with another boot manager to be able to boot an operating system other than Linux.

After installing LILO the Main Menu will look like:

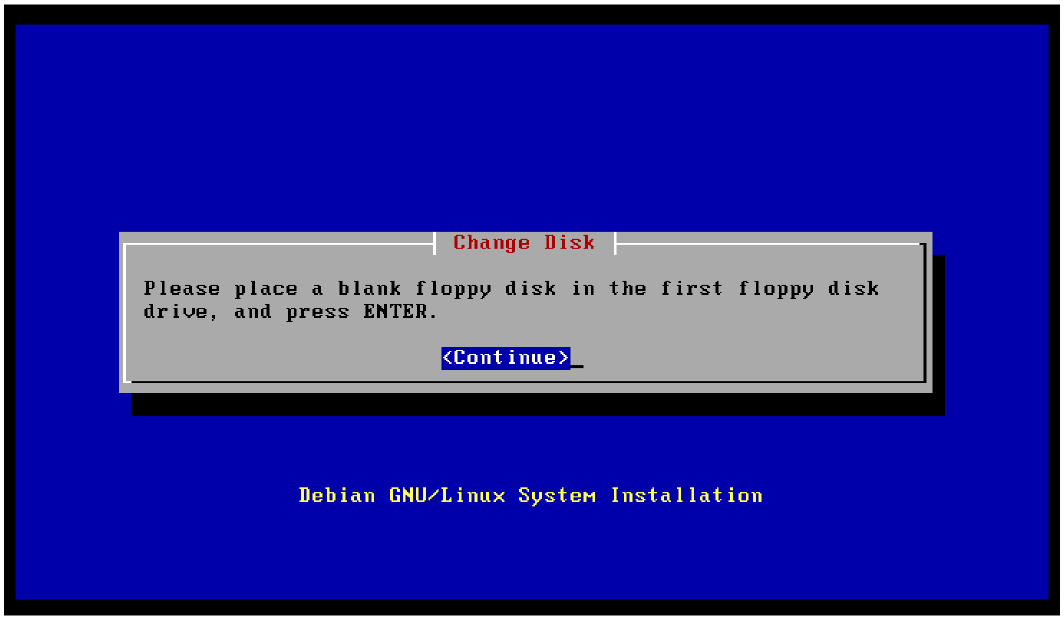


```
Debian GNU/Linux Installation Main Menu

You've built your system. You should build a boot floppy
that will start the system just in case it doesn't start
directly from the hard disk.  Please select "Next" to build
the 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
Mount a Previously-Initialized Partition

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

Screen 37: Main Menu: Make a Boot Floppy.

As suggested earlier it is a good idea to create a boot floppy as a backup boot capability. You may never need it, but it never hurts to have insurance.

Pressing ENTER presents the next screen:



```
Change Disk

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

<Continue>

Debian GNU/Linux System Installation
```

Screen 38: Change Disk

You will need to insert a blank floppy into the drive. When you press ENTER the drive will be formatted and a kernel installed and made bootable. On successful completion this will have produced a floppy that can be used to reboot the system.

Task 11: Finish Stage One

After building a boot floppy the only thing left to do for a base install is **Reboot The System**. The new system can be booted from the hard disk (assuming that LILO installed correctly) or using the new boot floppy. The DOS partition, if there is one, can be rebooted now and a loadlin installation can be set up if that is the desired boot path. No matter which method will be used, the final step in the base installation is “Reboot The System”!

Stage One Installation is now complete. **Stage Two Installation** begins on the next reboot of your newly installed **base system**. Be sure to remove the CD that you used to install the **base system**, as you do not wish to reboot from the CD at this time.

Dwarf wishes to point out that you can, indeed, use the CD to boot your new system, assuming that your hardware will boot the CD. When you get the **boot:** prompt, enter *linux /dev/<root partition>* where *<root partition>* is the device where the root file system resides. So, if you set **root** as **hda3** then the command would be:

```
linux /dev/hda3
```

This is a useful method for rebooting your system when the kernel is broken. You can also do a normal boot and use the installation system to recover from a more broken system, so this is a good recovery method. However, at this stage we are hoping our newly installed LILO boot process is going to work, so remove the CD before rebooting for **Stage Two**.

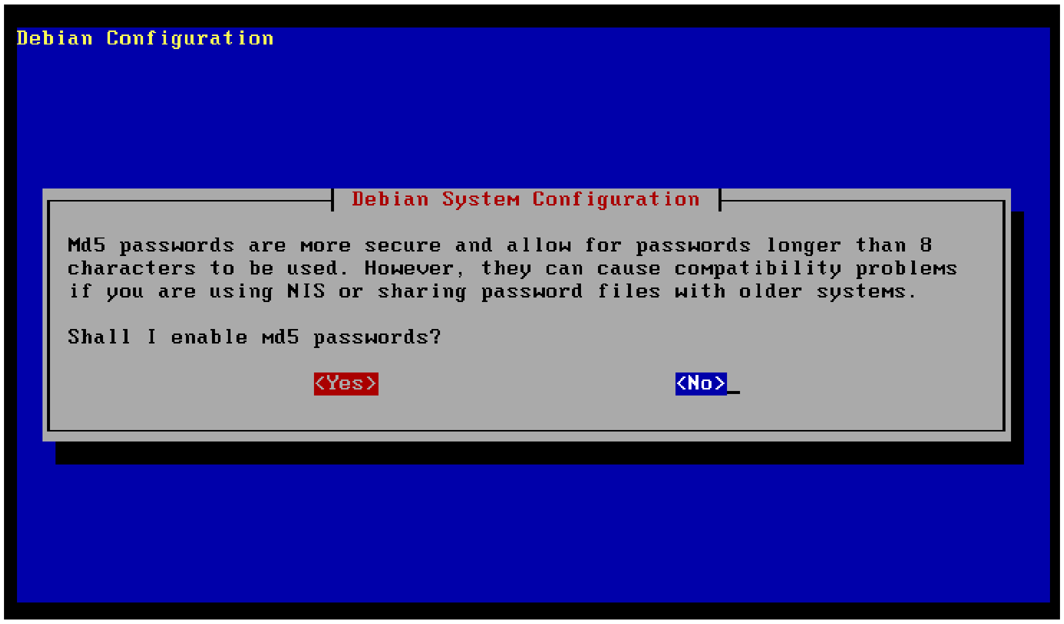
Stage Two: Package Installation

The base system installation is now complete and the rest of the system can be installed from various sources using a variety of methods. Once the system has been rebooted for the first time, the **Second Stage Installation** will begin. If, for some reason, this phase is interrupted, it can be restarted with the command:

```
perl -w /usr/sbin/dpkg-reconfigure base-config
```

General Configuration

Once this process begins the following screen is presented:

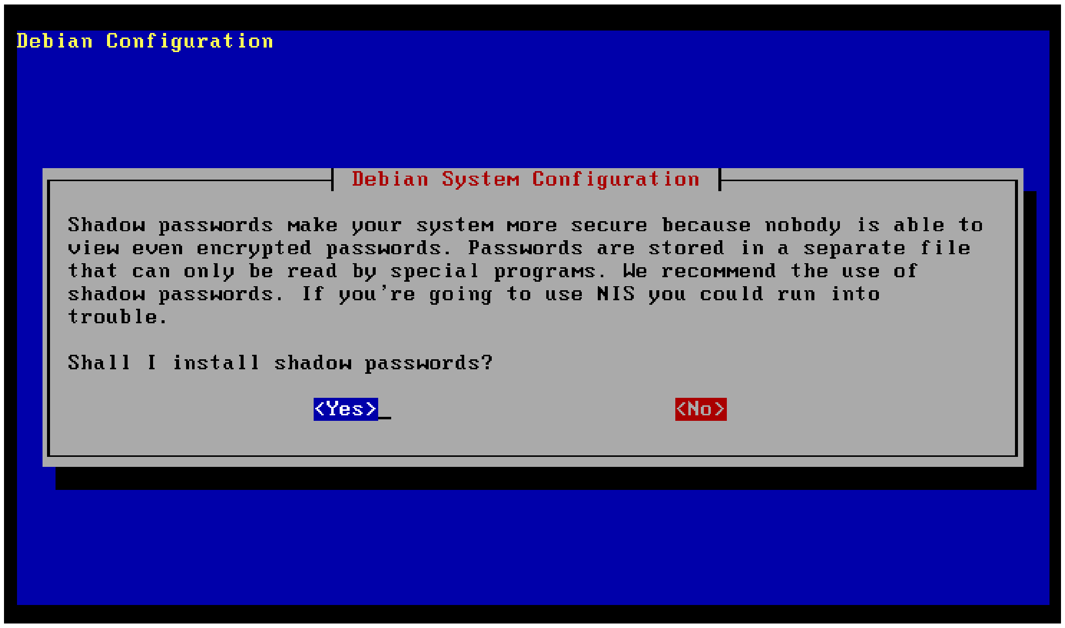


```
Debian Configuration

Debian System Configuration

Md5 passwords are more secure and allow for passwords longer than 8
characters to be used. However, they can cause compatibility problems
if you are using NIS or sharing password files with older systems.

Shall I enable md5 passwords?
```

Screen 39: MD5 Passwords?

The default reply is set to **No**, so only if you really know what you want and are willing to deal with the NIS problems presented by the use of MD5 Passwords, should you change this to **Yes**.

Whatever you choose will present the next screen:

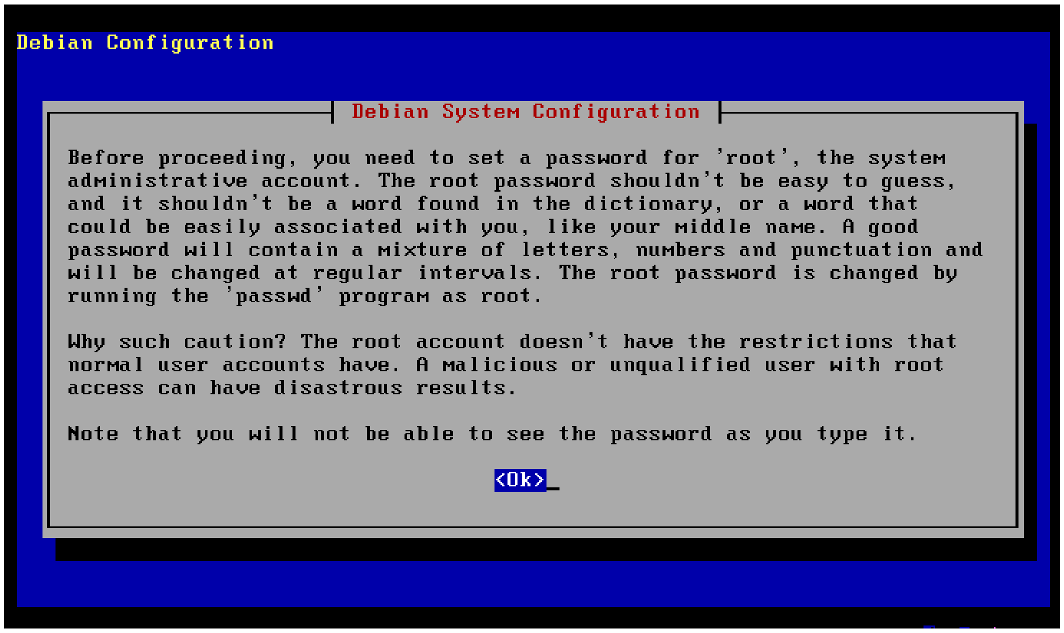


```
Shadow passwords make your system more secure because nobody is able to
view even encrypted passwords. Passwords are stored in a separate file
that can only be read by special programs. We recommend the use of
shadow passwords. If you're going to use NIS you could run into
trouble.

Shall I install shadow passwords?
```

Screen 40: Shadow Passwords?

As this screen implies, even with the potential NIS problems shadow passwords are a good idea. Because the actual encrypted password is kept separate from the password file, shadow passwords are considered more secure.

If this is the first time running base-config, then the following screen will appear next:



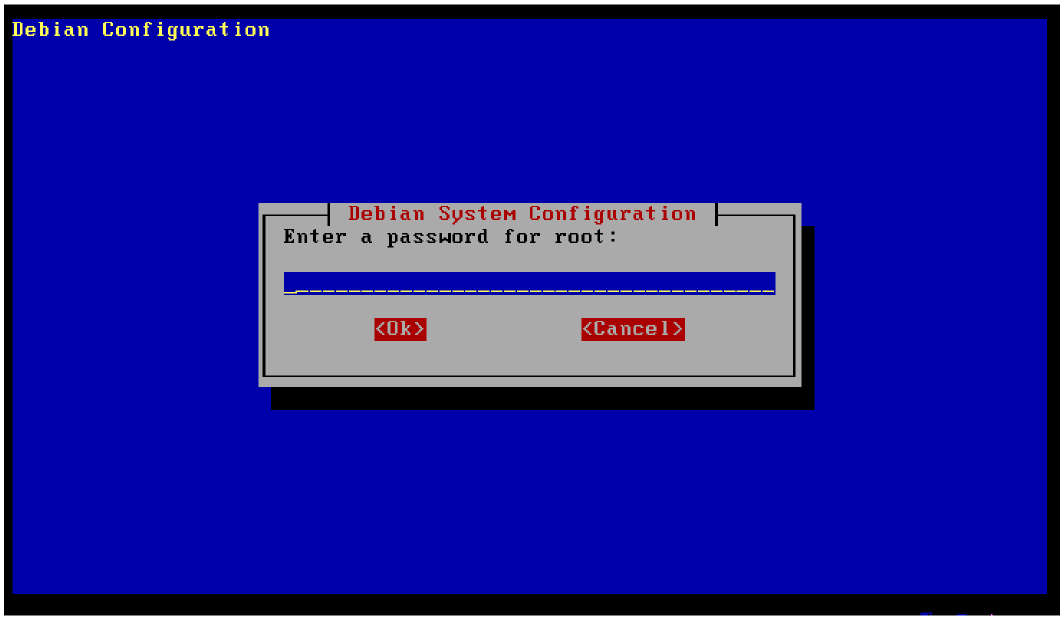
```
Debian Configuration

Debian System Configuration

Before proceeding, you need to set a password for 'root', the system
administrative account. The root password shouldn't be easy to guess,
and it shouldn't be a word found in the dictionary, or a word that
could be easily associated with you, like your middle name. A good
password will contain a mixture of letters, numbers and punctuation and
will be changed at regular intervals. The root password is changed by
running the 'passwd' program as root.

Why such caution? The root account doesn't have the restrictions that
normal user accounts have. A malicious or unqualified user with root
access can have disastrous results.

Note that you will not be able to see the password as you type it.

<Ok>
```

Screen 41: Root Password Information

If you have already been through this stage once, then the **root** password will already be installed, and these screens will not appear during a second pass.

Since there is no other choice on the introduction screen than <**Ok**>, pressing enter is the only way to proceed with the installation.

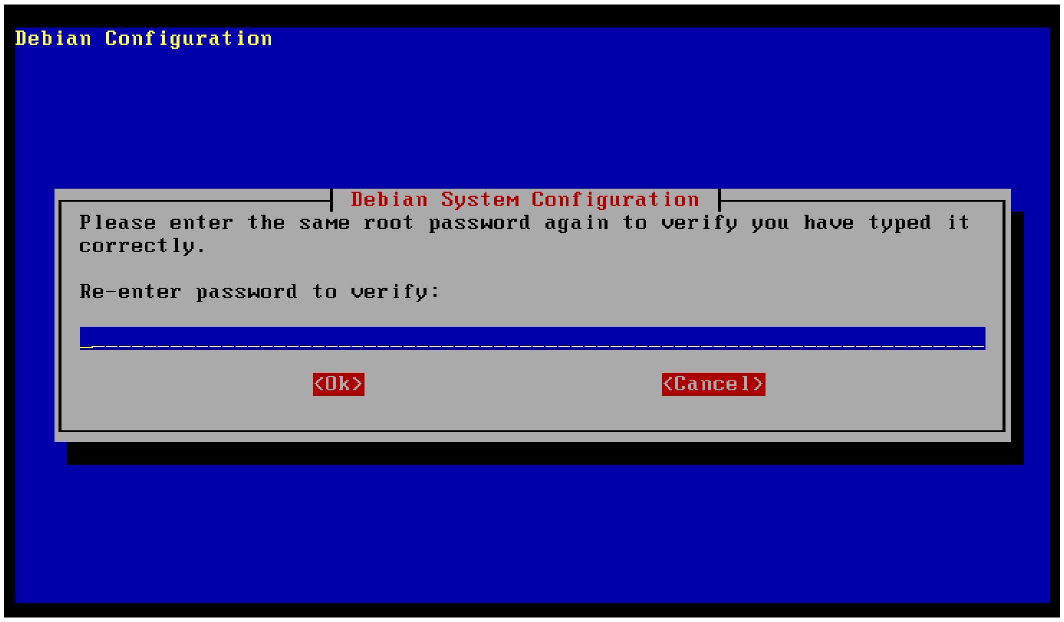
The next screen asks for the password that will be used to access the **root** account.



```
Debian Configuration

Debian System Configuration
Enter a password for root:

_
```

Screen 42: Enter Root Password

When you enter a password for **root** there will be an additional screen much like this one, asking you to verify this password.

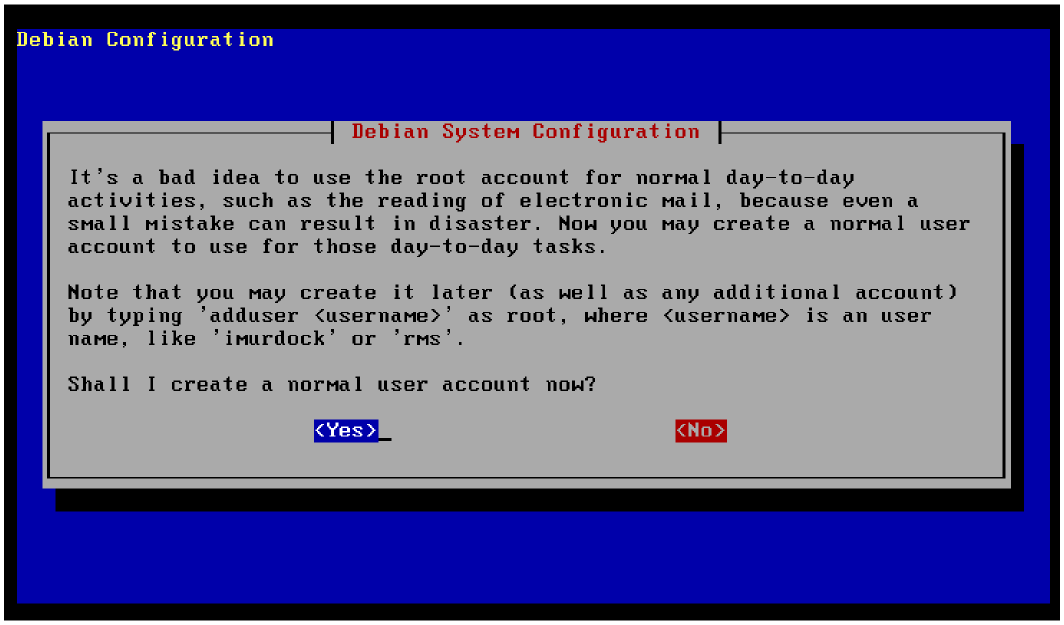


```
Debian Configuration

Debian System Configuration
Please enter the same root password again to verify you have typed it
correctly.

Re-enter password to verify:

_

<Ok>                       <Cancel>
```

Screen 43: Verify Previous Password

Simply type in the same password again and it will be assigned to the **root** account.

The next screen presents some very good reasons for having accounts with less permissions than the **root** account.



```
Debian Configuration

Debian System Configuration

It's a bad idea to use the root account for normal day-to-day
activities, such as the reading of electronic mail, because even a
small mistake can result in disaster. Now you may create a normal user
account to use for those day-to-day tasks.

Note that you may create it later (as well as any additional account)
by typing 'adduser <username>' as root, where <username> is an user
name, like 'imurdock' or 'rms'.

Shall I create a normal user account now?

                <Yes>                      <No>
```

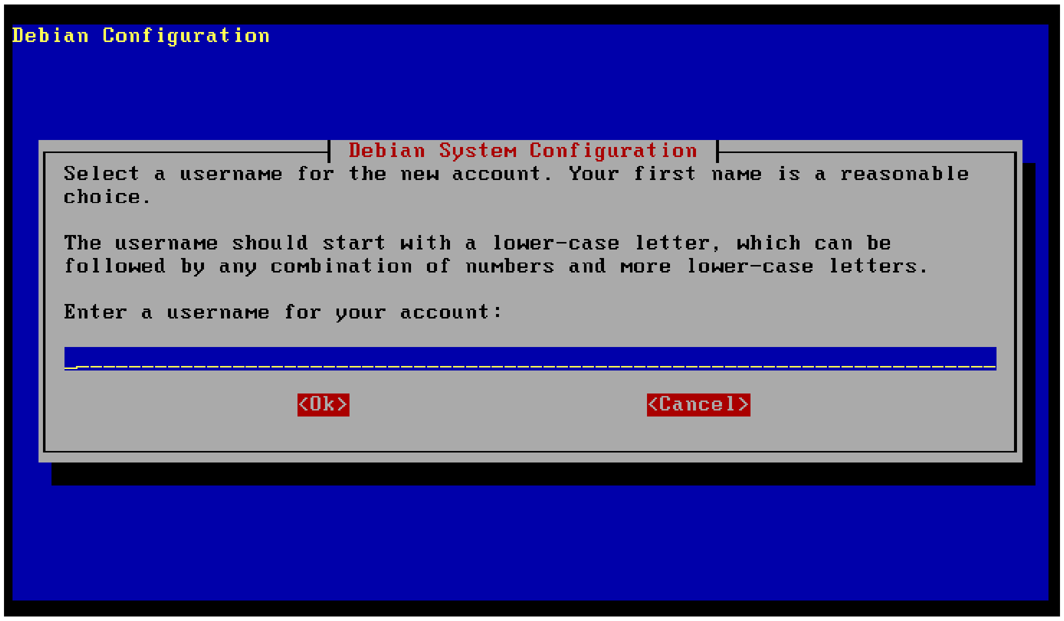
Screen 44: Additional user Accounts

There are many other good reasons to have a simple user account to use when using your Debian system to do the jobs you need a Linux system to do for you.

If you press enter to activate the <**Yes**> button you will be offered a screen for entry of the required information.

If you don't want to create a user at this time, press the tab key, highlighting the <**No**> button, and press **ENTER**. You will not be presented with the **add user** screen.

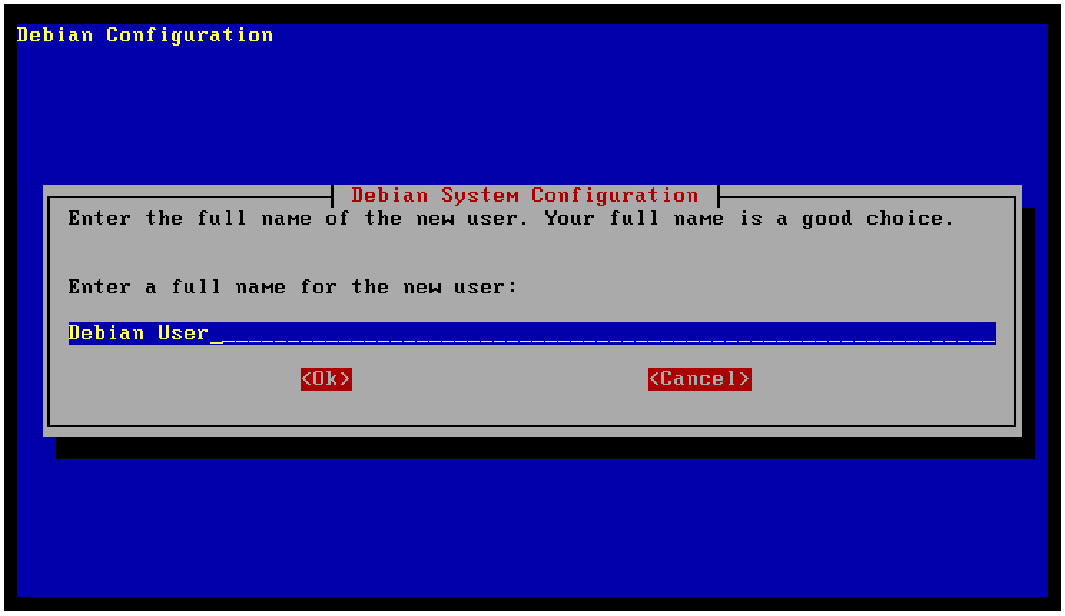
If you pressed the <**Yes**> button on the previous screen, the **add user** screen is presented next.



```
Select a username for the new account. Your first name is a reasonable
choice.

The username should start with a lower-case letter, which can be
followed by any combination of numbers and more lower-case letters.

Enter a username for your account:

_________________________________________________________________________
```

Screen 45: Adding a New User

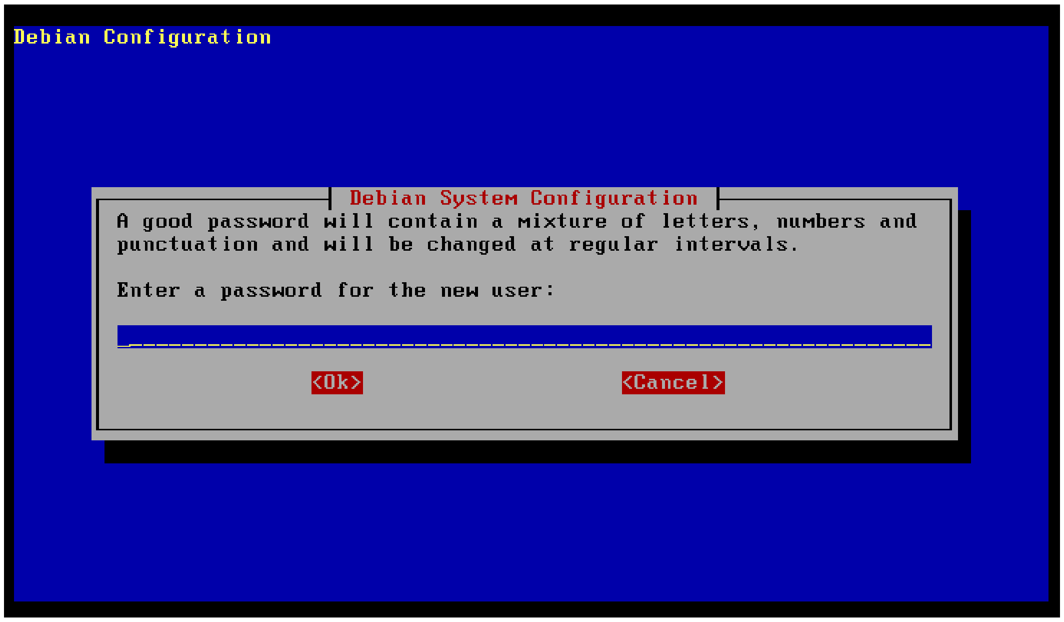
After you enter a **user name**, the next screen provides an entry for the full name of the user.



```
Debian Configuration

                  ┤ Debian System Configuration ├
 Enter the full name of the new user. Your full name is a good choice.

 Enter a full name for the new user:

 Debian User_____________________________________________________________

                  <Ok>                          <Cancel>
```

Screen 46: User's Full Name

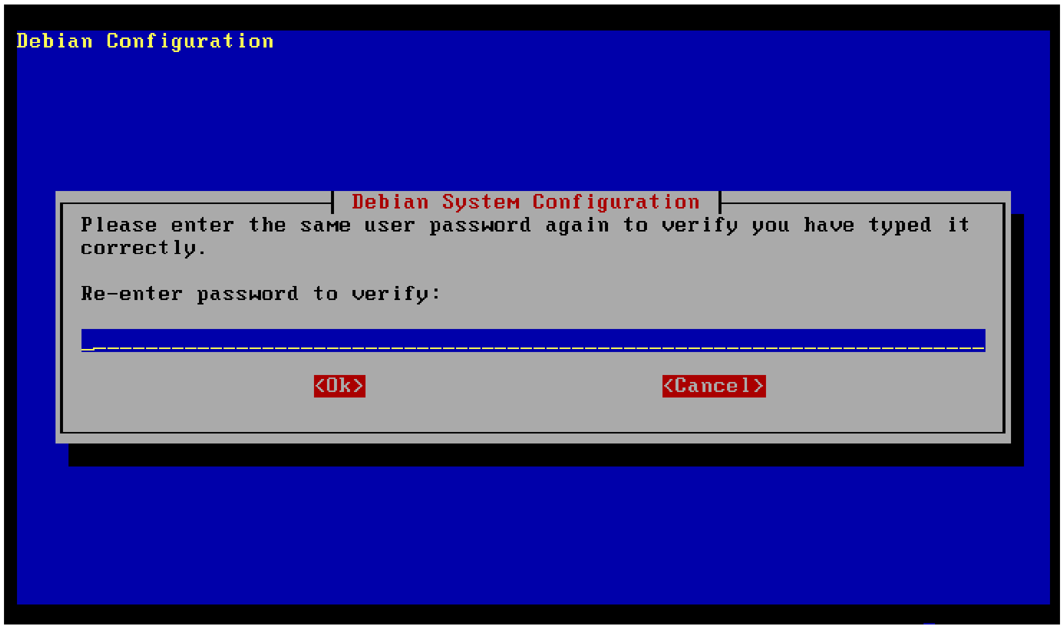
Dwarf usually leaves this field blank. Whatever you decide to enter, the next two screens will ask for the password for this user.



```
Debian Configuration

Debian System Configuration
A good password will contain a mixture of letters, numbers and
punctuation and will be changed at regular intervals.

Enter a password for the new user:
```

Screen 47: Password for New User

Enter the same password in each screen.

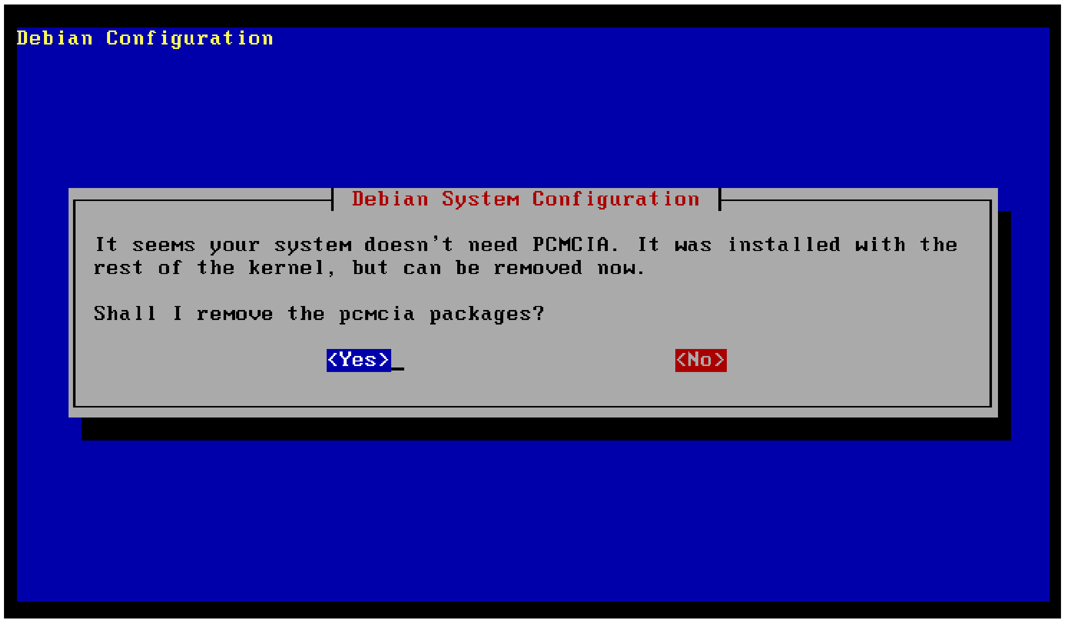


```
Debian Configuration

                    Debian System Configuration
 Please enter the same user password again to verify you have typed it
 correctly.

 Re-enter password to verify:

 _______________________________________________________________________

                  <Ok>                          <Cancel>
```

Screen 48: Verify New User Password

The second entry is to verify the password, so they must be identical for it to be accepted.

Unless the target machine is a laptop, it will most likely not be using pcmcia. If that is the case, the next screen is presented.



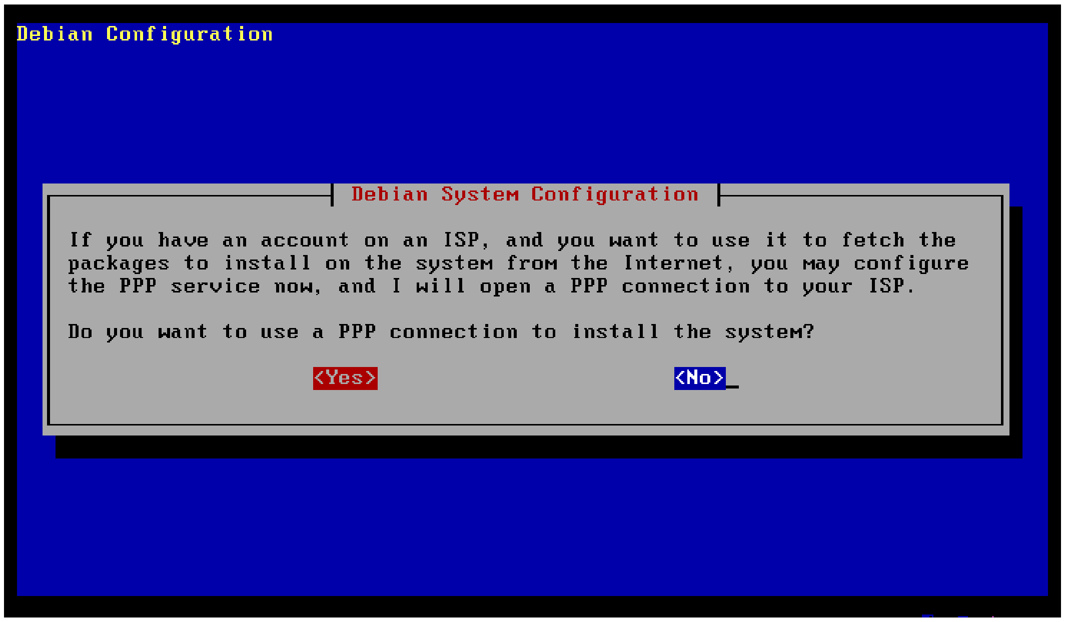
```
It seems your system doesn't need PCMCIA. It was installed with the
rest of the kernel, but can be removed now.

Shall I remove the pcmcia packages?
```

Screen 49: Remove PCMCIA Packages?

If you have no pcmcia hardware simply press **enter**. If you do have such hardware, tab to the <**No**> button and press **enter**.

PPP Configuration

Once the pcmcia software has been removed, the next offer will be to configure a PPP connection to your ISP. This option is offered at this time to enable the rest of the installation to proceed over the Internet. This installation is being done using the CD-ROM, so the connection is not needed yet, but this is as good a time as any to configure PPP.



```
Debian Configuration

Debian System Configuration

If you have an account on an ISP, and you want to use it to fetch the
packages to install on the system from the Internet, you may configure
the PPP service now, and I will open a PPP connection to your ISP.

Do you want to use a PPP connection to install the system?

<Yes>                    <No>
```

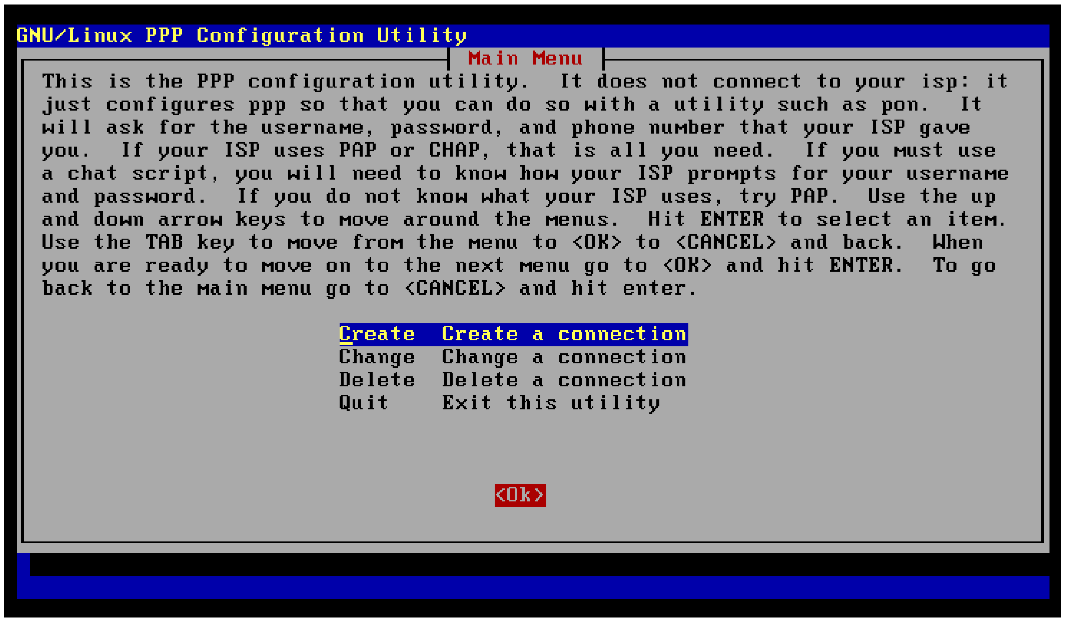
Screen 50: Set Up a PPP Connection?

The default option is set to <**No**>, so tab over to the <**Yes**> button and press **ENTER** if you wish to set up your PPP connections now.

If you answer <**No**> here, then skip ahead to page 156.

You can always configure PPP after the installation is complete by executing the script *pppconfig*. It is this script, used by the installation, that will be described in the next several pages.

When you answer <**Yes**> to the previous screen this script starts with the following screen.



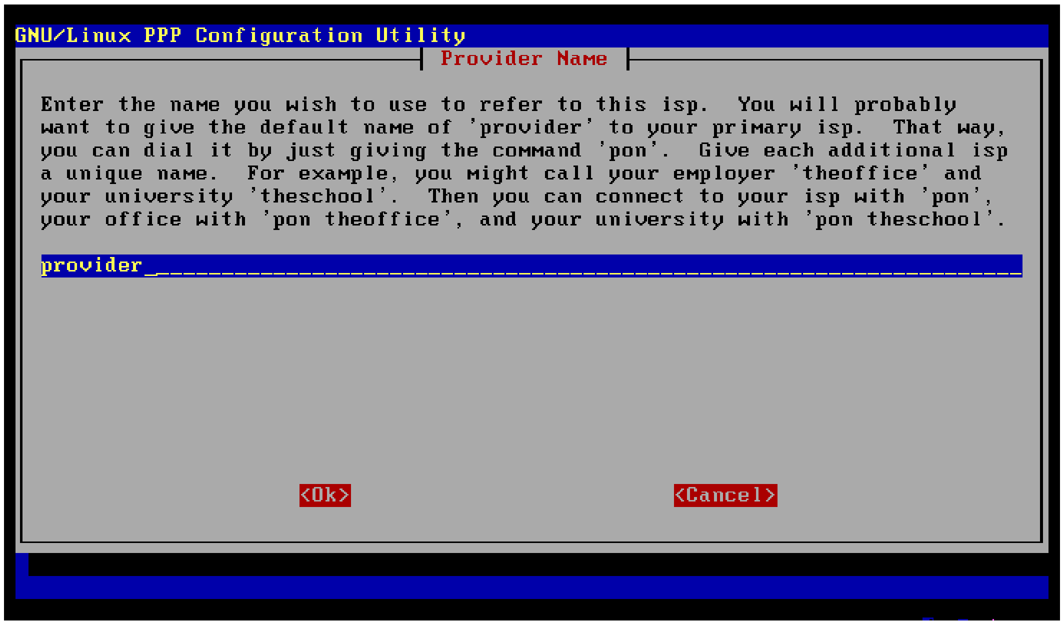
```
GNU/Linux PPP Configuration Utility
                              Main Menu
 This is the PPP configuration utility.  It does not connect to your isp: it
 just configures ppp so that you can do so with a utility such as pon.  It
 will ask for the username, password, and phone number that your ISP gave
 you.  If your ISP uses PAP or CHAP, that is all you need.  If you must use
 a chat script, you will need to know how your ISP prompts for your username
 and password.  If you do not know what your ISP uses, try PAP.  Use the up
 and down arrow keys to move around the menus.  Hit ENTER to select an item.
 Use the TAB key to move from the menu to <OK> to <CANCEL> and back.  When
 you are ready to move on to the next menu go to <OK> and hit ENTER.  To go
 back to the main menu go to <CANCEL> and hit enter.

                       Create  Create a connection
                       Change  Change a connection
                       Delete  Delete a connection
                       Quit    Exit this utility

                                   <Ok>
```

Screen 51: PPP Configuration Main Menu

This is the main menu. This configuration script creates a set of **peer** connection scripts that the PPP daemon will use to establish the connection with your ISP. This allows for the configuration of more than one ISP connection, which can be useful when you have more than one service provider.

Create a connection is the choice to make when creating a new peer configuration. If you made a mistake with a previous configuration you may edit it by choosing **Change a connection**. Remove a peer with the **Delete a connection** option and quit with **Exit this utility**.

Choose **Create a connection** and the next screen is presented, providing a field to name the peer you are configuring.



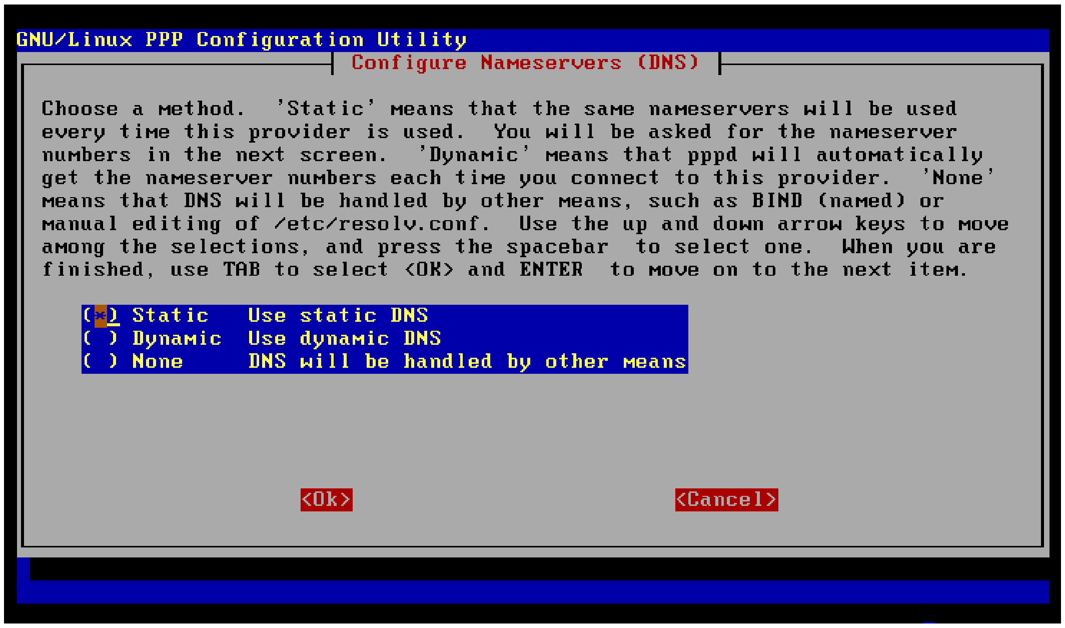
```
GNU/Linux PPP Configuration Utility
                              Provider Name

Enter the name you wish to use to refer to this isp.  You will probably
want to give the default name of 'provider' to your primary isp.  That way,
you can dial it by just giving the command 'pon'.  Give each additional isp
a unique name.  For example, you might call your employer 'theoffice' and
your university 'theschool'.  Then you can connect to your isp with 'pon',
your office with 'pon theoffice', and your university with 'pon theschool'.

provider
```

Screen 52: Name the peer

This name will be used to identify this peer configuration.

The name **provider** is supplied as the default. If you don't use this label for your connection (Dwarf prefers **home**), the installation program will not be able to establish a connection, as it expects the name **provider**. For the CD-ROM installation method this is not a problem, and you will be able to make the connection using **pon** with the correct name for the **peer** that you used during configuration.

Whatever name you enter here will deliver the next screen.



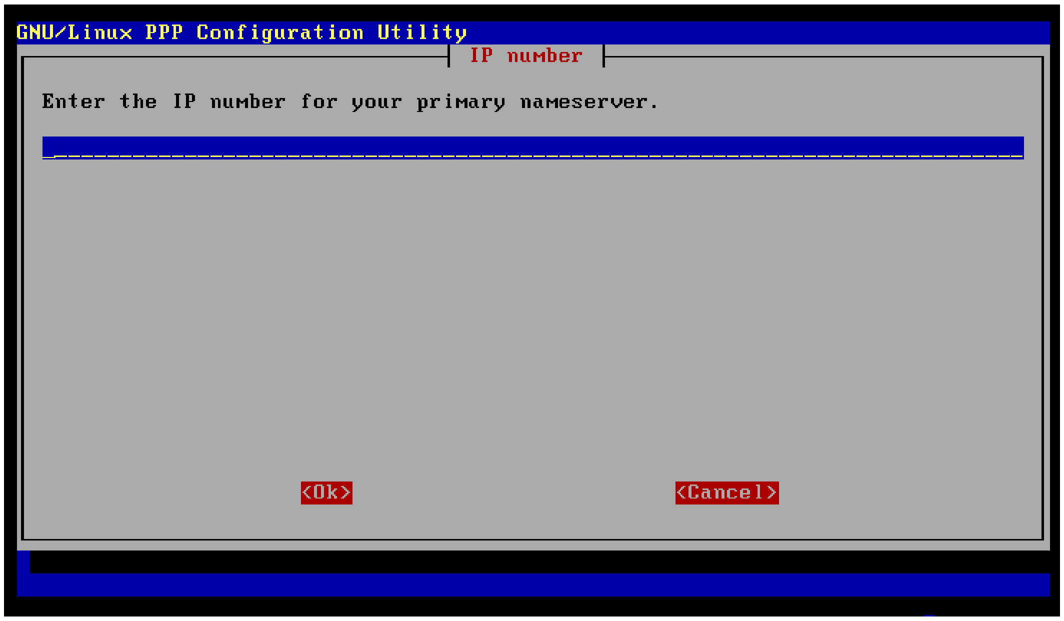
```
GNU/Linux PPP Configuration Utility
                      ┤ Configure Nameservers (DNS) ├

 Choose a method.  'Static' means that the same nameservers will be used
 every time this provider is used.  You will be asked for the nameserver
 numbers in the next screen.  'Dynamic' means that pppd will automatically
 get the nameserver numbers each time you connect to this provider.  'None'
 means that DNS will be handled by other means, such as BIND (named) or
 manual editing of /etc/resolv.conf.  Use the up and down arrow keys to move
 among the selections, and press the spacebar  to select one.  When you are
 finished, use TAB to select <OK> and ENTER  to move on to the next item.

      (*) Static    Use static DNS
      ( ) Dynamic   Use dynamic DNS
      ( ) None      DNS will be handled by other means


                        <Ok>                          <Cancel>
```

Screen 53: Choose DNS Type

This next screen is used to declare which sort of name service you will use with this provider.

The default, **static**, is the most likely. That is, you are always going to use the same **Designated Name Servers** for this ISP. If you are on a LAN, the system administrator for the LAN will tell you what these numbers should be.

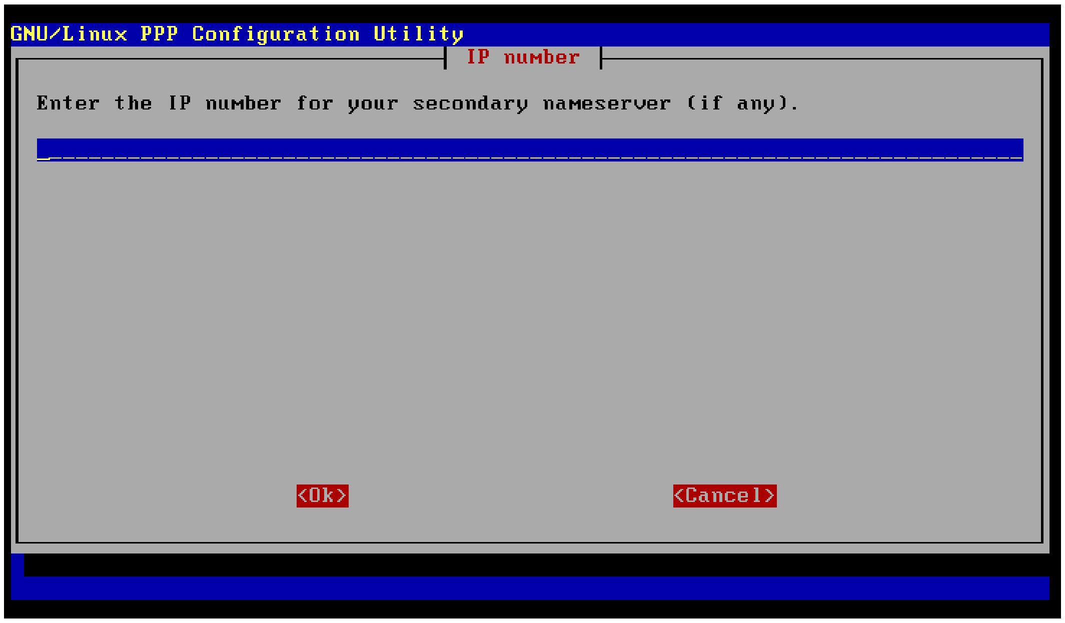
If you choose **Static** the next two screens will ask for the IP address of the primary and secondary name servers.



```
GNU/Linux PPP Configuration Utility
                              ┤ IP number ├

 Enter the IP number for your primary nameserver.

 ______________________________________________________________________________

                  <Ok>                          <Cancel>
```

Screen 54: First Designated Name Server

Place one IP address in the field provided on this screen for your primary name service.

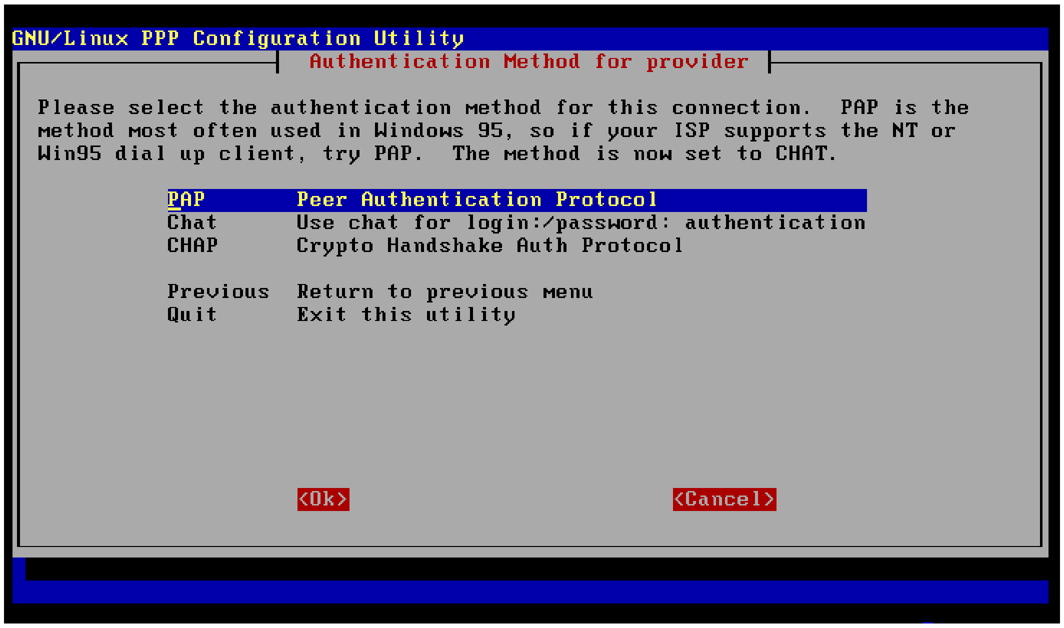


Enter the IP number for your secondary nameserver (if any).

Screen 55: Second Designated Name Server

Place the second IP address in this field for the secondary name service.

The next screen offers a selection of Authentication Methods to choose from.



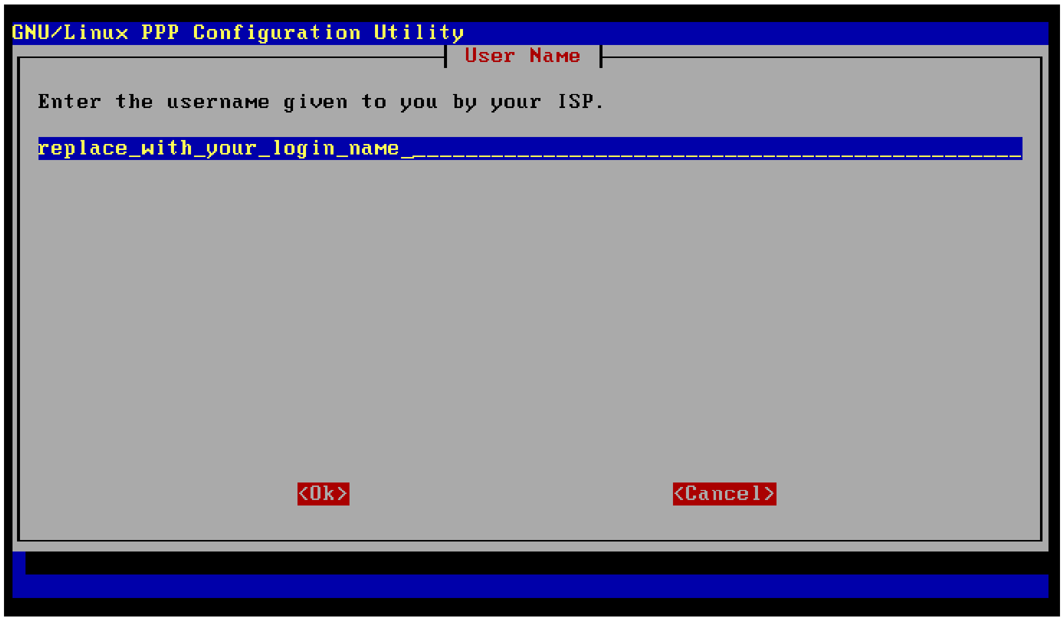
```
GNU/Linux PPP Configuration Utility
              Authentication Method for provider

 Please select the authentication method for this connection.  PAP is the
 method most often used in Windows 95, so if your ISP supports the NT or
 Win95 dial up client, try PAP.  The method is now set to CHAT.

            PAP       Peer Authentication Protocol
            Chat      Use chat for login:/password: authentication
            CHAP      Crypto Handshake Auth Protocol

            Previous  Return to previous menu
            Quit      Exit this utility
```

Screen 56: PPP Autorization Methods

The first choice, **PAP** or **Peer Authentication Protocol**, is the most likely choice. Unless your **Internet Service Provider** indicates otherwise, **PAP** can be expected to work just fine.

Depending upon which option you choose here, there will be minor differences in the questions. The following pages will follow the screens for the **PAP** option.

This next screen asks for the **User Name**. This label is the **account name** provided by your ISP.



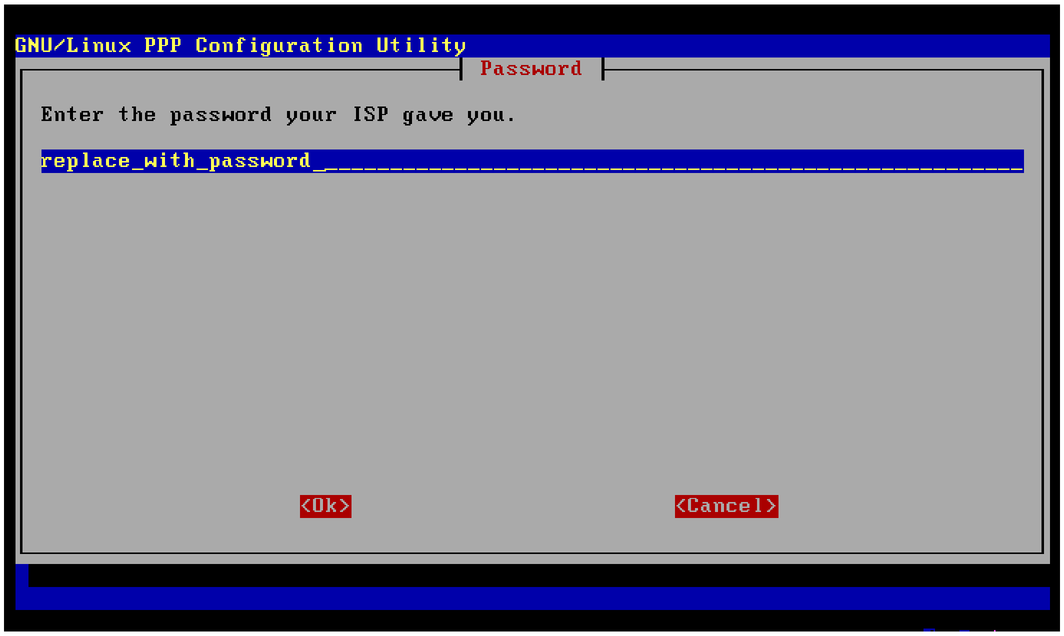
```
GNU/Linux PPP Configuration Utility
                              User Name

Enter the username given to you by your ISP.

replace_with_your_login_name_____________________________________________
```

Screen 57: PPP Login Account Name

Delete the contents of this line, and enter the **user name** provided for this account.

When the correct value is supplied on the entry line, you only need to press **ENTER** to move to the next screen. This can be a bit confusing. Some screens, without either button active, will press **OK** when the string has been entered. Other screens, such as ones providing multiple options, will require a **TAB** to the proper button for it to become active. Most of the single line string fields on a screen will activate the <**Ok**> when **ENTER** is pressed.

The next screen asks for your password. This is also a string provided by your ISP.

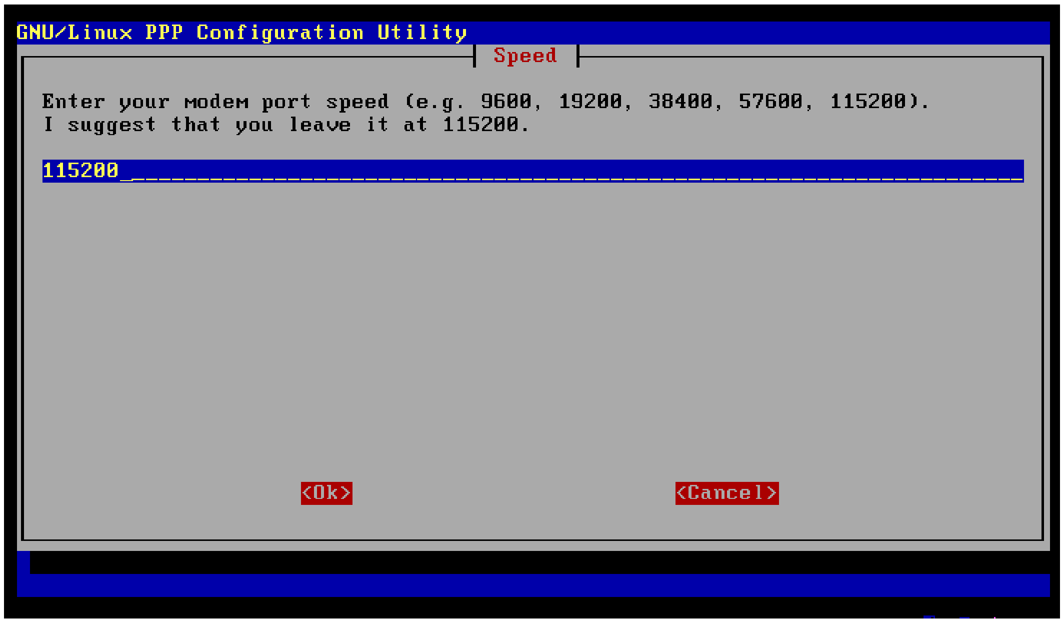


```
GNU/Linux PPP Configuration Utility
                              Password

 Enter the password your ISP gave you.

 replace_with_password_

                  <Ok>                        <Cancel>
```

Screen 58: PPP Account Password

Replace the default string with your password and press **ENTER** to move to the next screen.

This screen deals with the default speed of the modem, and provides a default value. This value indicates the fastest speed your modem port can use.

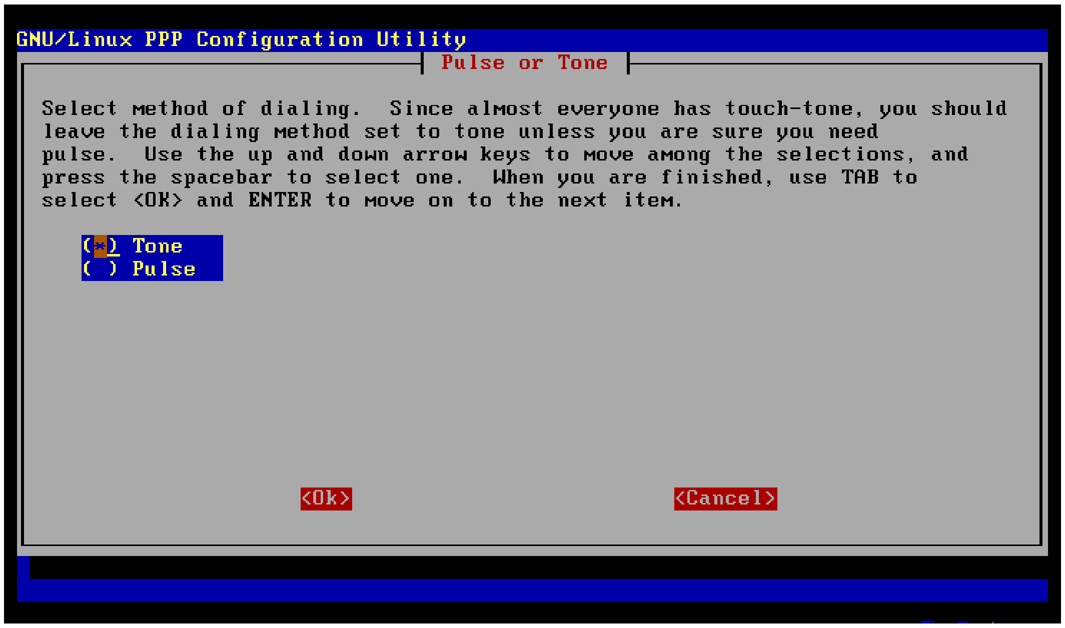


```
GNU/Linux PPP Configuration Utility
                              Speed

 Enter your modem port speed (e.g. 9600, 19200, 38400, 57600, 115200).
 I suggest that you leave it at 115200.

 115200_
```

Screen 59: PPP Default Modem Speed

Regardless of the speed of your actual modem, the default value presented is most likely adequate and should not be changed unless you know exactly why it should. Press **ENTER** to move on to the next screen.

The following screen presents the choice between Tone dialing and Pulse dialing.



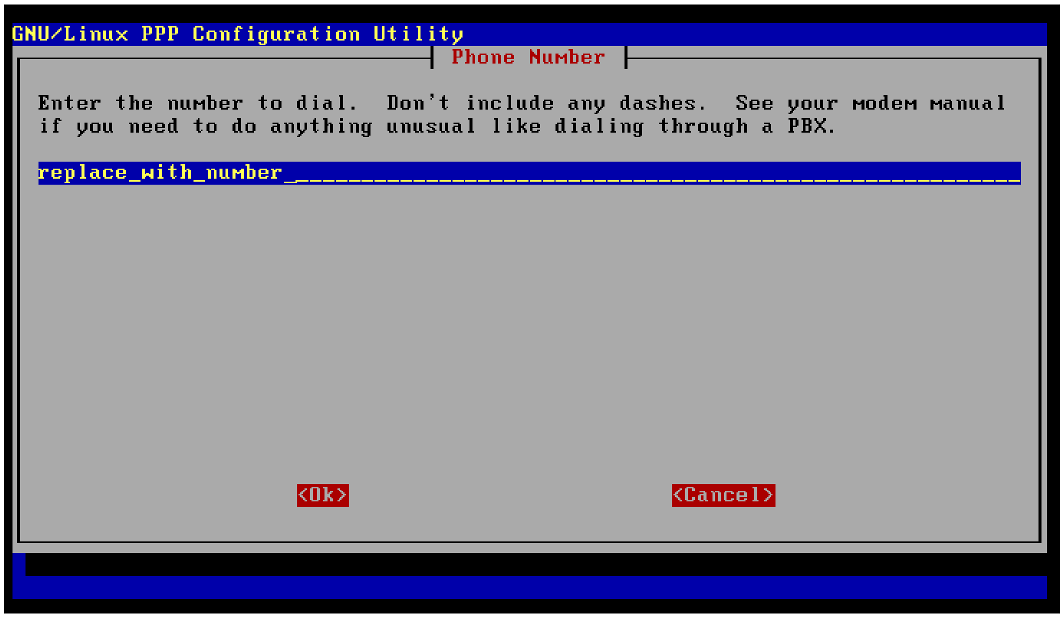
```
GNU/Linux PPP Configuration Utility
                              ┤ Pulse or Tone ├

 Select method of dialing.  Since almost everyone has touch-tone, you should
 leave the dialing method set to tone unless you are sure you need
 pulse.  Use the up and down arrow keys to move among the selections, and
 press the spacebar to select one.  When you are finished, use TAB to
 select <OK> and ENTER to move on to the next item.

     (*) Tone
     ( ) Pulse

                     <Ok>                          <Cancel>
```

Screen 60: Phone Dialing Method

Unless you are dialing through ancient phone equipment, you will most likely choose **Tone**. Dwarf is not aware of any place in the world where such equipment is being used to connect to the Internet. That doesn't mean that you will never need this option, which is why it is offered. If you find yourself in this circumstance, please contact the author and relieve his ignorance on this issue.

TAB followed by **ENTER** will accept the selected value and move on to the next screen.

This is a screen asking for the phone number used to connect to your ISP.



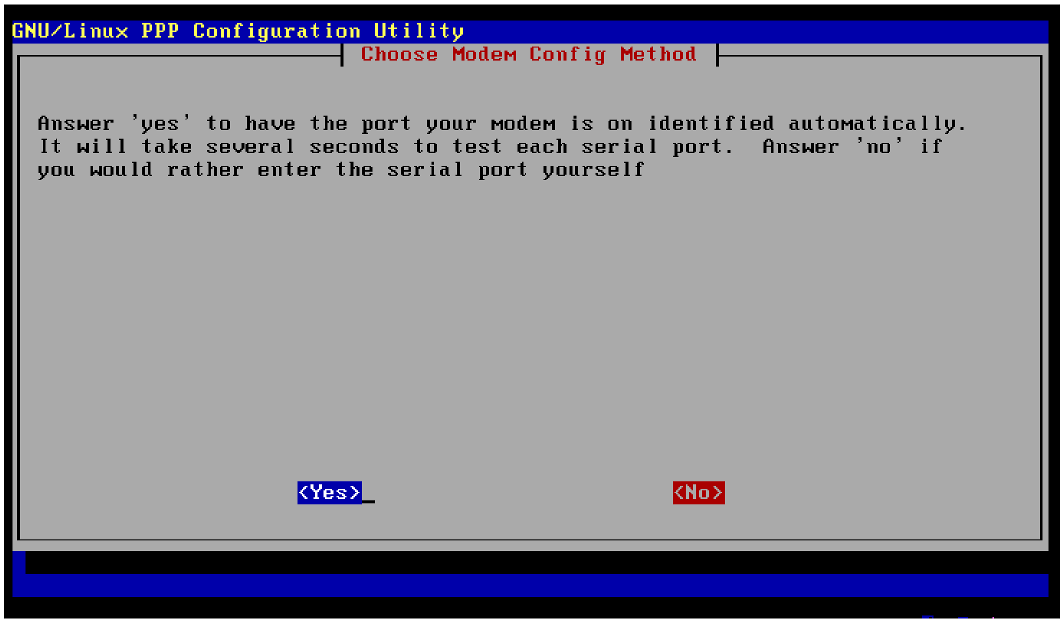
```
GNU/Linux PPP Configuration Utility
                              Phone Number

 Enter the number to dial.  Don't include any dashes.  See your modem manual
 if you need to do anything unusual like dialing through a PBX.

 replace_with_number_
```

Screen 61: Dialup Phone Number

This is not the number you use to talk to someone in the office of your provider, but rather the number used to dial the modem of your providers dialup machine.

Replace the placeholder string with the appropriate phone number, without any dash (-) or space between the numbers. Simply press **ENTER** when this value is correct.

The following screen is an introduction to the modem selection process to be used next.



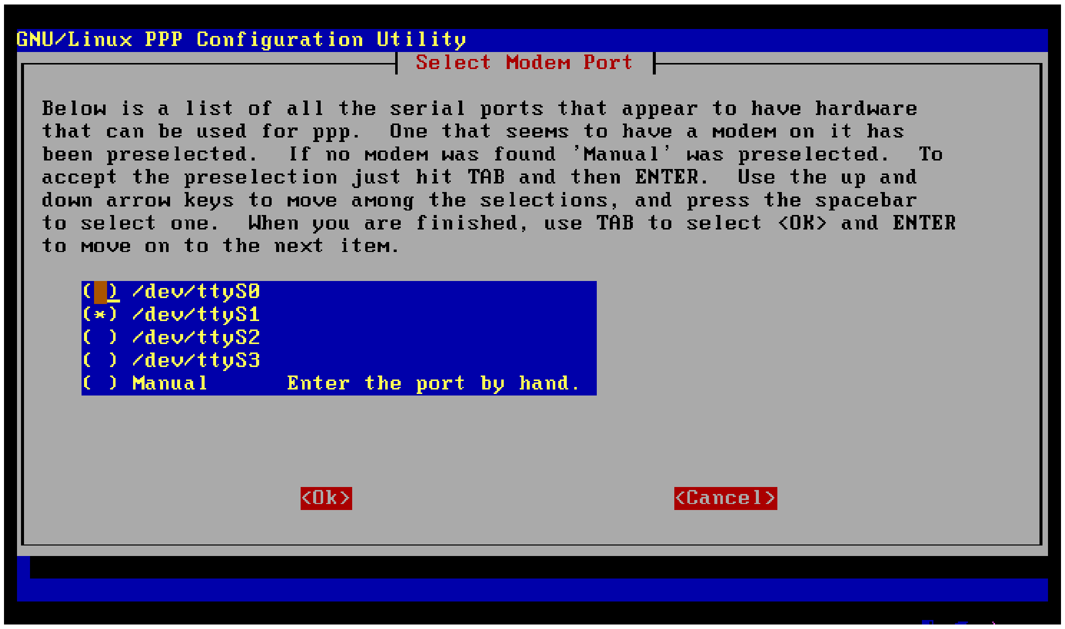
```
GNU/Linux PPP Configuration Utility
                    Choose Modem Config Method

 Answer 'yes' to have the port your modem is on identified automatically.
 It will take several seconds to test each serial port.  Answer 'no' if
 you would rather enter the serial port yourself
```

Screen 62: Manual or Automatic Modem Selection

The modem can be detected by the configuration script by pressing **ENTER** and selecting <**Yes**>. If you know the **device name** for your modem, you get the chance to enter this by selecting <**No**>.

The simplest choice is <**Yes**>. Although this may take a few seconds to probe each serial port, the choice is almost always correct. Even if it turns out to choose the wrong port (you may have two modems), you will still have the chance to change the port in the following screen. Choosing <**No**> will provide a screen detailing all the modem ports available, with the chance to enter the correct value, and is a bit less informative about the actual state of your hardware.

Assuming you have pressed <**Yes**> on the previous screen, the screen below is presented after each port has been probed.

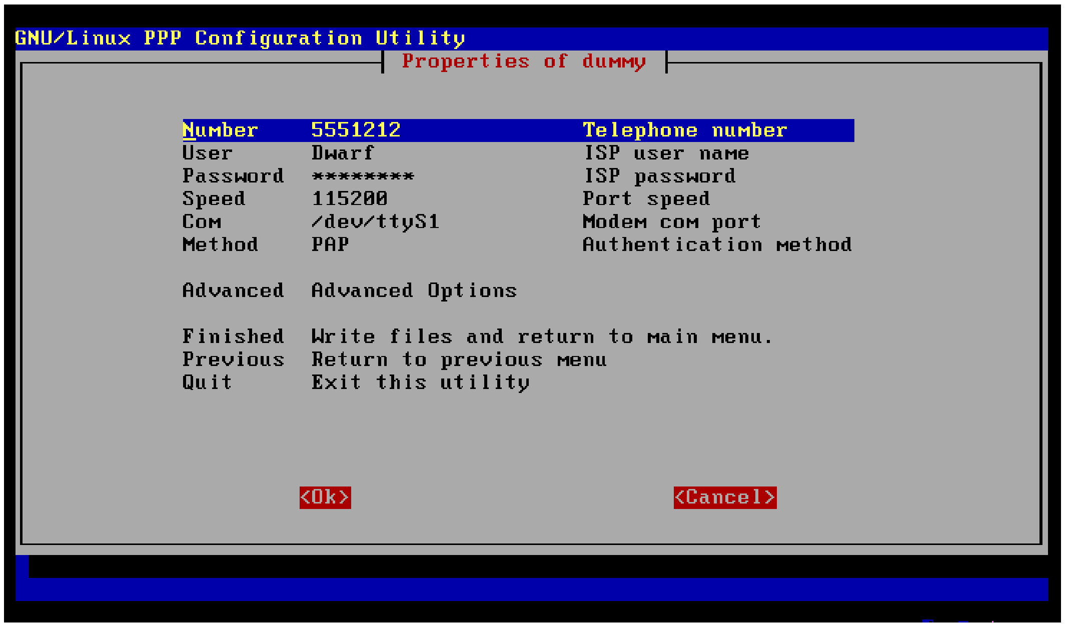


```
GNU/Linux PPP Configuration Utility
                        Select Modem Port

 Below is a list of all the serial ports that appear to have hardware
 that can be used for ppp.  One that seems to have a modem on it has
 been preselected.  If no modem was found 'Manual' was preselected.  To
 accept the preselection just hit TAB and then ENTER.  Use the up and
 down arrow keys to move among the selections, and press the spacebar
 to select one.  When you are finished, use TAB to select <OK> and ENTER
 to move on to the next item.

    ( ) /dev/ttyS0
    (*) /dev/ttyS1
    ( ) /dev/ttyS2
    ( ) /dev/ttyS3
    ( ) Manual       Enter the port by hand.


                 <Ok>                         <Cancel>
```

Screen 63: Selected Modem List

Use the space bar to mark the entry under the cursor. You can move down the list using the arrow keys, and **TAB** to the <**Ok**> button when the correct value is represented.

Finally the PPP configuration finishes with the following screen detailing the values you have just entered.

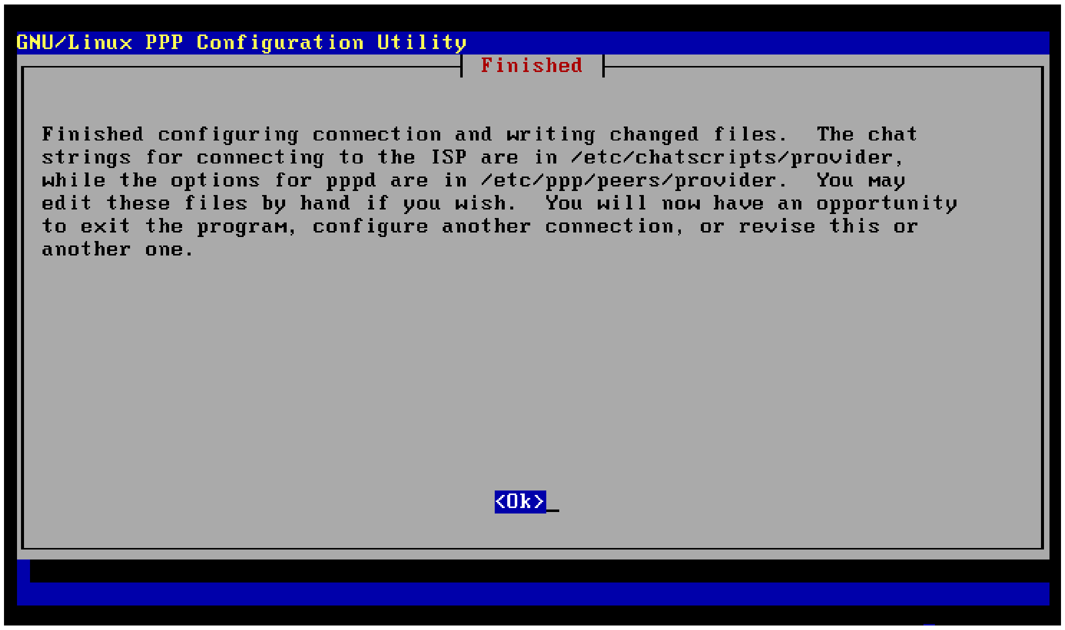


```
GNU/Linux PPP Configuration Utility
                         ┤ Properties of dummy ├

             Number    5551212                Telephone number
             User      Dwarf                  ISP user name
             Password  ********               ISP password
             Speed     115200                 Port speed
             Com       /dev/ttyS1             Modem com port
             Method    PAP                    Authentication method

             Advanced  Advanced Options

             Finished  Write files and return to main menu.
             Previous  Return to previous menu
             Quit      Exit this utility


                      <Ok>                          <Cancel>
```

Screen 64: PPP Configuration Confirmation

Unless you have special needs (like special **init** strings for you modem) you are finished and only need to arrow key down to the **Finished** option and press **ENTER**.

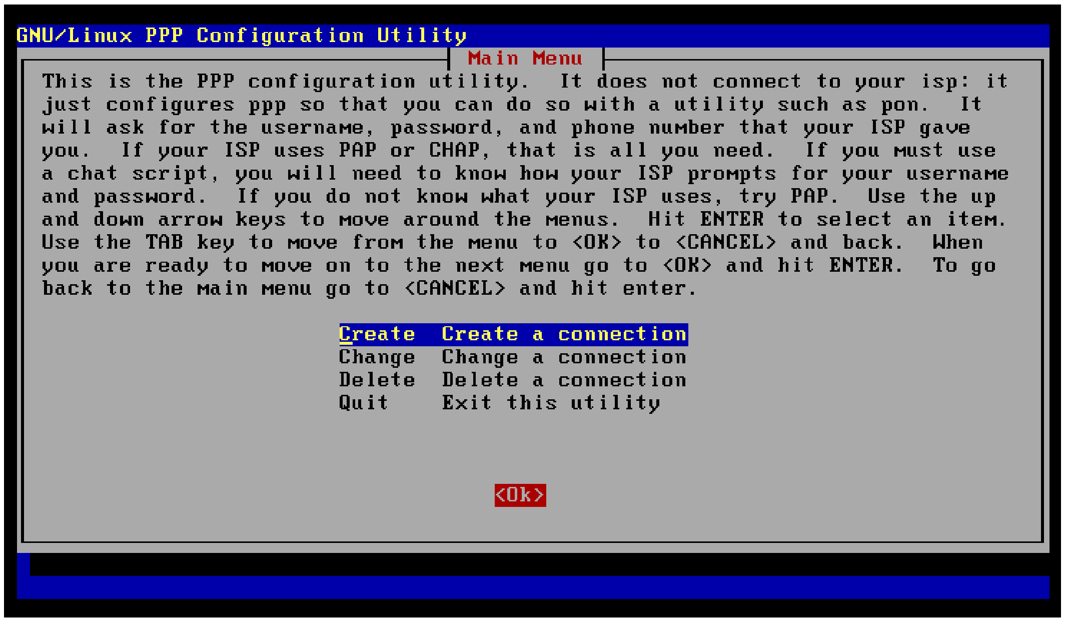
One final screen details what the configuration program has created on your system to produce the completed configuration.



```
Finished configuring connection and writing changed files.  The chat
strings for connecting to the ISP are in /etc/chatscripts/provider,
while the options for pppd are in /etc/ppp/peers/provider.  You may
edit these files by hand if you wish.  You will now have an opportunity
to exit the program, configure another connection, or revise this or
another one.
```

Screen 65: PPP Completion Dialog

This completes the PPP configuration and pressing **ENTER** returns to the **PPP Configuration Main Menu**.



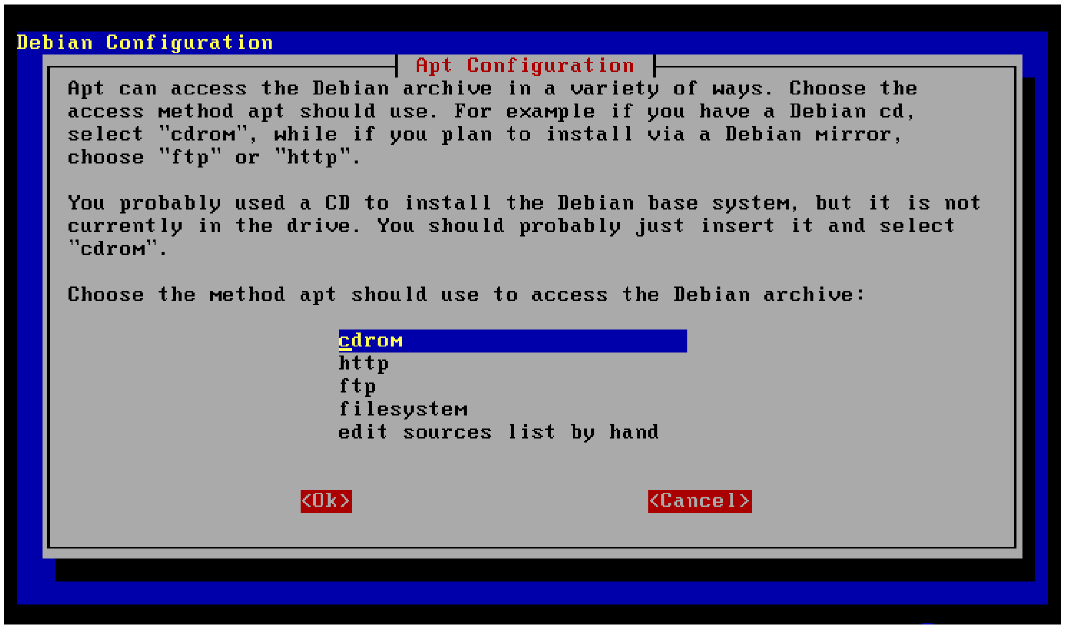
```
GNU/Linux PPP Configuration Utility
                              Main Menu
 This is the PPP configuration utility.  It does not connect to your isp: it
 just configures ppp so that you can do so with a utility such as pon.  It
 will ask for the username, password, and phone number that your ISP gave
 you.  If your ISP uses PAP or CHAP, that is all you need.  If you must use
 a chat script, you will need to know how your ISP prompts for your username
 and password.  If you do not know what your ISP uses, try PAP.  Use the up
 and down arrow keys to move around the menus.  Hit ENTER to select an item.
 Use the TAB key to move from the menu to <OK> to <CANCEL> and back.  When
 you are ready to move on to the next menu go to <OK> and hit ENTER.  To go
 back to the main menu go to <CANCEL> and hit enter.

                       Create  Create a connection
                       Change  Change a connection
                       Delete  Delete a connection
                       Quit    Exit this utility


                                  <Ok>
```

Screen 66: PPP Main Menu

Use the arrow keys to move the cursor down to the **Quit** option and the rest of the installation will proceed. You could, of course, configure another peer, or edit the peer configuration you just completed.

Configuring Apt

Whether or not you did a PPP configuration, the next screens will set up Apt for package installation. The first screen in this operation deals with the access method that will be used in the installation.



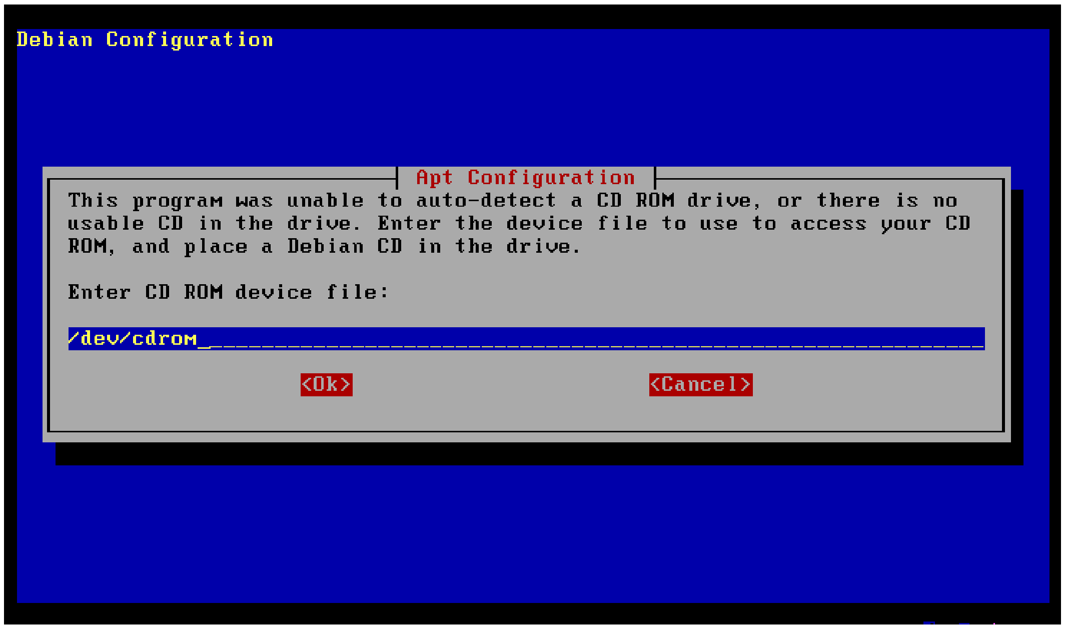
```
Debian Configuration
                          Apt Configuration
 Apt can access the Debian archive in a variety of ways. Choose the
 access method apt should use. For example if you have a Debian cd,
 select "cdrom", while if you plan to install via a Debian mirror,
 choose "ftp" or "http".

 You probably used a CD to install the Debian base system, but it is not
 currently in the drive. You should probably just insert it and select
 "cdrom".

 Choose the method apt should use to access the Debian archive:

                     cdrom
                     http
                     ftp
                     filesystem
                     edit sources list by hand

                  <Ok>                        <Cancel>
```

Screen 67: Choose Apt Method

This is a description of the CD installation so it will be assumed that you will press **ENTER** with the cursor on **cdrom**. Whichever access method you choose, the next screens will be pretty much the same as for the cdrom method. The minor differences should be obvious.

Be sure that the first CD is in the reader before you press enter here.

If there is no CD in the drive, or if the driver */dev/cdrom* is not correct for the CD-ROM drive, then the following screen will be displayed:

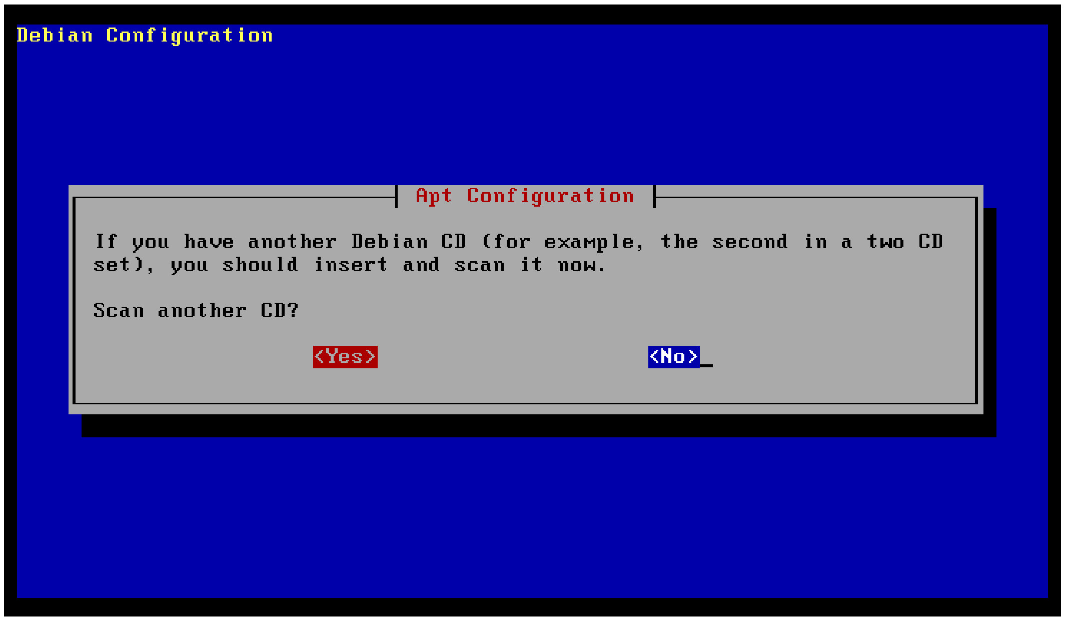


```
Debian Configuration

                        Apt Configuration
  This program was unable to auto-detect a CD ROM drive, or there is no
  usable CD in the drive. Enter the device file to use to access your CD
  ROM, and place a Debian CD in the drive.

  Enter CD ROM device file:

  /dev/cdrom_____________________________________________________________

                  <Ok>                             <Cancel>
```

Screen 68: Unoccupied or Unconfigured CD-ROM Drive

If this isn't correct, replace */dev/cdrom* with the name of the device driver for you specific CD-ROM. For instance, if you have a **SoundBlaster Pro CD** the correct value would be */dev/sbpcd0*. If the CD-ROM is the last IDE drive on the system, then the correct value will be */dev/hdd*. When you have entered the correct driver for your CD-ROM, and the first CD is in the drive, the configuration process will proceed to scan the CD. Various messages about the details of the CD will eventually result in the next screen.

Here, the first CD has been scanned successfully and the configuration process is ready for another CD.

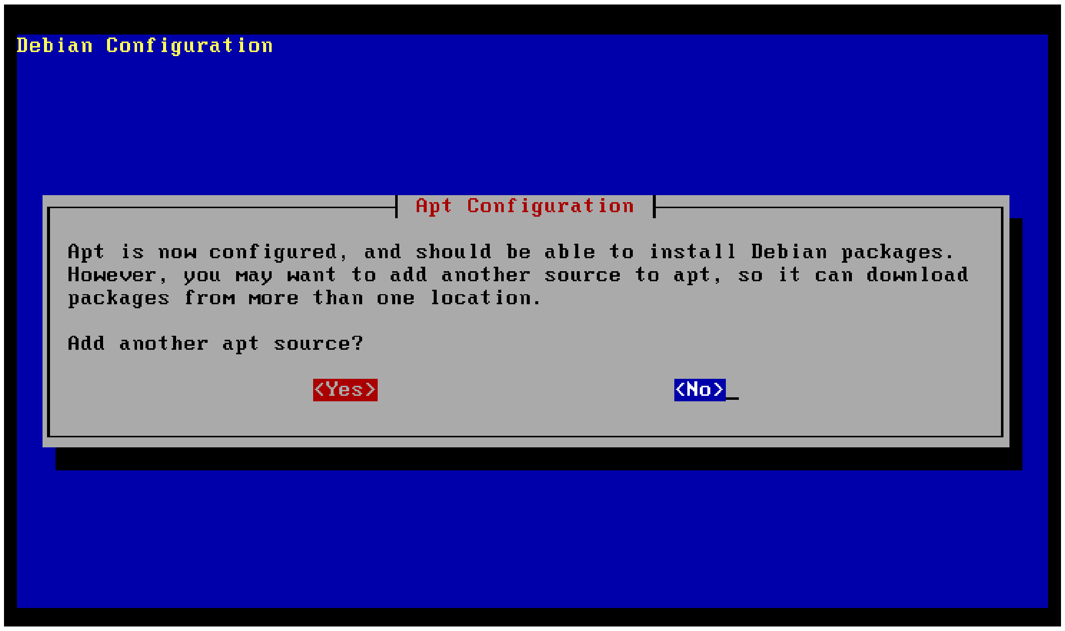


```
Debian Configuration

Apt Configuration

If you have another Debian CD (for example, the second in a two CD
set), you should insert and scan it now.

Scan another CD?
```

Screen 69: Another CD-ROM to Scan?

You may only have one installation CD, but the full distribution now takes 3 to 4 CDs to hold everything. Depending upon how many CDs you have, place the next one into the CD-ROM drive and **TAB** to the <**Yes**> button and press **ENTER**. Do this for each CD-ROM delivered, until all of them have been scanned by the system, then press **ENTER** over the <**No**> button to continue with the configuration process. Be sure to remove the last CD from the drive when you have completed this step. You will be asked for the specific CD when it is needed in the future segments of the installation.

After all the CDs have been processed, an additional opportunity to add to the source list.



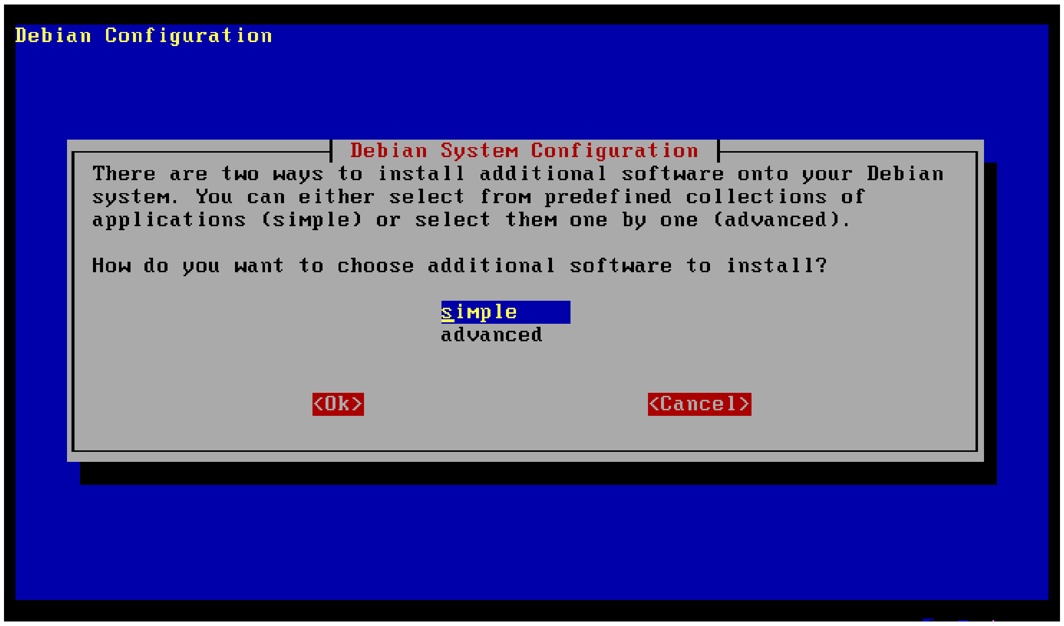
```
Debian Configuration

Apt Configuration

Apt is now configured, and should be able to install Debian packages.
However, you may want to add another source to apt, so it can download
packages from more than one location.

Add another apt source?
```

Screen 70: Another Apt Source?

Under these installations conditions this will not usually be necessary, and pressing **ENTER** will continue to the next step.

System Configuration

Now it is time to choose between two different installation processes. The following screen provides either the **simple** or the **advanced** process.



```
Debian Configuration

                 Debian System Configuration
  There are two ways to install additional software onto your Debian
  system. You can either select from predefined collections of
  applications (simple) or select them one by one (advanced).

  How do you want to choose additional software to install?

                           simple
                           advanced

                <Ok>                        <Cancel>
```

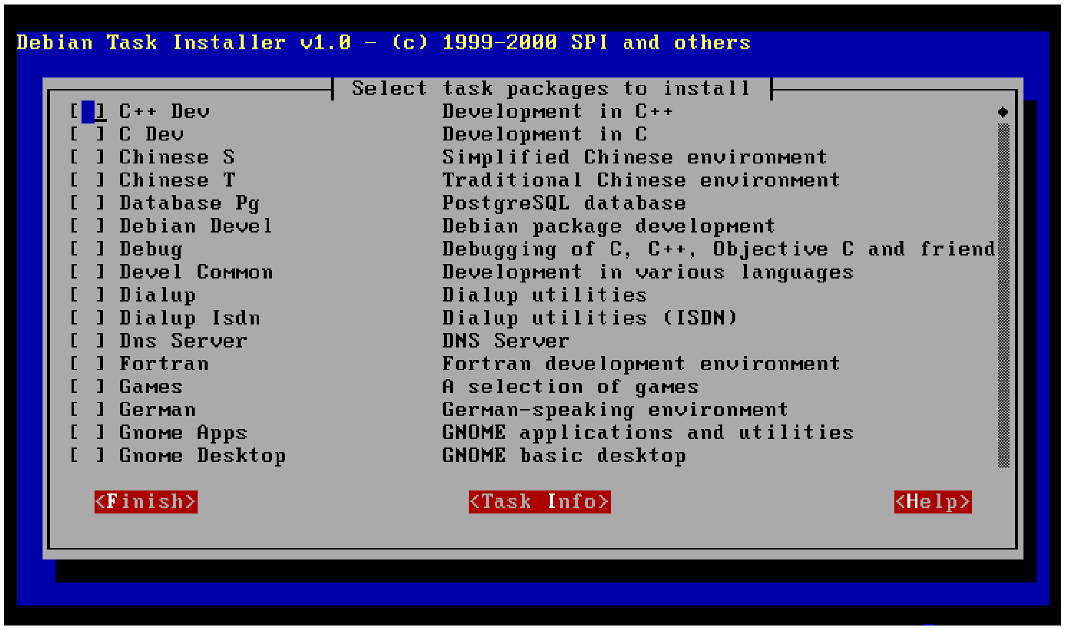
Screen 71: Simple or Advanced Install?

Unless you have some pressing reason to use the **advanced** option, Dwarf highly recommends the **simple** option. The advanced option provides **dselect** as the tool for choosing and installing the packages you want. While this lets you choose just those individual packages that you need, and their dependent packages, the process of selection can be tedious and complex. See the section on **dselect** starting on page 47 for more details.

Choosing simple provides a short list of task oriented package groupings that make it easy to get the features you want into the system you are building. Press ENTER here and, after a bit of clicking and whirring the task selection screen is presented.

The Debian Task Installer

This screen lists all of the tasks that are supported by a collection of packages.



```
Debian Task Installer v1.0 - (c) 1999-2000 SPI and others

              ┤ Select task packages to install ├
[ ] C++ Dev                Development in C++
[ ] C Dev                  Development in C
[ ] Chinese S              Simplified Chinese environment
[ ] Chinese T              Traditional Chinese environment
[ ] Database Pg            PostgreSQL database
[ ] Debian Devel           Debian package development
[ ] Debug                  Debugging of C, C++, Objective C and friend
[ ] Devel Common           Development in various languages
[ ] Dialup                 Dialup utilities
[ ] Dialup Isdn            Dialup utilities (ISDN)
[ ] Dns Server             DNS Server
[ ] Fortran                Fortran development environment
[ ] Games                  A selection of games
[ ] German                 German-speaking environment
[ ] Gnome Apps             GNOME applications and utilities
[ ] Gnome Desktop          GNOME basic desktop

  <Finish>                    <Task Info>                      <Help>
```

Screen 72: Task Package Selection

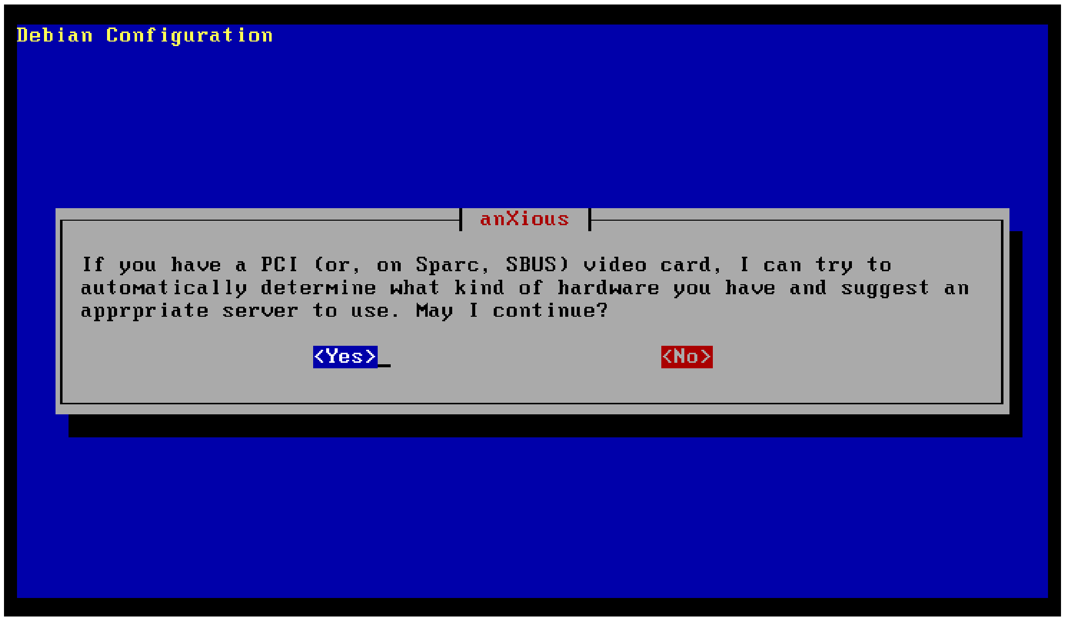
Items under the cursor can be selected with the space bar or the enter key. The cursor can be move up and down the list with the arrow keys. All of the entries in the list can not be viewed at once, and a scroll bar is provided on the right edge of the screen to view the remaining elements in the list. These include: Gnome Games, Gnome Net, Imap, Japanese, Laptop, Newbie Help, News Server, Objc Dev, Parallel Computing Dev, Parallel Computing Node, Polish, Python, Python Bundle, Python Dev, Python Web, Samba, Science, Sgml, Sgml Dev, Spanish, Tcltk, Tcltk Dev, Tex, X Window System, and X Window System Core.

The only limitations on which of these packages you may choose, are the disk storage limitations of your **root** partition. If you want to know more about a particular task package, simply **TAB** over to the <**Task Info**> button and press **ENTER**. The information about the task package that is currently under the cursor will be displayed.

When you have completed selecting the various task packages you wish to have installed, **TAB** to the <**Finish**> button and press **ENTER**.

X Configuration using anXious

If you choose the X Window System from the task packages, then anXious will be used to try to configure your X packages.


Debian Configuration

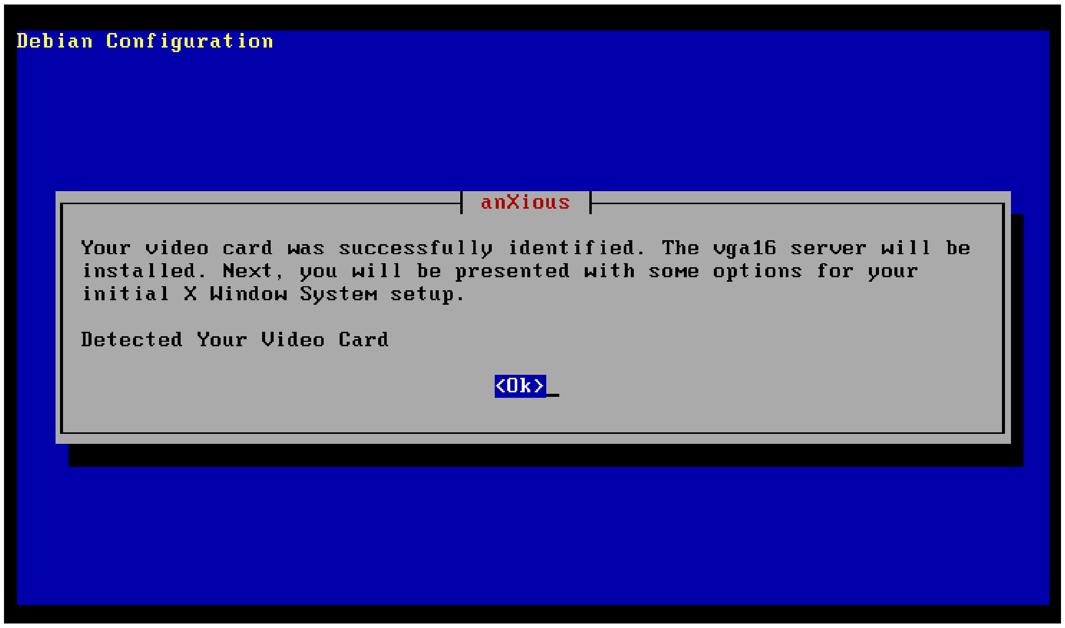
anXious

```
If you have a PCI (or, on Sparc, SBUS) video card, I can try to
automatically determine what kind of hardware you have and suggest an
apprpriate server to use. May I continue?
```

Screen 73: Probe X Hardware?

This is a new addition to the Debian installation process and it doesn't always get everything correct. You may still need to run **xf86Setup** or one of the other X configuration programs in order to get a proper configuration file. It is still worth the effort to run through these configuration steps, so you probably want anXious to probe your hardware and try to figure things out. For that reason simply press **ENTER** here and allow the script to continue.

If everything goes well, a screen like the following is presented:



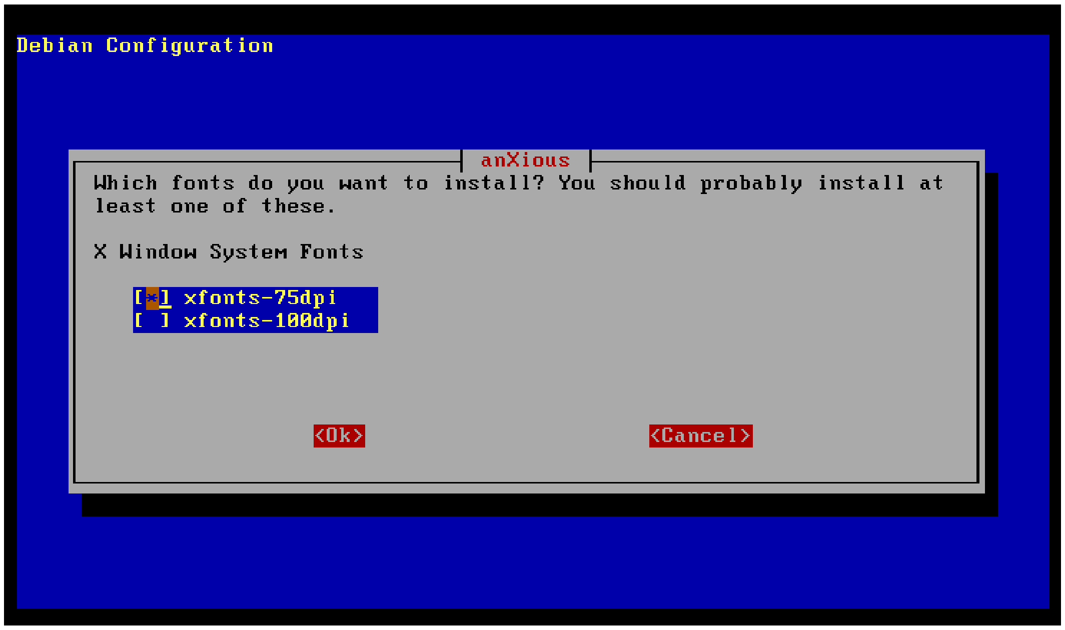
```
Debian Configuration

                              ┤ anXious ├

   Your video card was successfully identified. The vga16 server will be
   installed. Next, you will be presented with some options for your
   initial X Window System setup.

   Detected Your Video Card

                                 <Ok>
```

Screen 74: Hardware Detected

If the actual words on the screen don't make a lot of sense and major subject material seems to be missing (like the name of the server) then this step probably failed and there will be more involved in the configuration of X than this simple tool can provide.

Whatever the results of this screen, the only option is to press **ENTER** as there is only the <**Ok**> button available to press.

The next screen presented is the font selection screen.

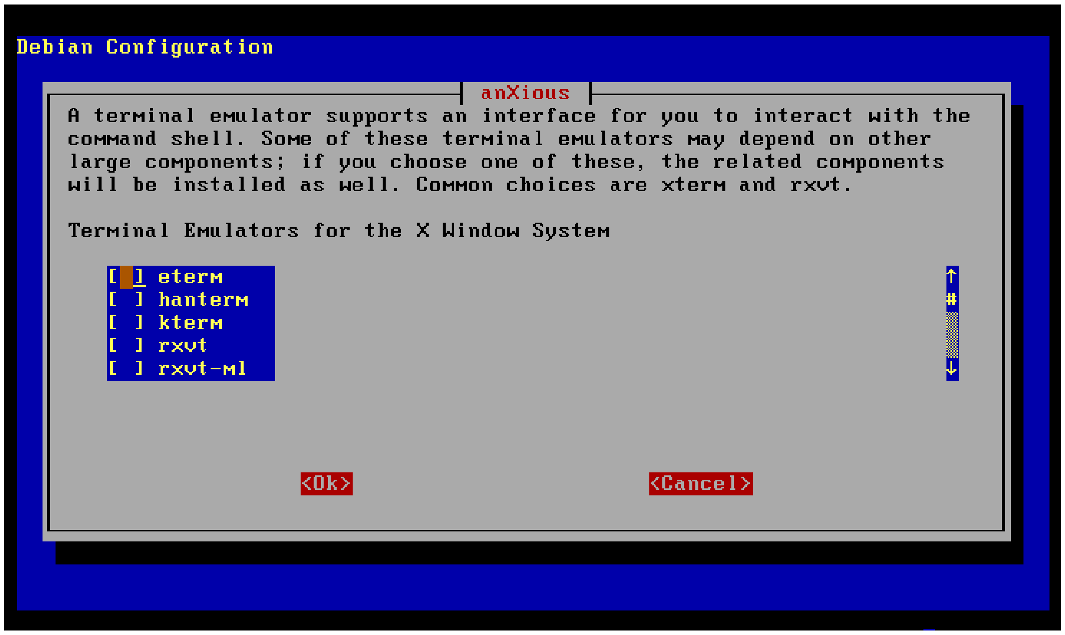


```
Debian Configuration

                              anXious
  Which fonts do you want to install? You should probably install at
  least one of these.

  X Window System Fonts

     [*] xfonts-75dpi
     [ ] xfonts-100dpi

                   <Ok>                       <Cancel>
```

Screen 75: Font Selection

Use the arrow keys to move up and down the list, the space bar to toggle the selection of any element under the cursor, and **TAB** to the <**Ok**> button when the selections are correct.

The next screen provides multiple choices for terminal emulators to use in the **X Window System**.

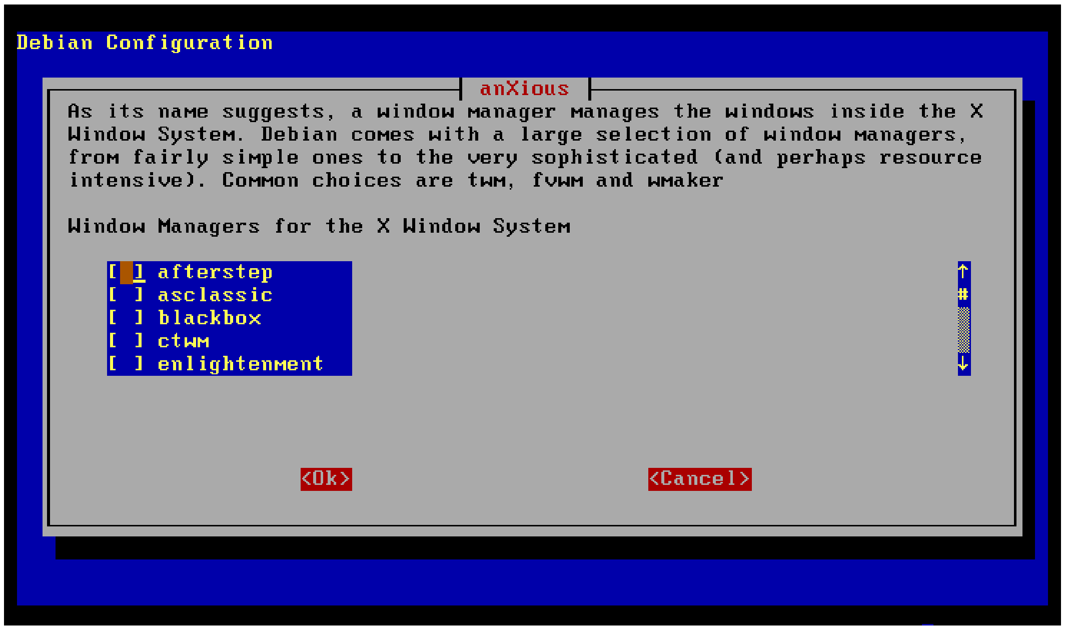


```
Debian Configuration

                          anXious
 A terminal emulator supports an interface for you to interact with the
 command shell. Some of these terminal emulators may depend on other
 large components; if you choose one of these, the related components
 will be installed as well. Common choices are xterm and rxvt.

 Terminal Emulators for the X Window System

    [ ] eterm
    [ ] hanterm
    [ ] kterm
    [ ] rxvt
    [ ] rxvt-ml
```

Screen 76: Terminal Selection

More than one emulator may be chosen, and the **xterm**, although not visible because it is off the bottom of the window, has already been selected. You can scroll down the selection window with the arrow keys or the scroll bar on the left hand side of the screen, but unless you want another terminal emulator in addition to **xterm**, simply **TAB** to the <**Ok**> button and press **ENTER**.

This very similar screen allows for the choice of additional window managers.



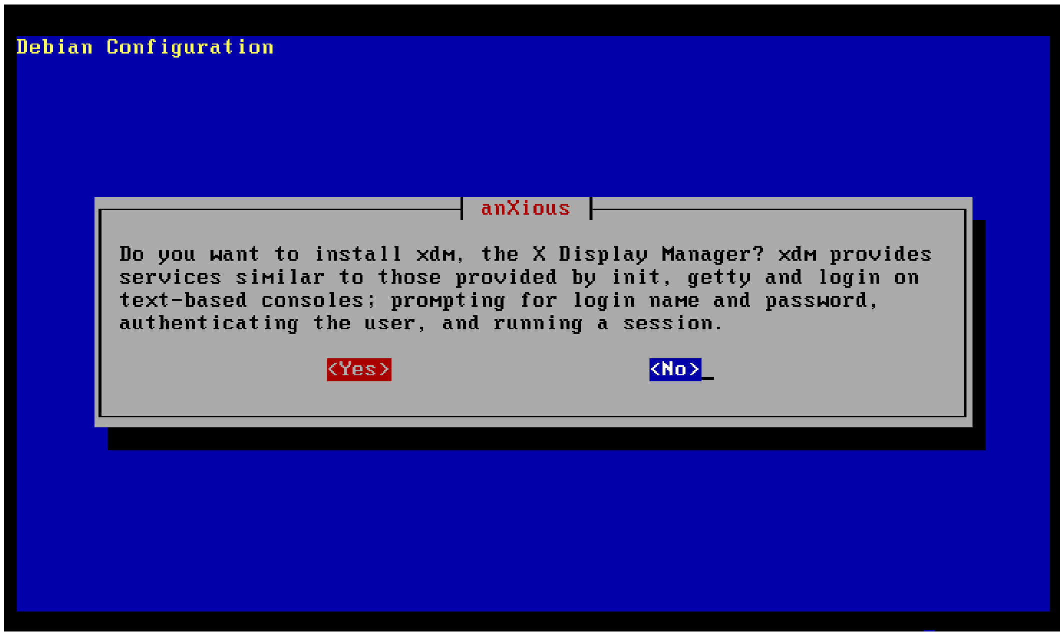
```
Debian Configuration

                              anXious
 As its name suggests, a window manager manages the windows inside the X
 Window System. Debian comes with a large selection of window managers,
 from fairly simple ones to the very sophisticated (and perhaps resource
 intensive). Common choices are twm, fvwm and wmaker

 Window Managers for the X Window System

    [ ] afterstep
    [ ] asclassic
    [ ] blackbox
    [ ] ctwm
    [ ] enlightenment
```

Screen 77: Window Manager Selection

The default selection is **twm** (visible farther down the list). The other choices not visible in the window are: flwm, flvwm, flvwm1, flvwm95, gwm, icewm, icewm-gnome, lwm, olvwm, olwm, qvwm, sawmill, scwm, twm, xtwm, wm2, and wmaker

Choose as many as you wish from this extensive list, then **TAB** to the <**Ok**> button to continue on to the next step.

This next screen offers to install the **xdm** server manager.



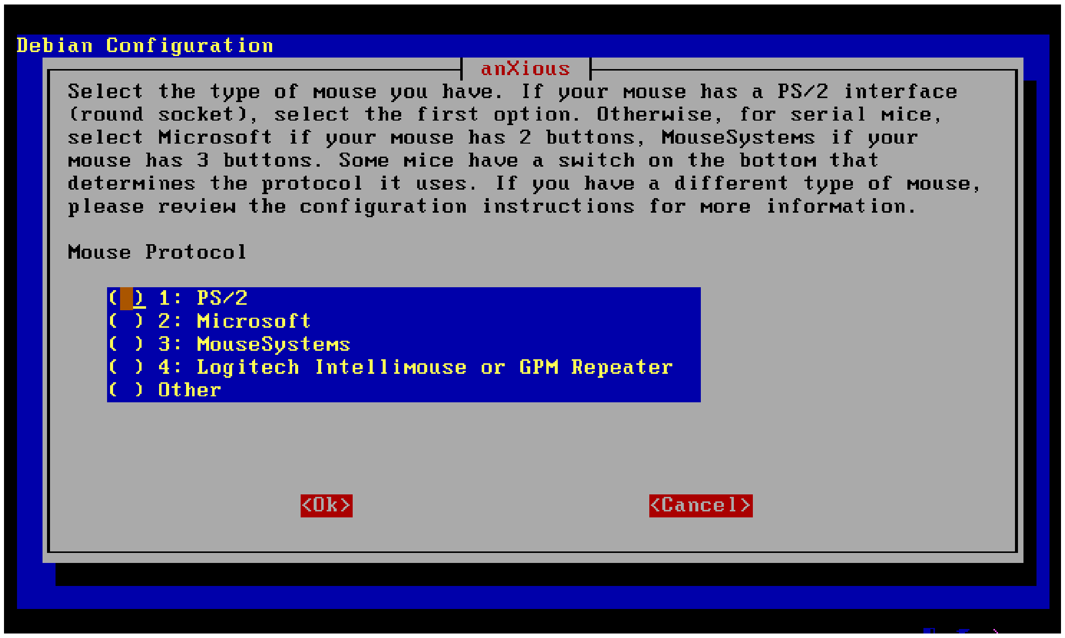
```
Debian Configuration

anXious

Do you want to install xdm, the X Display Manager? xdm provides
services similar to those provided by init, getty and login on
text-based consoles; prompting for login name and password,
authenticating the user, and running a session.
```

Screen 78: Install xdm?

Even if you eventually want the services of **xdm** this install should probably be put off. You can easily install it once you have X working to your satisfaction, and waiting will keep it from getting in the way of debugging your configuration.

The default is <**No**> so unless you really want to install **xdm** now just press **ENTER** to move to the next step.

Here is where you select your mouse type.

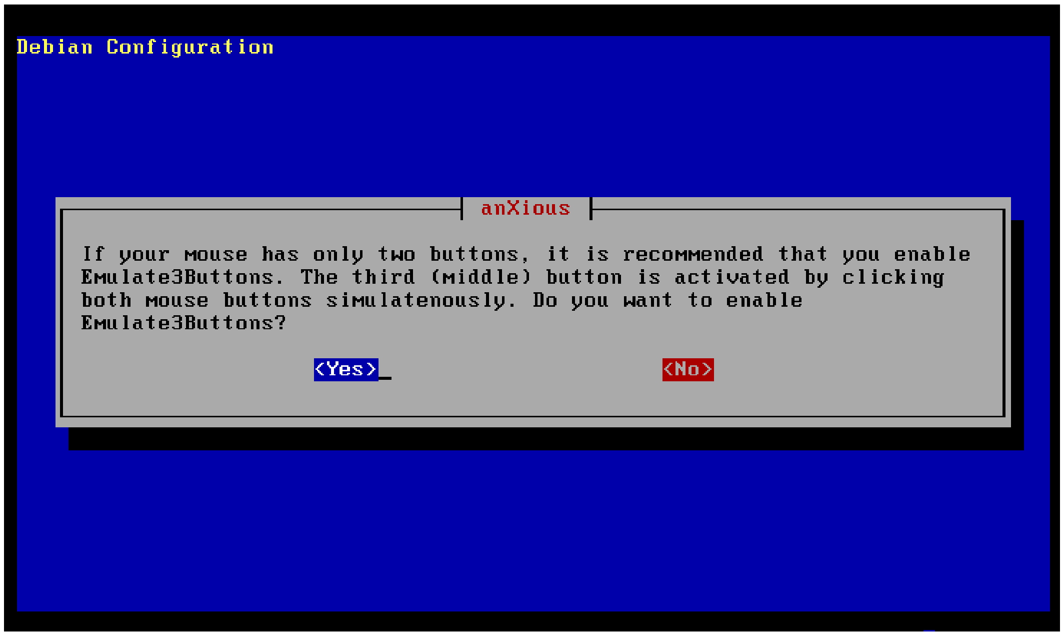


```
Debian Configuration
                              anXious
 Select the type of mouse you have. If your mouse has a PS/2 interface
 (round socket), select the first option. Otherwise, for serial mice,
 select Microsoft if your mouse has 2 buttons, MouseSystems if your
 mouse has 3 buttons. Some mice have a switch on the bottom that
 determines the protocol it uses. If you have a different type of mouse,
 please review the configuration instructions for more information.

 Mouse Protocol

    ( ) 1: PS/2
    ( ) 2: Microsoft
    ( ) 3: MouseSystems
    ( ) 4: Logitech Intellimouse or GPM Repeater
    ( ) Other
```

Screen 79: Mouse Type Selection

If you have a standard serial mouse, then choose **2: Microsoft**. Any other choice should be specifically shown on the list. Select your mouse type using the arrow keys to move the cursor and the space bar to toggle the selection. When you have the correct type identified in the list, **TAB** to the <**Ok**> button and press **ENTER** to continue.

Next you are asked about emulating the third button.

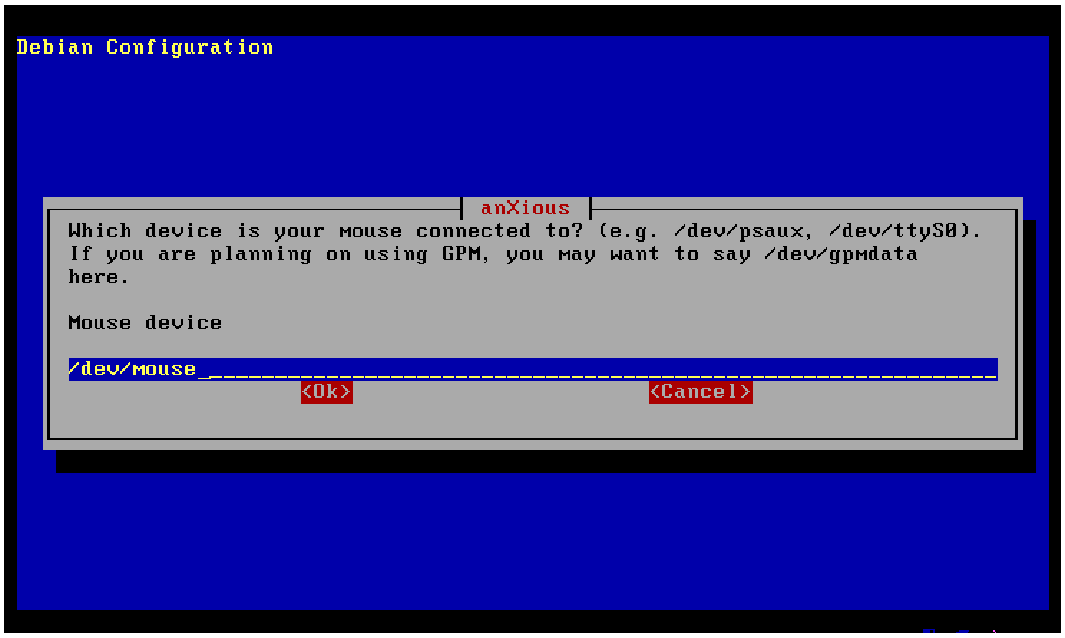


```
Debian Configuration

anXious

If your mouse has only two buttons, it is recommended that you enable
Emulate3Buttons. The third (middle) button is activated by clicking
both mouse buttons simulatenously. Do you want to enable
Emulate3Buttons?
```

Screen 80: Three Buttons?

If your mouse only has two buttons then the only way to get some window managers to provide all of their features will be to emulate the third button. If this is the case, simply press **ENTER** to accept this and move on to the next screen.

This screen asks that you declare a device for the mouse.

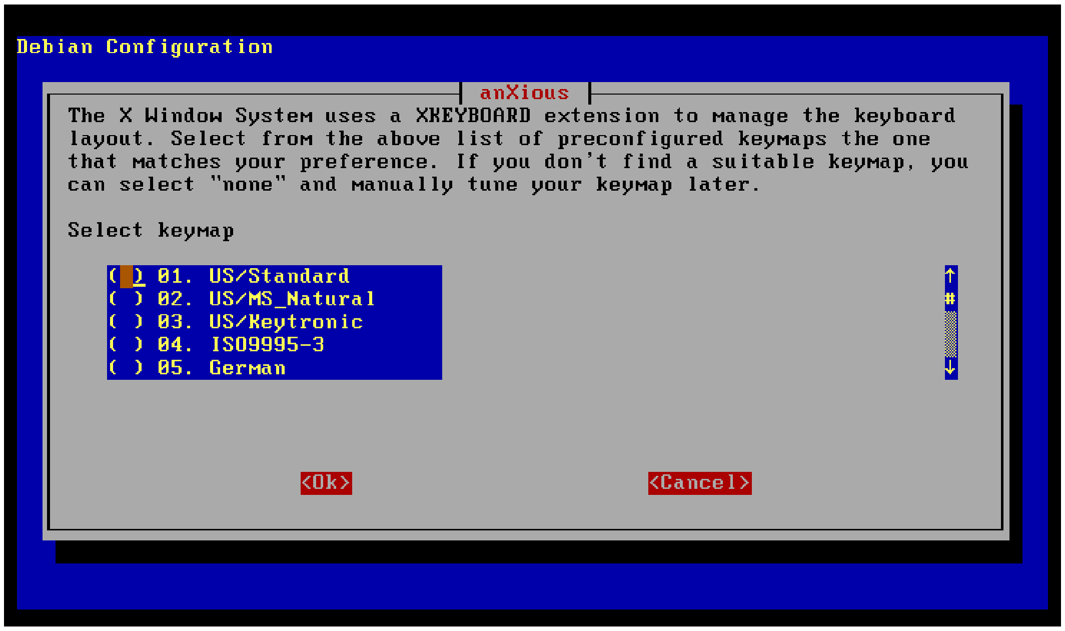


```
Debian Configuration

                              anXious
 Which device is your mouse connected to? (e.g. /dev/psaux, /dev/ttyS0).
 If you are planning on using GPM, you may want to say /dev/gpmdata
 here.

 Mouse device

 /dev/mouse_
                   <Ok>                          <Cancel>
```

Screen 81: Mouse Device Selection

This is the device that the X server will use to obtain mouse actions. If you have **gpm** installed then you will definitely want to enter the mouse as */dev/gpmdata* rather than the actual **tty device** where the mouse is connected. This is because **gpm** already has access to the physical port so X can't get to the device. **PS/2** mice get the */dev/psaux* device, and if you are not using **gpm** the mouse on the first serial port gets */dev/ttyS0* as its device. Once you have the correct device, press **ENTER** to accept this value and move on to the next screen.

Even though you already told the installation program what kind off keyboard you have, anXious wants to know this information again. The following screen asks that you select your keyboard from the list provided.



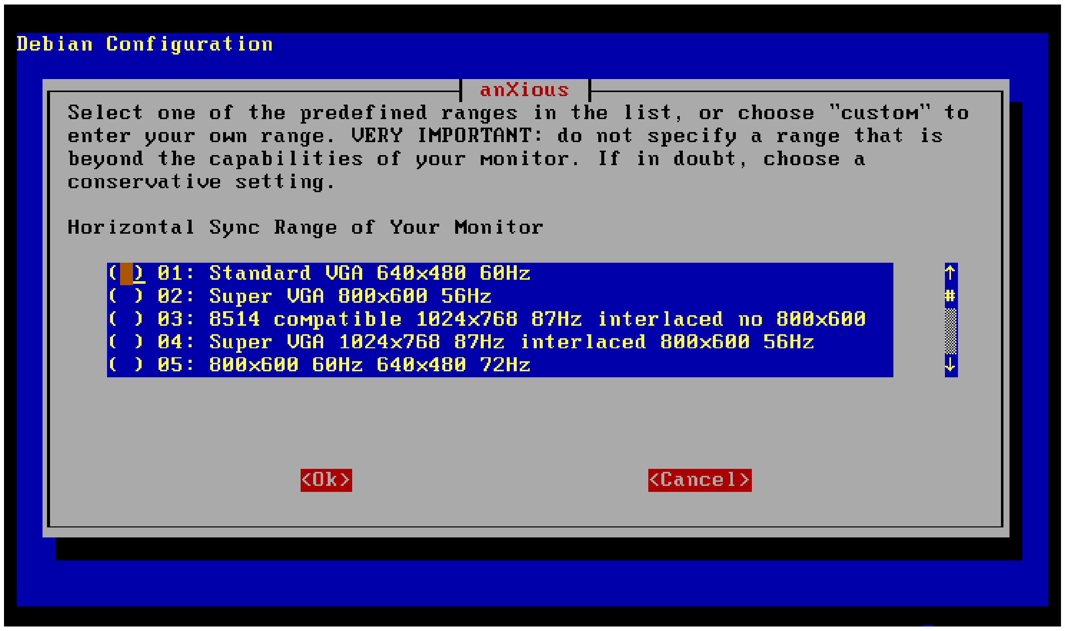
```
Debian Configuration

                              anXious
  The X Window System uses a XKEYBOARD extension to manage the keyboard
  layout. Select from the above list of preconfigured keymaps the one
  that matches your preference. If you don't find a suitable keymap, you
  can select "none" and manually tune your keymap later.

  Select keymap

     ( ) 01. US/Standard
     ( ) 02. US/MS_Natural
     ( ) 03. US/Keytronic
     ( ) 04. ISO9995-3
     ( ) 05. German

                    <Ok>                       <Cancel>
```

Screen 82: Keyboard Type Selection

The other keyboard extensions not visible in the above window are: French, Thai, Swiss/German, Swiss/French, US/International, Brazilian, disable, and none.

Arrow up and down the list, or use the slide bar on the left to move the list within the window to see more entries. Press **space** to select the item under the cursor and **TAB** to the <**Ok**> button when you have made your choice.

Now we get to the important stuff. This screen presents a selection of monitor horizontal sync ranges for you to choose from. As the screen implies, this is an important decision so have the correct values for your monitor ready when you get to this screen.



```
Debian Configuration

                              anXious
  Select one of the predefined ranges in the list, or choose "custom" to
  enter your own range. VERY IMPORTANT: do not specify a range that is
  beyond the capabilities of your monitor. If in doubt, choose a
  conservative setting.

  Horizontal Sync Range of Your Monitor

     ( ) 01: Standard VGA 640x480 60Hz
     ( ) 02: Super VGA 800x600 56Hz
     ( ) 03: 8514 compatible 1024x768 87Hz interlaced no 800x600
     ( ) 04: Super VGA 1024x768 87Hz interlaced 800x600 56Hz
     ( ) 05: 800x600 60Hz 640x480 72Hz
```

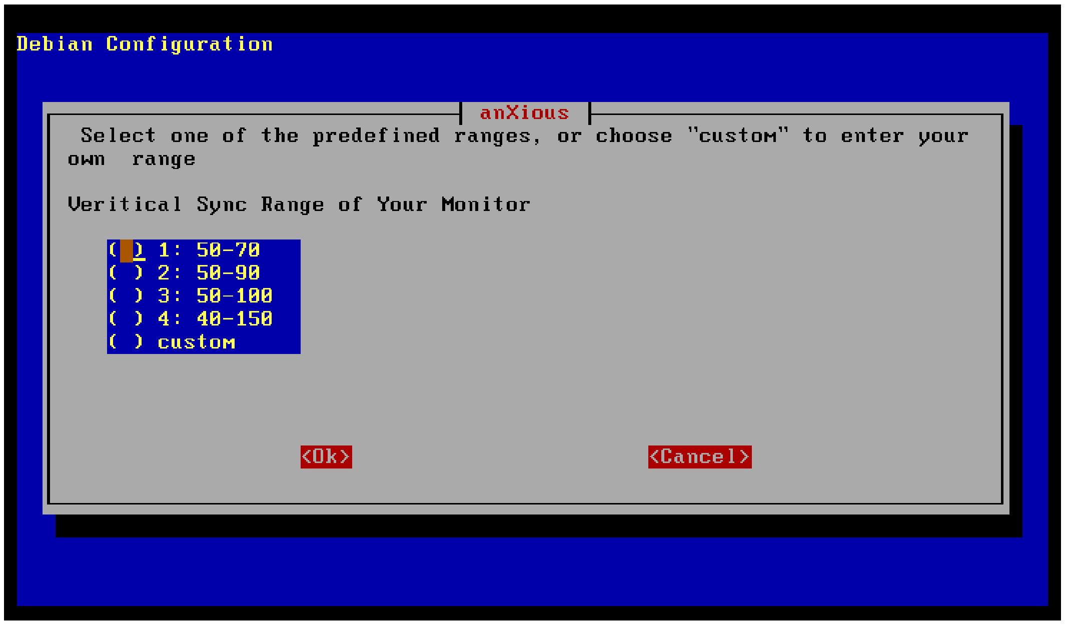
Screen 83: Monitor Horizontal Sync

Again, all the values available are not visible on the screen, but can be viewed by scrolling down the screen using the arrow keys or the provided scroll bar. The additional values available on the list are: 1024x760 60Hz, 800x600 72HZ, 1024x768 70Hz, 1280x1024 60Hz, 1280x1024 74Hz, 1280x1024 76Ha, or custom.

custom is not recommended unless you really know what you are doing. Placing an excessive value in the config using the custom option can very easily damage your monitor.

Choose a good value then **TAB** and **ENTER** to move on.

This screen expects the value for the vertical sync range of you monitor.

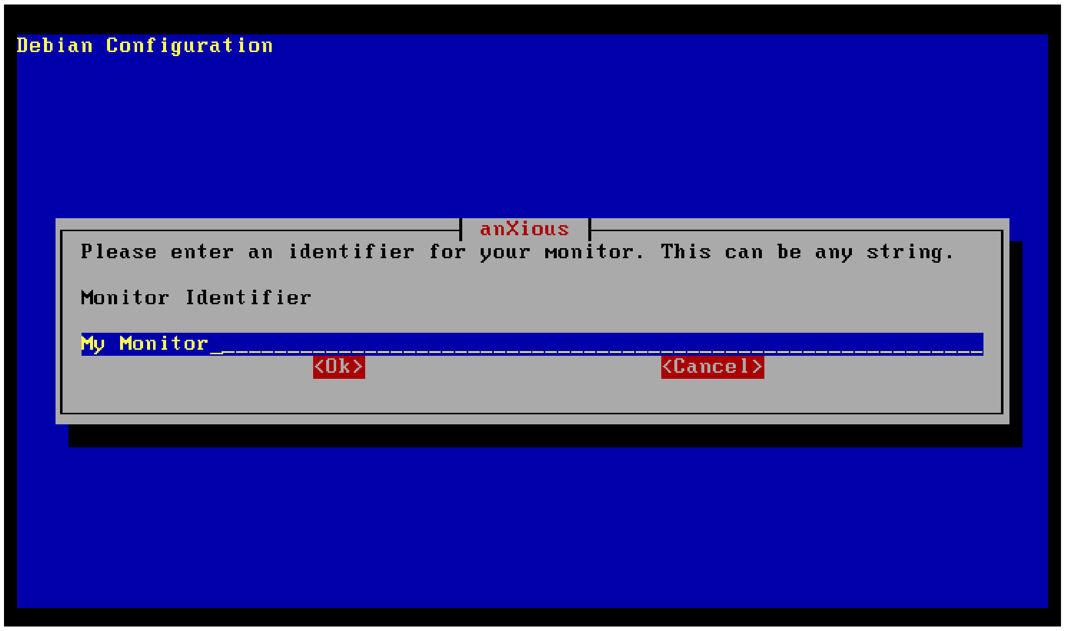


```
Debian Configuration

                              anXious
 Select one of the predefined ranges, or choose "custom" to enter your
own  range

Veritical Sync Range of Your Monitor

   ( ) 1: 50-70
   ( ) 2: 50-90
   ( ) 3: 50-100
   ( ) 4: 40-150
   ( ) custom
```

Screen 84: Monitor Vertical Sync

All of the possible values are visible in the window, so arrow down to the entry that describes your monitor's vertical sync rate, and press the **space** to select. Then **TAB** to the <**Ok**> button and press **ENTER**.

This next screen asks for a string to identify your monitor.

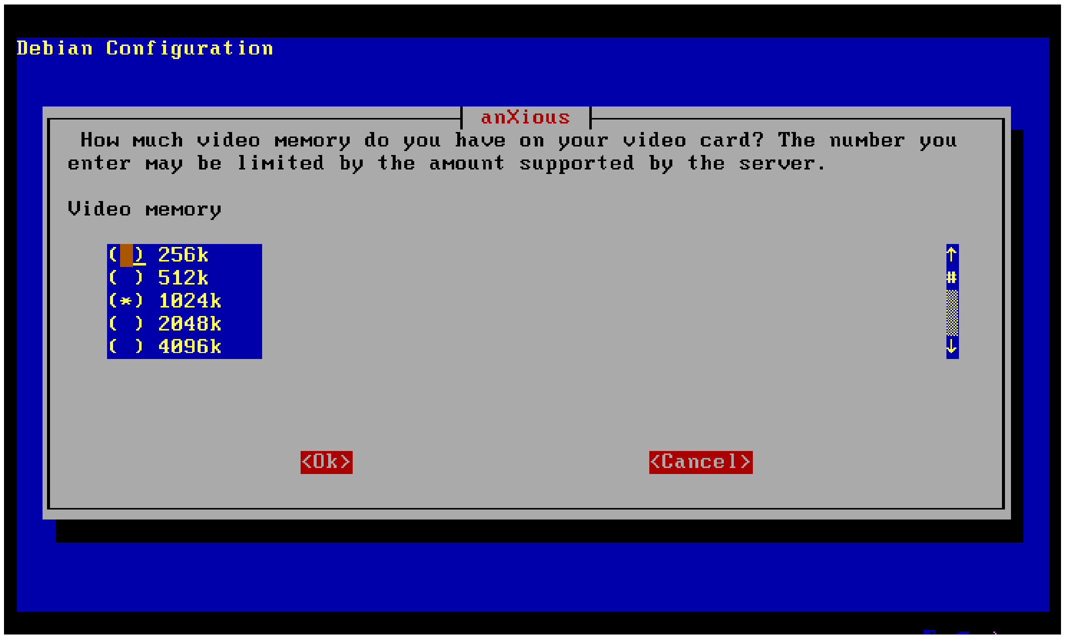


```
Debian Configuration

anXious
Please enter an identifier for your monitor. This can be any string.

Monitor Identifier

My Monitor_______________________________________________________
            <Ok>                              <Cancel>
```

Screen 85: Monitor Name Specification

This has no technical impact on the configuration of the X server, but will identify the monitor that has been configured. Replace the default string with something that more appropriately identifies your particular monitor. Simply press **ENTER** to accept the value you have typed.

The next screen deals with the amount of graphics card memory.

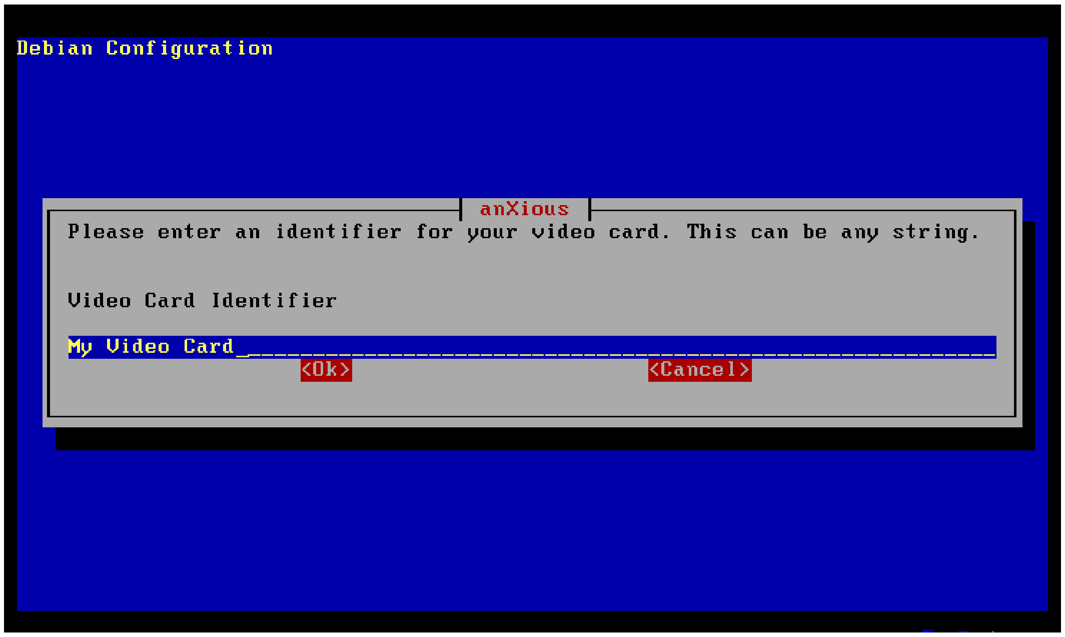


```
Debian Configuration

                    anXious
 How much video memory do you have on your video card? The number you
enter may be limited by the amount supported by the server.

Video memory

   ( ) 256k
   ( ) 512k
   (*) 1024k
   ( ) 2048k
   ( ) 4096k

            <Ok>                     <Cancel>
```

Screen 86: Graphics Card Memory

Since the card has already been probed, an item has already been selected. This is the amount of memory that the software thinks is available on your card. Unless you have some reason to believe that this is in error, simply **TAB** to the <**Ok**> button and press **ENTER**. If you must change this value, use the arrow keys to move the cursor to the correct item and press **space** to select that value. Then **TAB** to the <**Ok**> button and press **ENTER**.

Next your video card needs a name.

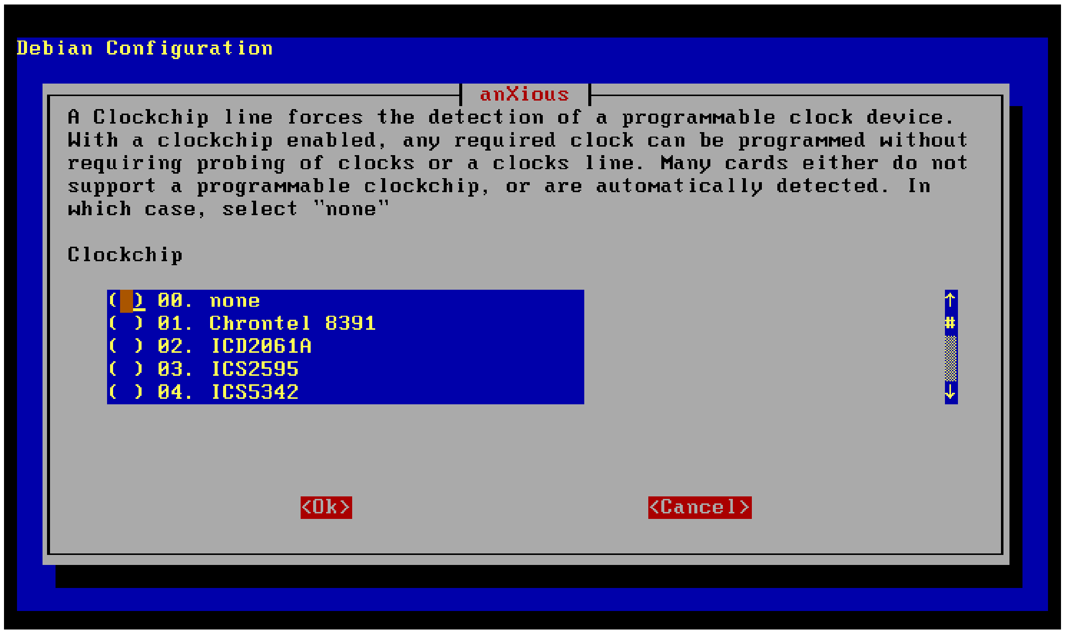


```
Debian Configuration

                              anXious
Please enter an identifier for your video card. This can be any string.

Video Card Identifier

My Video Card_
```

Screen 87: Video Card Name Specification

Replace the default string with one which identifies the video card being configured. Press **ENTER** to accept this value and move on to the next screen.

Next is the choice of clockchip.



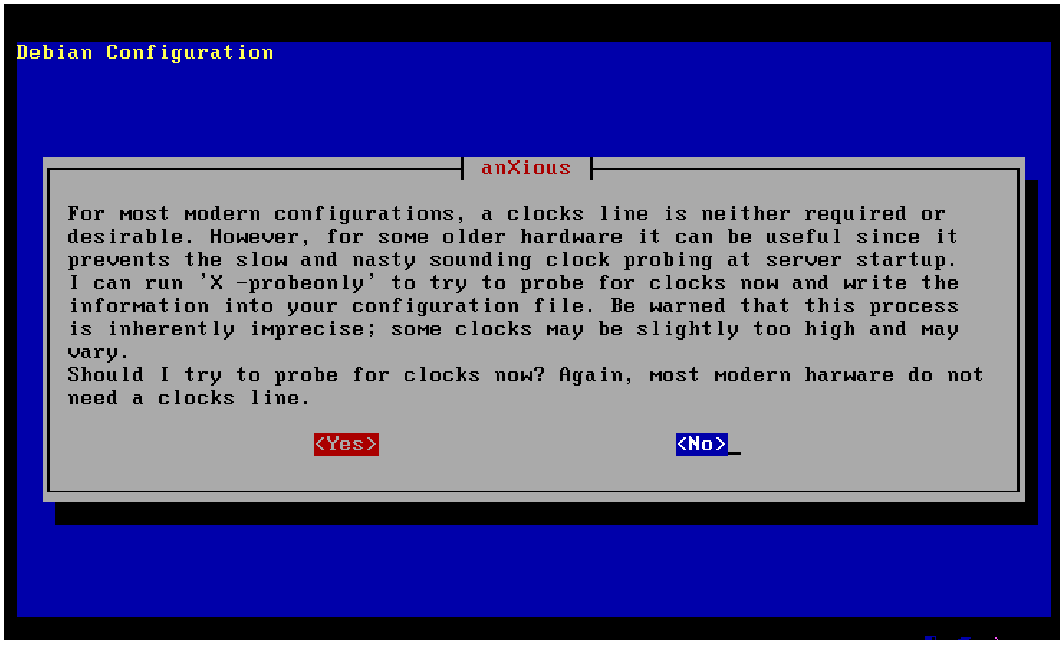
```
Debian Configuration

anXious

A Clockchip line forces the detection of a programmable clock device.
With a clockchip enabled, any required clock can be programmed without
requiring probing of clocks or a clocks line. Many cards either do not
support a programmable clockchip, or are automatically detected. In
which case, select "none"

Clockchip

( ) 00. none
( ) 01. Chrontel 8391
( ) 02. ICD2061A
( ) 03. ICS2595
( ) 04. ICS5342
```

Screen 88: Clock Chip Specification

Unless you know specifically that your card has and uses a particular clockchip, the most likely response here is none. Otherwise scroll down to the correct value and press **space** to select that item. The following items are available in the non-visible portion of the list: ICS5341, S3 GenDac 86C708 or ICS5300, S3 SDAC 86C716, STD 1703, Sierra SC11412, TI 3025, TI 3026, and IBM RGB 51x/52x.

Whatever your choice, once you have made it, **TAB** to the <**Ok**> button and press **ENTER** to accept the item chosen and move to the next screen.

Next is the request to probe for clock information.

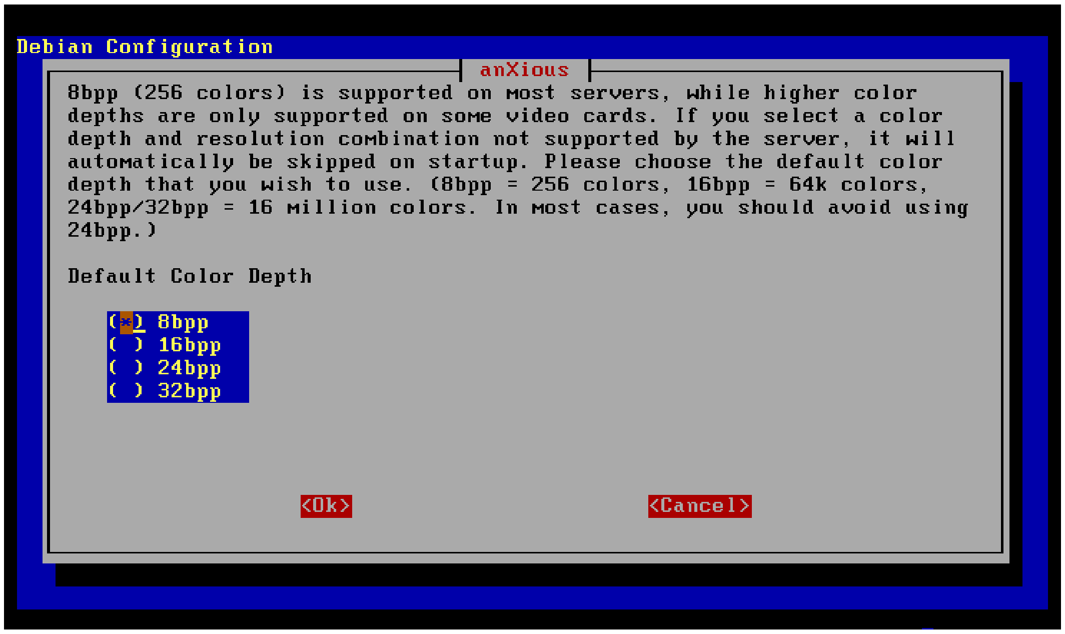


```
Debian Configuration

                              anXious

 For most modern configurations, a clocks line is neither required or
 desirable. However, for some older hardware it can be useful since it
 prevents the slow and nasty sounding clock probing at server startup.
 I can run 'X -probeonly' to try to probe for clocks now and write the
 information into your configuration file. Be warned that this process
 is inherently imprecise; some clocks may be slightly too high and may
 vary.
 Should I try to probe for clocks now? Again, most modern harware do not
 need a clocks line.

                  <Yes>                          <No>_
```

Screen 89: Probe for Clock Information?

Unless you are certain that your graphics card will need this information, accept the default value of <**No**> by simply pressing **ENTER**. If you wish to probe for clock information **TAB** to the <**Yes**> button and press **ENTER**.

Here we have the color depth selection screen.



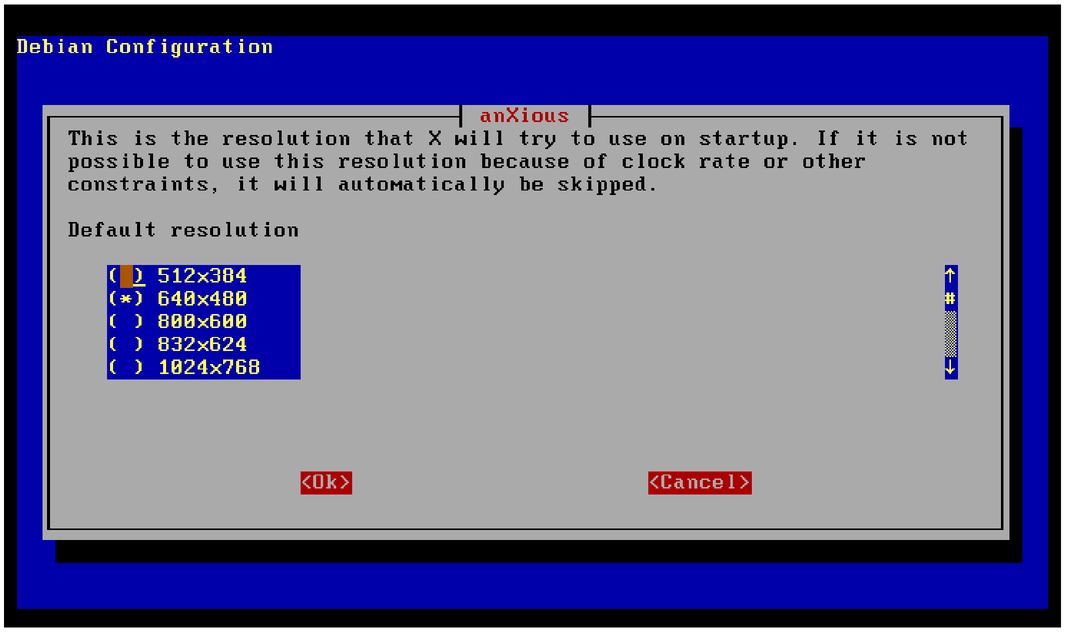
```
Debian Configuration
                                  anXious
  8bpp (256 colors) is supported on most servers, while higher color
  depths are only supported on some video cards. If you select a color
  depth and resolution combination not supported by the server, it will
  automatically be skipped on startup. Please choose the default color
  depth that you wish to use. (8bpp = 256 colors, 16bpp = 64k colors,
  24bpp/32bpp = 16 million colors. In most cases, you should avoid using
  24bpp.)

  Default Color Depth

     (*) 8bpp
     ( ) 16bpp
     ( ) 24bpp
     ( ) 32bpp


                   <Ok>                          <Cancel>
```

Screen 90: Color Depth Selection

8bpp is already checked as the default, so unless you need a different color depth, just **TAB** to the <**Ok**> button and press **ENTER**.

All of the values available are visible in the window, so just use the arrow keys to move to the item on the list you desire, and press **space** to select that item. Then **TAB** to the <**Ok**> button and press **ENTER**.

Use this screen to choose the default resolution.

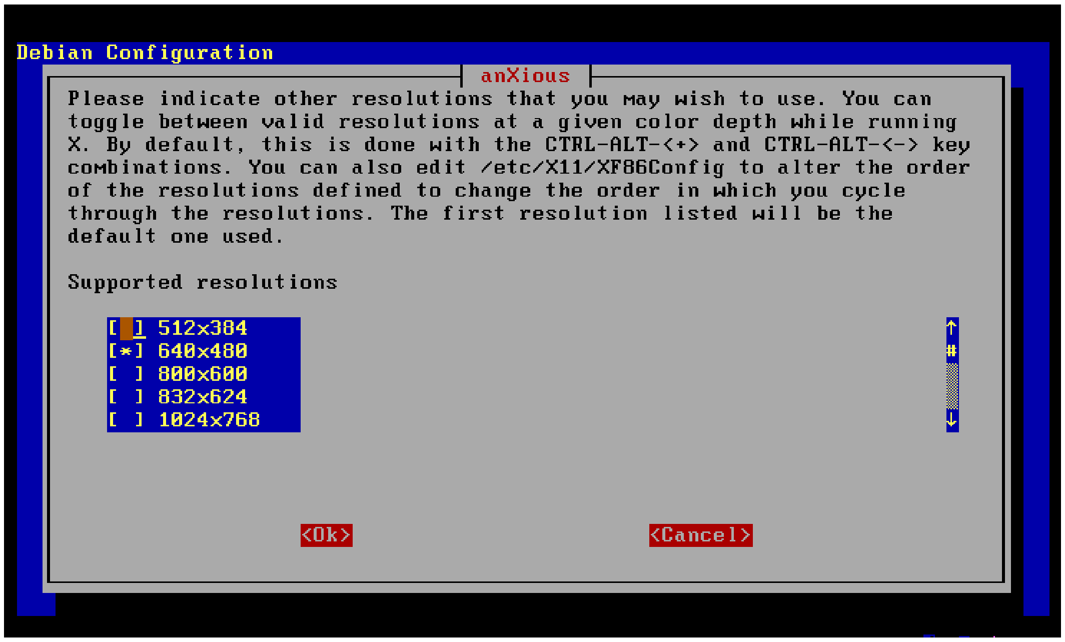


```
Debian Configuration

anXious
This is the resolution that X will try to use on startup. If it is not
possible to use this resolution because of clock rate or other
constraints, it will automatically be skipped.

Default resolution

( ) 512x384
(*) 640x480
( ) 800x600
( ) 832x624
( ) 1024x768

<Ok>                     <Cancel>
```

Screen 91: Default Resolution Selection

Your card may support several different resolutions. Here is where you decide which of these available resolutions will be used upon initial start-up of the X server. In addition to those values visible on the screen the values: 1153x864, 1280x960, and 1600x1200, are also available if you scroll down the window with the arrow keys or the scroll bar on the left. When you have selected the item you desire, by pressing the space bar, press **TAB** to move to the <**Ok**> button and press **ENTER** to move on to the next screen.

This screen looks just like the last one!



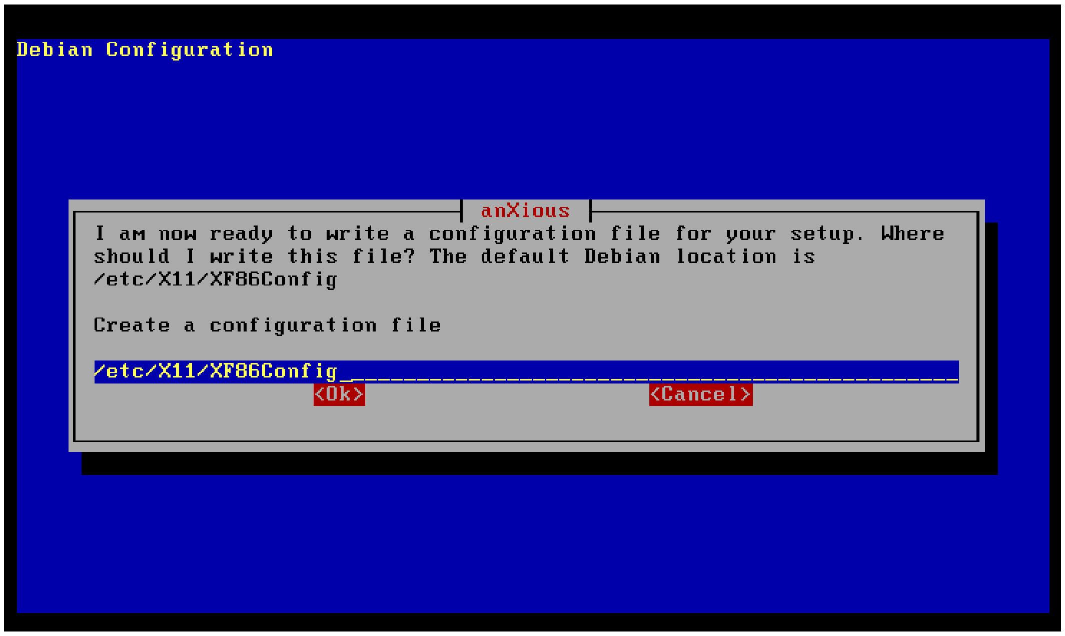
```
Debian Configuration
                              anXious
Please indicate other resolutions that you may wish to use. You can
toggle between valid resolutions at a given color depth while running
X. By default, this is done with the CTRL-ALT-<+> and CTRL-ALT-<-> key
combinations. You can also edit /etc/X11/XF86Config to alter the order
of the resolutions defined to change the order in which you cycle
through the resolutions. The first resolution listed will be the
default one used.

Supported resolutions

   [ ] 512x384
   [*] 640x480
   [ ] 800x600
   [ ] 832x624
   [ ] 1024x768

                 <Ok>                        <Cancel>
```

Screen 92: Usable Resolutions Selection

While it looks the same, it has a different purpose. Here you can choose as many different items as your monitor and video card will support. Each of these resolutions will then be available from the server. Pressing the **ALT** key at the same time as the control key, and either the plus or the minus key will cycle through these resolutions modes.

Use the arrow keys or the scroll bar on the left of the screen to move up and down the list. Select those resolutions to include by pressing the space bar. When all desired items have been selected, **TAB** to the <**Ok**> button and press **ENTER** to continue to the next screen.

Next we may choose the location of the configuration file.

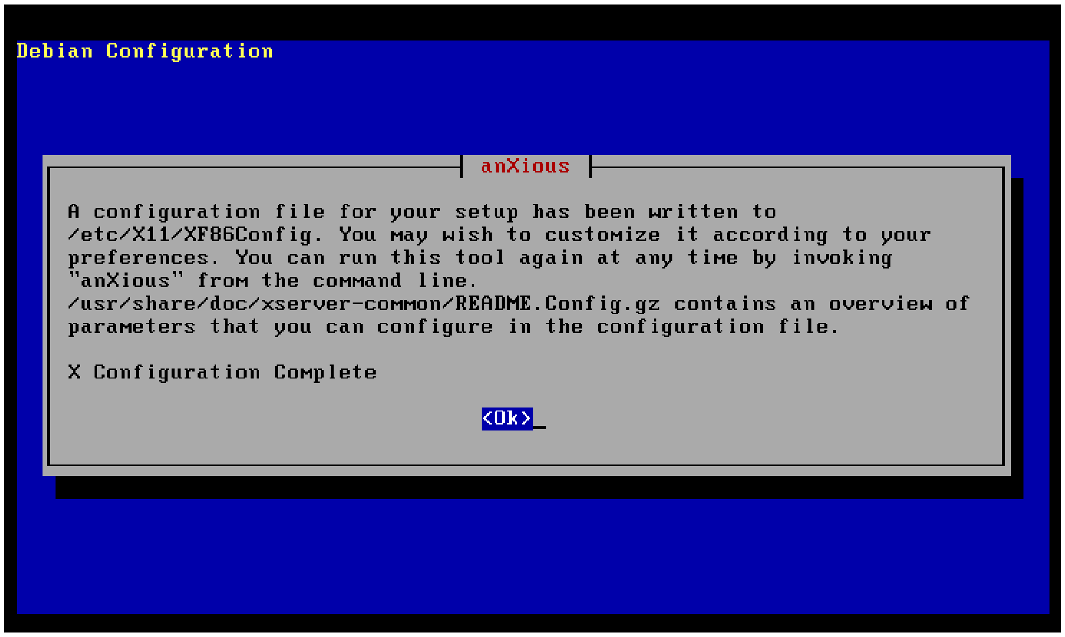


```
I am now ready to write a configuration file for your setup. Where
should I write this file? The default Debian location is
/etc/X11/XF86Config

Create a configuration file

/etc/X11/XF86Config_
```

Screen 93: Location of Config

X servers use this file to configure themselves at start-up. Unless you have some compelling reason to move the file somewhere else, the default value provided in the field is the correct place to put this file. All you need do is press **ENTER** to accept this value.

The next screen is an information screen, telling you that the X configuration is complete and giving some additional sources of information about how to configure your X servers.



```
A configuration file for your setup has been written to
/etc/X11/XF86Config. You may wish to customize it according to your
preferences. You can run this tool again at any time by invoking
"anXious" from the command line.
/usr/share/doc/xserver-common/README.Config.gz contains an overview of
parameters that you can configure in the configuration file.

X Configuration Complete
```

Screen 94: anXious Conclusion

You have no choice here but to press <**Ok**>. Do so by pressing **ENTER** and the X configuration is complete.

Apt Package Installation

If you did not choose to install X, then the previous **anXious** screens never happened. Either way the next activity is the installation of packages. After some clicks and whirs, **apt-get** will check for dependencies caused by the choices you made from the task selection screen. After assuring itself that all the packages it wants are in the archives it will list the ones it will be installing, followed by a synopsis of the results of its analysis. The list of packages to install may very well run off the top of the screen too fast to be read. Don't worry, dpkg can tell you exactly which package got installed, once the smoke clears.

On a typical installation, the synopsis lines might look like:

```
1 packages upgraded, 342 newly installed, 1 to remove and 0 not upgraded.
Need to get 0B/161MB of archives.  After unpacking 408MB will be used.
Do you want to continue?  [Y/n]
```

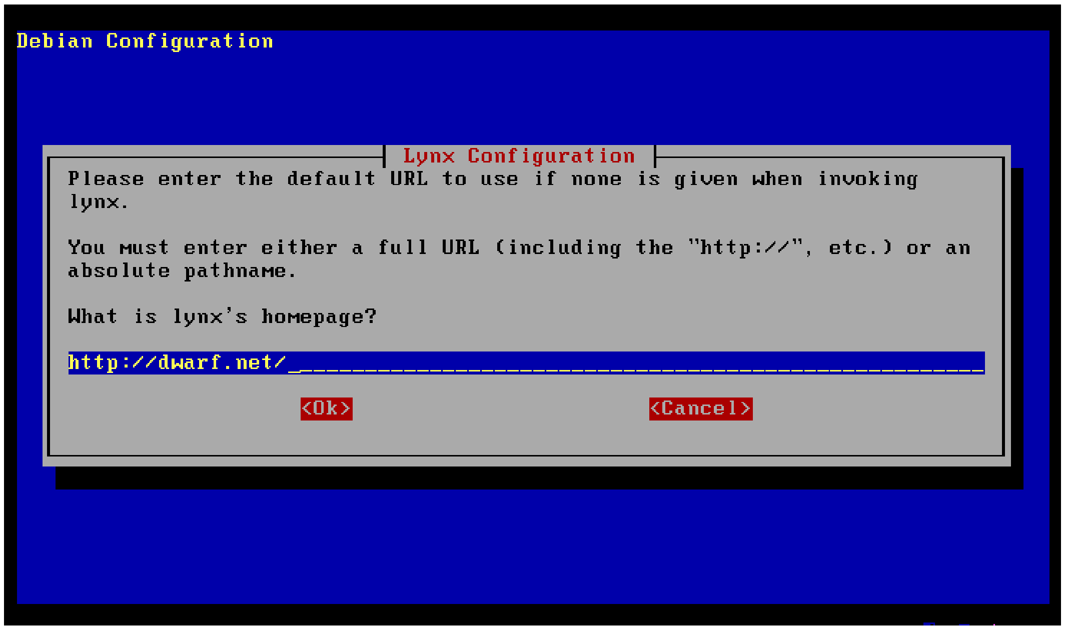
Pressing **ENTER** causes the installation to begin.

Since you removed the CDs after you registered them with apt-get there will be no CD to install packages from. **apt-get** will prompt you to put the correct CD into the drive and press **ENTER**. Read the label presented here very carefully. While often the first CD is needed first, sometimes, depending on the packages chosen, a different CD will be needed. Load the CD with the corresponding label into the CD-ROM drive and then press **ENTER**.

apt-get will scan the packages on the CD and then, most likely, drop back into configuration mode. Many packages now support the use of **debconf**, which allows configuration information to be gathered before it is needed by the package installation. Once you have answered all the questions in this phase, you might expect the rest of the installation to proceed without your assistance.

Only one example will be presented here. The number of additional configuration screens you will actually receive depends upon which packages you choose to install.

If **lynx** is being installed you will get the following screen.



```
Debian Configuration

Lynx Configuration
Please enter the default URL to use if none is given when invoking
lynx.

You must enter either a full URL (including the "http://", etc.) or an
absolute pathname.

What is lynx's homepage?

http://dwarf.net/_

<Ok>                            <Cancel>
```

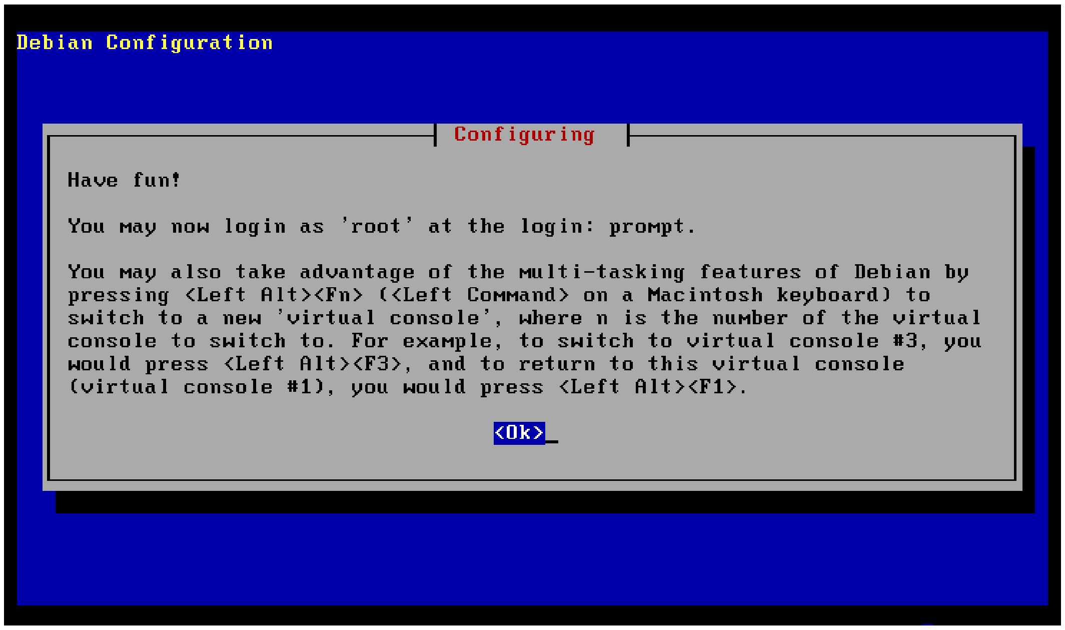
Screen 95: Lynx Configuration

Replace the default string with the URL of your desired home page. This can also be the absolute pathname of an HTML document on your local machine. Either way you want it, after typing the desired string simply press **ENTER** to accept the value and move on to the next configuration screen.

If a configuration screen is presented that you simply do not understand, either enter blank data, or, when provided, ask for no configuration at this time. All packages can be configured later, once you understand the necessary information required for their proper configuration.

Once all the configuration screens have been presented and completed, **apt-get** will proceed with the installation of the chosen packages. This will take some time, and will still require periodic intervention, telling those packages that don't yet use **debconf** what they need to know to complete their installation.

On those rare occasions **apt-get** will sometimes fail to install all of the packages correctly. This is most likely caused by an inconsequential error in the postinstall script, leaving an error code around for recovery by **apt-get**. Like **dselect** this requires an additional pass. When this happens, the offer to re-run the install will be made, so it is a bit more automatic than previous tools. Once all the packages have been installed the following information screen is displayed:



```
Have fun!

You may now login as 'root' at the login: prompt.

You may also take advantage of the multi-tasking features of Debian by
pressing <Left Alt><Fn> (<Left Command> on a Macintosh keyboard) to
switch to a new 'virtual console', where n is the number of the virtual
console to switch to. For example, to switch to virtual console #3, you
would press <Left Alt><F3>, and to return to this virtual console
(virtual console #1), you would press <Left Alt><F1>.
```

Screen 96: Installation Complete

Once **apt-get** completes the installation your new system is ready to use. You can add more packages in the future by using the command:

```
apt-get install <package name>
```

and that package will be installed from the proper CD.

You are now ready to begin using and hopefully enjoying your new **Debian GNU/Linux** operating system.

Chapter 4

Basic System Administration

Introduction

By now, if you have followed the directions provided so far, you should have a Debian GNU/Linux system up and running. The next most obvious question is: What do I do with it now that I have it running?

The answer is: First you must decide what the system is going to be used for.

In many cases, even a "private" workstation will have more than one user account active on the machine. Even if the machine is not connected to the Internet or any local net, there will be the need for more than one user account on the machine from time to time, so there is good reason to understand how to manage such accounts.

Managing many accounts

Your new Debian system was installed with a root account and possibly one user account of your choice. Even if you are the only one who will ever use the system, you will find that having multiple accounts can be useful at times, if only to help keep different projects separate. If the system is to be used by more than one person, then the following information is of more than passing interest.

Adding a user

The Debian system provides the adduser program as the primary tool for managing both user and system accounts. The simplest way to add the new user **fred**, is the command:

```
adduser fred
```

adduser will choose the next available **uid** (user ID) for **fred**, create a *user* and *group* with that *id*, create a home directory named fred, copy */etc/skel* into */home/fred*, and then ask for a new password for **fred**. While there are options that allow these items to be specified on the command line, this simple command is the best way to integrate a new user into the system.

If you use the **--disabled-password** option, the account will be created, but it will be disabled until a password is assigned to **fred**. In this case, *adduser* will not request a password at this time, and the *passwd* program must be used later to activate the account. This allows you to set up all the users in a class, and then assign them passwords as they come to you for them. This keeps the account from being an entry point into the system for someone who finds the “master” list. Once a password has been assigned to the user, they should be encouraged to change their password immediately making the “assigned” password useless to an intruder.

Finally, *adduser* will ask for the information to be presented when this account is "fingered" by someone. *finger* is a program that searches (even over the net when a different domain is specified) for the account specified, and returns the personal information found for that account. *adduser* inserts the data into the entry in */etc/passwd* that it creates for **fred**. So, now if you *finger fred* you will see something like the following:

```
Login:  fred Name:  Fred Smith
Directory:  /home/fred Shell:  /bin/bash
Office:  101, 555-1212 Home Phone:  555-5555
Never logged in.
No mail.
```

Removing a user

Just as you need to be able to add accounts to the system, it is also useful to be able to remove accounts that are no longer needed on the system. Debian provides the program *userdel* to remove a user account from */etc/passwd.* With the **-r** option, *userdel* will also remove the home directory for the account and all its contents. So, to remove **fred** from the system and all the files from */home/fred* use the command:

```
userdel -r fred
```

The entry in */etc/group* is not removed, and neither are any files on the system that are owned by the removed account. When the **uid** gets reassigned to another account, that account will end up owning those files. They should be removed by hand, or have their ownership changed, which will be described later in this chapter. Removing the entry from */etc/group* will require **root** privilege and the editor of your choice. Simply remove the line containing the account name, in this case **fred**.

Managing passwords

People forget things, like their password. It's human nature. So any system that is used by humans must have a way of fixing the problem, but it must work in a secure fashion. On a Linux system, like the Debian system you have installed, passwords are kept in an encrypted form that is not easy to break. The only recourse for the "loss" of a password is to create a new one for the account, and the only account that can do this is the **root** account. The program that does this is called *passwd.* Any user account can use this program to change the password for that account, but only **root** can change the password on another account.

To change the password for **fred** enter the following command as **root**:

```
passwd fred
```

Unlike when a user executes *passwd*, when **root** executes it, there is no request for the old password. This allows the password to be changed when the old password is not known, but it will only work for the **root** account. *passwd* will ask for the new password, and then ask for it again to verify correctness. Once the new account has been given a password by **root**, the individual who will use that account should use that password to log into the system and then use *passwd* to change the password to something known to no one else.

In the case of **fred**, suppose the initial password *#123fred* is assigned the account. Fred might then use this password to log into the system. Once the shell prompt appears he only needs to enter:

```
passwd
```

The *passwd* program first asks for the old password. Fred then enters:

```
#123fred
```

passwd does not echo the characters typed at the prompt. This would provide several ways to "steal" your password, including just looking over your shoulder as you enter it.

Once *passwd* verifies that the password is correct, the new password is requested. When it has been entered the new password will be requested again. If you can type the same password twice, it is assumed that you can type it again whenever you need to. This will fail if the two passwords don't match. This keeps the stray character from becoming part of the password. Sometimes when you press the key you wish to press you also press another by mistake. Since these keys are not echoed to the screen, you get no feedback that this has occurred. By asking for the password twice, these mistakes are filtered out. As a result, you may sometimes be asked to try again, even when you thought you got it right. No matter how clumsy you may be, *passwd* will let you keep trying until you get it right.

Once Fred types his new secret password twice correctly, *passwd* makes the entries that make that password the new one for this account. The password will not change unless Fred, or someone with **root** privilege, changes it. This makes the security of **root** privileges very important on a system with many users just like Fred. Each of these users expect their files to be secure and unaltered by another user account, relying on those with **root** privileges to not abuse those privileges. This trust between the system administrator (usually the only one with **root** access to the system) and the users is a relationship in which some are very good and some can be overbearing. However, the best administrator can have a password stolen or otherwise compromised. The person who would do this may not worry about the other people's files, making all precautions useless.

Managing Root Access

Aside from logging in as **root**, there are two different ways to grant a user account the superuser privileges of the **root** account. Debian delivers the two programs *su* and *sudo* as ways to provide **root** access for non-root accounts.

su

The *su* program changes the effective **uid** of the account to that of another account. In order to successfully change the **uid**, the password for that account must be entered. For this reason, in order to use *su* to gain **root** access, you must know the password of the **root** account. This means for each person who needs **root** access with *su*, yet another person must know (and remember) the password for the **root** account.

Now, if a user has their account password compromised, only that account is open for illegitimate access. When the **root** password is compromised, the whole system becomes vulnerable to attack. For this reason it is advisable to take more care with the **root** password than with anything else. Giving that password out to more than the System Administrator is begging for disaster.

So, what good is *su*? The safe usage of *su* is to use it to change from one user account to another. You, of course, must know the password of the account you wish to move to. This is sort of like logging into the other account, but with *su* the environment of the old account is carried over into the new account session. This is very useful if you have your user account set up with particular default environment conditions, and wish to work under an account that is set up different.

sudo

To allow more control over the privileges available to normal users, Debian provides the *sudo* command. This utility allows a specified user to execute commands as the **root** account. The users who may use *sudo*, and the programs they may have **root** privilege for, are specified in the file */etc/sudoers*. The amount of access that can be given to a particular user account ranges from ALL, which is equivalent to *su*, down to allowing only a single command to be executed as **root**.

Beside the finer granularity of the permissions provided, *sudo* also has the advantage of not requiring the **root** password. To invoke *sudo*, the user only supplies their own password, leaving account security in the hands of the person responsible for that account. So, to give **fred** the ability to shut down the system, the entry in */etc/sudoers* would look like:

```
fred /sbin/shutdown -[rh] now
```

In this way, a large variety of capabilities can be given to trusted users without compromising the **root** password. Since *sudo* can be used to give a user full **root** privilege, it is a potential avenue for a security breach if that user account password is compromised. Hence, it is advised that only those commands that absolutely must be provided are entered into */etc/sudoers*, providing maximum functionality, with minimal security risk.

Obviously security is of varying importance, depending upon what the system is being used to accomplish. This does not mean that there are times when system security can be ignored. Even a system that is completely disconnected from a network can be subject to a security breach. Most of the security measures of interest have to do with protecting users from making mistakes with files that do not belong to them. This keeps Joe from accidentally deleting any of Fred's files, and both of them from accidentally deleting important system files.

File Protection by Owner

It has been suggested that the system protects users from interfering with files that aren't theirs. This is done using the permission bits and ownership information that is kept by the system for every file in the file system. These permission bits determine who may read, write, and execute the file. These permissions are divided into owner, group, and others. If only the owner permission bits are set, then only the owner may perform those functions on the file. Debian creates user accounts, each with their own group, and defaults the **umask** value to 002, which causes all files created, to have default permissions that match the user only, thus protecting any files created by this user account from inadvertent tampering.

Only the owner of a file, or **root**, may change the ownership of a file. This is done with the program *chown* which modifies the ownership to be that of another user account. If the file is permitted for read by others, then when Joe copies such a file from one of Fred's publicly readable directories, the file copied into Joe's directory will now be owned by Joe's user account. This give Fred the ability to let others use his material without letting them change any of the original files. To prevent such "theft" of a file, for material that is private, the permissions on the file need to be set to restrict read permission for others.

The command *chmod* is used to change the permissions of files in the file system. Only the owner (and **root**) may change the permissions on a file. If Fred wants to keep the file **private.txt** from being read by others, he would enter the following command:

```
chmod go-wxr private.txt
```

This removes all permissions from either group or others, to either write, execute, or read the file **private.txt**. Now only the **fred** account can read, write, or execute the file **private.txt**.

File Protection by Group

A second level of security, that allows more than one user account to access a file, is the **group** permission. As previously pointed out, each user account is assigned their own **group**, but that is not the only use for **group** permissions.

If there is a group of user accounts, all of which are working on the same document project, it is possible to create a **group**, such as **doc**, and assign each user in the project to that **group**. This new **group** is added to the */etc/group* file by the *adduser* program. To add the group *doc* the command would look like:

```
adduser --group doc
```

While it is also possible to declare a specific ID for this **group**, it is not generally recommended. *adduser* will assign the next available ID to the *group*, and in most cases, this is just what is needed. There are a number of default groups already provided in */etc/group* for those programs that are expecting particular **gid** (groud ID) numeric values, so there should never be the need for a specific **gid** to be assigned to any particular **group**.

Once the **group** has been created, *adduser* is again used to assign each user in the documentation project to this new **group**. To add the **fred** account to this new **group**, **root** would execute the following command:

```
adduser fred doc
```

Now any documents that Fred creates may be assigned to the **group doc** and have its **group** permission bits set so that others in the documentation group can work on that file as well. To do this for the file **private.txt** Fred would first issue the command:

```
chown fred:doc private.txt
```

and then set the permission bits with the command:

```
chmod g=rwx private.txt
```

Now any member of the **group doc** can work on this file as well, but all others are still restricted from even looking at the file. Thus the permissions bits, when combined with file ownership and **group** information, can provide a great deal of flexibility while still maintaining a level of security at the same time.

Mounting and Unmounting File Systems

Everything your Debian system will ever use is found on one file system or another. When the system is first initialized, the kernel uses the file */etc/fstab* to mount any additional file systems needed. The root file system is initially mounted read only by the kernel so it can read */etc/fstab*. One of the things that this system file tells *init*, is which file systems need to be checked, and in what order. Normally the root file system is checked first, followed by any other file systems that need checking. *init* uses *fsck* to check the partitions it finds listed in */etc/fstab*. After the file system check is completed, the kernel re-mounts the root file system read/write and proceeds to mount the other devices it finds in */etc/fstab*.

If you want a device permanently mounted on the system, simply make an entry in */etc/fstab* specifying the mounting parameters and then execute the command:

```
mount -a
```

which causes all of the devices listed in */etc/fstab* to be mounted.

Additional devices can be mounted and unmounted as needed using the *mount* and *umount* commands. To mount an ATAPI/IDE CD-ROM on */dev/hdd* use the command:

```
mount -t iso9660 /dev/cdrom /cdrom
```

The device can be unmounted by either specifying the device (*/dev/cdrom*) or the mount point (*/cdrom*). So either of the following commands will unmount the CD-ROM drive:

```
umount /dev/cdrom
umount /cdrom
```

Monitoring System Activity

One of the advantages of a multiuser, multiprocess, operating system like Linux is the ability to run many programs simultaneously, as well as handling various users, and keep track of both at the same time. Under these circumstances it is necessary to be able to monitor the system for things like runaway programs, and memory hogs, so that these problems can be dealt with effectively.

Process Status

Executing the command *ps* returns a list of processes being run by the user account that invokes the command. This list includes the **process id**, the **command** that started the process, and the **terminal type** and **number**. Any process in the list may be terminated using the *kill* command with the associated **process id** obtained from the list generated by *ps*.

To see a list of all processes use the *ps -a* command. Unless done by the **root** account, only the listed processes which the user started may be killed by that user. The **root** account can *kill* any process on the system with *kill -9 <process id>*.

While the *ps* command includes a field for the **status** of the process, the only process in a R(un) status is the process ps itself. All other processes are marked S(uspended). Another tool is required to get a real-time picture of what these processes are doing.

top CPU Processes

The system utility *top* provides a continuous, real-time look at the system's consumption of memory and CPU resources. It lists the most consumptive process first, so finding that process that is gobbling machine resources is relatively easy. *top* also displays: the total operations time for the system since the last reboot; load averages; process counts for various states; the percentage of CPU time broken down between **user**, **system**, **nice**, and **idle**; memory and swap space usage; as well as the list of the processes using the largest amount of the machine resources.

fuser: Who has it?

Sometimes, when an attempt is made to *umount* a device, it is still in use by another account. *umount* will report "device busy" under these circumstances. So, "Who has it?", is the question which must be answered before that account can be contacted and asked to relinquish the device. *fuser* is the answer to this question.

Assume that a CD-ROM device has been mounted and now the material has been transferred, or installed, and it is time to unmount the device, but the *umount* command fails because the device is still busy. Very often the System Administrator has forgotten that some virtual console has the current working path somewhere on the CD. When memory doesn't resolve the issue it is possibly some other process and user account that are still using the device. The command:

```
fuser -muv /cdrom
```

will list each account using the device mounted on */cdrom*, along with the PID of the command that is currently executing in that device path. Once the account is identified, a message can be sent to the session indicating the System Administrator's desire to *mount* something else on that device.

Who is logged on?

The command *who* will return a list of all the accounts currently logged into the system, the terminal device being used, and the time of login. With the *fuser* command we might learn that **fred** is the account using the */cdrom* path. The *who* command would then tell us **fred** is logged into terminal **ttyS3**. You can then send **fred** a message by typing the following:

```
write fred ttyS3
How long do you need to use the /cdrom mount point?
I would like to mount a new CD.
<ctrl-D>
```

When the first line is entered, something like the following is printed at Fred's terminal:

```
Message from root@your.machine.net on ttyp1 at 14:52
How long do you need to use the /cdrom mount point?
I would like to mount a new CD.
```

Fred may then send a message back telling of his intention to comply.

Where is the printer?

The user who sends output to the printer may wish to know where the physical printer is located in order to retrieve their output. The System Administrator should be able to point you to the physical printer, if that is the problem.

But, how do I actually get my output to the printer? Debian uses *lpr* to place print jobs in a queue so that the printer daemon *lpd* can print it when the printer becomes available. The print behavior is defined in */etc/printcap*, and Debian installs this file with the default printer configured as a generic dot-matrix printer. This file declares the port the printer is connected to, among other things, and can be used to send plain ASCII text to most printers. If

Fred wishes to see a printed listing of the file *private.txt* he would enter the command:

```
lpr private.txt
```

If this is a busy system, Fred will not expect his job to print right away. He can look at the position of his job in the queue using *lpq* to list the jobs waiting in the print queue. This command lists the jobs in the queue, in the order they will be printed. A **job number** is associated with each entry in the queue. This number can be passed to *lprm* if it becomes necessary to remove a print job from the queue. Fred can, of course, only remove his own jobs from the queue, while the **root** account can remove any job found there.

If the file to be printed is more complex than simple ASCII text, then a filter will need to be applied in the */etc/printcap* file. Packages like **magicfilter** provide a wide range of filters for most printers. These filters will allow you to print PostScript files to a non-PostScript printer. In combination with programs like *dvips* (which converts a dvi file to PostScript output) the output of almost any Linux file format can be converted and sent to the printer.

Finding knowledge

In much of the above discussion there is only a hint of how some of the useful programs work for you to make using a Linux system possible. The Debian system comes with much “on-line” documentation. It is Debian Policy that every program has a **man** page. In addition, many packages also provide **info** pages.

How does this help when you may not even know the command name? Linux also provides the *apropos* command. If you give *apropos* a word, it will search certain portions of the **man** pages looking for that word. For instance the command:

```
apropos printer
```

will produce output something like:

```
banner (6) - print large banner on printer
lpc (8) - line printer control program
lpd (8) - liner printer spooler daemon
lprm (1) - remove jobs from the line printer queue
pac (8) - printer/plotter accounting information
print (3ncurses) - ship binary data to printer
printcap (5) - printer capability data base
```

Thus *apropos* can lead you to the **man** page entry for the topic of interest. In the previous list the command *lpc* looks interesting, and we didn’t discuss it when talking about printers. This program is mostly of interest to the System Administrator, but that could be you, so take a look at the **man** page by entering the command:

```
man lpc
```

The *man* program will format text to your screen through the **pager** assigned to that task (the default is *more*). *man* recognizes the **PAGER** environment variable, and will use the **pager** defined by this variable. Many people prefer *less* to *more* as far as pagers are concerned, they think *less* is better than *more*. (Dwarf agrees: *less* is more) If you only wish to set this for your own account, add the following lines to */.bash_profile*:

```
PAGER=less
export PAGER
```

and the next time you login *man* will use *less* as the pager.

Additional info

All GNU software provides its documentation in the info format. This is normally a different format from the **man** page, and uses a "menu" feature to allow pages of related material to be "linked" together, allowing cross-references to be placed with a page to provide broader discussion of the various topics. The *info* command will also display the **man** page if there is no index to the keyword you supply. For instance:

```
info lpc
```

will simply display the **man** page for the *lpc* program in much the same way that *man* did earlier. On the other hand, if the command had been:

```
info diff
```

The format of the **info** page for *diff* will be printed to the screen. **q** will exit *info*, and the *man* command with **diff** as the argument, will display the **man** page for *diff*, which you will see is quite different. Some folks like the terse structure found in **man** pages, while others like the expansive discussion provided by the **info** pages. Debian provides for both types of individual by providing both whenever possible.

Finding packages

Say, for example, you just heard about this really cool program, and you are pretty certain that it is provided in the Debian distribution, but a **man** or **info** attempt fails to provide any information. When you ask *dpkg* about the package, it says it isn't installed! So now what do you do?

Debian provides two files in its archives which are useful for this purpose: *Packages*, and *Contents*. The *Packages* file is usually located in the binary part of the archives, while the *Contents* file is usually in the outer directory of the main portion of the distribution. If you know the name of the package, then you can use *grep* to print its entry in the *Packages* file. However if you only know the name of a program that you are interested in, like *fuser* mentioned above, then the command:

```
grep fuser Contents
```

executed in the directory containing the *Contents* file, will tell us that */bin/fuser* is found in the package *base/psmisc*, in other words the **psmisc** package is located in the section **base**, so it is most likely already installed on the system (**base** packages are installed first). Checking with *dpkg -s psmisc* will indicate the status of the package, and provide information about the maintainer as well as a description of the package. If the package is not installed then the information from *dpkg* will be quite useless, as it can only tell you that the package is not installed. Once you know the package name, the *Packages* file can be *grepped* for the necessary information. So, look for programs in the *Contents* file, and their packages in the *Packages* file.

Searching the file system

When you work on a program source tree, or even just a collection of documents, the editor you use may well make a backup copy of the file, so you can recover to the last good version if the changes turn out to be really bad. When you are finished with all the edits, it is useful to be able to remove all those backup files. For instance *joe* adds the character to the end of the file name, so the command:

```
rm ‘find ./ -name ‘‘* ’’‘
```

will search all of the directories in the current path for file with a name that ends in and provide those names as a list of files for the *rm* command. This is a very useful method of using one command to produce output that is acted upon by another command. In this case the accent character (‘) is used to bracket the *find* command, causing it to be executed, with the output from the *find* command being used as the argument list for the *rm* command. Thus, each file found to end with the character is removed by the *rm* command.

A simpler use of the *find* command is to locate a configuration file on the system. Thus, the command:

```
find printcap
```

will list:

```
/etc/printcap
```

as the location for that file. If you forgot where you saved that important file, *find* will save the day. For more information on the *find* command, or any of the other commands discussed above, see your friendly **man** and **info** pages. With a little investigation you can learn all about the capabilities of your new Debian GNU/Linux system.

Building a Custom Kernel

Hardware

The current 2.0.x Linux Kernels will still run on 386 based Intel PCs, as well as all the subsequent CPUs of the 80x86 family from Intel. There are current, functioning ports of the kernel to Sparc and Alpha machines as well as the m68k port to the Atari and Amiga machines. With a proprietary micro-kernel there is even an m68k version for the MacIntosh machines. A wide range of SCSI, IDE, and special CD-ROM drivers are available with the "stock" kernel. As the kernel is the heart of a Linux system, it is also the fundamental infrastructure of any distribution. Debian is no different in this matter, and many times the flaws in the distribution can be traced back to flaws in the kernel being used.

Debian became a full ELF compliant system with the release of 1.1 and it has not changed the basic library infrastructure since then. There are a wide range of available kernels that will work. When the stock kernel does not work with the hardware configuration of the target machine, it becomes necessary to rebuild the kernel with a new configuration that better suits the hardware situation. When this can't be done with the current kernel, it is sometimes even useful to move to a newer, or older, version of the kernel. A standard Debian system (all packages with priority of Standard and higher) will have all of the development tools needed to perform the construction and installation of a custom compiled kernel.

The less memory the machine has, the more difficult it will be to compile a kernel. With as little as 8MB of memory, it is necessary to have at least a 20MB **swap** partition/file, but more memory will always improve this situation. Any error complaining about the exhaustion of virtual memory can be fixed by increasing the size of the **swap-space**.

Note: **swap-space** - All machines have a finite amount of Random Access Memory. Without a method for dealing with this limited memory the system would crash whenever available memory is exhausted. To resolve this issue, the kernel "swaps" memory to disk when not in use, to free memory for use by other processes. This requires a disk partition, or a file, that has been specially formatted to deal with memory swapping.

Building the Kernel

The Debian system provides access to the kernel in several ways. There is a separate **Kernel Image** package that will allow you to install a "pre-configured" kernel image on the current system. There is also a **Kernel Source** package that will allow the custom configuration of a kernel and build a **Kernel Image** package as its output.

There are no preconditions in a Debian system that require you to use these packages. The kernel for a Debian system can be built without these packages and will work just as well. A discussion of how to do this will be instructive to more than just the paranoid individual who needs direct control over such important matters as a kernel. The following section will first describe how to construct and install a kernel without using the Debian packages, followed by a description of the differences and advantages of using the Debianized versions. As there are trade-offs for either approach the choice of method will be dictated by local considerations.

Non-Debian Kernel Construction

Sometime before the advent of the 2.0 kernel the old method of building the kernel was abandoned. What was removed was the requirement that the kernel source reside in a particular location in the file system. Every time you installed a new kernel source it was necessary to remove and rebuild several **links** that pointed *gcc* and others to the proper header files and assembly routines needed to compile other programs as well as the kernel. At about the time that this practice ended, Debian began to distribute **Kernel Headers** that were stable and always found at these "predefined" locations. Thus satisfying the general need to compile programs with kernel header information. Since the new kernel *Make* files are now aware of the location of the kernel source tree (by doing a *pwd* early in the *make* process) that tree can be located anywhere.

Debian follows the standard on file system layout very closely, so you can be assured that anything that you install in */usr/local* will remain undamaged by any actions of the Debian Package System. The following procedure will allow the installation of kernel source; the compilation of that kernel source into an **image** and **modules**; and the installation of that **image** and those **modules** in the current system. Because the very few packages that depend on a particular kernel must interrogate the system to determine the running kernel, the registration with the packaging system of any given kernel is completely unnecessary.

- Create the directory */usr/local/source* if not present.
- Create the directory */usr/local/source/kernel-<version>*
- Change directory to */usr/local/source*
- Make a link between *linux* and *kernel-<version>*.
- *ln -s /usr/local/source/kernel<version> linux*
- Copy the *linux-<version>.tar.gz* file into */usr/local/source*.
- Untar the *linux-<version>.tar.gz* file

The steps above will create a kernel source tree in */usr/local/source*. This can, of course be any other place in the file system that seems appropriate, and need not be */usr/local/source*. This is only one of several safe places you could put this tree. Creating the target directory and the link to it from the *linux* directory allows for future kernel source trees to also install here. Setting up the link to the target before untarring the source files makes everything end up where it should, rather than having to rename directories after the fact. The need for all of this stems from the fact that the source **tar.gz** file unpacks itself into the local directory named "linux". This can also cause problems if there is already a full directory named linux. With the link, there is at least the chance of moving the link to a fresh directory before unpacking new kernel source over the old.

If there are no patches to be applied to the source you are ready to build a kernel. Begin by changing to the directory */usr/local/source/linux*. What gets done next depends on the state of the source tree and the desires of the builder. If the desire is that the build process should start from the "pristine" source, sometimes referred to as "distribution clean" then the first step is to execute *make mrproper*. From here the following steps are standard:

Configure the features of the kernel via:

```
make config
```

or

```
make menuconfig
```

or

```
make xconfig
```

then:

```
make dep
make clean
make zImage
```

or if the **image** is larger than 1MB

```
make bzImage
```

then:

```
make modules
make modules_install
```

The major caution during the configure phase is: Make sure that devices needed by the kernel, before the root file system is mounted, are always **built-in** and never made as **modules**. Any other features declared to be modularized in the configure menu can be built as **modules**. Once the configuration phase is complete, the rest of the steps should proceed with no errors. Use of *make zImage* will result in the kernel image being placed in *arch/i386/boot/zImage*. The normal place for this image is in */boot/vmlinuz<version>*, where it can be accessed using the link from */vmlinuz*. The kernel image must be moved to */boot* "by hand" and the new link created in */vmlinuz*. The system map file, *System.map*, found in */usr/local/source/linux* should also be placed into */boot* with the version number appended. Refreshing the link in */System.map* to point to this new file will complete the setup.

If you use **LILO** for loading the kernel there are several steps necessary for safe installation of the new kernel.

1. First copy the old kernel to */boot/vmlinuz.old*
2. Next copy the new kernel to */boot/vmlinuz*
3. Modify the link */vmlinuz* to point to the new kernel
4. Modify *lilo.conf* to provide another tag
5. Run **LILO**

Simply copying the new kernel over the old, will not result in the desired effect. Copying the old kernel to a separate location and copying the new kernel into the old slot also doesn't quite get the desired result, even though this is the correct first step.

Once the files have been moved, **LILO** still needs to be informed of the changes. When *lilo* is run, it reads its configuration information from the file */etc/lilo.conf*, so, to inform *LILO* of the change in kernels it is necessary to *edit/create* this file. If the file starts out like:

```
# /etc/lilo.conf - - single Linux installation.
  boot=/dev/hda1

root=/dev/hda1
compact
install=/boot/boot.b
map=/boot/map
vga=normal
delay=20
image=/vmlinux
root=/dev/hda1
label=Linux
read-only
```

then you need only add the following section:

```
image=/vmlinuz.old
root=/dev/hda1
label=Old
read-only
```

next, run *lilo* and the new kernel will be installed with access to the old kernel as a fall back.

Debian kernels are often built with all the SCSI low level drivers built in. This makes the uncompressed kernel greater than 1 Meg, requiring the use of **bzImage** instead of **zImage**. **bzImages** can sometimes cause problems for some hardware and for some loaders. *loadlin* has been known to have problems with these kernel images in the past, although most of those difficulties have already been fixed.

If *loadlin* is used as the boot loader, then none of the above actions will make the new kernel available. You must copy the **zImage** file to the file that is used on the **DOS** partition as the kernel **image** file. If you use *loadlin* to boot the file *linux* from your **DOS** partition, then you must copy the new kernel **image** into the file *linux* on the **DOS** partition. Also make sure that *make modules_install* has been run, and you are ready to reboot the new kernel.

If you boot from a floppy, then mount the floppy as a **DOS** disk, and copy the kernel image file from */arc/i386/boot/zImage* to the file *linux* on the floppy. This is the kernel image file on the boot floppy and replacing it with the new kernel image will allow the system to boot under the new kernel.

Debian Kernel Construction

The kernel for Debian comes prepackaged as a *kernel-image-<ver>_<ver>.deb* file.

Installing this with *dpkg* or *dselect* will deliver a pre-compiled kernel of that version to the system. In many cases this is perfectly adequate and results in a new functional kernel with little effort. When this fails to be adequate to the needs of the system there is a **kernel-source** package to help resolve the problems.

Installing the **kernel-source** package with either *dpkg* or *dselect* will result in the kernel source being unpacked into */usr/src/linux* via a linux symlink similar to that used in the */usr/local/source* example to link the *linux* directory to the correct *kernel-source-<version>*. There is also the file *.linux-versions* placed in */usr/src*. This file contains a list of all the kernel source and kernel header packages that have been installed on the system. The reason the version number is embedded into the package name is to allow more than one version of kernel source (or header or image) to reside on the system at the same time.

If you have installed all of the tools needed to build a kernel, in addition to the **kernel-source** package the **kernel-package** package must also be installed. This package contains the scripts needed to build the kernel and make image, header, and source packages. Once this package is installed the following results can be obtained when *make-kpkg* is executed from the kernel-source directory:

- make-kpkg build Will build a kernel.
- make-kpkg binary Will make all the kernel packages.
- make-kpkg kernel-image Will make only the kernel-image package.

If the intent is to construct a "custom" kernel, rather than just rebuild the delivered one, then *make config* (or one of its varieties) must be executed to re-configure the kernel to the new requirements. Once this has been done:

```
make-kpkg kernel-image
```

will construct a **kernel-image** package that can be installed using *dselect* or *dpkg*. This **image** package is only an advantage if **LILO** (or some other boot manager) is used. If *loadlin* is being used to boot the kernel, then it will be necessary to move the kernel image file by hand to its proper location.

Although the benefits may seem limited, use of the packaging system can make life much easier. With the kernel this is not as necessary as it is with other packages, so the decision is left to the System Administrator as to how this issue should be dealt with.

Modules

Sometime before the 2.0.x kernels, modularized drivers were first introduced into the kernel. This was a pretty shaky interface at first, and some drivers worked very well while others didn't work at all. By the 2.0 kernel the use of **modules** was a well understood method. While there are still occasional problems associated with modular drivers, this feature of the Linux kernel have become one of the stable workhorses of the present day operating system.

Advantages

- Smaller kernel image
- More versatility in installation environments
- Driver modification/testing without kernel reboot
- Reduction in memory usage.

Almost every **driver** can be built as a **module**. If the kernel was built with **module** support and **kerneld** enabled, then most modules will not need any special entries in the */etc/modules* file. *kerneld* will load them when they are needed and unload them when they are not being used. Some modules need to be loaded and initialized by the kernel before *kerneld* is started by *init.* There is a line in the provided */etc/modules* file, after the general comments, that has been commented out. This is the "auto" line. Removing the comment (# character) from the "auto" line, will tell the kernel to load those modules automatically, without the aid of *kerneld*.

Some device drivers, like the low level drivers for specific SCSI cards, must be installed explicitly. When a CD is mounted, the system (and *kerneld*) know that the modules that support the SCSI device must be loaded. The kernel has no knowledge of what card the machine has installed. This information is provided to the kernel in the */etc/modules* file. An Adaptec 1542 SCSI controller needs a line added to the modules file (anywhere after the "auto" line) that declares this driver. In this example the */etc/modules* file might look like:

```
# /etc/modules:  kernel modules to load at boot time.
#
# This file should contain the names of kernel modules that are
# to be loaded at boot time, one per line.  Comments begin with
# a '#', and everything on the line after them are ignored.
#
auto
aha1542
```

This file will cause all the necessary modules that support the controller to be loaded by *kerneld*, while the aha1542 driver is loaded at boot time and left loaded until reboot.

The device driver that accesses the root file system must be built into the kernel or the system will be unable to boot. If the device driver is a **module**, the kernel has no driver to read the root file system, as this is where it expects to find the **module** for that driver. This Catch 22 is, for instance, why the Debian installation kernel has the MS-DOS FAT driver built into the kernel, rather than available in the modules installation section. Because the rescue disk is an MS-DOS format disk the kernel must have the FAT file system drivers already installed before it can read the root file system from the floppy (**minix** is also built in because the *root.bin* image file is a **minix** file system).

Any devices the kernel will need before the root file system has been mounted, in order to get to the point of having a mounted root file system, must be built into the kernel. All other devices can be built as modules. The Debian installation kernel is built in this fashion, making it possible to fit the installation into complex hardware configurations where blanket support causes failures.

With the recent growth in the size of the kernel image this configuration allows the kernel to stay below the size limitations of some boot loaders and still maintain the highest level of functionality.

Many of the configuration problems associated with device drivers involve finding the correct set of options to pass when installing the driver.

Without modules each test would require rebooting the system in order to get the newest test options to take effect. With the **module** utilities supplied with a Debian system, a **module** can be unloaded with the *rmmod* command. The module's options can be edited in */etc/conf.modules* to reflect the new test values, and the **module** reinstalled with *insmod*, all without rebooting the system!

With the use of *kerneld*, **modules** can be loaded and unloaded as needed. When no access has been made to the VFAT file systems for awhile, kerneld will unload the module and free up the memory it used. If, by chance, there is the need to read that VFAT file system in the future, the request to the kernel will trigger the loading of the required **module** by *kerneld*. In this way the kernel only takes up as much memory as it actually needs to continue with operations while maintaining the capability of accessing a wide variety of devices and systems.

Disadvantage

- Broken or missing modules can cripple the system
- Provide opportunities for security violations
- Erratic operation of, and timing problems with, some devices

If a driver file in */lib/modules/<version>/<device-type>/*.o* is either corrupted or removed, the system can find itself without critical resources. This can be fatal for critical processes and possibly result in a system crash. Recovery is not always obvious, but with a **rescue disk** the system can be rebooted and the state of */etc/modules* and */lib/modules* can be investigated. It is not always obvious that the problem is with the modules. But, having a printed listing of the files in */etc/modules* (along with their file sizes) is a good tool to have in the recovery file folder when you start looking for solutions to problems.

Modules also provide a potential security risk. Using the "exploit of the day", a system cracker need only gain **root** access long enough to copy "his" version of the system PPP module to provide an "undetectable" TCP channel into the target machine with full **root** access. Any number of other drivers could be subverted in this fashion. This makes security even more important with a modular kernel at the heart of the system. Obviously the security hole is actually in the "exploit of the day" and not in the modules as delivered. Protecting the system from attacks of this nature is no more difficult than protecting from the "exploit of the day". Debian and Linux have a continued focus on security. Security breeches are announced in the appropriate news groups. Patches and their locations are announced as well.

Most important of the disadvantages of modular drivers and kerneld has to do with timing problems and unusual hardware configurations. With some devices, under certain load conditions, the time it takes kerneld to install and initialize the driver is long enough for the device to time out, resulting in a failed installation. So, if an FTP client calls for a network connection, and *pppd* isn't running, *kerneld* will kick off the appropriate processes and, depending on configuration, dial the number and establish the connection. In the mean time, however, the FTP client has timed out, waiting for the connection. The solution here is to try again with the FTP client. By this time the connection should be established and the connection will go through with no delay. This is not the best example, since the time delays are very large for a diald-type PPP connection even when everything is compiled into the kernel. If, however, this system did not time out the FTP connection without kerneld and modules, it would be likely to do so under some level of system load because of the additional time needed to load the modules from disk.

The other point has to do with special hardware configuration. If there is a modem that uses a nonstandard interrupt for the given slot location, the system will install the **driver** and configure it properly using *setserial*. However, if the driver isn't used for several minutes, it will be unloaded. When the driver is next needed, *kerneld* will load it, but knows nothing of the special configuration needs of the driver. This results in the driver trying to use the wrong interrupt for that device, and failing. The resulting symptom is that if the system establishes a PPP connection within the first several minutes of operation, the connection will be made in the standard fashion and work as expected. If, however, the system is allowed to sit quietly for several minutes after reboot, the PPP connection will never get established. The *chat* program (if that is how the connection is established) will time out waiting for the first response from the modem because it isn't servicing the proper interrupt. This can be fixed temporarily by issuing another call to *setserial* with the proper interrupt value for the card. The better fix is to explicitly name the device driver in */etc/modules*. Adding **serial** to the list of modules in */etc/modules* will cause the **module** to be loaded by *init* and, later in the boot process, configured with *setserial*. It will also have the beneficial side effect of causing *kerneld* to leave the module alone and never unload it. Thus, the configuration initially provided for this driver remains unaffected and the device continues to work properly.

Chapter 5

Appendix

The following sections comprise the **appendix**. Each section will cover one particular topic in some detail, or introduce subjects that are not explicitly covered in the body of this book.

Appendix 1 - Gives details on various useful UNIX commands.
Appendix 2 - Provides a short introduction to several useful editors.
Appendix 3 - Explains the various uses of the **loop** device.
Appendix 4 - A simple solution to some multiple OS installs.
Appendix 5 - Shows you to build a **Packages** file.
Appendix 6 - Some hints on how to use Linux as a server on the LAN.
Appendix 7 - Lists the options that can be passed to the Linux Kernel.
Appendix 8 - Contains the body of the Debian Social Contract/DFSG.
Appendix 9 - Presents the full text of the GNU Free Documentation License.

Appendix 1: Common UNIX Commands

Introduction

This section covers most of the useful commands available on any UNIX operating system. None of the commands are covered in great detail, but enough information is given to be a useful introduction for those not familiar with the various UNIX commands. For more details on these and other commands, there are two standard commands for getting documentation on programs installed on the system: the **man** command and the **info** command. If information is desired for command **foo**, either *man foo* or *info foo* will provide the available documentation. Not all packages provide **info** pages, while many GNU packages don't provide **man** pages. Depending on the program involved one or the other of these will be available. When the command is unknown but some idea of functionality is known, a key word search can be made of the **man** pages using the *apropos* command or *man -k*.

Note: The term **program** and **command** have been used interchangeably in the previous discussion. This is because in UNIX, all executable commands are, in fact, programs that execute to perform their various tasks. Every program, and thus, every command, is documented in some fashion or another within the system. In a Debian system each package places its copyright, changelogs, and any additional examples or documents into the directory */usr/share/doc/<package name>* and is a good place to start after you have exhausted the **man** and **info** pages.

chgrp

This command is used to change the group ownership of a file or files. The form of the command is: *chgrp <options> <group> <files>.* The *<group>* will be assigned to the *<files>* that are specified. As usual wild cards are permitted, allowing multiple files to be specified. Several useful options are:

-c	describe only the files who's ownerships have actually changed.
-f	don't print error messages on unchanged files.
-R	recursively traverse the sub-directories changing group permission on the appropriate files.

Example:

```
chgrp -R dwarf /home/dwarf
```

Will change all of the files in all of the directories below */home/dwarf* to belong to the group **dwarf**.

See also: **chmod**, **chown**

chmod

When it is necessary to modify the permissions on a particular file or group of files this command is the one to use. It takes the form:

```
chmod <options> <mode> <files>
```

<mode> has the form: [ugoa...][[+-=][rwxXstugo...]...][,...]

u	the user who owns the file
g	users in the files group
o	other users not in the group
a	all users
+	causes permissions to be added to those already present for the file.
-	remove the specified permissions from the file
=	make those permissions specified as the only permissions defined for the file.
r	read permission
w	write permission
x	executed permission
X	execute only if the file is a directory or already has permission for some user.
s	set user or group id on execution.
t	save program text on swap device
u	set the permission to those of the user who owns the file.

g	set the permissions to those of the group.
o	set the permissions to those other than the group.

<options>

-c	give a verbose reporting of only the files changed.
-r	recursively change permissions of directories and their contents.
-f	do not print errors about files that were not changed.

Example:

```
chmod g-w ./*
```

Will remove the group write permissions from all the files in the current directory.

See also: **chown**, **chgrp**

chown

To change the ownership of a file or group of files the *chown* command is used. This command has the form:

```
chown <options> [user][:.][group] <files>
```

If only **user** is supplied, then the ownership of the file is changed to that user. If a **:** or **.** is followed by a user name, then the ownership of the file is changed to the specified user and the group is changed to the login group for that user. If **user:group** or **user.group** is provided, not only the ownership of the file is changed to user, but also the group assigned to the file is changed to **group**.

Files can be designated with wild cards so that a set of files can be changed at the same time.

Several useful options are detailed below. See the **man** page for complete details:

-c	describe only those files that have been changed.
-f	don't print error messages when files can't be changed.
-R	recursively change files in sub-directories.

Example:

```
chown fred ./document.1
```

will change the ownership of **document.1** to user **fred**.

See also: **chmod**, **chgrp**

cp

This command is used to copy files from one location in the file system to another. The form of the command is:

```
cp <options> <from files> <to location>
```

Wild cards are permitted in the <**from files**> parameter. If this parameter refers to more than one file, the <**to location**> must be a directory. There are a wide range of options for this command. Several are listed here.

-a	This option is the same as the **-dpR** and attempts to retain the permissions and ownerships of the files being copied. This option is the best way to “replicate” a file system on another partition, or to move a sub-tree from one place to another without changing its permission structures.
-d	Copy symlinks as symlinks rather than the file pointed to and preserve hard link relationships between files in the copies.
-p	Preserve the owner, group, permissions, and time-stamp found on the original when making the copies.
-R	Recursively copy all sub-directories and their file contents. This will allow a complete limb of a file system to be replicated somewhere else in the file system.

Example:

```
cp -a /home /mnt
```

will copy all of the files and directories in */home* into the directory **/mnt** and will maintain the permissions and ownerships of the files during the copy.

df

This command displays file system information for each device mounted on the system. The information displayed includes the size of the partition, the amount used, and the amount available. The device name and mount point are also provided in this display.

Example:

```
 df
Filesystem            1024-blocks   Used    Available  Capacity   Mounted on
/dev/hdb3             649339        396690  219108     64%        /
/dev/hda1             1054176       784352  269824     74%        /mnt/DOS
/dev/hdb1             649307        542629  73139      88%        /mnt/image/1
/dev/hdb7             331767        180711  133922     57%        /mnt/image/2
/dev/hdb5             347375        258195  71240      78%        /Debian
/dev/hdb6             347375        303564  25871      92%        /home
/mnt/a/deb.iso        617388        617388  0          100%       /mnt/loop1
/mnt/image/1/1.3.iso  540506        540506  0          100%       /mnt/loop2
```

The order in which the file systems are listed, is the order that they were mounted. Except for the last two entries, the above were all mounted by *fstab* at boot time. The last two are **loop** mounted image files.

du

The *du* command reports the disk usage for the specified directory. The value reported for each sub-directory and the total are given in 1024 byte blocks.

Example:

```
du /lib
 1     /lib/modules/2.1/fs
 1     /lib/modules/2.1/net
 1     /lib/modules/2.1/scsi
 1     /lib/modules/2.1/block
 1     /lib/modules/2.1/cdrom
 1     /lib/modules/2.1/ipv4
 1     /lib/modules/2.1/misc
 8     /lib/modules/2.1
 77    /lib/modules/2.0.22/block
 55    /lib/modules/2.0.22/net
 133   /lib/modules/2.0.22/scsi
 321   /lib/modules/2.0.22/fs
 45    /lib/modules/2.0.22/misc
 635   /lib/modules/2.0.22
 80    /lib/modules/2.1.5/block
 58    /lib/modules/2.1.5/net
 139   /lib/modules/2.1.5/scsi
 335   /lib/modules/2.1.5/fs
 48    /lib/modules/2.1.5/misc
 664   /lib/modules/2.1.5
 477   /lib/modules/2.0.30/net
 974   /lib/modules/2.0.30/scsi
 265   /lib/modules/2.0.30/fs
 245   /lib/modules/2.0.30/cdrom
 228   /lib/modules/2.0.30/misc
 2195  /lib/modules/2.0.30
 3503  /lib/modules
 5730  /lib
```

This declares */lib* to contain 5730, 1k (1024 byte) blocks.

ls

This command provides information about the contents and permissions of files within the file system. The bare command *ls* will list the files in the current directory. In addition to the many options that can be used with this command, the last element on the command line can be a file or directory specification. Wild cards are appropriate here, so *ls m** will list all files in the current directory that begin with **m**. There are many options that can be used with the *ls* command. A few of the more useful are:

-l	display listing in the "long" format, giving file type, permissions, the number of hard links, owner and group name, byte size, and the time stamp, by default the modification time.
-a	display all files, including those that start with the '.' character.
-R	give a recursive listing of the contents of all subdirectories.
--	color-code files according to file type.
-S	sort by file size.
-r	reverse the order of whatever sort has been chosen.

Example:

```
ls -l /dev/l*
brw-rw---- 1 root cdrom 24, 0 Jan 17 09:45 /dev/lmscd
srw-rw-rw- 1 root root , 0 Jun 8 23:49 /dev/log
brw-rw---- 1 root disk 7, 0 Sep 23 1996 /dev/loop0
brw-rw---- 1 root disk 7, 1 Sep 23 1996 /dev/loop1
brw-rw---- 1 root disk 7, 2 Sep 23 1996 /dev/loop2
brw-rw---- 1 root disk 7, 3 Sep 23 1996 /dev/loop3
```

```
brw-rw---- 1 root disk 7, 4 Sep 23 1996 /dev/loop4
brw-rw---- 1 root disk 7, 5 Sep 23 1996 /dev/loop5
brw-rw---- 1 root disk 7, 6 Sep 23 1996 /dev/loop6
brw-rw---- 1 root disk 7, 7 Sep 23 1996 /dev/loop7
crw-rw---- 1 root lp 6, 0 Jan 17 09:45 /dev/lp0
crw-rw---- 1 root lp 6, 1 Jan 17 09:45 /dev/lp1
```

lists all the files in */dev* that begin with **l** and lists them in the long format.

mkdir

This command is used to create sub-directories within the file system. The command expects at least a directory specification and has several options. The most useful of which is:

-m mode
This option allows the specification of permissions on the directory being created

Example:

```
mkdir /mnt/DOS
```

will add the subdirectory **DOS** to the */mnt* directory.

See also: **rmdir**

mv

The move command *mv*, like copy, is of the form:

```
mv <options> <source> <dest>.
```

The <**source**> parameter can contain wild cards, and when referring to more than one file the <**dest**> must be a directory. Several options include:

-b	make backups of the files to be removed.
-i	prompt before removing files.
-v	print the name of the files being moved.

Example:

```
mv myfile myfile.old
```

will move the file **myfile** to the new file named**myfile.old**.

rm

The remove command *rm* is used to delete files from the file system. After options this command expects a file name. Wild cards are permitted here, so the command can remove more than one file at a time. Several options for this command are:

-r	recursively remove files from sub-directories.
-f	never prompt for files to remove and ignore nonexistent files.
-i	prompt for each file before removing it.

Example:

```
rm -r /home/fred/*
```

will remove all the files in all the directories of */home/fred*.

rmdir

This command is the reverse of the **mkdir** command. It causes the directory specified to be removed. There is only one option for this command:

-p	removes any parent directories that become empty by removing the specified directory.

Example:

```
rmdir /mnt/DOS
```

will remove the sub-directory **DOS** from the path */mnt*.

See also: **mkdir**

Appendix 2: Text Editors

Introduction

Linux systems provide many different kinds of editors. Debian is no different. From the tiny **ae** used in the base system, to the gigantic and powerful **emacs**, there are all the editors you might expect to find in any UNIX, and then some. The following section will give a brief description of four editors found in Debian. Each description section will tell how to start the editor, how to save a file, and how to exit the editor. Some of the primary edit features will also be discussed.

For more detailed information see the **man** pages and other documentation provided with each of the editors.

ae

Anthony's Editor **ae** is an exceptionally small program (23.9K) that provides basic edit capability for the rescue disk and the base system. The default condition for the editor is with the help screen visible. This is one of the features that makes this editor so easy to use.

The current release of Debian uses **slang1** instead of **ncurses**, because the library is much smaller, allowing other things to go on the boot file system. To accommodate the new library constraints, and to give **ae** functionality on many terminal types, the key-bindings have been changed to be more like **emacs**. Most keys are control keys, with some use of the escape key. For those terminals without an escape key, <**ctrl-**[> will act as an escape.

All of the possible commands available in the editor are displayed on the screen. <**ctrl-X**><**ctrl-H**> will toggle the help screen off and on to provide more screen area for editing. The standard commands are:

ˆX I	Read a file into the editor.
ˆXˆS	Write the contents of the editor to a file.
ˆ@	Block select to cursor movement.
ˆW	Cut selected block.
ˆY	Paste a cut block.
ˆXˆC	Exit the editor only if file has been saved. Prompts to save file.
ˆQ	Exit the editor without saving the file first. Discards changes

ae is the only editor available in the base system, so until the system is upgraded to "standard" there may not be an editor available that is completely familiar. **ae** is the compromise between size and familiarity.

joe

Joe's Own Editor **joe** is a **WordStar** clone with all the features found in that ancient piece of CP/M code. The **control K** sequences are the same as in the original, so anyone familiar with **WordStar** can just use **joe** without thinking about learning new commands. For those not familiar with this word processor the following will be a good start:

ˆK H	Toggle the help screen on and off.
ˆK D	Save the current edit session.
ˆK X	Save the current edit session and exit.
ˆK Z	Suspend the editor for the shell.
ˆC	Abort the current edit session. (don't save the file)

With the help screen you can find out how to select a block of text, move or copy the block. Word search and replace are also available along with a spell checker.

When a file is saved with **joe** it automatically moves the original file to original-file-name . The makes it easy to identify the backup edit session, but it only reflects the last edit session, so older contents will be lost.

vi

One of the standard editors found on any UNIX system, **vi**, is a modal editor that provides many useful features. This section is not spacious enough to do justice to **vi**'s many features, so only the basic commands needed to do simple editing will be covered here. A few of the more useful commands include:

:viusage<**CR**>
: Show a list of vi commands.

:exusage<**CR**>
: Show the list of ex commands.

:q! Unconditional exit back to the shell.

:wq Write the file and quit.

i Enter insert mode (characters can only be entered when in input mode).

<**esc**> Return to command mode.

Modal refers to the way that **vi** operates in different modes. When first invoked, **vi** is in command input mode. The cursor can be moved around the screen, but any other keys typed are expected to be commands. Some of those commands, like the **i** command, switch the editor into text input mode. For the insert command, the characters typed are inserted to the left of the cursor.

The escape key will return to command mode, either from text input mode or to abort a command in the middle. If you type several characters that do not create a command, no other command can be entered until these characters are cleared. The escape key returns to command input mode and clears the partial command.

emacs

emacs is another, larger editor that comes with most standard systems. It has many features not found in other editors, not to mention a built-in LISP interpreter. Because of its many features, many of the **emacs** commands are multiple control or escape sequences. Some of the more common are listed below:

ˆXˆC	Exit emacs
ˆHˆT	Presents a Tutorial on emacs
ˆHˆI	Displays the GNU Info pages on emacs
ˆXˆS	Save the current buffer to a file

emacs can be used to edit several buffers at one time, and provides many different "source" environments helpful to creation of source code. These areas are far too broad to be covered in these few paragraphs. The **Tutorial** is highly recommended as a good way to get to know the features of **emacs**. The **info** pages will give other, useful information not found in the tutorials. Both of these should be enough to give you a good feel for how this editor works, and what it can provide.

Appendix 3: The Loop Device

The **loop device** is a device driver that allows an **image file** to be mounted as though it were a normal block device. The question that immediately jumps to mind is, "So, how do I use this beast". As an example, let's look at mounting, and examining the contents of, the rescue floppy image file.

First the CD-ROM, or other device containing this **image file** needs to be mounted like:

```
mount -r -t iso9660 /dev/scd0 /cdrom
```

Note: The **-r** option declares the device to be read only. Also */dev/scd0* assumes the drive is the first SCSI device. If the machine uses a SoundBlaster Pro/Creative Labs combo for the cdrom this device would be */dev/sbpcd0* instead. The other possibilities depend on the particular interface being used.

If this is an "Official CD" that was mounted, the file of interest is found as:

```
/cdrom/debian/hamm/main/disks-i386/current/resc1440.bin
```

So the mount command using the **loop device** will look like:

```
mount -t msdos -o loop /../../resc1440.bin /mnt
```

After this an *ls* of */mnt* will show all the files that will appear on the floppy disk when this image file is transferred to it. This is a "live" file system. That is, it can be modified just like any other read/write file system mounted as a block device. The changes made to the file system while mounted become part of the image file and will be reflected on the floppy constructed from that image. Of course, in our example the **file image** resides on a read-only medium so changes aren't possible, but when the image file resides on a writable medium, like the hard disk, then those kinds of changes become possible.

OK. We can mount such file systems, but how are they constructed in the first place? This, of course requires a *nix environment. The tools are **dd** and **losetup**, and the process goes something like this:

First it is necessary to create an empty file of the desired size. This is done with **dd** and the **zero device** in a command like:

```
dd if=/dev/zero of=/../image.file bs=1k count=100000
```

This will create a file with 100 MB of space. Note that the file size is equivalent to the partition size when creating partitions on a block device like the hard disk. This file will thus hold a file system that can reach 100 MB in size before "device full" errors occur.

Before a file system can be created on this **image file** and mounted, it needs to be made to look like a block device. This is done with the **losetup** program in the following fashion:

```
losetup /dev/loop0 /../image.file
```

There are 8 loop devices to choose from, so you may need to check to see if a particular device is already in use. *losetup /dev/loopN*, where **N** ranges between 0 and 7, will return the status of the Nth **loop device** if it is mounted and will return an error message if it is not.

Now that */dev/loop0* is a legitimate block device, **mke2fs** (or any of the other file system creation utilities) can be used to create a file system on the looped **image file** with a command like:

```
mke2fs -c /dev/loop() 100000
```

The **-c** option checks the device for bad blocks and the value on the end specifies how many blocks are on the file system. These two options can be left off the command line and results will be seen faster, but there is no guarantee that the resultant file system will be useful. Take the time to check for bad blocks. It is time you will not spend looking for the cause of problems later. Once the file system has been created, and a **loop device** associated with file, it can be mounted using the following command line:

```
mount -t ext2 /dev/loop0 /mnt
```

When this is later unmounted make sure to release the device with the command *losetup -d /dev/loop1.* Another alternative is to release the **loop device** as soon as the file system creation is complete, and mount the file as in the previous example with a command like:

```
mount -t ext2 -o loop=/../image.file /mnt
```

In either case this produces a “file system” mounted on */mnt* that has 100 MB of storage space. Whatever this file system is intended to contain can now be copied, or untarred, or otherwise created on the new file system.

It should be evident that **loop devices** are very useful critters. The installation software makes use of one to perform several sets in the installation by mounting image files from the CD, instead of requiring a floppy disk to mount. This is not only faster, but physically easier than the floppy installation method.

Appendix 4: Multiple OS Installation

There are different ways that more than one Operating System can reside on the same machine. DOS and Windows can easily reside on the same machine with Linux, and each can be booted at the choice of the operator. Windows 95/98 is a little bit trickier since it demands its own boot manager. Installing Windows 95/98 after installing Linux will result in LILO getting “bumped off” the system. Installing Linux AFTER Windows 95/98 will work. You will need to configure your Linux system to be able to boot Windows 95/98 with LILO however.

LILO can be used as long as there is a Linux partition on the first physical drive. With a modified boot record and LILO, the system can be made to boot either Linux or DOS/Windows 95/98. There are other boot loaders available for Intel machines. Most of these can be used to boot a variety of operating systems.

The more versatile method of installation uses the DOS program **loadlin.exe**. This software can boot a kernel image from a DOS partition and mount any partition on the system as the root partition for that kernel. For automatic operation, the best method is to use the config.sys menu system in DOS to offer the choices of the different OS available on the machine. Then **autoexec.bat** can determine whether to go ahead and boot DOS or run **loadlin** and boot Linux. The following example should clarify this process:

```
config.sys:
[Menu]
menuitem=NewLinux, Linux 2.0.30
menuitem=OldLinux, Linux 2.0.27
menuitem=DOS
[DOS]
DEVICE=C:\DOS\SETVER.EXE
DEVICE=C:\DOS\HIMEM.SYS
```

```
DOS=HIGH
FILES=30
STACKS=9,256
[NewLinux]
[OldLinux]
autoexec.bat:
cd \boot\linux
goto %config%
:DOS
cd \
C:\DOS\SMARTDRV.EXE
@ECHO OFF
PROMPT $p$g
PATH C:\WINDOWS;C:\DOS
SET TEMP=C:\DOS
goto End
:NewLinux
loadlinx vmlinuz2 root=/dev/hdb3
goto End
:OldLinux
loadlinx vmlinuz root=/dev/hdb3
goto End
:End
```

Note: This system allows the root file system to reside on another area besides the first device on the machine.

This process will work for Windows 95/98 as well as DOS/Windows and OS/2.

Appendix 5: Building Packages Files

Files with the extension **.deb** contain the packages managed by **dpkg**. For information on how these packages are built, the *programmer.html/* and *policy.html/* directories will provide much useful information. These files are found on a Debian system in the */usr/doc/dpkg.*

What you need to build a Packages file:

1. Debian GNU/Linux system with dpkg-dev installed.
2. A collection of .deb package files.
3. The override file for the distribution involved.

The Debian package **dpkg-dev** provides the script **dpkg-scanpackages** which is used to produce **Packages** files. With this tool an arbitrary collection of Debian binary packages can be converted into a "distribution". Normally the Debian distribution resides in the sub-directories of the directory *binary-i386.* Many times however, even with a "standard" archives of this distribution, the **Packages** file is not in sync with the archive, and a new **Packages** file will be required. The proper operation of **dselect** depends on an accurate and up-to-date **Packages** file.

Another reason for building a **Packages** file relates to the construction of a "custom" installation. Collecting a specific subset of the Debian distribution for installation using **dselect** will require a **Packages** file for this subset. Once these packages are collected into a sub-tree, a **Packages** file can be created.

dpkg-scanpackages has two required parameters, and sends its output to **stdout**. The first parameter is the path to the archives while the second points to the **override file** for the release being used. The 1.3 release is code named **bo**, while the previous release was **rex** and the next one was **hamm**

(all characters from the movie dgbToy Story). These **override** files are found in the *indices/* directory of any Debian mirror with the name **override.bo** or **override.rex** or **override.hamm** depending on which distribution the packages came from. The command line would look something like:

```
dpkg-scanpackage ./bo/binary-i386 ./indices/override.bo >Packages
```

and will run for quite some time. Upon completion the script will print information about any differences found between packages and information found in the **override** file. This is typically caused by the fact that the maintainer changed and the **override** file does not reflect the new maintainer. These errors are never a problem to the correct operation of the **Packages** file, but are informative only. The final information provided by the script gives the total number of packages added to the **Packages** file. A copy of this file along with a gzipped copy should be placed in the outer directory of the distribution tree. For the standard distribution the **Packages** file goes into *../binary-i386*. When a group of packages are collected into one directory that is the directory where the **Packages** and **Packages.gz** files should be placed as well. Once this has been accomplished the distribution is ready to be installed using **dselect**.

Appendix 6: Linux as a Server

One of the many versatile features of Linux is its ability to act as a server in various network environments. Linux, and therefore Debian, provides features that allow it to act as a server for Windows and Windows 95/98 machines. It can also act as a server for an AppleTalk network, or even a Novell network.

SAMBA is the name of the software that provides the tools for serving Windows machines. A very comprehensive discussion of this tool can be found at:

```
http://lake.canberra.edu.au/pub/samba
```

There is also a **HOWTO** that can be found at:

```
http://www.interweft.com.au/other/samba/smb_se.html
```

as well as a **FAQ** that can be found at:

```
http://sunse.jinr.dubna.su/local/samba/samba.faq.html
```

AppleTalk, the Macintosh network protocol, can be supported by the kernel if the proper **appletalk.o** module is provided, or compiled into the kernel. This network is also called **netatalk** and a **HOWTO** can be found at:

```
http://www.anders.com/projects/netatalk/
```

Linux can also be run directly on a Mac with the product called **MKLinux**. Information on how to set up **netatalk** on **MKLinux** can be found at:

```
http://www.primate.wisc.edu/people/dubois/mklinux/netatalk-setup.html
```

Linux can even act as a server on a **Novell** network. The kernel needs to have **IPX** support compiled in, or as a module. For more information see:

```
http://electron.phys.dal.ealmini/HTTP+Netware
```

or

```
http://www.inet.co.th/cyberclub/karnedp/http-4.html
```

The **IPX** module is also discussed in the **Modules HOWTO** which can be found at:

```
http://linuxwww.db.erau.edu/ldp/HOWTO/Module-HOWTO-s.html
```

or at most mirrors of the **Linux Documentation Project**.

All of the above sites were found using one of the many search engines that the Internet provides. If you point your browser at:

```
http://altavista.digital.com/cgi-bin/query?pg=aq
```

the first blank line on the screen that comes up is for a list of keywords with logical operators that define the way they are selected. The real power of this page is the second field for **Result Ranking Criterion**. Key words in this list are used to bring more likely hits to the top of the list. This is very useful when your keywords return 2,000 hits.

There are many other search engines available on the net. Using the above site and the keyword "search engine" produces 500,000 hits, so choices for **Result Ranking Criterion** can be very useful.

Appendix 7: Arguments

The kernel accepts arguments of the following form:

no387=
: This argument tells the kernel not to use the math coprocessor, even when present. The kernel must, of course, be compiled with math emulation support for this to work.

root=
: Tells the kernel which partition to use as the root device. This value can be entered as a text string, like */dev/hda4*, or as the **major/minor** number for the device, which for **hda4** is “0x304” for major 3 and minor 4, **fsck** has been run. This argument can be set with **rdev**.

debug=
: Under normal circumstances the kernel logs only those messages of debug level 7 or below to the console. Using this argument will send all debug messages to the console. **klogd** also has options that it will accept to modify what debug level will get logged. See the **man** page for details.

reserve=
: Protects various i/o ports from being probed. It takes the form: *reserve=iobase,extent[,iobase,extent]...* and will keep one device driver from probing sensitive ports on another card. Since that card will eventually need to be initialized by its own driver, this command is usually used in conjunction with another. If the device **foo** is caused to lock up by a probe from another device, the

argument list would look like:

```
reserve= 0x300,32 foo=0x300
```

This will keep all other device drivers from probing 0x300 while allowing the drive that uses that port to be properly initialized.

ramdisk=
This argument tells the kernel how big a ramdisk to create. To put a file system from a **floppy** into a **ramdisk** would require an argument like: *ramdisk=1440*. Note that the units of size for this argument is kilobytes. This is one of the few parameters that can be set in the kernel using **rdev**.

mem= Since the BIOS of most PCs can only report 64 MB there are times when the kernel needs to be told that there is more memory than this. This argument should be used with great care, since, if you lie to the kernel and tell it that it has more address space than it actually has, the system will crash miserably at some time in the near future. Also make sure that your system does not use the highest memory for caching the BIOS code, as use of this memory by the kernel will ultimately crash the system as well. The value given after the equal sign should be the highest ram address available to the kernel, so a 96 MB system would get an argument like 'mem=0x6000000'.

SCSI device arguments

General Notation

iobase The first I/O port that the SCSI device occupies, normally between the values: 0x200 and 0x3ff

irq The hardware interrupt that the SCSI card is configured to use.

scsi-id The host adapter's ID on the SCSI bus. Some adapters allow this value to be changed but most don't. Many adapters use ID of seven, however, the **Seagate** and dgbFuture Domain TMC-950 cards used ID of six.

parity Specifying a 1 tells the card to expect the SCSI devices to report parity. Setting this to 0 has the opposite effect. As with the scsi-id not all adapters accept this argument.

SCSI argument list

max_scsi_luns
Some SCSI devices are poorly designed and will lock up when probed for LUNs (Logical Unit Numbers) other than zero. As the kernels now default to only probe LUN 0, this argument allows for the probing of other LUNs as well.

st= This is used to configure SCSI tape devices and has the form: *st=buf_size[,write_threshold[,max_bufs]]*.

buf_size:
The value is given in units of kilobytes with the default value of 32 and a max value of 16384.

write_threshold:
Is the size of the buffer that will commit a write to tape and defaults to 30 kilobytes.

max_bufs:
Normally defaults to 2 but the proper number depends on how many devices are present.

aha152x=iobase[,irq[,scsi-id[,reconnect[,parity]]]]
These values are as described earlier and are order dependent, so to specify parity, the rest of the parameters must appear on the line. If the reconnect value is non-zero the device will be allowed to disconnect/reconnect.

aha1542=iobase[,buson,busof[,dmaspeed]]

The iobase for this card is usually, 0x130, 0x134, 0x230, 0x234, 0x330, and 0x334. Some clone cards provide other addresses as well. **buson** and **busof** specify the time the card spends on and off the buss. Default values are 11μs on and 4μs off so that other cards get a chance at the ISA buss. **dmaspeed** refers to the rate at which the DMA transfers proceed. The default value is 5 MB/s, although some cards allow setting up to 10 MB/s, but care should be taken to make sure that the machine is up to the task.

aic7xxx=extended,no_reset

A nonzero value for extended enables extended translation for large disks. A nonzero value for no_reset tells the card not to reset the card at boot up.

buslogic=

The only parameter that this argument accepts is the **iobase**. The expected values are: 0x130, 0x134, 0x230, 0x234, 0x330, and 0x334.

tmc8xx=mem_base,irq

The probe code for these **Future Domain** cards looks for a BIOS on the card. If one is not found, or the signature is not recognized, this card will go unrecognized. Specifying the **mem_base** address and the **irq** in this argument will force recognition of the card. The **mem_base** address specifies the memory mapped I/O region of the card and expects values: 0xc8000, 0xcc000, 0xce000, 0xdc000, 0xde000.

pas16=iobase,irq

This card uses the **NC5380** SCSI chip and usually expects the **iobase** to be 0x388. If the **irq** value is given as 255, then the card will operate without interrupts, at a reduced capability.

st0x=mem_base,irq

This probe code also expects to find a bios and will not find the card if there is no BIOS. This argument will force detection of the card. **mem_base** is the memory mapped I/O region of the card and uses values: 0xc8000, 0xca000, 0xcc000, 0xce000, 0xdc000, 0xde000.

t128=mem_base,irq

These cards also use the **NC5380** SCSI chip and expects the following values for **mem_base**: 0xc8000, 0xcc000, 0xd8000, 0xdc000.

The following cards do not accept arguments:

Always IN2000	Adaptec aha1740	EATA-DMA
EATA-PIO	Future Domain 16xx	NCR5380 (generic)
NCR53c7xx to	NCR53c8xx	Qlogic
Ultrastor	Western Digital wd7000	

Hard Disk Arguments

The hard disk arguments are used to do special setup on IDE devices. This includes a number of CD-ROM devices as well as the normal hard drives. Drive-specific options have the form:

```
hda=, hdb=, hdc=, hdd=.
```

Non drive-specific options have the form: **hd=**. This option can be used to specify the next drive in the **a**, **b**, **c**, **d** order. Drive specific options can be used in place of non-specific ones with the expected result. In the following discussion the **hd=** will be used exclusively and can be replaced by any drive specific option that applies.

hd=cyls,heads,sects[**,wpcom**[**,irq**]]

This option is used to specify the physical geometry of the disk. Only the first three parameters are required. The **wpcom** (write pre-compensation) parameter is ignored by IDE devices. The **irq** value is the **irq** of the controller of the device and as such is not strictly a drive parameter.

hd=serialize

Many of the older kernels had a broken CMD-640 chip driver design which resulted in corrupted data when the second controller was operated at the same time as the first. This option forces the driver to do sequential accesses rather than simultaneous activity. You will only need to use this option if you have a drive installed on the second IDE controller.

hd=dtc2278

If you have a **DTC2278D** IDE interface this option causes the driver to attempt to enable the second controller as well as faster transfer options.

hd=noprobe

If the machine has an old IDE drive that suffers from being probed this option will keep that disaster from happening. For example, if the drive is **hdb** then the following will allow the drive to not be probed but also provide the drive information for proper identification of the drive: *hdb=noprobe hdb=1166,7,17* will do the trick.

hd=nowerr

As some drives appear to have the **WRERR_STAT** bit permanently stuck, this option allows the drive to work anyway.

hd=cdrom

If there is an ATAPI compatible CD-ROM installed on one of the IDE controllers and it is not recognized by the kernel, this option will allow the kernel to recognize the drive.

xd=type,irq,iobase,dma_chan

If the system has an old 8 bit xt card, the driver probes the card for a BIOS. If none is found, or the signature is not recognized, then the drive will not be recognized. This option will allow the recognition to occur. **type** specifies the card manufacturer and has the values: 0=generic; 1=DTC; 2,3,4=Western Digital; 5,6,7=Seagate; 8=OMTI. **irq**, **iobase** and **dma_chan** are all required parameters, so don't leave any of them unspecified.

Other CD-ROMS

This section contains options for CD-ROM devices that are neither SCSI or IDE/ATAPI devices.

aztcd=iobase[,magic_number]
The **Aztech** interface uses these options. If the magic number is set to 0x79 the driver will continue to try to function even when the firmware version is unknown. Any other magic number will be ignored by the driver.

cdu31a=iobase[,irq[,is_pas_card]]
This argument is used to configure the **Pro Audio Spectrum** cards as well as some **Sony** supplied interface cards. Passing an **irq** value of 0 tells the driver that the card does not support interrupts. If the card does support them you are encouraged to use the interrupt, as it will ease the CPU load. If the card is a **Pro Audio Spectrum** then the **is_pas_card** parameter should be **PAS**.

sonycd535=iobase[,irq]
Used by the **Sony CDU-535** interface card, the **iobase** passed as 0 to act as a placeholder when an **irq** needs to be specified.

gscd=iobase
Used by the **GoldStar** interface.

mcd=iobase[,irq[,wait_value]]
Used by **Mitsumi Standard** interface cards, the **wait_value** is for devices that are having trouble with time-outs, but should be used with care as some implementations of the driver don't support this option. Whether or not it is implemented depends on a compile time DEFINE.

mcdx= For the **Mitsumi XA/Multisession** interface experimental.

optcd=iobase

The **Optics Storage** interface uses this argument.

cm206=[iobase][,irq]

Used by the **Philips CM206** interface, this driver expects **irq** values between 3 and 11 and **iobase** values between 0x300 and 0x370, so these can be supplied in any order, with either value missing. There is also the **cm206=auto** which tells the driver to auto probe for the information.

sjcd=iobase[,irq[,dma_channel]]

This argument is used by the **Sanyo** interface card.

sbpcd=iobase,type

Used by the **Soundblaster Pro** interface card, the value of **type** can be: 0 for **LaserMate**, 1 for **SoundBlaster**, 2 for **SoundScape**, and 3 for **Teac16bit**. The **iobase** is the base for the CD interface and not the sound card.

Floppy Drive Arguments

All of the information in this section can be found in **README.fd** in *linux/drivers/block*. The following information is taken directly from this file.

Several points of interest include:

- Using **floppy=** arguments on the command line as well as in the LILO config file will result in the two arguments being concatenated.
- The **floppy=** argument can be used with **insmod**, however the syntax is slightly different.
- For **insmod** the **floppy=** is only entered once with the parameters following it in quotes as, *floppy"daring two_fdc"*.
- **floppy=mask,allowed_drive_mask** and **floppy=all_drives** are obsolete and should be replaced by: *floppy=<drive>,<type>,cmos*

Specifying a drive is mandatory when using more than 2 floppy drives, and sets the CMOS type for <**drive**> to <**type**>, where the CMOS types are:

```
0 - Use the value found in the physical CMOS
1 - 5 1/4 DD
2 - 5 1/4 HD
3 - 3 1/2 DD
4 - 3 1/2 HD
5 - 3 1/2 ED
6 - 3 1/2 ED
16 - unknown or not installed
```

Note: Type 5 is for use with **AMI** floppy drives. Under other circumstances this type was intended for use by floppy "tapes".

floppy=asus_pci
Sets the bitmask to allow only devices 0 and 1. This is the default.

floppy=daring
Indicates a “well-behaved” floppy controller, which allows the driver to provide more efficient and smoother operation and will speed up certain operations. However this may cause failures on some controllers.

floppy=one_fdc
Tells the driver that you have only one controller. This is the default.

floppy=[address,]two_fdc
Indicates to the driver that there are two floppy disk controllers and that the second controller resides at **address**. If **address** is not entered then 0x370 is assumed.

floppy=thinkpad
Informs the driver that this is a **Thinkpad** and will deal with the inverted disk change line used by these machines.

floppy=0,thinkpad
Tells the driver that this is not a **Thinkpad**.

floppy=omnibook and floppy=nodma

As the **HP Omnibook** has no workable DMA channel and other hardware deliver "Unable to allocate DMA memory" error, this argument tells the driver not to use DMA transfers with the drive. The machine must be at least a 486 to use **nodma**, and the FIFO threshold should be set to 10 or lower to limit the number of data transfer interrupts.

floppy=dma

Allows the driver to use DMA and is the default.

floppy=nofifo

Used to disable the FIFO entirely. If the Ethernet card or other device declares "Bus master arbitration error" while using the floppy, you will need to use this argument to disable the FIFO.

floppy=fifo

Enables the FIFO and is the default.

floppy=threshold,fifo_depth

This argument is relevant in DMA mode and sets the FIFO threshold. A higher value allows the floppy driver to tolerate more interrupt latency, while generating more interrupts. A lower value should lower the interrupt latency and produces fewer interrupts.

This threshold can be tuned using the *floppy control --messages* and then later looking for *Over/underrun - retrying* messages when accessing the floppy drive. If there are a huge amount of these messages then the threshold is set too low. Set the threshold value higher until you only see an occasional error message. Building the kernel with the floppy driver as a module will allow you to remove the driver and reinstall it with a different threshold value without the need to reboot the machine. Remember to reissue the *floppycontrol --messages* command after each reload of the driver. Usually this "tuning" will not be necessary, as the default value of 0xa is a reasonable value in most cases.

floppy=unexpected_interrupts
This argument causes a warning message to be printed whenever an unexpected interrupt is received from the floppy. This is the default behavior.

floppy=no_unexpected_interrupts and floppy=L40SX
These arguments causes no message to be printed when an unexpected interrupt in received. The **IBM L40SX** laptops have an interaction between certain video modes and the floppy controller that result in unexpected interrupts. These interrupts only effect performance and can be safely ignored.

fdutils and *mtools* provide additional run-time configuration capabilities for the floppy drive, including reading high capacity disks.

Other Device Drivers

Any drivers not falling into the above categories will be found here.

ether=irq,iobase[,param1[,param2....[,param8]]],name
For configuring Ethernet cards. The various cards have differing parameters, but they all require an **irq** and **iobase** address. The first non-numeric value is taken to be the name of the device. This argument is typically used to force the detection of a second Ethernet card, and, for that purpose, would look like: *ether=0,0,eth1*. Note that 0 is again used as a place holder for this argument.

sound=device1[,device2.....[,device11]]
Provide a means to override the compiled in values. This is somewhat complex and should be avoided if possible. The **deviceN** parameters are of the form 0xTaaaId where the bytes are used as follows:

T	device type: 1=FM, 2=SB, 3=PAS, 4=GUS, 5=MPU401, 6=SB16, 7=SB16-MPU401
aaa	I/O address in hex
I	Interrupt in hex
d	DMA channel

Using a ‘sound=0’ disables the sound card all together.

busmouse=irq
Used by the busmouse driver.

Appendix 8: What is Free Software

The following definition of Free Software is copyright © 1997 "Software in the Public Interest" and is printed here with the author's permission. The project participants overwhelmingly voted to adopt this declaration on the 4th of July, 1997. What follows is the entire document.

DEBIAN'S "SOCIAL CONTRACT" WITH THE FREE SOFTWARE COMMUNITY

We are **Software In The Public Interest**, producers of the **Debian GNU/Linux** system. This is the **social contract** we offer to the free software community.

1. Debian Will Remain 100% Free Software

We promise to keep the **Debian GNU/Linux** distribution entirely free software. As there are many definitions of free software, we include the guidelines we use to determine if software is "free" below. We will support our users who develop and run non-free software on Debian, but we will never make the system depend on an item of non-free software.

2. We Will Give Back to the Free Software Community

When we write new components of the Debian system, we will license them as free software. We will make the best system we can, so that free software will be widely distributed and used. We will feed back bug-fixes, improvements, user requests, etc. to the "upstream" authors of software included in our system.

3. We Won't Hide Problems

We will keep our entire bug-report database open for public view at all times. Reports that users file on-line will immediately become visible to others.

4. Our Priorities are Our Users and Free Software

We will be guided by the needs of our users and the free-software community. We will place their interests first in our priorities. We will support the needs of our users for operation in many different kinds of computing environment. We won't object to commercial software that is intended to run on Debian systems, and we'll allow others to create value-added distributions containing both Debian and commercial software, without any fee from us. To support these goals, we will provide an integrated system of high-quality, 100

5. Programs That Don't Meet Our Free-Software Standards

We acknowledge that some of our users require the use of programs that don't conform to the Debian Free Software Guidelines. We have created "contrib" and "non-free" areas in our FTP archive for this software. The software in these directories is not part of the Debian system, although it has been configured for use with Debian. We encourage CD manufacturers to read the licenses of software packages in these directories and determine if they can distribute that software on their CDs. Thus, although non-free software isn't a part of Debian, we support its use, and we provide infrastructure (such as our bug-tracking system and mailing lists) for non-free software packages.

THE DEBIAN FREE SOFTWARE GUIDELINES

1. Free Redistribution

The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form _only_ if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (This is a compromise. The Debian group encourages all authors to not restrict any files, source or binary, from being modified.)

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to Debian

The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

9. License Must Not Contaminate Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

10. Example Licenses

The "GPL", "BSD", and "Artistic" licenses are examples of licenses that we consider "free".

The concept of a Linux distribution stating its "social contract with the free software community" was suggested to me by Ean Schussler. I composed a draft, and then it was refined by the Debian developers in e-mail conference during most of June. They then voted to approve it as our publicly stated policy. We hope that other software projects, including other Linux distributions, will use this document as a model. We will gladly grant permission for any such use.

Respectfully Submitted

Bruce Perens
Debian Project Leader

Appendix 9: GNU Free Documentation License

What follows is the verbatim text of the **GNU Free Documentation License**. This is the license under which this book has been released.

Verbatim Copying

prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index