Working Centre Linux Project: Administrator's Guide

Paul "Failure" Nijjar

RCS Version 1.18

Contents

1 Introduction

1.1 How to read this manual

The aim of this manual is to document *everything* about the Working Centre Linux Project, so that if the project volunteers are eaten by locusts the project can continue. Eventually, we want this manual to contain enough information so that somebody could recreate the project from scratch if necessary.

Fortunately, you probably do not have to read this entire manual to use the project:

- If you are a user who has had our version of Linux installed onto your system, you should not need to read this manual at all. (You are welcome to read it, of course!)
- If you are a volunteer installing Linux onto refurbished systems, you should read Sections 3, 4 and 5 .

  Additionally, you may want to refer to the Appendices as weird installation situations crop up.
- If you are a developer or administrator, you should start by reading about the installation process in Sections 3, 4 and 5 .

  To learn about the installer itself, you should really read the FAI manual in addition to Section 6. You can find the FAI manual in the directory

  `/usr/share/doc/fai/`

  on the server.

1.2 This documentation is broken!

In the process of getting together a working installer, we have made many changes to the infrastructure in this project. We try to keep this documentation up to date, but errors slip through. Every so often you will probably read sections of this document that make no sense. In such cases, *please* let us know so that we can fix the errors and improve this document.

At the risk of creating yet another section that will quickly grow out of date, here is a list of the problems of which we are aware:

- Section 7, which covers the local Debian mirror on the server, needs to be rewritten:
  - We now use *two* mirrors on the server. One is an unstructured mirror as described in this section. Its location is

    `/usr/local/mirror/localdebs/`

    The other mirror is structured. It is created by some local scripts and the **apt-move** utility. The scripts can be found in

    `/usr/local/mirror/bin/`

    and the mirror is

    `/usr/local/mirror/debian/`
  - Because we now have a structured mirror, we no longer need a `base_woody.tgz` file.
- We patched the **make-fai-bootfloppy** utility to take an extra `-i` parameter. This changes the way that the utility is used to make floppies. See Section 9.6.1
- We took **gnumeric** out of the standard installation. It is *usually* installed (currently for any machine with a 1Gig or bigger hard drive), but not always. Thus, systems do not always contain a spreadsheet.
- The location of the FAI files on our server is

  `/usr/local/share/fai/installer/`

  and not

  `/usr/local/share/fai/`

  However, if you downloaded our software, you will find the files in

  `/usr/local/share/wclp-`*`version`*`/installer/`

  where *version* is the version of WCLP you downloaded.
- We now use GRUB as the default bootloader instead of LILO. Boot floppies still use LILO, but we install GRUB to the hard drive, because dual-booting under GRUB is easy.
- We no longer use **pdq** as our printing system. Currently we use **lprng** instead, but even that has a few problems. Eventually we may switch to the CUPS printing system.

1.3 Conventions and Terminology

In this section we discuss some terms you should be familiar with. We also outline the conventions we use in the manual.

1.3.1 Terminology

Here are some terms you will want to know:

client: A *client machine* is a machine onto which we will install Linux.

FAI: FAI stands for "Fully Automated Install." It is the program that we use to install Linux onto computers.

NFS: NFS stands for "Network Filesystem". It is a networking protocol that allows computers to use files and directories on remote computers. To access the remote drive, the computer *mounts* the drive using NFS.

FAI relies on NFS to mount a temporary Linux filesystem, which it uses to install a copy of Linux on the client machine.

server: Our *server machine* is a Linux server that contains the local package archive and the FAI program files. Client machines connect to the server in order to get Linux installed on them.

(More terms? -P.)

1.3.2 Conventions

To disambiguate filenames, when ending a sentence with a file we always put a space between the period and the filename. For example, I will end this sentence by mentioning the file `foo` .

We use a number of different typefaces to indicate special terms and instructions:

- We indicate files or directories as follows: `/usr/local/share/fai/`
- We indicate text you should type into the computer as follows: `type this exactly`
- We indicate messages that the computer prints out as follows: `Hello world`
- We indicate program names as follows: **aptitude**
- We indicate menu items as follows: **File** . To indicate a sequence of menus you should follow, we use →: **File** → **Save As**
- We indicate keys to press as follows: ⟨enter⟩

2 Background Information

This section includes some general information about the project, operating systems, and about Linux. Its purpose is to give the Gentle Reader an idea of the philosophies and social movements that have made this project possible.

2.1 What is an operating system?

An *operating system* is made up of software that manages resources on a computer. The computer's hardware provides these resources – CPU cycles, memory, hard drive space, input devices such as the keyboard, and output devices such as the monitor signal. A wide variety of applications and system maintenance programs need these resources to get their jobs done. The operating system's job is to manage the interactions between the hardware and software programs.

For example, when **Abiword** starts up, it asks the operating system for some memory so that it can load. The operating system then decides whether there is enough memory available, and possibly re-organizes memory usage to free up space. Then it gives this memory to **Abiword** so that the program can run and you can type a letter. If not for the operating system, **Abiword** would have to manage its own memory, which is an error-prone process.

Here's another example: say **Abiword** and **Gnumeric** both want to save backup files to the hard drive. If they both tried to save to the hard drive at the same time, the data on the hard drive could become corrupted. It is the operating system's job to control access to the hard drive, first giving one program permission to write to the drive, and then giving the other permission. Without the operating system, these programs would have to manage these issues themselves.

People have written lots and lots of operating systems for computers. You may be familiar with the Microsoft Windows family of operating systems. Many Macintosh computers use an operating system called Mac OS. Other operating systems of note include BeOS and VMS.

In the academic and networking worlds, a family of operating systems called UNIX is popular. There are many variants of the UNIX operating system available, too: Solaris, BSD and its free variants FreeBSD, OpenBSD, and NetBSD, HP-UX, AIX... the list goes on and on. Linux is a version of UNIX that runs on personal computers.

2.2 Is this software legal?

Yes. All of the software we include in the Working Centre Linux project is legally available free of charge. Most of our software is *open-source* software. Some programs, such as **Opera** are not open-source, but still available free of charge.

2.3 What is free software? What is open-source software?

Free software is a term that was invented by a group called the Free Software Foundation back in the early 1980s. The "free" in free software does not refer to price, but to the idea of "liberty". According to the Free Software Foundation, liberated software is software that is distributed with its *source code* and the author's permission that users of the software can modify and redistribute that source code.

In order to understand the concept of source code, you have to know a bit about how software is written. In order to write a piece of software, a programmer first designs the software, and then chooses a *programming language* to implement the program. A programming language is a set of commands that tell the computer what actions it should take to accomplish a task. The programmer types commands from the programming language into some text files. These text files make up the source code of the program.

In theory, humans that understand the programming language used to write a program can read that program's source code and understand how it works. They can also modify the source code to add new features or fix programming errors.

In contrast, computer cannot follow the instructions in the source code directly. Instead, the computer uses a program (called a *compiler*) that takes the source code and translates it into a form the computer can understand. The translated source code is known as *machine code* or *binary code*. Computers can understand machine code, but humans have a hard time understanding what is happening. Furthermore, while it is straightforward to turn source code into machine code, translating machine code into source code understandable by humans is hard.

This makes source code a powerful thing. If Bill writes a program and gives Sarah the program under an open-source license, then Sarah can both run the program and – in theory at least – Sarah can easily modify the program to make it better. However, if Bill gives Sarah only the binary code, then Sarah can run the program, but – in theory at least – cannot change it. Making source code available means that people all over the world can independently change and improve Bill's software, which makes it easier to improve the quality of software available. Unfortunately, releasing source code makes it hard for Bill to make money, which is why most companies keep their source code hidden. Such software is known as *proprietary*, or *closed source*.

Why do we care about any of this? It turns out that lots of people are willing to write open-source software and release it free of charge. Furthermore, the existence of the Internet means that they can distribute their software all over the world cheaply. In practical terms, that means that there is a lot of good-quality software available free of charge, which in turn means we can install that good quality software on our refurbished computers legally and cheaply. Without free software, we might never have been able to provide legal, cheap software for our refurbished computers.

An important aspect of open-source software is its license. One of the most common free software licences is the GNU Public License, also known as the GPL. You can read the text of the GPL for yourself in the file

`/usr/share/common-licences/GPL`
Some other free licenses are also contained in this directory.

2.4 What is Linux? What is GNU/Linux?

In the early 1990s, a Finnish university student named Linus Torvalds initiated a project to write a free software operating system for his 386. Soon after he started the project, other programmers around the world offered their help. Together, they developed the *kernel* (core) of an operating system, which eventually came to be known as Linux. Today, many people refer to this kernel and its application programs as "Linux".

However, the success of Linux has its roots in another project, called GNU. In the early 1980s, a software developer named Richard M. Stallman initiated the GNU project as an attempt to create a free (open source) UNIX-like operating system. Stallman also created an organization called the Free Software Foundation to aid in the development of GNU.

The Free Software Foundation wrote many free software components for GNU, including a compiler (**gcc**), many utilities (including **ls**, **make**, **bash**) and some important libraries (including **glibc**). These GNU tools became very popular in the UNIX world – many UNIX vendors included these tools with their operating systems. However, the GNU project had no kernel of its own; the Free Software Foundation designed a kernel called Hurd to complete its dream of a totally free-software operating system, but to this date the Hurd is not complete. On the other hand, the Linux project used GNU tools extensively to write and run their kernel.

Together, the GNU tools and Linux kernel form a usable operating system, which many people refer to as "Linux". Other people refer to the operating system as "GNU/Linux", which acknowledges the GNU project's contributions to making Linux usable.

Which label is correct? We don't know – some people get very uptight about calling the operating system GNU/Linux, and others get very uptight about calling the operating system Linux. In this manual we usually refer to the kernel, its utilities and software programs that run on the operating system "Linux", but feel free to substitute "GNU/Linux" for the name if you wish.

2.5 What is Debian?

Debian GNU/Linux is a *Linux distribution*. A Linux distribution consists of the GNU/Linux operating system and applications to run on that operating system, all packaged so that the components run well together. Debian is just one Linux distribution; hundreds of Linux distributions exist. Some of the more popular include Red Hat Linux, Mandrake GNU/Linux, Slackware, SuSE Linux and Caldera OpenLinux.

Our project uses Debian Linux as the foundation of our project; in essence, we install a configuration of Debian that suits our needs. That means we provide a fully-operational Debian GNU/Linux system, which is convenient for users because they can find documentation and install new software on their systems fairly easily.

Our choice of Debian was more-or-less arbitrary; the project's founders used Debian on their home machines. As it turns out, though, Debian is well-suited to our project; it supports older hardware, it is a stable, well-established distribution, and it offers a huge selection of software.

To learn more about Debian, visit
`http://www.debian.org`
or read the Debian FAQ:
`/usr/doc/debian/FAQ/index.html`

2.6 What is the Working Centre Linux Project?

Located in Kitchener, Ontario, Canada, the Working Centre is an employment centre with a twist. In addition to helping people find paid work through employment counselling, job searches and resume critiques, the Working Centre is dedicated to providing resources and support to help people develop new skills and find self-worth regardless of their employment

status. Some supports the Working Centre provides include a community kitchen (where volunteers prepare and serve food), urban agriculture projects (which help people who live in apartments grow some of their own food), a bicycle resource centre (where people can learn how to repair their bicycles, and where they can obtain refurbished used bicycles freely), simplicity circles (where people get together to explore fulfilling lives less dependent on money), and a computer recycling project.

The purpose of the computer recycling project is to refurbish donated computers, then to distribute them cheaply to the community. Almost everybody wins in this situation: donors feel good because their "obsolete" machines will be used instead of thrown into some landfill, volunteers learn how to refurbish used computers and diagnose problems, and members of the community who might otherwise not have access to a home computer can practice marketable computer skills in their own homes.

The computer recycling program faces one serious problem, however: while many donors freely donate hardware, very few donate software licenses for operating systems or applications. Without these software licenses, the computer recycling project cannot legally install commercial software on donated computers. In the end, people purchasing computers through computer recycling are faced with the choice of buying software elsewhere (which often costs more than the system!), or pirating software from their friends.

That is where the Working Centre Linux Project comes in. It is an attempt to install low-cost, legal, useful software on machines the computer recycling project receives.

2.7 What is the design philosophy of the project?

We designed the project around the following principles:

Software should run on donated computers: Many of the systems we currently receive are low-end Pentiums or 486s with smaller hard drives and (most importantly) limited RAM. This rules out KDE or GNOME immediately.

Our minimum requirements for this project are:

- A 486 processor. Speed is not that much of a concern – the system has successfully been installed on a 486DX/33.
- A 400MB hard drive. Currently we do not meet this requirement very well, although 520MB hard drives work fine.
- A VGA-compatible display. 800x600 resolution is nice, but not necessary.
- 16MB of RAM. This turns out to be the most severe bottleneck.

We are working on some package sets that work with smaller hard drives (the TINY selection) and package sets that take advantage of better hardware. As the computer recycling project receives better computers, we will be able to increase the amount of software on the system.

The software should be inexpensive or free: "Free" in this sense refers to zero cost. All of the software on our system is available at no charge, although at least one important component of the system – Opera – is closed-source.

The software should feel familiar: Many of our customers have never used computers before, and those that have computer experience have usually only used some version of Microsoft Windows. For the most part, we want to emulate this look-and-feel – not because it is the best user interface, but because it is the least scary for the typical user.

This principle has a few important side-effects: first of all,

1. Much of our software consists of Windows-lookalikes. Often, this comes at the expense of memory consumption and hard drive space.

2. We decided that the user should not have to drop down to the command line for any day-to-day tasks (although users might have to drop down to the command line to carry out administration).
3. We took the risk of confusing users by offering them a system that *almost* behaves like Windows. This could potentially be more confusing than presenting a system that is completely different from Windows.

The system should not hide UNIX: Although users should not have to drop down to the command line for daily operations, the command line environment should exist and be usable.

This provides a way for interested people to learn more about computers in general and the UNIX design philosophy in particular.

Unfortunately, we were not able to adhere to this principle religiously. In order to save hard drive space, our smaller distribution omits some UNIX mainstays, such as the **gcc** development tools and **Emacs**.

Data files should be compatible with Windows apps: On their home machines, users should be able to save their data files to a floppy, then load those files at the Working Centre, at the library or at any other computer where they might want to work. Since most of those other computers will be running Windows applications, compatibility is really important.

In practical terms, this means that in our applications we tried to set default save formats to ones that Microsoft products would be able to read. See Section 8.1 for more information about the changes we made.

Volunteers must be able to install the system: Many of the volunteers at the computer recycling projects are not familiar with Linux. We designed the system so that they would easily be able to put our software on their machines.

To this end, we created an automated installer based on FAI (Fully Automated Install). This installer removes a lot of the flexibility of other Linux installers, but makes installation a lot easier. To learn more about this installer, see Section 6

The project should be easy to administer: There is no point in setting up a software project if the resources do not exist to maintain and administer it. Since the computer recycling project is volunteer based, it was important to set up a system that could be maintained by volunteers.

Notice that none of these guiding principles force us to use Linux or open-source software. Indeed, we explored several different alternatives to getting cheap software on our systems: DOS-based freeware and open-source software, "abandonware", and licensing arrangements with Microsoft. In the end, we chose Linux because a lot of quality, no-cost, permissively licensed software was available for it, and because some of the volunteers (namely the ones putting together the distribution) were familiar enough with Linux to put a good set of packages together. In that sense, this project is an example of unintentional Linux advocacy.

3 Pre-installation tasks

3.1 Does the user really want Linux?

(Flesh this out. -P.)

We set up this project in the hope that users would be able to get free, useful, easy-to-use software for their new computers. However, some users may not want to use Linux. You should point out the advantages and disadvantages of installing Linux on their new computer. Be honest. Don't try to force Linux onto people who do not want it: that hurts the images of the Computer Recycling program, Linux and the Working Centre.

Some reasons why users might want our Linux selection on their system include:

- The user does not have software of his or her own to install. Our standard installation allows users to carry out the following tasks:
  - Word processing
  - Web surfing, e-mail and newsgroup access, and instant messaging (Note that the user has to arrange ISP service by themselves to do this.)
  - Game playing

  There are all sorts of other programs available for our Linux distribution as well. See Section 11.1 for other examples.
- The user does not have money to purchase software elsewhere. All of the software in our distribution is available free of charge.
- The user wants to "learn about computers." Our software allows people to learn about computers at a high level (by using graphical applications) and at lower levels (by playing with command line utilities, or by programming).

  Many of the skills users learn on their Linux system are transferable to other operating systems.
- The user has heard of Linux and wants to use it, but does not know where to start.

Some reasons why users might not want our Linux selection on their system include:

- The user wants to run Windows software on the system. Linux is not Windows, and for all practical purposes you cannot run standard Windows software on these systems. We provide some software that produces files compatible with those produced by Windows applications, but we do not provide standard Windows applications (for example, Microsoft Office) themselves.

  Also keep in mind that compatibility between our programs and their Windows counterparts is not always perfect. For example, **Abiword** will read Rich Text Format files, but it does not handle tables very well.

  Another consideration is that the user might be studying for Microsoft-specific certifications, such as MOUS (Microsoft Office User Specialist). In such cases, the user is better off using Microsoft software, which corresponds exactly to the training material.
- The user already owns software for the system, or plans to get software for the system themselves.
- The user wants UNIX-like Linux on their system. In such a case, the user might be better off installing a more typical Linux configuration from a standard distribution. Since we do not hide the UNIX layer, our software will work – but to conserve space we do eliminate some common UNIX software (for example, Emacs and gcc) found on most Linux systems.
- The user wants to run intensive applications, such as
  - Modern 3D games
  - Heavy programming – compiling big projects will take a long time on lower-end processors. However, you can *learn* programming on one of our machines.
  - MP3 playing
  - MPEG decoding for watching movies

  In these cases, I question why the user wants one of our systems, since the hardware limits the user's capabilities.

  It is worth noting that 486 systems will have problems playing MP3s, but higher-end Pentiums (133MHz and up) will be able to manage. Also, any computer can play CDs from a CD-ROM.

- The user already owns hardware that is not supported by Linux. Some printers and scanners fall into this category, but Linux support is getting better for these things.

3.2 Get user configuration decisions

We have a form to gather the user preferences listed below. It is titled "Your System's Configuration." You can have users write down their preferences on this sheet.

Some of the preferences you will need to gather include:

- Whether the user wants separate accounts for each user, or whether the household will use a single account.

  If there will be separate users, then it makes sense to get user names for each person who will have an account. That way you can create accounts for the user.

- A name for the machine. This is called a *hostname*. It is not so important, but hostnames are easy to set in Linux and it's fun to give machines names.

- Whether the system should log in automatically upon bootup. This makes the most sense for a single-user account, but it is possible to configure **gdm** to log in a user after waiting for a specified amount of time. The most usual way to do this is by enabling the AUTOLOGIN class. For more information, see Section 4.3.1 and Section 5.3.2 .

- Whether the user will be using this computer to connect to the Internet. If the user plans to connect to the Internet, you should find out whether the user plans to use a dial-up account or broadband (cable or ADSL) access. The user might not know the difference between the two; in this case you should install both PPP configuration software, and any broadband configuration software we have.

  If the user already has ISP information available, you should take down this information so you can pre-configure the system to connect to the Internet. See Section 5.9.1 for more information.

- What additional peripherals – if any – the user will use with this machine. Some common peripherals include printers, scanners and webcams. If the user already owns the peripheral in question, you should see whether the user is willing to bring its information up so you can look it up in the Linux Compatibility HOWTO. **(Where is this? -P.)**

- What additional software the user would like installed. Often users will not know the names of specific programs, but if they do you can install these packages. More often the user will know what they want to do on the machine. Then you can look around for appropriate packages to install. We have listed some software we have tried out in Section 11.1 . Section 8.6 lists some ways to search for new software packages. **(point to synaptic-searches instead -P.)**

  Feel free to demo packages on a test machine (or the server), so that the user can decide whether they want the programs or not. If you are installing packages on the server, however, you should limit yourself to Debian packages – and don't break the server!

  You should *not* install illegal software on the user's machine.

  Also keep in mind hard drive limitations when deciding to install software. A system that comes with a 400MB hard drive will not have the capacity to store much additional software.

- Whether the user *does not* want any of the standard components installed. In most cases the answer to this will be "no." However, some users may know that they will never use certain software, in which case you may be able to remove it. However, the standard installer currently does not support this level of customization; you have to go "under the hood" and edit some files yourself. To learn more about this, see Section 8.6

3.3 Get hardware information

In order to configure the system you need some hardware-specific information. Information you need includes:

- The make of the monitor, and its vertical and horizontal refresh rates. If you cannot find this information (In Appendix A.2 or on the Internet) then you have two choices: either get a new monitor or hope that the monitor has reasonable ranges.
- The type of mouse (PS/2 or serial), and preferably the mouse protocol it uses. There exists a program called **mdetect** that might help you figure out mouse protocols.

  Often, serial mice will use the "Microsoft" or "Mousesystems" protocols, and PS/2 mice will use the "PS/2" protocol.
- Whether the keyboard is a standard 101-key type, or whether it is a 104-key keyboard. The latter type comes with extra "Windows" keys.
- The make of the video card. You need this to decide which *X-server* package to use, and to figure out the maximum resolution and colour depths the card supports. On the server, type

  ```
  zcat /usr/share/doc/xserver-xfree86/Status.gz | less
  ```

  to see information about X-servers. You can also use the **detect** program to determine the make of the video card.
- The size and number of hard drives. Currently we only support installations on one hard drive, but this may change in the future.
- The make and model of the sound card, if one is to be installed.

Information that is not always needed but is nice to have includes:

- The make of any peripherals, such as printers.
- The make and model of the network card that will be used for installation.
- The make and model of the modem, if one is installed. ISP information is useful as well.

3.4 Install a network card

Since the installer works off the network, you will have to ensure that the client machine has a network card. If there is no network card in the machine already, then you should install one. Most ISA or PCI network cards will work with the installer.

4 Using the Installer

We designed the installer to be fairly easy to use. Most of the installation process consists of waiting while the client downloads and installs packages. However, once the installer has finished you need to carry out some manual configuration.

The installation procedure takes about 45 minutes on a Pentium 90, and about an hour and a half on a 486-66. Fortunately, most of the installation can be left unattended.

This section describes the steps you need to follow in order to complete a typical installation.

4.1 Booting with a floppy

Each FAI boot floppy should be marked with an IP address. This IP address will be assigned to the client. Make sure that no other computer is using this computer (that is, that no other currently-installing client used this boot floppy).

In the BIOS of the computer, check that the boot sequence will boot from floppy first. Then put in the boot floppy and (re) start the machine.

4.2 Startup

The computer should boot from the floppy. First the computer loads a Linux kernel from the floppy. At this point you should see a message similar to:

```
LILO 22.1:  Booting FAI....
```

Once the kernel has loaded off the floppy, the client boots into Linux and tries to connect to the server. At this point you will see a message like the following flash by:

```
------------------------------------------------------
FAI 2.3.1, 16 apr 2002
Fully Automatic Installation for Debian GNU/Linux

Copyright (c) 1999-2002, Thomas Lange
lange@informatik.uni-koeln.de
------------------------------------------------------
```

At this point, the installer takes control of the machine.

4.3 Customization

You should now be presented with a few configuration screens. In each of these screens, use the space bar to toggle selections, and press ⟨enter⟩ to move to the next screen.

(Screenshots? -P.)

4.3.1 Additional classes

The first screen allows you to select additional FAI classes that should be installed on the system. Depending on the size of the system's hard drive, some classes available on this screen will be defined regardless of your choice. For more information about this process, see Section 6.3.1

These selections do not take into account hard drive size, so you should make sure that the hard drive is sufficiently big to hold any extra packages you install.

Unless the user specifically requested certain packages, it is best to stick with the default applications. However, you may need to select additional classes in the following situations:

- If you are installing a sound card, you will want to select classes to help you configure and use the card. Read Section 5.8 for more information.

By default, this screen defaults to selecting the following classes:

AUTOLOGIN: This class configures GDM to log in a user account automatically. See Section 5.3.2 for more information.

4.3.2 X-server selections

In order for the X-Window graphical environment to work, a video card driver (called an "X-server") needs to be selected and installed. There are several versions of X-Windows available for Linux. The version we use is called XFree86.

Selecting the right X-server can be tricky. One complication is that Debian currently supports two versions of XFree86 – version 3.3.6 and the newer version 4.1.0 . Many older video cards that work with XFree86 3.3.6 do not work with 4.1.0 . Thus, you have to do some detective work to find out which package to install:

1. First, figure out the make and model of your video card. Ideally, you have done this before you started the installation. If not, switch to the second virtual terminal by pressing ⟨alt⟩-⟨F2⟩ and type

   ```
   discover --format "Card %V %M, Xserver:  %S" video
   ```

If you are lucky, this will show you the make and model of the video card. If you are unlucky your display will go blank, and you will have to reboot the system to see anything.

Another detection program is called **detect**. To use it, type

```
detect
```

If all goes well, a file called `report.txt` will be generated in the current directory.

2. Once you know the make and model of the video card, go to the server, open a terminal window, and type

   ```
   zcat /usr/share/doc/xserver-xfree86/Status.gz | less
   ```

   This will display a file containing the status of all video cards supported by XFree86. Look up the card in this list. You will discover one of three things:

   (a) The card is supported by XFree86 4.1.0 . In this case, you want to note the driver for the card, and select the XSERVER-XFREE86 package (which contains all drivers for XFree86 4.1.0).

   (b) The card is not supported by XFree86 4.1.0, but is supported by 3.3.6 . In this case, you have to look up the xserver package you need to install.

   (c) The card is not supported by either XFree86 versions 3 or 4. In this case you are out of luck, and you should swap the video card for something that is supported, or swap the computer altogether.

3. Once you know the X-server you need, go back to the installer (by pressing ⟨alt⟩-⟨F1⟩) and see whether the package you need is available. If it is, select it. If it isn't, then you will need to install the X-server later. See Section 8.6 to learn about installing extra packages.

It is possible to install multiple X-servers. However, this will just waste disk space, since you can only *use* one X-server at a time.

Keep track of which X-server you installed. You will need this information when configuring the system (see Section 5.2).

4.3.3 Hostname

In this screen, you are asked to provide a hostname for the machine. A hostname is the computer's name. If your user named the new system, type in this name. Otherwise stick with the defaults.

Conventionally, hostnames are made up of a combination of lower case letters, numbers, and dashes ('_' and '-'). You may be able to use other characters as well, but definitely should not use periods.

(Restrictions? -P.)

4.4 Automated install

After configuration, the installer takes over. First, it partitions and formats the hard disks:

```
   Calling task_partition
Partitioning local harddisks
```

You will probably see a few screenfuls of a line similar to the following:

```
modprobe:  modprobe:  Can't locate module block-major-105
```

These messages are harmless.

Next, the installer starts installing software. You will see the installer print out a large list of software that needs to be installed. The installer will first download each of these packages from the server. Then it will install these packages. First, packages are unpacked:

```
   Selecting previously deselected package libgtk1.2-common.
Unpacking libgtk1.2-common (from .../libgtk1.2-common_1.2.10-11_all.deb) ...
```

When all packages have been unpacked, they are configured:

```
Setting up libgtk1.2-common (1.2.10-11) ...
```

Finally, the installer runs some customization scripts and makes the system bootable. When the installer has completed you will hear a beep and see the message:

```
FAI finished.
Calling task_chboot
Calling task_savelog
```

At this point you can reboot the system.

4.5 Checking the installation process

4.5.1 Restoring a blank screen

When the keyboard has not been touched for a few minutes the screen goes blank. This is just Linux saving energy. To restore the screen press an arrow key (pressing ⟨enter⟩ is not such a good idea because you will reboot the system if the installer has finished).

4.5.2 Using the second virtual terminal

During the install, you can check what is going on by pressing ⟨Alt⟩-⟨F2⟩ . This activates the second *virtual terminal*. You should see a command line prompt, which you can use to check the system.

Some useful things you might want to do include:

- Running the **top** program to check on memory and CPU usage. You can start **top** by typing `top` , and get out of it by pressing `q` .
- Looking at the messages the installer has printed. To do this type

  ```
  less /tmp/fai/rcS.log
  ```

  To get out of **less**, type `q` .

  There are lots of other log files and configuration files in `/tmp/fai` as well. See Section 6.1.3 for more information about these logfiles.
- Looking at disk usage. Type `df` to see this. The local hard disk(s) will be mounted to directory `/tmp/target` .

To get back to the installer's main screen, press ⟨Alt⟩-⟨F1⟩.

4.6 Reboot the system

Remove the boot floppy from the computer's drive, then press ⟨enter⟩. The system should reboot.

Before rebooting, you may see the following message:

(write down the message -P.)

It is safe to ignore this.

The system should boot into Linux, and you should get a login prompt. If the system does not boot, the installer failed.

If the system does boot properly into Linux, you can log in as the root user and continue configuring the system.

5 Post-installation tasks

We were not able to automate the Linux installer completely. When you reboot the system, you have to configure some things manually. This section documents things you have to do to complete the computer's configuration.

5.1 Log in

To configure the system you will need to be the root user. The initial password for this account is "fai". You will want to change it once the rest of the configuration is done. See Section 5.6 to see how to do this.

5.2 Configure X-Windows

During the installation process you chose an X-server to match the video card in the system.

At this point, you want to configure the X-server so that graphical mode will work. First, you need to gather information about your computer. Next, you will need to configure the X-server using `dpkg-reconfigure` . Finally, you have to test the configuration out.

5.2.1 Information you want

In order to do this, you will want the following information:

- The X-server you installed. This is the name of the package you made in Section 4.3.2, specified in lower case letters. For example, if you selected an X-server package called XSERVER-FOO, the corresponding X-server is called "xserver-foo".

  At a command prompt, you can see which X-server packages are installed by typing

  ```
  dpkg -l xserver-*
  ```

- Information about which resolutions and colour depths your video card supports, if you can find it.

- Information about the monitor that will be used with this system. Having the vertical and horizontal refresh rates of the monitor are best, if you can find them. See Appendix A.2 for more details.

- Information about the mouse you will use. You will need is whether the mouse is PS/2 or serial. For serial mice, you will want to know which serial port the mouse will use.

5.2.2 dpkg-reconfigure

Once you have as much useful system information as you can find, type

`dpkg-reconfigure` *`xserver`*

where *`xserver`* is the xserver you are using (in lower case). For example, if we had selected XSERVER-FOO, we would type

```
dpkg-reconfigure xserver-foo
```

This will unleash a storm of configuration screens. Fortunately, you will be able to use the default settings for all screens except the following:

Keyboard: You will have the option of entering a keyboard type. The default will be labelled "us104". This corresponds to a keyboard with extra "Windows" keys. A regular keyboard without these keys is "us101".

In fact, it does not seem to make much difference which keyboard type you use.

Resolutions: This screen will ask which screen resolutions you want to support. Most video cards we see support at most 800x600, and many support only 640x480 resolution. Try unchecking all resolutions except 800x600 and 640x480.

The up and down arrow keys move through the list. The space bar toggles selections. Press ⟨Tab⟩ to select **Ok** option and move on.

Color depth: This screen will suggest using a color depth of 24 bits, which corresponds to "millions of colours" in Windows. Many video cards don't support this depth. A colour depth of 8 (256 colors) works for any VGA card, and better cards work with a color depth of 15 or 16.

Monitors: If you have the vertical and horizontal refresh rates, choose the "Advanced" configuration and enter in these numbers. Otherwise, use the "Simple" configuration option and follow instructions.

Mouse device: This screen presents a cryptic list of `/dev` files, and expects you to know which of these files is connected to your mouse. This is easier than it looks:

- `/dev/psaux` is the PS/2 port, so select this if you have a PS/2 mouse.
- `/dev/ttyS0` is the first serial port (COM1). Most serial mice are connected here.
- `/dev/ttyS1` is the second serial port (COM2).

Mouse protocol: This is one of the most frustrating steps in the installation process. You are presented with a list of mouse protocols, and you have to choose the right one.

Most mouse manufacturers hide the fact that mice speak to computers in different ways. Windows hides this fact as well. In the end, you have to experiment. The following guidelines might help:

- PS/2 mice often use the PS/2 protocol. (Surprise!)
- Most serial mice use either the Microsoft or MouseSystems protocol. Try the Microsoft protocol first, then if that does not work try MouseSystems.

For more mouse-y fun, see the Mouse-HOWTO.

(where is the mouse HOWTO?? -P.)

5.2.3 Testing the configuration

Once you have finished configuring the X-server, it is time to test the configuration out. At a command prompt, type

```
startx
```

You will see a whack of messages flash by. Then one of the following things will happen:

- Your configuration will work. You will see a grey screen first, and then the IceWM window manager will start up and present the desktop. You will be able to move the mouse. All of the mouse buttons will work properly. In this case you are done. Hooray!
- The computer sits and think for a while, then returns you to the command prompt with an error message. **(Document the messages -P.)** This could be caused by a number of things:
  - You chose an incorrect X-server. Find the right X-server package and install it. See Section 3.3 for more information.
  - You chose a colour depth that was too high.
  - You configured the monitor so that it would not synchronize with the video card. Sometimes this is because your monitor configuration was too strict. Sometimes the monitor and the video card just don't work together very well. In the latter case, it is easiest to find a new monitor to use with the computer.
- The screen is garbled. This often means that you chose a resolution too high for the monitor to support.

- The mouse pointer does not move. In this case you specified the wrong port for the mouse, you chose the wrong mouse protocol, the mouse does not work at all, or the mouse does not work under Linux. Unfortunately, all of these things happen.
- The mouse moves, but acts crazy – you left-click once to open a terminal window, and it opens two or three instead, or the movement is completely erratic. In this case you chose the wrong mouse protocol.

The keystroke ⟨ctrl⟩+⟨alt⟩+⟨backspace⟩ is your best friend when configuring X-Windows. This keystroke gets you out of graphical mode and back to your command prompt.

If your configuration was incorrect, you have to repeat the configuration process by using `dpkg-reconfigure` again. Note that the configuration program will remember your previous selections and offer them as defaults if you re-reconfigure the X-server.

Configuring X-Windows in Debian is frustrating and time-consuming. Fear not! It gets easier and faster with practice.

If you are using the 3.6 versions of XFree86, you may want to trap the output of the X-Server. To do this, type

`startx >& /tmp/x.out`

and look at the file `/tmp/x.out` . For version 4 of XFree86 and higher, a log of the X-server output is kept at

`/var/log/` **(finish me! -P.)**

5.3 Enable and configure GDM

GDM stands for "GNOME Display Manager". For a long time we thought the name was misleading and that GDM did not require GNOME to run. We were wrong – GDM costs several dozen megabytes of extra libraries to run. However, it is the only login manager we know of that permits easy automated graphical logins.

The Debian bootup process is divided into *runlevels*. Runlevel 0 is run when halting the computer. Runlevel 1 is for single-user (administrator) mode. Runlevels 2-5 are user-level runlevels. By default they are identical. Runlevel 6 is used when rebooting the computer. You can learn more about runlevels by looking at the Debian FAQ, located in the directory:

`/usr/doc/debian/FAQ/`

Ordinarily, GDM program runs when the system is booted into runlevels 2-5. Its job is to start X-Windows on bootup and manage a graphical login screen. It is also capable of automatically logging in some user account on bootup.

5.3.1 Re-enable GDM

The installer *disables* GDM during installation by removing symbolic links from the runlevel directories. Allowing GDM to run before X-Windows has been configured properly causes a lot of frustration – if X-Windows is not working correctly, it can be nearly impossible to log in and fix things.

Before re-enabling GDM, it is a good idea to get X-Windows working.

To re-enable GDM, become the root user and type

`update-rc.d gdm defaults 99`

GDM will then run the next time the computer is rebooted.

5.3.2 The AUTOLOGIN class

In the class selection dialog (documented in Section 4.3.1) there is an option labelled AUTOLOGIN. If this option is selected, the installer will create an account named `linuxuser` , and configure GDM to log in this account upon bootup.

Selecting this class makes a lot of sense for users who do not want to bother remembering a username and password, so long as they can live with a single account for the whole system.

5.3.3 Configure GDM to log in an account automatically

The user may have specified that he or she wants an account to be logged in automatically upon startup. If you did not choose the AUTOLOGIN class (or if the user wants a different account logged in) you can configure this behaviour in GDM:

1. At the GDM login screen, you can choose the **System** → **Configure** menu item.
2. **gdmconfig** will prompt you for the root password. Enter it.
3. You will see a configuration menu. On the left there is a sidebar with selections **Basic**, **Expert**, and **System**. Select **Basic**.
4. In the right window you will see a row of tabs. Select **Automatic login**.
5. In the **Timed login** section, click the "Login a user automatically after a specified number of seconds" option. In **Timed Login** enter the login name of the account to log in. Setting the **Seconds before login** box to 15 seconds is a good idea so that other users (such as the root user) have an opportunity to log in manually.
6. Click **Apply**, then **Restart after logout**.
7. Click **Ok**, and you are done.

(PICTURES -P.)

You can also run the GDM configuration program when logged in. Start X-Windows, log in, and then run the **gdmconfig** program.

5.4 Create user accounts

Some users will want several accounts created on their systems. This makes sense when several different people will be using the system. In such cases, you will want a list of logins and names of the people associated with those logins.

Say we want to create the following account:

LOGIN NAME	FULL NAME
`daveb`	Dave Bright

To create this account, type the following at the command line:

```
adduser
```

The system will copy all the configuration files, and prompt you for a password and real name. It will also prompt you for a "Room Number", "Work Phone", "Home Phone", and "Other". You do not need to fill any of these in.

Often it is best to let the users to type in passwords for their own accounts if possible. That way you will not need to see the passwords they have chosen.

If you want to create an account with no password (for example, an account that will be logged in using gdm), then type

```
adduser --disabled-password
```

Currently we do not force users to choose sensible passwords. This may change in the future.

5.5 Reboot and test

Now it is time to test the system out. At the command prompt, type

```
shutdown -r now
```

This will reboot the system. At the end of the boot process the system should enter graphical mode. If you configured GDM to log in an account automatically, the account should log in. Otherwise, you should see the GDM login screen. Try logging in with a user account if you can. Otherwise, log in as the root user.

5.6 Change the root password

If you logged in under a user account, the first thing you will have to do is become the root user. Click the computer icon on the taskbar to open a terminal window. Then type

```
su
```

and enter the current root password. The command prompt should have a hash mark (#) at the end.

Next, you want to change the root password. Typing

```
passwd
```

will allow you to do this.

Once you have changed the root password, open up another terminal window, and type

```
su
```

again. Then log in as the root user using the new password. This allows you to make sure that the change took effect properly.

Once you have changed the root password, you will want to write this password down on the "Your System's Configuration" page.

5.6.1 Sensible passwords and security

First, a lecture: The root account is really powerful. It provides open access to the entire Linux system. Using the root account people can do all sorts of destructive things, such as deleting the contents of the hard drive, or snooping around wherever they want.

Security is a big concern for systems that will be accessing the Internet – especially for systems that connect to the Internet using DSL or cable modems. In such cases it is really important to choose good root passwords – a weak root password defeats any other security on a system. **(refer to SECURITY HOWTO -P.)**

Why is this important? For one thing, a cracker that breaks into systems connected to the Internet can use those computers to launch *Denial of Service* attacks. Put simply, a Denial of Service attack is a way of taking lots of computers and using them to clog up a site on the Internet, crashing the target computers or slowing them down immensely.

How do you choose a good password? The `passwd(1)` manual page provides some hints. Type

```
man passwd
```

and scroll down to the section labelled "Hints for user passwords". Other things to try include:

- Mixing lower and uppercase letters with numbers and punctuation.

5.6.2 Automatic password generation

You can use the **apg** command to generate relatively-secure passwords. To do this, log onto the server and type

```
apg -M SN
```

This will generate a few passwords which you can use for the root account and/or user accounts. The passwords are designed to be pronounceable, and a suggested pronounciation follows each password.

The **apg** program has other options as well, which you might want to investigate. See the **apg** manual page for more information.

5.7 Set up printers

In version 0.5 of the WCLP project, we switched our printing system from **pdq** to **lprng** . It turns out that **lprng** performs much better on the low-end computers we refurbish.

To set up **lprng** we use a graphical installation program called **lprngtool** . To control printers, users can use the program **printop**, which they can access as the **Printer Tool** in the IceWM "start" menu.

You can use **lprngtool** to set up a printer as follows:

1. In graphical mode, become the root user and type

   `lprngtool`

2. You will get a message stating that the `/etc/printcap` file has been modified. This is fine.

3. You will get a message that `A SMB client does not appear to be installed` . Select **Ignore** and the message will go away.

4. Select **Add**

5. Select **locally attached printer**

6. You will be presented with a window of configuration options. Leave all options at their defaults except the **Input filter** option. Select that.

7. Yet another window will appear. Select the printer's model from the list. For dot matrix printers, the Epson dot-matrix printer drivers almost always work – select 9 pin or 24 pin as required. Many other common printers are supported.

   Unfortunately, we are not really certain how to add new printers to the standard **lprngtool** database, so if your printer is not listed you may be out of luck. On the other hand, many printers can emulate a more common printer (for example, many PCL laser printers emulate HP Laserjets). If you can figure out which printers are similar to yours, you may be in luck. The website

   `http://www.linuxprinting.org`

   may be of some help. Alternatively, search for the model name of the printer and the word "linux".

8. You may want to select the **Send EOF at end of job to eject page**, especially for dot matrix and bubblejet printers.

9. Select **OK** then **Modify** . Then select **File** → **Restart lprng**

10. Once you have restarted the daemon, click on the new printer definition and select **Tests** → **Print Postscript test page**. If this fails try printing an ASCII test page – this page contains useful diagnostic information you can use to fix the print filter options.

11. Once you are happy, exit the program, log in as a regular user, and verify that you can print. You should be able to print from any application, such as **abiword** or **opera** .

    The print job should come out relatively quickly. If everything seems to hang, you probably want to use **printop** to see whether the job has been sent to the printer, and where **lprng** is getting stuck.

If you encounter weird situations with printers, please let us know so that we can document those situations in an Appendix.

5.8 Set up a sound card

Some users request that we set up sound cards with their systems. In this case, you need to set up that card. In Linux, sound card configuration is a royal pain. However, with a little luck and a lot of patience it can be done.

In order to get a sound card working, you need to set up the following things:

1. A working sound card supported under Linux.

2. A way for Linux to recognise the card's settings. This may mean setting up jumpers, or you may have to use the **isapnp** tools.

3. Low-level sound card drivers. Two classes of drivers are available: OSS/Lite drivers and ALSA drivers. We prefer ALSA drivers, but there are advantages to both driver systems.

4. User permissions so that regular users can use sound facilities.

5. Programs to use the sound card, such as CD-ROM players, MIDI and MP3 programs.

In the following sections we explain each of these steps in more detail.

5.8.1 Getting a working sound card

Before installing a sound card, you should check that it is supported under Linux. Ways to do this include:

- Consulting the Linux Compatibility HOWTO on the web. **(link? -P.)** Note that this site lists OSS/Lite support for sound cards.
- Visiting `http://www.alsa-sound.org` to look up your card's ALSA support. **(Can this information be found locally? -P.)**
- Installing the `libdetect0` package, and running the **detect** program as root. This will generate a file called `report.txt` in the current directory. If you are lucky, this will tell you the make of the sound card and what OSS driver it uses.

5.8.2 Setting up resources using isapnptools

Like any other hardware, sound cards need to use resources in order to work properly. Unlike other hardware, Linux does not do a great job of automatically allocating or recognising sound card resources automatically. In order to make your sound card work, you may have to explicitly set the IRQ, IO ports, and DMA channels for your sound card.

If your sound card is an ISA card with jumpers, you should set the card so that it uses a free IRQ and IO port address.

If your sound card is an ISA Plug-and-play card, you may have to use **isapnp** and **pnpdump** to set the card's resources. The easy way to do this is to type the following:

```
pnpdump -c > /etc/isapnp.conf
isapnp /etc/isapnp.conf
```

What is happening? First, the **pnpdump** utility is detecting all plug-and-play cards on the system, and attempting to automatically generate a configuration file that will Just Work. Then the **isapnp** utility tries to use that file to set the resources on the card and let Linux know that the card exists.

If you are lucky, **isapnp** will print out messages that indicate the sound card configured properly:

(Include these messages here.. -P.)

If you are not lucky, the program will indicate an error. In this case, you will have to edit the `/etc/isapnp.conf` manually. First, regenerate the file:

```
pnpdump > /etc/isapnp.conf
```

Next, edit the file:

```
nano /etc/isapnp.conf
```

Basically, this involves reading the comments and following the instructions to uncomment (remove the hash marks from) certain lines. It is not hard, but you have to be aware of the structure of the file: each card is broken down into one or more *logical devices*. There are usually about four logical devices in a sound card. Each logical device consists of a block starting with the line `LD` and ending with `ACT Y`. In between there are a bunch of `INT`, `IO`, and `DMA` lines that are commented out with a hash mark.

For each `LD` line, you want to uncomment one `INT` line, one `IO` port line, and one `DMA` line. Then you have to uncomment the `(ACT Y)` line at the bottom of each `LD` section.

Here is an example:

```
#
# Logical device id CMI0003
#     Device supports vendor reserved register @ 0x39
#     Device supports vendor reserved register @ 0x3a
#     Device supports vendor reserved register @ 0x3c
#     Device supports vendor reserved register @ 0x3e
#     Device supports vendor reserved register @ 0x3f
#
# Edit the entries below to uncomment out the configuration required.
# Note that only the first value of any range is given, this may be
# changed if required
# Don't forget to uncomment the activate (ACT Y) when happy

(CONFIGURE CMI8329/67109120 (LD 3
```

This specifies logical device LD 3. Following this are several possible configurations:

```
#     Start dependent functions: priority preferred
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x0530
#             Maximum IO base address 0x0530
#             IO base alignment 1 bytes
#             *** Bad resource data: Base alignment 0 - changed to 1
#             Number of IO addresses required: 8
    (IO 0 (SIZE 8) (BASE 0x0530))
#       IRQ 11.
#             High true, edge sensitive interrupt (by default)
    (INT 0 (IRQ 11 (MODE +E)))
#       First DMA channel 0.
#             8 bit DMA only
#             Logical device is not a bus master
#             DMA may execute in count by byte mode
#             DMA may not execute in count by word mode
#             DMA channel speed in compatible mode
    (DMA 0 (CHANNEL 0))
```

This configuration uses an IO address of 0x0530, an IRQ of 11 and DMA channel 0. This configuration option is selected, because the lines beginning with IO, INT, and DMA are uncommented. Note that you want to uncomment lines in the same block – don't mix and match between configuration blocks.

Finally, you have to uncomment the following line for each logical device:

```
#     End dependent functions
 (NAME "CMI8329/67109120[3]{AD-CHIPS Audio Adapter}")
# (ACT Y)
))
```

In this example, (ACT Y) is still commented, so **isapnp** will ignore this section. You want to make sure the (ACT Y) lines for each logical device are uncommented.

When you have edited the file, run **isapnp** on the configuration file:

```
isapnp /etc/isapnp.conf
```

If you are lucky, each logical device will be recognised, and appropriate status messages will print on the screen.

If this does not work, you have to re-edit the file, re-comment some of the lines you uncommented, uncomment some new lines, and try again.

When you have a working configuration, write down the resources you have used.

For hints on how to determine which IRQs and IO ports are free, see the "Determine resource usage" section in the **isapnp** FAQ. To access this FAQ, type the following:

```
cd /usr/share/doc/isapnptools/
zcat isapnpfaq.txt.gz | less
```

5.8.3 Low level drivers: ALSA vs. OSS/Lite

Crazy hackers have developed two popular sets of sound drivers for Linux. The OSS/Lite driver system came first. This system supports a lot of sound cards and is integrated into the Linux kernel. However, it is a pain to configure, and (supposedly) the sound quality is not as good as the ALSA drivers.

The ALSA project was an attempt to improve sound support under Linux. ALSA does not support as many sound cards as OSS/Lite does, but the setup is a bit easier and the quality is supposedly better. ALSA support will be integrated into the Linux kernel for the 2.6 series of kernels.

Our preference is to use the ALSA drivers whenever possible. However, the drivers you use probably do not matter all that much.

5.8.4 Configuring ALSA drivers

We support the ALSA drivers in the SOUND_CONFIG class. This class contains the ALSA drivers themselves, and some tools to help set up these drivers. You should select this class during the install, in the "Additional Classes" screen (see Section 4.3.1 for more information).

To set up the ALSA drivers, first configure the resources for the card, then type

```
alsaconf
```

You will first have to select your card from a list of supported cards. At this point, the configuration program should autodetect the resources you set up, and set up the drivers. Finally, the program plays a sound sample.

If any of these steps fail, you probably either chose the wrong sound card to set up, or you did not configure sound card resources properly.

Once ALSA has been set up, you can use the following programs to play sounds and control the card:

- The ALSA mixer **aumix** allows you to control the volume of the sound card.
- **aplay** plays sound files from the command line.
- A program called **alsaplayer** supposedly plays all kinds of sound files, including CDs and OGG files.

5.8.5 Configuring OSS/Lite drivers

We support the OSS/Lite drivers through the OSS_CONFIG class. It contains the **sndconfig** utility to help you configure the drivers. As OSS/Lite sound is integrated into the kernel, you do not actually need this package to configure sound support. Indeed, this package takes up several megabytes of hard drive space. However, you might find the **sndconfig** utility easier to use than setting up sound manually.

As with the SOUND_CONFIG class, choose this class in the "Additional Classes" setup screen.

To use the **sndconfig** program, type

```
sndconfig
```

You must then select the sound card, and select the resources the sound card should use. These resources should match the jumper settings or **isapnp** settings.

Once you have completed your selection, the program will play a sound sample for you. If you do not hear the sample, run **sndconfig** again and change your configuration settings.

Once you have successfully set up the card, you may want to remove the **sndconfig**, **kudzu** and **hwdata** Debian packages from the system. This will save a few megabytes of space. Removing these packages should be safe, but we have not checked this.

5.8.6 Installing sound programs

(SOUND_APP class. -P.)

5.9 Set up a modem

Modems are a pain to set up under Linux. In order to get a modem working, you need to go through the following steps:

1. If the computer's BIOS supports Plug-and-Play, then you may want to disable PnP support for the modem and its resources.
2. If the modem is internal, you may have to set jumpers on it.
3. You must physically install the modem (which should be straightforward).
4. If the modem does not have jumpers, you must set up `/etc/isapnp.conf` to configure the modem.
5. You must get Linux to recognise the serial port the modem is using. This involves editing the file `/etc/setserial.conf`
6. You must configure software to recognise the modem and dial out to the user's ISP. Currently, we use **wvdial** or **pppconfig** for this.
7. You might consider setting up a graphical dialer program to make life easy for the user. Our current favorite is **gkdial** . Another option is to try **diald**, which sets up PPP connections on demand.

In the following sections, we will go through modem setup in some detail. Modems are tricky beasts, however, and these instructions certainly don't cover every case you will run into. To learn more about setting up modems, you will want to read the Modem-HOWTO . You can look at this HOWTO by typing

```
zcat /usr/share/doc/HOWTO/en-txt/Modem-HOWTO.gz | less
```

on the server.

In this section we focus on setting up internal modems. Setting up external modems is similar, but easier – you can usually plug in the modem into a free COM port, then configure dialer and ISP information directly.

5.9.1 Information you need

Before you start, you need some information:

- A guarantee that the modem is a hardware modem, and *not* a Winmodem (otherwise known as a software-based modem). Winmodems are cheap, but more-or-less useless to us. Some Winmodems are now supported under Linux, but even these drivers consume a lot of CPU cycles, and are probably inappropriate for the computers in our project.
- Whether the BIOS of your computer supports Plug-and-Play.
- Whether the modem uses jumpers for configuration, or whether it is Plug-and-Play. If the modem uses jumpers, you need to find out what the jumper settings are.

- Which IRQs are free on your system. To see which IRQs are currently being used, type

  `less /proc/interrupts`

  Note that this only shows IRQs in active use. If there are devices that use IRQs but are not active, they will not show up in this list.

- The serial ports in use. This is especially important if you are using a serial mouse – if your mouse lives on `/dev/ttyS0` (aka COM1), you will likely want to avoid both `/dev/ttyS0` and `/dev/ttyS2` for the modem, because these serial ports tend to share an IRQ.

  You will know that you have a conflict if you set up a working modem, but you find that the mouse freezes when the modem is active.

- ISP information for your customer. You will want this to set up the dialer program. You will want to know the customer's login and password, the phone number for the ISP, and the protocol that the ISP uses for authentication (CHAP, PAP, chat or something else).

  If the customer does not have an ISP yet, then you can still set up the modem partially. You may even be able to use a test account to see whether the modem authenticates. However, this will not guarantee that the customer will be able to configure their modem to work with their intended ISP easily.

5.9.2 Fiddling with PnP BIOS settings

The BIOS in many Pentium-class systems natively supports the Plug and Play specification. Unfortunately, the support is often not that good for ISA devices. When configuring modems, this causes great headaches. Linux and the BIOS fight for control of the PnP devices, and Linux ends up losing.

Our solution to this problem is to disable PnP configuration for the modem. With Award BIOS sets you do this indirectly: you tell the BIOS to reserve certain IRQs for "Legasy ISA" cards. Here is how we set the BIOS:

1. Figure out the resources you want the modem to use. This may require configuring `/etc/isapnp.conf` to set the PnP jumpers.
2. Reboot the computer and get into the BIOS.
3. Find the BIOS section that controls Plug-and-Play devices and select that menu option.
4. On Award BIOS systems, you can set each IRQ to one of two options: **ISA or PCI PnP** or **Legacy ISA** . Set the IRQ for the modem to **Legacy ISA** .

   (Check these menu messages -P.)
5. Save the configuration and exit.

If you are lucky, this will free up the IRQ you want, so that **isapnp** can allocate the resource to the modem properly.

5.9.3 Setting modem jumpers

Older modems may not use the Plug-and-Play protocol. On these modems you will have to set up the modem manually. You will need to know what the jumper settings mean. Often the legend is printed somewhere on the modem, but sometimes this means logging onto the Internet and finding specifications.

Once you have figured out what the jumpers do, you will want to set the following jumpers:

- You need to set the serial port to something unused. If you have a serial mouse on `/dev/ttyS0`, you will want to set this jumper to either COM2 (`/dev/ttyS1`) or COM4 (`/dev/ttyS3`) to avoid mouse conflicts.

- You need to set the IRQ to something free. Often IRQ 3 will work, but it varies a lot from system to system.
- Sometimes you can set the IO address the modem uses. Set this to match the serial port you have set, as listed in the first four lines of `/etc/serial.conf` (see the setserial information in Section 5.9.4 for more information).

Set the jumpers to values you think are free. Be prepared, however, to take out the modem and try again.

5.9.4 Using setserial

At first, we thought we needed to edit the `/etc/serial.conf` file to tell Linux the serial port the modem was using. Apparently, this is not the case: editing `serial.conf` appears to have no effect on making the modem work properly. What *is* important is to set the IO address the modem uses to an IO address corresponding to the serial port you want to use.

According to the Modem-HOWTO you do need to modify `/etc/serial.conf` if you change the IRQ or IO address of the modem to something non-standard. We have not verified this, however.

Near the top of the `/etc/serial.conf` file, you will find the following lines:

```
These are the standard COM1 through COM4 devices
#
#/dev/ttyS0 uart 16450 port 0x3F8 irq 4
#/dev/ttyS1 uart 16450 port 0x2F8 irq 3
#/dev/ttyS2 uart 16450 port 0x3E8 irq 4
#/dev/ttyS3 uart 16450 port 0x2E8 irq 3
```

The file `/etc/serial.conf` does not really configure anything. Supposedly, it registers serial ports, but we have not seen evidence that editing the file makes any difference. However, we don't understand modems well yet, so, editing `/etc/serial.conf` might yet turn out to be important.

In order to register a serial port, you have to know how you configured your modem. You need to know the IRQ, the COM port the modem is using and (if using `/etc/isapnp.conf`) the IO port associated with that COM port. Then go through the following steps:

1. Edit the `/etc/serial.conf` file:

   ```
   nano /etc/serial.conf
   ```

2. Find the line corresponding to the COM port that you want to use. COM1 is `/dev/ttyS0`, and COM4 is `/dev/ttyS3`.

   Note that `/etc/isapnp.conf` will tell you the serial port to use if you used **isapnp** to configure the card. Look at the BASE address used by the card. For example, if your `isapnp.conf` file had the following uncommented block:

   ```
   #       Start dependent functions: priority acceptable
   #       Logical device decodes 16 bit IO address lines
   #             Minimum IO base address 0x02e8
   #             Maximum IO base address 0x02e8
   #             IO base alignment 8 bytes
   #             Number of IO addresses required: 8
     (IO 0 (SIZE 8) (BASE 0x02e8) (CHECK))
   #       IRQ 3.
   #             High true, edge sensitive interrupt (by default)
     (INT 0 (IRQ 3 (MODE +E)))
   ```

then you would know that the card was using IRQ 3, and `/dev/ttyS3`, since this serial port corresponds to IO address 0x02e8 .

3. Copy the line and paste it. In **nano**, ⟨ctrl⟩-k copies a line and ⟨ctrl⟩-u pastes the line(s) you copied most recently.
4. Uncomment one copy of the line by removing the hash mark at the front of the line.
5. Now edit the line. The "irq" should be set to the IRQ. If you matched the IO port to the appropriate serial port, you should not need to change the "port" line.
6. Save the file.
7. Restart the setserial program. If you are lucky, this should register the port. To restart the program, type

   ```
   /etc/init.d/setserial restart
   ```

If all worked well, you should see some lines similar to the following on bootup:

```
Loading the saved-state of the serial devices...
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16450
/dev/ttyS1 at 0x02f8 (irq = 3) is a 16450
```

5.9.5 Using isapnptools (revisited)

Most new modems use the Plug-and-Play specification. Instead of setting hardware jumpers manually, the jumpers on PnP boards are registers that can be set using software.

Some modems have jumpers that allow you to choose whether to use Plug-and-Play or not. Disabling Plug-and-Play and setting jumpers manually can sometimes make your life easier, but it does not always work. If you have other Plug-and-Play cards in the computer, it might be less work to set jumpers manually, because you will have to fiddle with the `/etc/isapnp.conf` file less.

See Section 5.8.2 for information on how to use the **pnpdump** program to set Plug-and-Play jumpers.

The tricky part here is to co-ordinate IO addresses in `/etc/isapnp.conf` with the IO addresses of the serial port you want to use.

You can see standard IO addresses for the serial ports by looking at the `/etc/serial.conf` file. Here is the relevant part of the file again:

```
These are the standard COM1 through COM4 devices
#
#/dev/ttyS0 uart 16450 port 0x3F8 irq 4
#/dev/ttyS1 uart 16450 port 0x2F8 irq 3
#/dev/ttyS2 uart 16450 port 0x3E8 irq 4
#/dev/ttyS3 uart 16450 port 0x2E8 irq 3
```

From this snippet we can see some important things:

- `/dev/ttyS0` and `/dev/ttyS2` typically share an IRQ. Thus, if there is a serial mouse on COM1 (`/dev/ttyS0`) you want to avoid setting the modem to `/dev/ttyS2` .
- Similarly, `/dev/ttyS1` and `/dev/ttyS3` share an IRQ.
- The IO addresses for the serial ports are specified by the `port` section. For example, the IO port for `/dev/ttyS0` is `0x3F8` .

When editing `/etc/isapnp.conf`, you need to set the IO address the modem uses to an IO address corresponding to one of these serial ports.

In the following example, the IO address the card uses is 0x02e8, which corresponds to `/dev/ttyS3`.

```
#       Start dependent functions: priority acceptable
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x02e8
#             Maximum IO base address 0x02e8
#             IO base alignment 8 bytes
#             Number of IO addresses required: 8
  (IO 0 (SIZE 8) (BASE 0x02e8) (CHECK))
#       IRQ 3.
#             High true, edge sensitive interrupt (by default)
    (INT 0 (IRQ 3 (MODE +E)))
```

5.9.6 Using wvdialconf to test modem detection

The **wvdialconf** program can quickly determine whether Linux can see the modem. Type

```
wvdialconf /tmp/wvdial.conf
```

A successful modem detection will produce output similar to this:

```
ttyS1<*1>: ATQ0 V1 E1 -- OK
ttyS1<*1>: ATQ0 V1 E1 Z -- OK
ttyS1<*1>: ATQ0 V1 E1 S0=0 -- OK
ttyS1<*1>: ATQ0 V1 E1 S0=0 &C1 -- OK
ttyS1<*1>: ATQ0 V1 E1 S0=0 &C1 &D2 -- OK
ttyS1<*1>: ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0 -- OK
ttyS1<*1>: Modem Identifier: ATI -- 33600
ttyS1<*1>: Speed 4800: AT -- OK
ttyS1<*1>: Speed 9600: AT -- OK
ttyS1<*1>: Speed 19200: AT -- OK
ttyS1<*1>: Speed 38400: AT -- OK
ttyS1<*1>: Speed 57600: AT -- OK
ttyS1<*1>: Speed 115200: AT -- OK
ttyS1<*1>: Max speed is 115200; that should be safe.
ttyS1<*1>: ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0 -- OK
```

If Linux does not see your modem, you will get output like this for each detected serial port:

```
ttyS0<*1>: ATQ0 V1 E1 -- failed with 2400 baud, next try: 4800 baud
ttyS0<*1>: ATQ0 V1 E1 -- failed with 4800 baud, next try: 9600 baud
ttyS0<*1>: ATQ0 V1 E1 -- failed with 9600 baud, next try: 19200 baud
ttyS0<*1>: ATQ0 V1 E1 -- failed with 19200 baud, next try: 115200 baud
ttyS0<*1>: ATQ0 V1 E1 -- and failed too at 115200, giving up.
```

Once Linux detects your modem properly, you are ready to configure a dialer.

5.9.7 Setting up a dialer

A *dialer* is the program that actually dials your modem to connect to the outside world. It will negotiation with an ISP, and start the PPP daemon so that the user can start surfing the Internet.

To set up ISP information, you actually want to configure two dialers: **pppconfig** and **gkdial** . The first actually sets the dialing information. The second dialer is graphical, and uses the information set by **pppconfig** to dial out.

Another possibility is to use the **wvdial** program to dial out, but **wvdial** does not work in every situation, and it does not play well with **gkdial** .

Here is how to use **pppconfig**

1. Start the program:

   ```
   pppconfig
   ```

2. Choose **Create a connection**
3. Leave the account name as **provider**
4. Select the default options for everything except the DNS (which should be **Dynamic DNS** and the connection protocol (which should be **PAP**) .
5. If you have ISP information for your customer, then you can enter that information when prompted for the username, password and phone number fields. Otherwise we may have a test ISP account you can use (but you will have to undo that ISP information once you have finished testing the modem).
6. When you have finished setting the account, select **Finish** to save the configuration. If you do not do this you will lose everything and have to start over.

To dial the modem out, first disable the network card connection if it is active. To do this, type:

`ifdown eth0`

Then connect the computer to a phone line and type

`pon`

The modem should dial and authenticate with the ISP. To verify that the connection is up, type

`ifconfig`

You should see an entry for interface `ppp0` .

You can also test the connection by connecting to an Internet site, or pinging another host. For example, to ping Google you could type:

`ping www.google.com`

Then type ⟨ctrl⟩-C to quit ping. If all goes well you will see messages similar to:

```
PING www.google.com (216.239.33.101): 56 data bytes
64 bytes from 216.239.33.101: icmp_seq=0 ttl=40 time=141.4 ms
64 bytes from 216.239.33.101: icmp_seq=1 ttl=40 time=138.8 ms
64 bytes from 216.239.33.101: icmp_seq=2 ttl=40 time=153.8 ms
```

Otherwise you will see nothing, and you will have to debug the problem. **(how? -P.)**

Once you have verified the connection works, end the connection by typing

`poff`

Once you have a working connection, you need to go through a few more steps to make the dialer work conveniently with **gkdial** and to eliminate traces of the test account if you used one:

- The **gkdial** program looks at the PPP connections defined in `/etc/ppp/peers/` directory and generates a drop-down list of them. We want the "provider" connection to be first in the list, but by default the "dsl-provider" connection takes precedence. We can eliminate that by renaming the "dsl-provider" connection:

  ```
  cd /etc/ppp/peers ; mv dsl-provider dsl-provider.orig
  ```

- After renaming the "dsl-provider" account, you can test out **gkdial** . Start graphical mode and select the **Modem dialer** option from IceWM's start menu. The program should start.

  Check that "provider" is the default entry in the dialer list. Then test out the connection by selecting **Connect!** . The modem should dial out and connect to the ISP. To stop the connection you have to select **Disconnect!** – simply exiting the program is not enough.

- If you used a test account, we want to delete this account's information. Otherwise users could use the test account at home to surf the Web, which means we get in trouble and lose the test account, which means that your life as a modem-configurator becomes more difficult, which will make you sad. You don't want to be sad, so go through the following steps to remove the account information:

1. Run **pppconfig** again. This time select **Change** to change the connection.
2. Change the **Number**, **User** and **Password** settings to bogus information. I find that "5555555" makes a good phone number, "replace-with-username" makes a good user name, and "replace-with-password" makes a good password.
3. Select **Finished** to save changes, and then exit the program.
4. We are not done yet – **pppconfig** makes backup files that contain the test account information. We need to remove that evidence. Type the following:

   ```
   cd /etc/ppp
   rm pap-secrets.bak
   cd peers
   rm provider.bak
   ```

   After doing this, all evidence of the test account should be gone.

It is nice if your user has ISP information handy when setting up the modem. Then you can configure the modem once and not have to remove any evidence. If the user does not have ISP information on hand they will either have to run **pppconfig** themselves to enter the ISP information (which is bad) or they will have to bring in their computer so you can enter this information (which is inconvenient).

If you are having problems getting the modem to connect with the user's ISP, refer to Appendix A.4 to see whether somebody has already run into your problem.

Note that in order to use a dialer program as a user, the user has to be a member of a privileged group. The two groups we have seen are "dip" and "dialout" . Currently, the installer is configured so that all default user accounts (and any subsequent accounts created with the **adduser** command) will be automatically added to these groups. See Section 8.1.9 for more information.

To see which groups a user belongs to, type

`groups` *`username`*

where *`username`* is the name of the user. To add a group called *`groupname`* to a user named *`username`*, type

`adduser` *`username groupname`*

5.10 Deal with the network card and configuration

What you do at this point depends on whether you are supposed to keep the network card in the computer or not. Sometimes this decision will be made for you – you will be told to take the network card out, or customers will want to buy network cards with their systems.

Advantages of keeping the network card installed include:

- You will be including a network card that you know works with this computer. This card could be used for high-speed Internet access.
- You can easily download programs to the computer later on if need be.

Disadvantages of keeping the network card in the computer include:

- The default configuration of the network card will not work if the user takes the computer home.
- We probably do not have enough network cards to provide one with every computer.
- Apparently, Sympatico installs its own network cards for subscribers to ADSL access (see Section A.3.2, removing existing network cards in the process. Thus, installing our own cards may be futile for Sympatico subscribers.

If you want to *keep* the network card, you should make sure that it continues to work:

1. Check the file `/etc/modules` . The network card module should be listed in it. (It is not `auto`, if you are wondering.)

   If there is no network card module, you have to find out what module should be inserted, and then you need to insert an appropriate line by editing `/etc/modules` as the root user.

   For help in discovering which network card driver to use, see Section A.3.1 .

2. If the user knows the network settings he or she needs (gateway, IP address, DNS servers and netmask) you will want to edit the following configuration files:

   - Enter the new DNS server in `/etc/resolv.conf`
   - Enter the IP address, gateway and netmask in `/etc/network/interfaces` . (For notebooks with PCMCIA network cards, look in `/etc/pcmcia/` instead.)

     You can also specify that IP addresses should be specified by DHCP or bootp in `/etc/network/interfaces` . See the `interfaces`(5) man page to see how to do this.
   - Change the IP address in `/etc/hosts` . You can also put the domain name in.

   If the user does not know the new network information, it is best to leave all these configuration files as is.

3. At some point, reboot the machine and verify that the network card is detected. You can see this by typing

   ```
   ifconfig
   ```

If you do not want to keep the network card, you need to make sure that Linux does not try to detect the network card on bootup:

1. Edit `/etc/modules` . Comment the network card driver with a hash mark ("#").
2. Edit `/etc/network/modules` . Comment out the line

   ```
   auto eth0
   ```

   with a hash mark.

If you do not go through the above steps, the computer will probably still work – but it will complain that it cannot find the network card.

5.11 Finishing up

At this point, the system should be working and ready to be used and loved. Now all you need to do is prepare the user for his or her purchase:

- Show the computer how to turn the computer on, how to log in (if necessary) and how to shut the system down properly.
- Show the user how to start programs, and where to find them.
- Show the user how to save to the floppy drive.
- If you configured a printer, then show the user how to use it.
- Make sure the user gets the following documentation:
  - The Quick Reference, which provides basic tips.
  - The User's Guide, which provides more in-depth information.
  - The Configuration Information sheet. As this contains sensitive information (namely, the root password), make sure the user keeps this in a safe and secure place. Also tell the user that we will need this sheet if he or she ever brings it in for service.

Of course, you should make sure that these sheets have been filled out appropriately.

- Answer any questions that the user might have.
- Tell the user about support options. **(which are? -P.)**
- Record the user's contact information in our database. **(which does not exist yet.. how do we do this? -P.)**

6 The Installation Backend: FAI

We have designed our Linux system so that it is easy for volunteers to install. As such, you don't need to know very much about the installation process in order to get Linux on a computer.

Having said that, understanding the machinery behind the install can be helpful. If nothing else, it will help you troubleshoot the installation process when installs fail. Learning about the installation process helps the Linux project, too: it helps volunteers understand Linux better, and it gives the project security in case the current administrators stop volunteering.

The next section introduces FAI, and documents some resources that can help you understand the package. Next, we explain the installation process – from bootup to configuration – in some detail.

6.1 FAI overview

The software we use for our backend is called FAI (Fully Automated Install). It is a set of scripts that allows Linux to be installed on many workstations at once, without human intervention. FAI can be customized to suit local needs, and we have customized FAI to install our subset of Debian automatically.

In our project, FAI scripts are responsible for:

- Partitioning and formatting hard drives.
- Installing Debian packages.
- Carrying out some customizations specific to our project. For example, we copy configuration files to make the local window manager look like Windows 95.

We do not yet exploit the full functionality of FAI. One problem is that we face many different hardware configurations. We have not figured out how to automatically detect and select packages specific to hardware, monitors and mice, so we rely on volunteers to configure X. Similarly, volunteers must create user accounts on the system. As our Linux installation matures, we may be able to automate some of these things as well.

6.1.1 FAI resources

The homepage for FAI is `http://www.informaik.uni-koeln.de/fai` . This page offers the latest version of the FAI manual, and archives of the FAI mailing list.

The FAI manual is also installed locally on the server. You can find it at
`/usr/share/doc/fai/`

Working through the manual and setting up FAI is the best way to understand the system. Failing that, working through the manual and comparing that to the files on our server is good too.

6.1.2 FAI tips

- All configuration files need to be world-readable, and all directories need to be world-executable. Otherwise FAI will not read the configuration files, and installs will fail for strange reasons.

  This problem can probably be fixed by specifying proper ownership privileges, but we don't know how to do that yet.

6.1.3 FAI logfiles

FAI logs configuration information and runtime messages into a directory full of logfiles. During the installation process, you can access these logfiles in the directory
`/tmp/fai/`
After installation, you can find the logfiles in the directory
`/var/log/fai/`*`hostname`*`/last-install/`
on the client machine, where *`hostname`* is the name of the machine.
Log files of interest include:

- `rcS.log` , which logs all the messages that appear on screen after the kernel has booted. If you are wondering whether a step of the FAI installation failed, this is the first place to look.

- `dmesg.log` , which logs all the information printed out during the booting process. This is the place to look for information about drivers detected by the system, such as network cards.

- `FAI_CLASSES` , which shows all the classes defined for this installation. This is the first place to check if something was mysteriously misconfigured or not installed – often the required class was not defined.

- `hda_sfdisk` , which shows how the first IDE drive was partitioned.

6.2 Startup

The boot process of our FAI installation goes through the following steps:

1. The BIOS finds LILO on the boot floppy.

2. LILO loads a Linux kernel from the floppy.

3. LILO connects to the network and NFS-mounts a Linux filesystem that sits on the server. This filesystem contains the FAI installer, which begins execution.

4. The installer NFS-mounts the FAI configuration files off the server. **(Check this -P.)**

After these steps, the FAI installer is ready to partition the local hard disks, mount local hard drives and install program files.

The following subsections explain the startup process in more detail.

6.2.1 BIOS and the boot loader

The BIOS of a computer contains basic information the computer needs to interact with its components. Part of the BIOS's job is to look for boot information for the computer's operating system.

The BIOS is not very smart. It knows some locations where the boot information may exist – the *boot sequence* – and it knows how to load enough information from those locations to get the rest of the boot loader going.

Most BIOSes allow you to boot from the floppy or the hard drive. Some allow booting from CD-ROMs, and a few allow direct booting off a network, using a network card.

Our installation uses a bootable floppy for installation. The first few sectors of the floppy contain LILO, a boot loader for Linux. The BIOS finds LILO, loads it into memory and executes it.

6.2.2 LILO

LILO stands for LInux LOader. Its job is to tell the computer where to find Linux components. LILO can also be used to boot other operating systems (such as Windows 95), but we do not use that functionality in FAI.

In order to boot Linux, LILO needs two pieces of information:

Kernel location: The Linux *kernel* is the core of the operating system. It is a program that serves as an interface between the computer's hardware and software, managing system resources, allowing access to hardware, and interacting with application programs.

There may be several different versions of the Linux kernel on a computer. LILO needs to know which one to use, and where to find it. In our installations, the kernel is located on the boot floppy disk.

Root location: The *root directory* is the top of the directory structure. The directory structure contains all of the user programs and data present on the system. The directory structure may be spread over many hard drives, but it has a single root directory. (The hard drive partition that contains the root directory is called the *root partition*.)

LILO needs to know the location of this root directory, so that it can *mount* the directory tree. Other filesystems (on hard drives, CD-ROMS or the network) can then be mounted relative to the root.

On standalone Linux systems, the root directory is often located on the same partition as the kernel. However, this is not required, and in our installation the root directory is located on the server. It is accessible via a protocol called *NFS*. LILO is smart enough to know about NFS, and it is able to tell the kernel where to find the root directory that should be used for booting.

On our server, the NFS root directory is:
`/usr/local/share/fai/nfsroot/`
Permission for the server to export this directory is contained in the `/etc/exports` file.
(Where should I be talking about NFS? Here, or elsewhere? -P.)

6.2.3 rcS_fai

Traditionally, when Linux boots it executes startup scripts. These scripts start services, such as X-Windows and the printing daemon. On Debian, these scripts can be found in `/etc/init.d/` .

To start the install, FAI includes a script in the `/etc/init.d` directory called `rcS_fai` . The Linux kernel finds this script (or rather, a symbolic link to this file) and starts executing it.

`rcS_fai` contains the control code for FAI. It is the installer program; its job is to co-ordinate the installation process, calling subroutines that carry out each stage of the install.

6.3 Installation

Once the installer has started, installation proceeds in the following steps:

1. Classes are defined.
2. Local hard drives are partitioned and formatted.
3. Packages get installed and configured.
4. Local configuration changes are made.
5. GRUB is installed onto the local hard drive.

The FAI manual goes through each of these steps in detail. Over the next few sections, we will describe the basic ideas, and document local changes we made for our project.

6.3.1 FAI classes

FAI has the capability to install many different Linux configurations at the same time. Different configurations are specified by *classes* – labels that specify which configuration files FAI should use in the installation.

Each installation starts out with three classes defined: DEFAULT, LAST and a class labelled with the client's hostname.

Once an installation starts, one of the installer's first jobs is to figure out what additional classes (if any) should be defined. To do this, FAI looks in

`/usr/local/share/fai/installer/classes/` .

Three different kinds of files are allowed in this directory:

- Perl or shell scripts, with names that match the regular expression 'S[0-9]*.sh,pl,source' . Each of these scripts is executed in alphabetical order. FAI considers the output of these scripts to be a new class name.

  This would be the natural way to define classes for based upon detected hardware. We currently use it to define classes based on hostname (`S01alias.sh`), to decide disk partitions (`S07disk.pl`) and to run configuration menus (`S30menu.sh`, `S40xserver.sh`).

- Files with names matching an existing class. Each word (token) in these files is considered to be a new class name. However, the contents of this file will *not* be executed.

- Files of the form *classname*.var . These files are scripts that are executed if and only if *classname* is a defined class.

  We use the pre-existing file `DEFAULT.var` for our installations, but do not define other variable files.

The installer executes any scripts matching the name of an existing class. This may define new classes, so the process iterates until no more classes have been defined. Finally, scripts of the form *classname*.var are executed.

We explain relevant classes in the parts of the installation where they are used. Hostnames in the `classes/` directory which directly pertain to class definition are:

wc_custom: This hostname does not call any of the standard packages or classes. It is meant as a testing hostname, so that administrators can fiddle around with new features while other people are doing standard installations.

wc_191 ... wc_198: Each one of our standard installation FAI floppies is given one of these names. Hosts with these names use the standard disk partitioning and formatting, and choose the amount of software to install based on the local hard drive size.

The names themselves are defined in `S01alias.sh` .

6.3.2 Partitioning Hard Drives

In this phase, the installer looks in `/usr/local/share/fai/installer/disk_config` for partitioning information. It should partition the local hard drive(s) according to the information specified in exactly one of these files, which means exactly one class should match a filename in this directory.

FAI partitions the local hard drive(s) according to the information it finds in this file. It also formats the partitions appropriately.

One big advantage of these files is that they are robust with respect to hard drive sizes. They can size partitions proportionally according to the size of the hard drive.

(What happens if there are multiple matches? -P.)

The format of these files is described in the FAI manual. We have defined the following classes:

TWCN_DISK: This configures a single IDE drive. It wipes out the entire drive, then creates three partitions:

- A swap partition, which ranges between 32-64MB depending on the hard drive size. (FAI determines the size of the swap partition by looking at the size of the drive.)
- A root partition, which is at least 300 MB, and can expand to the entire disk.
- A boot partition called `/fai-boot`, which is 2MB big.

As well, this file specifies that all partitions should be checked for errors as they are formatted. (This is the "-c" option.)

TWCN_DISK_SMALLSWAP: Like TWCN_DISK, except that the swap size is fixed at 32MB.

TWCN_DISK_BIGSWAP: Like TWCN_DISK, except that the swap size is fixed at 64MB.

TWCN_DISK_TEMP: Like TWCN_DISK, except that hard drives are not checked for errors. For new installs, you should not use this class. Checking for hard drive errors adds time to the install, but it saves a lot of headaches later on.

Other classes we should add include:

- A class to deal with SCSI drives.
- A class which preserves Windows partitions. This would allow us to dual boot our installations.
- A class that deals with multiple hard drives.

After the hard drives have been partitioned and formatted, they are *mounted*. In Linux, hard drives must be mounted before the system can access their data.

The root partition is mounted to `/tmp/target/` . Any other partitions are mounted relative to this point. For example, if we create a partition that should end up at `/usr/local/` , the installer will mount this partition to `/tmp/target/usr/local/` .

6.3.3 Base package installation

A certain subset of the Debian distribution is labelled "required", and another subset is labelled "important". These packages make up a barebones Debian installation.

A program called *debootstrap* is responsible for downloading these packages and unpacking them into `/tmp/target/` .

The packages that *debootstrap* installs are listed in `/usr/lib/debootstrap/scripts/` on the server.

6.3.4 Package installation

Once the local hard drives have been prepared, package installation begins. The installer NFS-mounts the local mirror on the server. It then downloads and installs packages according to the classes defined in

`/usr/local/share/fai/installer/package_config/`

The contents of every file matching a defined class name are used.

Again, the format of these configuration files is described in the FAI manual.

(Where can readers find out about Debian packaging policies? -P.)

Below, we document some of the more notable package configuration classes we use. This list is undoubtedly incomplete:

DEFAULT: We have changed this file significantly from the version shipped with FAI. We consider the default installation to include very few packages: basically just a few useful utilities not provided by UNIXLIKE:

- The printing utility **lprng**
- The text-based web browser **w3m**
- **bzip2** and **zip** and **unzip**, which are used for file compression.
- **sudo** , which allows regular users to execute superuser commands safely. **(We don't use this at all.. -P.)**

(Is this it? -P.)

X11: This package contains X-Window functionality; all the packages needed for graphical mode *except* the X-server packages.

UNIXLIKE: This includes almost all packages in the "standard" Debian distribution. We want this because FAI will (correctly) not install these packages by default.

These packages provide most of the functionality that one would find on a standard Linux or UNIX system, with the following exceptions:

- The text editor Emacs, provided by the EMACS package.
- Development packages, which are used to compile programs. These are provided by the DEVEL package.
- A few utilities that are annoying:
  - **gpm**, which controls mice in text terminals. This package is a huge pain because it interferes with regular mouse operation.
  - **lpr**, which is an ancient UNIX printing system. We use **lprng** instead.
  - **mtools**, which are a means of accessing files on floppy drives. These are ordinarily very useful tools, but because we automatically mount the floppy drive using the Supermount protocol, it is not safe to use **mtools** on our systems. (see Section **(where? -P.)** for more information).
- TEX packages. TEX and LATEX are powerful typesetting systems that take up a lot of disk space.

WINLIKE: This package contains applications that make Linux look like Windows. Here is an incomplete list of some of the things it includes:

- Productivity applications: **abiword**, **opera**, and **gmc** .
- The window manager **icewm**, and **gdm** (which provides a graphical login).
- Games: the ace-of-penguin suite of card games, the arcade games **koules**, **xkobo** and **xbubble**, and a few other small puzzle games.
- Smaller applications: **xpaint**, **eeyes** (an image viewer), **sylpheed** (a mail reader), **gaim** (ICQ/AIM/MSN client)...
- Printing utilities **printop** and **lprngtool**, and the modem dialer **gkdial**

These packages are graphical. They have a lot of dependencies.

DEFAULT_ORIG: This is the DEFAULT class that ships with FAI. We do not use it.

EMACS: **emacs** is a super powerful text editor. However, it is large and thus optional.

DEVEL: The development tools (**gcc** and its friends) that we removed from the default package installation.

SGML: SGML development. No installation uses this yet.

TEX: TEX and LATEX development, which is removed from the default installation.

GNUMERIC: The **gnumeric** spreadsheet. Originally we included this spreadsheet by default, and it is installed in STANDARD installations, but we took it out of WINLIKE to save space on systems with smaller hard drives, and because it appears that many people do not use spreadsheets.

SCRIBUS: A rudimentary desktop publishing program. It is more cute than functional now, but is useful for making posters.

ABIWORD_EXTRAS: The **abiword** word processor has some extra components not necessary for operation, such as clip art. We take these packages out for smaller installations.

SIAG_OFFICE: Currently only used by the TINY installation, this is an alternative office suite. It consists of a word processor, a spreadsheet and an animation program.

PKG_MANAGE: Some bigger but useful utilities for managing Debian packages. **aptitude** is a useful front-end command-line **apt-get** utilities. **deborphan** is used to discover packages that can be safely removed from the system.

This package is obsolete, because we now include **aptitude** in the DEFAULT package.

Packages we should add include:

- Possibly a package for music, or for web development, or for the GIMP and its friends – whatever is popular with users.

FAI's package configuration system is powerful and useful, but it has its quirks:

- FAI automatically satisfies dependencies of all the packages specified in the package configuration files. However, it does not satisfy "recommended" dependencies. You must satisfy them explicitly.
- The "remove" command will fail if the package in question has not been installed.
- It used to be the case that if FAI could not find a package it needed, the installation process would halt. Thankfully, this behaviour has been changed, and FAI will simply print a warning:

  `These unknown packages are removed from the installation list:`

  Unfortunately, when package dependencies are wrong FAI can still get confused, and continue the installation process without installing important packages. See Appendix B.2.1 to learn more about this problem.

  It is a good idea to check the `/tmp/fai/software.log` file on the client before rebooting the system, to ensure that all packages installed correctly.

- FAI will by default install all packages with priority "Required" and "Important", but will install no packages with priority "Extra", which includes such useful utilities as `less` .

  To get around this, we had to manually install Debian on a machine and create a package configuration file with the standard packages. To see how to do this, see Section 8.4.

  (Wrong formatting -P.)

6.3.5 Configuration

In order to make our setup look like Windows, we have to tweak some configuration files. You can find the code that executes these tweaks in the `installer/hooks` and `installer/scripts` directories.

To learn about specific tweaks we make to our applications, see Section 8.1.

6.3.6 Finishing up

The LAST class triggers operations that clean up the Linux installation.

The GRUB and BOOT classes together trigger the BOOT script, which makes the computer bootable. The kernel to install is specified in `installer/class/DEFAULT.var` . The corresponding kernel package is located in `installer/files/packages/`

7 The local Debian mirror

Installing packages over the Internet takes a long time. To speed up installations, we keep a local archive of Debian packages in the `/usr/local/mirror/` directory. Clients NFS-mount this mirror and download packages from it.

(This section needs to be completely rewritten! We now have two mirrors – one unstructured, and another structured. -P.)

7.1 Maintaining the mirror

"Real" mirrors are structured according to a complicated hierarchy. Because we only need a small subset of the entire Debian archive, we just throw all of our packages into

`/usr/local/mirror/localdebs/`

When we run APT or dselect on our server, it automatically downloads security updates for its installed packages. It stores these new packages in

`/var/cache/apt/archives`

To move these files to the mirror, become root, then type

`mv /var/cache/apt/archives/*.deb /usr/local/mirror`

In addition to .deb files, the mirror contains a file called `Packages.gz` . This file must be updated, or APT will not be able to see the new packages.

To update the `Packages.gz` file:

1. Become the root user.
2. Go to the `/usr/local/mirror/` directory:

   ```
   cd /usr/local/mirror
   ```

3. Type

   ```
   dpkg-scanpackages debs override | gzip > ./debs/Packages.gz
   ```

   the "override" file happens to be empty, but is nonetheless a required argument. Ordinarily, its job is to override information specified in .deb files.

Note: When installing new packages on the server, APT will ask you whether you want to "Delete the downloaded .deb files." You should respond NO, so that these files can be added to the mirror later.

For more information about using local Debian mirrors, refer to the APT HOWTO. **(reference? -P.)**

7.2 Using the mirror

7.2.1 APT and the mirror

APT can use our mirror to download packages. In order to do so, we must modify

`/etc/apt/sources.conf`

On the server, the entry for the local mirror is as follows:

`deb file:/usr/local/mirror localdebs/`

(WRONG WRONG WRONGchange this -P.)

On a client machine, the entry for the local mirror is:

`deb file:192.168.1.190:/usr/local/mirror debs/`

The IP address is the server's. In order to use this mirror, the `/usr/local/mirror/` directory should be NFS-mounted. To see how to do this, please see Section **(where?? -P.)**

7.2.2 fai.conf

The FAI configuration file `/etc/fai/fai.conf` also contains an APT entry for the server. Clients use this entry to download packages.

7.3 base_woody.tgz

In Section 6.3.3, we discuss the need for a `base2_2.tgz` file. This file can be found at

`/usr/local/mirror/base2_2.tgz`

It is a symlink to the more descriptively-named `base_woody.tgz` . (Debian 2.2 was the old Potato distribution. The base file for Woody should be called `base3_0.tgz` , if we were to follow the naming scheme.)

The `base_woody.tgz` file contains all the base packages for the debian archive, unpacked into a directory. This archive is created by the **debootstrap** command.

(Put down the command to build this. Assume we unpack to /tmp/basedebs/ -P.)

debootstrap connects to a Debian server, downloads the base packages for a distribution (Woody in our case), and unpacks it to the specified directory. We then use **tar** to pack these files up.

To create a new base debs file:

(Put full instructions here -P.)

8 Other tasks

In this section we document noteworthy tasks that did not fit in well anywhere else.

8.1 Local Package Customizations

We made several customizations to our software. Most of these customizations are performed automatically by the installer.

Why do we make customizations? In many cases it is to set default save formats so that they are compatible with the save formats of Windows products.

8.1.1 Abiword

Abiword does not appear to have a system-wide configuration file. However, users can specify their preferences in

`~/.AbiSuite/Abiword.Profile`

In this file, we add the following line:

`DefaultSaveFormat=".rtf"`

to the "_custom_" scheme. This sets the default save format to Rich Text Format, which is readable by most modern word processors.

We then take this configuration directory and stuff it into `/etc/skel/` , so that this preference file will be copied into the home directory of each user account.

The version of `Abiword.Profile` that we install can be found in the file

`skel-winlike.tar.gz`

in the `installer/local.files/` directory.

8.1.2 Gaim

As with **Abiword**, **Gaim** does not have a system-wide configuration file, but rather uses a `.gaimrc` configuration file that lives in each user's home directory.

We again use the `/etc/skel/` hack by copying a customized `.gaimrc` file to `skel-winlike.tar.gz` .

In this case we want the configuration to load all common chat protocols that **Gaim** supports. We add the following lines to `.gaimrc`, in the `plugins` section:

```
plugin  /usr/lib/gaim/libyahoo.so
plugin  /usr/lib/gaim/libicq.so
plugin  /usr/lib/gaim/libirc.so
plugin  /usr/lib/gaim/libmsn.so
```

8.1.3 Gnumeric

(That xml file -P.)
(/usr/lib/gnumeric/*-bonobo/plugins/excel/ -P.)

8.1.4 Opera

opera offers two global configuration files that can be used to set preferences for all users on the system:

- `/etc/opera6rc` contains preferences that users can override in their own preferences files.
- `/etc/opera6rc.fixed` contains preferences that users cannot override. This is useful for system administrators who want to lock down preferences in their browsers.

As indicated by their names, both these files are specific to version 6 of **opera**. We only modify the `/etc/opera6rc` file.

The syntax for these files matches the syntax in users' personal configuration files, which you can find in `~/.opera/opera.ini` . **(CHECK THIS -P.)**

We wanted to make the following changes to Opera's preferences:

- We wanted to increase the screen real estate as much as possible, by hiding the hotlist and reducing icon sizes as much as possible.
- We wanted to get rid of the annoying question that Opera asks on each startup.

To create the `/etc/opera6rc` file, you can modify a user's preferences file as follows:

1. First, create a dummy account (or delete the `~/.opera/` directory in an existing dummy account.
2. Next, start Opera, then shut it down right away without changing any options. This creates a fresh `~/.opera/opera.ini` file.
3. Make a backup copy of this clean preferences file:

   ```
   cd ~/.opera/
   cp opera.ini opera.ini.orig
   ```

4. Start Opera again. Now make all the changes you want. Then shut down the browser again. This changes the `opera.ini` file.
5. Look for differences in the two configuration files:

   ```
   cd ~/.opera/
   diff opera.ini opera.ini.orig > /tmp/opera.diffs
   ```

6. Looking at the `/tmp/opera.diffs` file you created should indicate which lines changed in the file. Simply enter these new lines into the `/etc/opera6rc` file. Order is probably important: try to match the section headings in the `opera.ini` file. **(example? -P.)**

(Where can people find out more information about this? -P.)

8.1.5 IceWM

(Additional themes -P.)
(Toolbar, Preferences, Menu files -P.)

8.1.6 GMC

GMC is the file manager. It also puts icons on the "desktop" – that is, the root window. We want to put icons on the desktop without opening a browsing window. To do this, we put the following lines into the file `.xsession` :

```
gmc --nowindows &
exec x-window-manager
```

This `.xsession` file lives in `/etc/skel/`, and it is put there by `skel-winlike.tar.gz` .

8.1.7 Symbolic links

Sometimes, applications call other applications in order to perform certain tasks. For example, a spreadsheet might launch a web browser to display the spreadsheet's help files. There is nothing wrong with that, but some programs make a stupid mistake: instead of realizing that there are many different programs that accomplish the same task, they hard-code an alternative. An example of this are programs that expect the **netscape** browser to be installed. These programs call **netscape** to view online documentation explicitly – which is a mistake, because there are lots and lots of HTML browsers available.

To get around this problem, we use *symbolic links*. We make a symbolic name from our replacement program (for example, **opera**) to the expected program (for example, **netscape**). This way, the offensive program that calls **netscape** to view documentation will still work – it will indirectly call **opera** instead.

In our distribution, we make the following symbolic links. Unless otherwise stated, the code that makes these links is in the `hooks` directory of the installer:

- In the file `configure.WINLIKE`, we make a symbolic link from the web browser **opera** to `/usr/local/bin/netscape`
- In the file `configure.DEFAULT`, we make a symbolic link from the terminal emulator **rxvt** to `/usr/local/bin/xterm`

(Check these files! They may have changed! -P.)

8.1.8 Local menu entries

8.1.9 adduser configuration

(This first bit is basic UNIX information. Why is it here? -P.)

In UNIX, just about everything is a file. For example, a CD-ROM can be accessed by reading from the file `/dev/cdrom` and the floppy drive is accessed as the file `/dev/fd0` .

Every file in UNIX belongs to one user and one group. For example, in many UNIX systems configuration files are owned by the user `root` and the group `wheel` . There should be only one user called `root`, but many user accounts can belong to group `wheel` .

For security purposes, access to special device files (floppy drives, CD-ROM drives, sound cards, printer ports and so on) are typically restricted. In order to access these devices, users

must belong to the group that owns the file in question, and the file must give the group read and write permissions.

By default, user accounts do not belong to groups that can access devices. This is a problem, so we fixed it by writing a script, which can be found in `/usr/local/bin/adduser.local` . This script is executed by the **adduser** program when creating users.

Our `adduser.local` script is simple: we add every user that is created to all groups that can access devices. Specifically, our script consists of a bunch of lines of the form:

`/usr/sbin/addgroup $1` *group*

`$1` refers to the user that has been created. We add users to the following groups:

floppy for floppy drive access.

cdrom for CD-ROM access.

lp for parallel port access. This is needed for printing.

dialout for modem access.

dip which is also needed by some modem programs.

audio for sound card access.

(HTML links -P.)
(Extra games to run in small mode -P.)

8.2 WCLP In-house additions

Most of the software we use comes straight from the Debian distribution. However, we have created a few WCLP-specific resources and applications for our users.

8.2.1 Our feedback form

(Document Charles' application here -P.)

8.2.2 Offline help

New computer users often have a lot of trouble finding documentation and support for the applications they use. UNIX-like systems have a lot of documentation available, but that documentation can be hard to find and hard to use.

We don't have a great solution to this problem. Debian offers some packages to provide easier access to documentation (such as the **dhelp** package), but we thought even this was too complicated. We opted to set up a few local HTML pages that would point to some of the more relevant documentation available on the system.

On the user's system, this page is located in
`/usr/local/share/wclp/html/`
The pages contain the following information:

- Pointers to any HTMLized program manuals that exist (for example, **opera** and **ace-of-penguins** have such manuals).
- A few tips on how to find documentation on the system.
- A few tips on how to read the installed documentation (for example, how to read gzipped files).

(Which file is this on the local.files directory? -P.)

On the system's menu, users can access this help via the **Programs** → **Help** → **WCLP Help** menu option.

8.3 NFS Configuration

(/etc/exports -P.)

(kernel server or user server? -P.)

8.4 Creating the UNIXLIKE package configuration file

FAI will by default install all packages with priority "Required" and "Important", but will install no packages with priority "Extra", which includes such useful utilities as **less** .

In order to specify these packages explicitly, we had to manually install Debian GNU/Linux on a machine, then use its package selections for the UNIXLIKE file.

To get a listing of package selections on a machine, type

`dpkg --get-selections > /tmp/select.txt`

This will put the selections into the file `/tmp/select.txt` . This file will consist of each installed package, followed by the word "install". For example, here are the first few lines of `select.txt` on my machine:

```
acroread                                        install
adduser                                         install
ae                                              install
anacron                                         install
apt                                             install
at                                              install
base-config                                     install
base-files                                      install
base-passwd                                     install
```

In order to use this file with FAI, you need to get rid of the "install" lines. To do this, either search and replace using a text editor or type

`sed 's/deinstall//;s/install//' select.txt > select2.txt`

This command first gets rid of all invocations of the word "deinstall", and then gets rid of all invocations of "install". Note that if either of these words occurs as some part of a package name, the word will be removed, which will cause errors.

You can then use `select2.txt` by putting the following line at the top of the file:

`packages INSTALL`

and then saving the file as `UNIXLIKE` in the following directory:

`/usr/local/share/fai/installer/package_config/`

(Which of the standard packages do we not need? Move from the UNIXLIKE section? -P.)

8.5 Compiling the kernel

The Linux *kernel* is the core of the operating system. It is the program responsible for managing programs, processes, system devices, and memory. An interesting aspect of Linux is that you can compile a customized version of the kernel, which includes the functionality you need and omits the functionality you don't.

Compiling the kernel is a big topic, so we will not attempt to describe it all here. For more information about the kernel and how to compile it, see the Kernel-HOWTO, and the documentation in `/usr/share/doc/kernel-package` .

We use a custom kernel with our distribution. To create this, we download Debian packages We use the following kernel patches:

1. The "supermount" patch allows users to access floppies without having to use the **mount** command first. The patch varies from kernel release to kernel release, so make sure you get the patch that corresponds to your kernel version. **(from where can you download this patch? -P.)**

2. Patches to enable ext3 filesystem support. This is included in the standard kernel sources for 2.4 series kernels. A Debian package called `kernel-patch-ext3-2.2` provides the ext3 patch for 2.2 series kernels.

To build the kernel, go through the following steps. In all cases, assume VERSION is the version of the kernel we want to compile. For example, in our case VERSION was 2.2.20 .

1. First, download the kernel sources we wanted. To do this, install the `kernel-source-VERSION` Debian package.
2. Install the `kernel-package` Debian package. This package provides scripts that make kernel compilation a lot easier.
3. Download any Debianized patches you wish to apply.
4. Unpack the kernel sources. To do this, type:

   ```
   cd /usr/src/
   tar xvzf kernel-source-VERSION.tar.gz
   patch -p1 < /tmp/supermount.patch
   ```

5. If you have any other patches, the Debian kernel system can make them automatically. Simply add the following line to `/etc/kernel-pkg.conf` :

   ```
   patch_the_kernel := YES
   ```

6. You want to apply these automatic kernel patches. To do this, type:

   ```
   make-kpkg debian
   ```

7. Now apply the Supermount patch (which as of this writing was not Debianized). Say that the patch was in the file `/tmp/supermount.patch` . Then we would type the following:

   ```
   cd /usr/src/kernel-source-VERSION
   ```

8. Now would be a good time to read the kernel package documentation so that you know what you are doing:

   ```
   zcat /usr/share/doc/kernel-package/README.gz | less
   ```

9. You now have two choices: you can manually configure the kernel, or you can modify an existing kernel configuration file. We chose to modify the standard Debian configuration for their "vanilla" kernels.

   You can download kernel configuration files from any mirror. For example, as of the Potato release you could find the "vanilla" kernel configuration in the following file on any Debian mirror (for example, http://http.us.debian.org

   debian/dists/stable/main/disks-i386/current/kernel-config

   Download this file someplace accessible. We'll assume that you save it to `/tmp/kernel-config.vanilla`

   We chose to modify the "vanilla" kernel because it includes a lot of device drivers, which makes life easier when trying to get obscure hardware to work.

10. Now you need to configure the kernel:

    - Go to the kernel source directory:
      `cd /usr/src/kernel-source-VERSION/`
    - Start the kernel configuration program:
      `make xconfig`
      A pretty Tcl/Tk interface should pop up.

- Select **Load configuration file (check this -P.)** and type the location of the kernel configuration file: `/tmp/kernel-config.vanilla`
- In the **Filesystems** section, select **Supermount removable media support** . You want to compile this directly into the kernel, not as a kernel module.
- In the **Filesystems** section, select **Second extended fs development code** .
- If you want, in the **Processor Type and Features** menu, change the **Processor Family** to the minimum processor type you want to support. We selected "486". Note that changing this means that you will not be able to run this kernel on 80386 machines.
- In the main menu, select **Save kernel configuration and exit**

11. As the root user, type

    `make-kpkg --bzimage --revision=`*`name`*` kernel_image`

    Where *name* is the name of the kernel. For example, our kernels are named

    `custom.wclp.486.vanilla.`*`x`*

    where *x* is the revision of our kernel – for example "1.1" .

    Do *not* run

    `make-kpkg clean`

    or you will undo the kernel patches, and furthermore you won't be able to apply them correctly (because the Supermount patch will conflict with the ext3 one). With 2.4 series kernels this may not matter so much.

12. Wait. If you have a slow processor, wait and wait and wait. (Kernel compilation takes about four hours on our Pentium II 233MHz system.)

13. If all went well, a `.deb` file should be created in the `/usr/src/` directory. Copy this kernel to your mirror, and specify its name in the FAI default configuration file:

    `wclp/installer/class/DEFAULT.var`

8.6 Package Management

Debian programs are broken down into *packages*. A Debian package filename ends with `.deb` . Packages most often contain program files or libraries. Sometimes packages do not contain any data, but rather just specify dependency information to make other packages work together.

Often, programs in Linux require other programs or libraries to work properly. Such requirements are called *dependencies*. A primary purpose of packaging systems is to help make program installation easier by automatically managing dependencies. The `.deb` format contains the following dependency-related fields:

Depends: If package A depends on package B, then B must be installed for A to work properly.

Suggests: If package A suggests package B, then B adds some important functionality to A, but A will work without B.

Note that **apt-get** will *not install* "suggests" dependencies by default.

Recommends: This is a weaker form of "suggests". It indicates that packages A and B fit together in some way. "recommends" dependencies are also not installed by default.

Conflicts: If package A conflicts with package B, then A and B cannot both be installed at the same time. Trying to install package A when package B is installed will fail.

Provides: Sometimes many packages provide the same functionality. These packages can state that they provide this functionality with a "provides" dependency. Say that package A provides C and package D provides C, and that package B depends on C. Then either package D or package A will satisfy B's dependency.

The `.deb` format provides some other information which can be useful:

- A written description of the functionality the package provides.
- The size of the `.deb` file, and the unpacked size. The latter information is especially useful when trying to figure out whether you will have enough hard drive space to install a package.

There are several levels of abstraction that control package management. The interface we will use is called APT – "A Package Tool". A lower-level interface is provided by the **dpkg** program.

For more information on packages and dependencies, see the APT-HOWTO. **(Where?? -P.)**

8.6.1 apt-get

The classic Debian interface to APT is a command-line based utility called **apt-get** . It is great for installing packages easily, but not so good for browsing through packages or learning the names of packages you want to install.

Before installing packages, you will want to refresh the local package database to make sure that it is current. To do this, type

```
apt-get update
```

To install packages `foo` and `bar` and all their "depends" dependencies (but *not* their "suggested" or "recommended" dependencies) type

```
apt-get install foo bar
```

As usual, you need to be the root user to do this.

For more information, see the **apt-get**(8) man pagman page.

8.6.2 aptitude

aptitude is a console-based front end to Debian's package management system. It allows you to search for packages to install, select and deselect packages, and inspect package dependencies. It also shows estimated disk usage for packages that are to be installed.

In many cases you will want to use **aptitude** on the *server* to check dependencies and install sizes. Then you can install packages on the client using the command-line based **apt-get** utility.

aptitude is a wonderful program, but it is hampered by some really weird keystrokes. Here are the options I use most, and some of the quirks I have to work around:

- To start the program type `aptitude` on the command line. You need not be the root user to do this, which is nice if you only want to browse packages. If you try to install or remove packages, though, you will be prompted for the root password.
- The program is mouse-sensitive. If you click in a window, pressing keys will affect only that window.
- The most confusing key is `q` . This is the "back button" of the program – it takes you to the previous screen. If you press `q` enough times you will eventually be able to exit the program.
- The most important key is `h` . This brings up the online help. To get out of help, press `q` (Back button, remember?).
- The second most important key is `/` . This allows you to enter a search string so that you can find packages by name quickly. The third most important key is \, which repeats your last search.

  Searching is quite sophisticated. By prepending the string `~d` to a query, you can search via description. Before doing this, however, you will want to turn off partial matching: press ⟨F10⟩, then select **Options** → **UI Options** . Unselect the **Show partial search results** option.

- If you have highlighted a package, pressing ⟨enter⟩ will being up a more detailed description. of the package. This is great for inspecting dependencies.
- To select a package, highlight it and press `+` . To unselect a package, press `-` . Note that `+` and `-` keystrokes cancel each other out, so if you mistakenly select a package for installation, you can unselect it with `-` .

  There is also a `=` keystroke that puts packages on hold, so they will be forced to remain in their current state.
- To refresh your package list, press `u` .
- To actually install or remove packages after selecting them, press `g` . You may have to press `g` several times – once to show you the list of packages that will be installed, and once to install packages. If you are prompted for the root password, you will have to press `g` a third time to actually install the package list.

8.6.3 synaptic

synaptic is another frontend to APT. This one is GUI-based. From the installer, selecting the SYNAPTIC class (from the screen documented in Section 4.3.1) will install it.

This program will only work if you are logged in as the root user. The root user must have access to the X-display. **(provide more info -P.)**

In addition, the program uses up a lot of colours. It is unusable on a display with 256 colours.

synaptic provides about the same amount of functionality as **aptitude**. The interface is relatively straightforward to figure out, so I will only document things I found confusing here:

- There is a button that toggles between "search" and "filter" functionality. To access search mode, click on the magnifying glass icon.

8.6.4 packages.debian.org

Unfortunately, none of these package managers make it easy to search for packages based on their *description*. It is easy to find packages if you know the *name* of the package, but hard to learn the names of those packages if you don't know what you are looking for. You can look through packages one by one, but with over 10 000 packages available for Debian, this can take a while.

UPDATE: I have learned that **aptitude** allows you to search for package descriptions; see Section 8.6.2. However, searching via `http://packages.debian.org` seems to be faster.

To search for promising programs, I use `http://packages.debian.org` . This website provides three nice interfaces for package search:

- You can browse packages by category, if you know which Debian distribution (stable, testing or unstable) you want.
- You can search packages by keyword, either by package name or package description. This is the interface I use the most. Often, I will set the **distribution** field to "any" to determine whether newer versions of the package in question might be available.
- You can search packages by filename. This can be handy if a program complains that it cannot find a particular file.

Of course, standard web searches for programs works well too. You can also look in Section 11.1 to look through installation candidates we have already inspected.

9 Installing the project from scratch

Although we have released our project's configuration scripts as an open-source project, actually setting up the project after downloading those scripts takes some work. This section outlines the steps you would need to take in order to set up an installation server with our project files. If you already have access to a functional server (as we do at the Working Centre), you probably don't need to concern yourself with this section too much. However, if our server ever breaks or needs to be re-installed, this section might prove useful in getting the project re-installed.

Similarly, if you are a project administrator who downloaded our scripts from SourceForge and are wondering what to do with them, this section is for you.

9.1 Hardware you need

FAI is currently a network-based noninteractive installer. Thus, in order to get the installer working, you will want:

- A machine to act as a server. This server need not be fancy; our first server was a Pentium 90 with 64MB of RAM – and we could probably have scraped by with a lesser machine. The installer does not take a lot of computational power on the server. However, if you are planning to compile your own packages (or your own kernels) then having a fast processor and lots of memory can really speed things up.

  One thing that will help is to have lots of hard drive space available on the server. You will want at least 1GB of free space to install programs on the server, and 2GB is useful if you can get it. In addition, you will want to have at least 500MB of space to host a local NFS mirror for Debian packages – and preferably you want more space.

  You will want to install Linux on this server. Since our project is a Debian-based subdistribution, it is probably most convenient to install Debian on the server. This may not be strictly necessary, but it helps a lot. We'll assume that you have managed to manually install Debian on the server.

  For added security, you may want to partition the drive so that NFS-mounted directories are on their own partition. You might consider separate partitions for your local package mirror, for the NFS root directory that clients mount (which should not be bigger than 100MB), and for the WCLP configuration files (which is currently less than 15MB and should not be bigger than that unless you have a lot of custom kernels to install).

- A way to get Debian packages to the server. Broadband access to the Internet is nice, but not strictly necessary; if you have the distribution on CDs you can use those as well.

- Some kind of Internet connection really helps, especially when getting security updates. Again, this is not strictly necessary, but it can help a lot.

- Some kind of network. Our installation system is hooked up to our local network, but in fact the only parts of the network we actually use for installations are an eight-port hub and some RJ-45 cables.

- Network cards for the client machines. You can re-use network cards if they are scarce. The cards need not be fancy; 10 megabit ethernet cards work fine for us.

9.2 Install the configuration files

Once you have installed Debian onto your server, you can download our project's files from `http://wclp.sourceforge.net` . You will want all of the configuration files, and you may want our customized kernels as well.

Let's assume you unpack these scripts into the `/usr/local/share/wclp` directory.

9.3 Set up NFS

You need to configure NFS – the network file system – server in order to allow client machines to connect to the server.

First, install a NFS server package. If you have a kernel with NFS service support installed, install the `nfs-kernel-server` package. Otherwise you will want the `nfs-user-server` package.

We found that the `nfs-user-server` worked with a stock Debian kernel, and the other server did not.

To configure the server, you will need to set up the `/etc/exports` file. This file specifies the clients that are allowed to mount your directories, and which directories they are allowed to mount.

Our `/etc/exports` file looks like this:

```
/usr/local/mirror 192.168.1.0/255.255.255.0(ro,root_squash)

/usr/local/share/fai/installer 192.168.1.0/255.255.255.0(ro,root_squash)

/usr/local/share/fai/nfsroot 192.168.1.0/255.255.255.0(ro,root_squash)

/usr/local/mirror 192.168.1.190(ro,root_squash)
```

The first three lines make the server directories containing the mirror, installer configuration files and nfsroot accessible to any client with an IP address beginning with 192.168.1.

`ro` stands for "read only". `root_squash` means that the client cannot execute any commands in these directories as the root user.

The fourth line should be unnecessary. It specifies that the server can access the `/usr/local/mirror` directory.

For security purposes, you are supposed to put directories you export on their own partitions.

9.4 Set up a mirror

A local mirror of Debian packages speeds up installations considerably. If you want to mirror the entire Debian distribution for your architecture, you can use the **debmirror** script (available from the FAI home page), or the **demish** package.

We did not have the space to mirror the entire Debian archive, so we set up a partial mirror. To do this, we used the **apt-move** package. This involved using APT to download (and install) all the packages we wanted on a machine (in our case the server), and then using **apt-move** to transfer the files from the APT cache directory to our local mirror. See Section 7 and the **apt-move** manual for more details.

You will want to register your mirror with APT. We use the following lines in our `/etc/apt/sources.list` to register our mirror:

```
 % APT lines
deb file:/usr/local/mirror/debian/ woody main non-free contrib
deb file:/usr/local/mirror/debian/ woody/non-US main non-free contrib
deb file:/usr/local/mirror localdebs/
```

These lines should appear at the top of the `sources.list` file, so that APT considers them before accessing remote sites.

The first two lines access the structured mirror. The third line looks at the unstructured mirror. See the APT-HOWTO for more information. **(where? -P.)**

9.5 Set up FAI

In order to install anything you will need to set up the FAI installer. To do this, first install the Debian packages **fai** and **fai-kernels**. You will then need to configure FAI by editing the file `/etc/fai/fai.conf` .

9.5.1 fai.conf

By default, FAI wants to use either DHCP or BOOTP to assign IP addresses to clients. However, getting this to work properly involves collecting MAC addresses of network cards, which is bad for us because each computer we install could potentially have its own network card, and volunteers would have to modify the DHCP server settings every time they wanted to install the distribution on a new machine.

Instead of DHCP or BOOTP, we assign IP information via a boot floppy. Unfortunately, we ran into a snag: we could not append all the FAI options to the boot floppies using LILO's "append" parameter. Thus, we put the following options into the `fai.conf` file:

```
FAI_LOCATION="192.168.1.190:/usr/local/share/fai/installer"
FAI_FLAGS="verbose createvt"
FAI_ACTION="install"
```

Note that this is dangerous and wrong. It means the "sysinfo" option is permanently disabled, and FAI will attempt to wipe out a hard drive and install Linux whenever called.

In addition to these options, we modify the following lines:

NFSROOT_PACKAGES: We take out **reiserfsprogs** and add **dpkg-dev**, **discover**, **libdetect0**, **mtools** . Some of these will likely be included by default in future FAI versions.

FAI_DEBOOTSTRAP: Used to set up a core set of packages for the installer.

FAI_DEBMIRROR: We have a local mirror, so we specify it here.

FAI_SOURCES_LIST: These are the `sources.list` lines that use t he local mirror.

Note that you can also change the default root password for clients here. This password is encrypted; to encrypt your root password replacement, use the **mkpasswd** program.

You can override this password by specifying a different one in
`wclp/installer/class/DEFAULT.var`

(Hacks to make the boot floppies work.. -P.)

(do you have to change some FAI files for this to work? -P.)

9.5.2 make-fai-nfsroot

Once you have set up the `/etc/fai/fai.conf` file, you should be ready to set up the *nfsroot* – a stripped-down Debian installation that clients will mount to do their installs.

To make the nfsroot, become root and type the following command:

```
make-fai-nfsroot -v
```

FAI should start downloading packages and installing them into the nfsroot directory. It will also copy the `fai.conf` you set up, which means that changes to `/etc/fai/fai.conf` on the server will not be reflected in the nfsroot unless you explicitly copy the file over – and depending on the change, you may have to call **make-fai-nfsroot** again.

9.6 Finish up

9.6.1 Making FAI boot floppies

Our project boots clients using startup floppies, so if you are using a DHCP or BOOTP server, you can skip this step.

We created several boot floppies so that we could do multiple installs at once. Each boot floppy specifies a unique IP address and a unique hostname.

FAI provides a command called **make-fai-bootfloppy** to create a boot floppy. However, we modified this program so that we could specify an IP address explicitly. Our patched version is currently located in the `wclp/installer/local.files/scripts/` directory. **(Check this -P.)**

To use our version, simply move our **make-fai-bootfloppy** to `/usr/local/bin` , and the manpage file to `/usr/local/man/man1` .

To make a boot floppy, carry out the following steps:

1. Choose a hostname and IP address for the floppy. We chose to relate hostnames to IP addresses: if we chose 192.168.1.191 for the floppy IP, we called the host `wc_191` .
2. Put a floppy into the floppy drive, become the root user, and execute the command:

   `make-fai-bootfloppy -v -s` *`host`* `-i` *`ip`*

   where *`host`* is the hostname you chose, and *`ip`* is the IP address for this floppy.
3. To register this floppy, you may want to edit the `wclp/installer/class/S01alias.sh` file in the installer. You will want to configure your hostnames to select the MENU, AUTOCHOOSE and XSERVER-SELECT classes, as we do in the following fragment:

```
case $HOSTNAME in
    wc_???)
        echo MENU ;
        echo AUTOCHOOSE ;
        echo XSERVER-SELECT ;;
esac
```

At this point, you should be able to boot a client machine connected to the network with your new floppy. The client should be able to access your server and install Linux.

(Of course, it won't work this smoothly. Do I have any troubleshooting tips to offer? -P.)

10 Our Project's SourceForge Account

SourceForge is a web-based open source development resource. It provides tools, web hosting and source repositories that open-source developers can use to speed up their project development.

We currently use the following SourceForge features:

- We use SourceForge to host our project's homepage.
- We use the software repository to make releases of our installer and manuals available for download.
- We use SourceForge CVS to develop our documentation.

We may expand our SourceForge usage as the project matures. For example, we would like to use the bug-tracker to monitor problems with the distribution.

(Should this be a subsection of "Other tasks"? -P.)

10.1 URLs

Our project's homepage is located at

http://wclp.sourceforge.net

Our Sourceforge project page is located at

http://www.sourceforge.net/projects/wlcp/

The current project administrators are Paul Nijjar, Daniel Allen and Charles McColm.

(What do you do with these URLs? How do you edit the webpage? How do you upload files? -P.)

10.2 Creating your user account

You need a SourceForge account in order to do any of the following things:

- You want to use CVS to upload changes to our manuals.
- You want to take responsibility for bugs or tasks listed on our SourceForge project page.
- You want to develop our webpage.

(how about bug reports? -P.)

Specifically, you do *not* need an account if you are just using the installer, or even if you are editing installer files.

Creating an account is easy: just go to `http://www.sourceforge.net` , click on "New user via SSL", and follow the instructions.

However, an account alone is not enough to become a project developer. You also have to be registered as a WCLP project developer. To do this, you have to contact one of the current project administrators. Either talk to whomever is in charge of the Linux project at the Working Centre, or send an e-mail to one of the "Project Admins" shown on the WCLP project homepage. If the administrator agrees that it makes sense for you to become a developer (which is highly likely) then the administrator will add you to the WCLP developer list.

10.3 Using CVS

CVS stands for "concurrent version control". It is software designed to track changes to source code (the "version control" part) and to allow many different people to work on a single piece of software at the same time (the "concurrent" part).

Basic information on using SourceForge to upload and download our sources can be found in the SourceForge documentation:

`http://sourceforge.net/docman/display_doc.php?docid=763&group_id=1`

An introductory CVS manual can be found at:

`http://cvsbook.red-bean.com/`

You should *definitely* do a little bit of reading before uploading anything to CVS.

CVS is complicated software. It looks overwhelming at first. However, there are only a few commands you will use regularly:

- `cvs update` refreshes the sources from the CVS server.
- `cvs commit` uploads your changes to the CVS repository.
- `cvs add` adds new files to the repository. Warning: don't use this command unless you know what you are doing. CVS has quirks that make it tricky to upload binary files. It is probably best to run any additions you are going to make by the project administrator first.

All of these commands should be executed in the current CVS working directory.

10.3.1 Our CVS layout

Currently, we only store documentation on SourceForge CVS. The CVS directoryname for the documentation is called `manuals` . The CVS projectname is `wclp` .

The `faiadmin` account currently uses *anonymous* access to download manual sources. This is so that we can associate developer names with changes to the CVS source. It also means that you should not try to upload changes to the manuals from the

`/usr/local/share/fai/manuals/`

directory. Rather, you will want to download a private copy of the LaTeX sources from CVS, and edit that. In order to do this you will need to create a SourceForge account and register yourself as a developer for our project.

(To make a release, faiadmin should download the freshest version of the sources.. -P.)

10.3.2 CVS Tips and Quirks

If for some reason you cannot anonymously download the manual sources, try visiting the manuals via the web interface. Browse the CVS repository using your browser, then try anonymously downloading the sources again.

No. This *shouldn't* work. But for some reason it seems to help.

10.4 Bug Tracking

10.5 Releasing our software

Only project administrators can release our software to SourceForge.

(Show how to use the mkrelease script -P.)

(Show how to compile the documentation -P.)

11 Software notes

In this section we document software we have tried out, and document problems we have had with the different software packages.

11.1 Additional software to try

In deciding upon packages to include with the distribution, we had to look through many packages. We rejected a lot of good programs because they did not meet all of our criteria. However, other people might be interested in some of the interesting alternatives available for Debian.

In most cases, we just list Debian package names.

Window managers: There are many many window managers available. Some of the ones I like are: **wm2**, **blackbox**, **afterstep** . You can find others in the following ways:

1. Look at the dependencies of **xfree86-common**.
2. Look through the X11 section of packages on `http://packages.debian.org`
3. Visit `http://www.plig.org/xwinman/`

File managers: **xnc**, **xfm**, **gentoo**, **gmc**, **rox**, **worker**, **dfm**

There is a program called **foXcommander** that resembles Windows Explorer, but it is not available on Debian. If you want to try it, check out

`http://www.sourceforge.net/projects/foxdesktop/`

Password generators: **apg**, **pwgen**, **makepasswd**, **gpw**.

I like the first two of these the best.

Package managers: **aptitude**, **diety**, **stormpkg**, **synaptic**

I like **aptitude**, but it is console-based. **stormpkg** depends heavily on GNOME libraries, and is unusable on a machine with 16MB of RAM. **synaptic** is a better bet if you are looking for a graphical package manager.

You can read some hints on using the different package managers in Section 8.6

Web browsers: There are tons of web browsers available. Some of the graphical ones I like include **skipstone**, **opera**, **dillo**, **mozilla**. Note that **skipstone** and **mozilla** are not suitable for machines with 16MB of RAM.

Good text-based browsers include **lynx**, **w3m**, **links**. I like **w3m** the best.

No list of "web browsers" is complete without mentioning without **wget** . This utility downloads webpages from the command line.

Text editors: The two most popular editors are **emacs** and **vi**. Two popular flavours of **emacs** are **xemacs**, and GNU **emacs**. Three popular flavours of **vi** are **vim**, **elvis**, **nvi**

Beyond the big two, other popular text editors include **nedit** (graphical and powerful), **nano** (a clone of PICO, which is the text editor used in the **pine** e-mail program).

There are lots and lots of others. Look in the "Editors" section of
http://packages.debian.org

Drawing and Image Manipulation: **gimp** is well known in the Linux world as a Photoshop clone. It is better suited to higher-end machines. **xfig** is an excellent two-dimensional geometric drawing program. **scribus** wants to be Pagemaker when it grows up.

Spreadsheets: **siag**, **oleo** (text-based) are good alternatives to **gnumeric**

Word processors: **xpw**, **ted** are both good. Both save to RTF format. The **tetex-base** package provides LaTeX, which is a non-graphical typesetting system. **lyx** is a good LaTeX frontend.

Music and Sound: If you have a sound card, here is some software that you can use to take advantage of it.

- **mpg123** and **mpg321** are command-line MP3 players.
- **cdtools** is a set of command-line utilities for playing audio CDs in a CD-ROM drive.

Typing programs: **gtypist** is text-based, but is fairly easy to use and works on any machine. **tipptrainer** is more graphical, but seems to require a large display (800x600) in order to run properly. **xletters** is a fun typing game; **tuxtype** is a fancier version of the same game that uses the SDL libraries.

Games: Many many games are available. Many games use the SDL (Simple DirectMedia Layer) libraries, which make it easy for game programmers to access hardware. Unfortunately, the SDL libraries take up 15MB of storage space. Game listings with an "(SDL)" after their name depend on the SDL libraries.

Puzzle games: Most of the pre-installed games are puzzle games. However, there are some good additions:

- **frozen-bubble** (SDL) is a neat Puzzle-Bobble clone.
- In **mirrormagic** (SDL) you redirect laser beams to solve puzzles.
- **gemdropx** (SDL) is a Jewels/Tetris like game where you match up blocks.

Action games: Some of these games involve blowing things up, but in a "cutesy" way (which for some reason makes the violence less violent).

- **hatman** is a somewhat cute PacMan clone.
- **heroes** (SDL) is just weird; none of us have figured out exactly what its objective is yet – but the graphics are great!
- **zblast** is a space-shooter with simple graphics and great sound. It reminds me of the Sierra game Sylpheed. To use it properly on 640x480 resolution, you need to run it in "small" mode – see the manpage for more information.
- **xgalaga** is a clone of space shooter Galaga packaged for Linux.
- **nighthawk** is a clone of an odd Commodore 64 game called Paradroid. The objective is to traverse a ship by taking control of robots.
- **circuslinux (check name -P.)** (SDL) is a vaguely disturbing Breakout-type game where you bounce clowns around to pop balloons.
- **madbomber** (SDL) is a tiny little SDL game where you catch bombs in pails of water. It is not as violent as it sounds. (Nor is it a Bomberman clone.)
- **xdigger** is a smallish Boulderdash-type game. Run around in a maze and collect diamonds.

Simulation games: We include both simulation games and sports games (such as they are).

- **lincity** is a SimCity clone that takes only 2MB of disk space. **freeciv** is a much bigger (and slower) Civilizations clone that probably requires more than 16MB of memory to run well.
- **orbit** is a space-flight/combat simulator.
- **gtkpool** is a pool game. (Surprise!)

Violent games: These games are violent. Some of them depict graphic blood and gore. You have been warned.

- **nethack** is the most addictive game ever. Fortunately, its complicated interface and simple graphics scare people away, but it is not known as "GradeWrecker" for nothing. Roam a dungeon and kill things.
- **clanbomber-x11** is a Bomberman clone. **freecraft** is a WarCraft clone. Both are better suited to higher end systems, but they might be playable on 16MB with a fast processor and good video card.
- **xbill** is the classic human-squashing game. It is a lot more fun than the similar **bugsquish**
- **abuse-sdl** (SDL) is a side-scroller ported from DOS.
- **lxdoom** is DOOM packaged for Linux.
- **xevil** is fun, but features explicit sexual and drug references in addition to splattering blood.

Unusable games: Some games looked promising, but turned out to be unusable on our systems. Some games seem hopelessly broken (but may get better in the future) and others are just hopelessly slow.

- **pingus** appears to be broken, which is too bad because it promises to be a neat Lemmings clone.
- **powermanga** is a great vertical space-shooter – but it requires a Pentium to run. Apparently it requires 32MB of RAM to run as well, but I have gotten the game working on a machine with 16MB with few ill effects.
- **chomium** is supposed to be another vertical shooter, but it is way slow.
- **flightgear** is a flight simulator that is too slow for our server.
- **tuxracer**, **tuxcart** are slow without accelerated video cards.
- **xsoldier** needs a resolution higher than 800x600 to run. **WINGS** is another vertical space shooter that also requires a high resolution.
- **pipenightdreams** is a neat clone of the puzzle game PipeDreams. Unfortunately it has some bugs.

Miscellaneous: Other neat software that is available:

- **mgp**, a lightweight PowerPoint-type tool.
- **kali**, which is used to generate tesselations (think M.C. Escher)
- **octave**, which advertises itself as a MATLAB clone.
- **linneighborhood** is a great GUI SAMBA file browser.

(Mail programs, ICQ, IRC, sound and music... -P.)

12 Wishlist and Future Work

If we are lucky enough, this project will have been a success – customers will allow us to install Linux on their new systems, and they will find the software useful. If the project is a wild success, customers will *ask* us to install Linux on their systems.

As the project matures, it could grow in many different ways. There are a number of improvements we would like to see, which we will outline in the sections below.

12.1 Support Improvements

Currently, only a handful of volunteers understand the Linux project well enough to help users out when they are stuck. At the very minimum more volunteers should become familiar with the installation process, and with answering frequent Linux-related questions.

Also important is technical knowledge of the backend. Currently only *one* on the team understands the FAI backend enough to make modifications. We need more people to become familiar with the installer's workings.

A third area in which we need to improve is in online support. People should be able to go online and ask questions that volunteers can answer. (The KWLUG website may be the answer to this, but people are still too afraid to ask questions.)

12.2 Technical Wishlist

If you are looking for a challenge, perhaps you could implement one of our wishlist items.

Better software: As better versions of our existing software come out, we will probably want to upgrade it. One danger is that the packages we have chosen will bloat. In this case, we should be looking for alternatives.

We always want to be on the lookout for better software packages that are appropriate for our system. It would also be nice to recompile packages to make them leaner and meaner, but this is a lot of work to maintain.

Dual-booting: Some people might want our configuration in addition to some other operating system. This challenge involves finding *foolproof* ways to program hard disk partitioning schemes that will preserve existing data on hard disks while allowing us to install our system on another partition automatically. The current installer wipes out the existing hard drive contents and repartitions it before installing Linux.

Better documentation: We need to make our documentation available on local harddrives, and we need to make sure that users can access this information easily.

We also want to incorporate user experiences into our software, adding FAQs and tips to our documentation as we continue to learn about Linux and our user base.

Move documentation to docbook: The docbook SGML DTD is quickly becoming the standard documentation format for Linux technical issues. At some point we will want to convert our LaTeX documentation to SGML. LaTeX is a nice input language, but it is ill-suited to producing HTML-ized documentation.

GNOME and KDE: As newer computers come into the computer recycling project, we may eventually be able to install GNOME or KDE on some of our systems. This "challenge" may be as easy to solve as adding a Debian virtual package to a FAI package configuration file.

It is important to leave lower-end options available, though, because the computer recycling project is likely to get older computers for some time yet.

Better automation: It would be nice to automatically detect hardware, then use that information to select packages and create configuration files automatically. Other distributions (such as Mandrake) can do this, so maybe we can too.

PCMCIA installs: We have problems putting the distribution on notebook computers, because FAI does not currently support PCMCIA very well. Getting around this means changing FAI or using somebody else's modified version of FAI.

CD-ROM based installs: Marc Shaeffer has been working on modifying FAI so that you can burn a CD-ROM that contains a FAI installer and the software it is going to install. This

would be a neat thing to use in our project if we feel confident enough to move away from standard FAI.

If Shaeffer's changes get incorporated into the main FAI packages, then it would make a lot of sense to support a CD-based install.

A Common hardware configurations

A.1 Systems

A.1.1 AST Bravo LC 5100

Systems in the AST Bravo line sometimes fail to install LILO properly. The installer will complete successfully, but the system will not boot. You may see a lot of numbers on the screen.

What is the solution? We don't really know. Here are some things you can try:

- Boot to the system using a DOS floppy. Then format the hard drive and transfer the system to the hard drive (by typing `sys c:`). Then try the installer again.
- Again at a DOS prompt `fdisk /mbr` to overwrite the existing information.
- Use a single-floppy distribution (such as YARD) to boot into the system, then mount the hard drives manually and run LILO manually.

Since we have moved from LILO to GRUB, this problem may be gone. We have not verified this.

None of these solutions are very good. If you find a better solution, please report it!

A.1.2 Award BIOS 4.50G

The Award BIOS version 4.50G is *not* Y2K compliant. This BIOS can be found in several higher-end 486 models. You will notice this problem when you set the time: the date will revert to 1994 every time the computer reboots.

There is no BIOS fix for this, but fortunately the **hwclock** program can deal with this situation. Go to

`/etc/defaults/rcS`

and add the line

`BADYEAR=yes`

This will tell the **hwclock** command to ignore the year in the BIOS and manage the year manually. The script that uses this variable is `/etc/init.d/hwclock.sh`

Once you have added the line, reboot the system and reset the time. Then reboot again to see if the system holds the right time.

A.2 Monitors

There are some good online resources that list monitor refresh rates and settings. Some of the best ways to find monitor information online are:

- Visiting `http://www.monitorworld.com` As of this writing, this is the best resource available. It has a fairly comprehensive database containing lots of monitors.
- Performing a search engine search for the monitor make and model.

A.3 Network Cards

A.3.1 Finding network card information

FAI will detect almost every network card that works under Linux, so the first place to look for network card information is the `dmesg.log` file in the FAI logs. Look through this file to see what configuration information the network card driver reported. Here is an example of what to look for:

```
tulip.c:v0.91g-ppc 7/16/99 becker@cesdis.gsfc.nasa.gov
eth0: Digital DS21140 Tulip rev 32 at 0x6100, 00:80:C8:34:A8:48, IRQ 11.
eth0:  EEPROM default media type Autosense.
eth0:  Index #0 - Media MII (#11) described by a 21140 MII PHY (1) block.
eth0:  MII transceiver #8 config 3100 status 7829 advertising 01e1.
```

This tells you that the network card uses the `tulip` driver, that the card uses IRQ 11, and that the IO address is (probably) 0x6100 . This information is valuable if you need to specify parameters for kernel modules.

See Section 6.1.3 for more information about FAI log files, such as where to find them.

A.3.2 Linksys LNE 100TX

The autodetection tools **detect** and **discover** do not yet detect these cards properly. However, they are supported by the `tulip` driver.

These cards are important because apparently Sympatico installs these cards for its ADSL subscribers – removing any existing network cards in the process! However, Sympatico technicians "do not support Linux", so we have to fix their damage by installing the right driver in `/etc/modules` .

A.3.3 NE 2000

These network cards use the `ne` kernel module. The FAI kernel detects them properly, but **discover** does not.

If you want to use the network after FAI has finished installing and you have rebooted, you have to find out the IO address of the card, and you have to manually add a line to `/etc/modules` :

`ne io=`*`io-address`*

where *`io-address`* is the IO-address of the card. I have found that `io=0x240` usually works.

A.4 Internet Service Providers

Even though ISPs are not hardware, it is worth mentioning information we have learned about different ISPs, as most ISPs do not officially support Linux for dial-up connections.

Since service providers are always changing, each entry here includes the date on which we entered the information.

A.4.1 Gateway

Gateway `http://www.gate-way.net` does work with Linux, but **wvdial** has problems dialing to it. Apparently, Gateway uses PAP authentication.

We set up a connection using **pppconfig**, and it worked well.

This information was entered October 22, 2002.

A.4.2 Sympatico dialup

In addition to providing DSL service, Sympatico currently manages a dial-in pool as well. We had no problems connecting to Sympatico using **wvdial** .

This information was entered October 22, 2002.

A.5 Resource usage tables

This section documents common resource allocations for devices under Linux. It may be useful when configuring modems and sound cards.

A.5.1 Serial ports

This is taken from the file `/etc/serial.conf`

```
These are the standard COM1 through COM4 devices
#
#/dev/ttyS0 uart 16450 port 0x3F8 irq 4
#/dev/ttyS1 uart 16450 port 0x2F8 irq 3
#/dev/ttyS2 uart 16450 port 0x3E8 irq 4
#/dev/ttyS3 uart 16450 port 0x2E8 irq 3
```

A.5.2 Typical IRQ allocations

This is taken from the Modem-HOWTO, which you can find at
`/usr/share/doc/HOWTO/en-txt/Modem-HOWTO.gz`

You can sometimes break some of these assignments (for example, I often set network cards to IRQ 5) but be aware that you could generate IRQ conflicts by doing so.

You really really want to leave IRQs 1, 2, 6, 8, 9, 13, and 14 alone. You probably want to avoid reallocating IRQs 3, 7 and 15 unless you know what you are doing.

```
Standard IRQ assignments:

        IRQ  0    Timer channel 0 (May mean "no interrupt".)
        IRQ  1    Keyboard
        IRQ  2    Cascade for controller 2
        IRQ  3    Serial port 2
        IRQ  4    Serial port 1
        IRQ  5    Parallel port 2, Sound card
        IRQ  6    Floppy diskette
        IRQ  7    Parallel port 1
        IRQ  8    Real-time clock
        IRQ  9    Redirected to IRQ2
        IRQ 10    not assigned
        IRQ 11    not assigned
        IRQ 12    not assigned
        IRQ 13    Math coprocessor
        IRQ 14    Hard disk controller 1
        IRQ 15    Hard disk controller 2
```

B Troubleshooting

This section documents error messages and situations you might run into when installing the system, and whatever solutions we have come up for those problems. If you know of other problems and/or solutions, please let us know so we can add to this list.

B.1 Problems when starting the install

These are problems that crop up from the time you boot the client until the client reboots for the first time.

B.1.1 `Neighbor table overflow`

If you boot a client machine and get this message, it usually means that the client cannot NFS-mount the FAI directories from the server. This could be caused by a number of things:

- The client's network card may not have been recognised. To check this, type ⟨shift⟩+⟨page up⟩ to scroll through the boot messages. Make sure that the kernel has recognised the `eth0` network interface. **(what does the message look like? -P.)**

  If it seems that the network card has not been recognised, try switching network cards.

- The client may not be seeing the network. Often this is as simple as forgetting to plug a network cable into the client's network card. (I have done this many times..)

B.1.2 `Kernel panic: could not mount root fs`

This also means that the client could not NFS-mount directories from the server. The same fixes as the "neighbor table overflow" message apply here.

B.2 Problems during configuration

This covers problems you might run into when trying to configure the X-server and set up the system. You might also look at Section 5 for more troubleshooting information.

B.2.1 None of the software on the system is right!

If you notice that a significant amount of software is missing, or a lot of weird packages were installed, make sure that you did not use the boot floppy labelled "Custom boot floppy! Experts only!" If you did this, then you inadvertently used the boot disk the project administrators use to develop the installer. You should have known better; now you will have to re-install the system using one of the regular boot floppies.

B.2.2 `Sorry, the following packages failed to install:`

At one point, some Debian packages were broken – the packages themselves work, but FAI did not install them properly. The problem turned out to be that swap space was not enabled, so the installer was running out of memory.

If you see any messages like:

`Sorry, the following packages failed to install:`

then the configuration was unsuccessful. This indicates some bug with the installer; you should report a problem and write down the configuration for the installed system – especially the packages you installed, the amount of memory the system had, and the hard drive size. You might see this package during the first installation phase (when packages are being downloaded or configured) or when you are trying to configure the system.

Running `dpkg --configure --pending` might fix matters.

Since we now activate swap space, one possible cause of this problem is that some of the package dependencies are actually broken. Sometimes this is because packages are missing from the local mirror, and sometimes this is because upgraded packages have errors. In these cases please contact the project administrator: this is an important problem that will affect all installations.