The Debian GNU/Linux FAQ

Susan G. Kleinmann (sgk@debian.org),
Sven Rudolph (sr1@inf.tu-dresden.de), and
Joost Witteveen (joostje@debian.org)

24 August 1996

This document answers questions frequently asked about the Debian distribution of GNU/Linux. (See section 1.4 for the Copyright details.)

Contents

1 Meta-questions

1.1 In what format is this document written?

This document was written using Linuxdoc-SGML. The Linuxdoc-SGML system can be used to create files in a variety of other formats, e.g., GNU info, HTML, LaTeX, TeX .dvi format, and Postscript (TM) files.

1.2 Where do I get the latest version of this document?

The latest version of this document can be viewed in HTML format at the *Debian home page* `<http://www.debian.org/FAQ>`.

It is also available in plain text format at any of the Debian mirrors in the directory `doc/FAQ` as `debian-faq.txt`.

The original SGML file used to create this document is available as part of the Debian package called doc-debian_VVV-RRR.deb, where VVV is the latest version number and RRR is the latest revision number. This package is normally stored in the directory `Debian-1.1/binary-all/doc`. The `linuxdoc-sgml` package contains tools to transform this document into one or more other formats, as required by the user: GNU info files, HTML, LaTex, TeX .dvi format, and Postscript.

1.3 Where do I send questions, corrections, etc. about this document?

Comments on this document are welcome. Please send Email to debian-faq@debian.org.

1.4 (How) Can I redistribute this file?

This document is copyright 1996 by 14.3.1.

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this document under the conditions for verbatim copying, provided also that the section entitled "GNU General Public License" is included exactly as in the original, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this document into another language, under the above conditions for modified versions, except that the sections entitled "GNU General Public License" and this permission notice may be included in translations approved by the Free Software Foundation instead of in the original English.

1.5 What information resources were used to obtain the questions and answers presented here?

This FAQ previously was maintained by J.H.M. Dassen (jdassen@wi.LeidenUniv.nl) and Chuck Stickelman (stick@richnet.net).

Parts of the information came from

- The *Debian-1.1 release annoucement* `<http://www.debian.org/1.1/release.html>` by Bruce Perens.
- The *Linux FAQ* `<http://www.cl.cam.ac.uk/users/iwj10/linux-faq/>` by *Ian Jackson* `<http://www.cl.cam.ac.uk/users/iwj10/ijackson.html>`.
- The *Debian Mailing List Archives* `<http://www.debian.org/Lists-Archives/>`
- The `dpkg programmers' manual`, which is incorporated into recent versions of the Debian package known as `dpkg`.
- The `Debian policy manual`, which is incorporated into recent versions of the Debian package known as `dpkg`.
- The many developers, volunteers, and beta testers.
- The flaky memories of its authors.

The authors would like to thank all those who helped make this document possible.

2 Definitions and Overview

2.1 What is Debian GNU/Linux?

Debian GNU/Linux is a particular *distribution* of the Linux operating system, and numerous packages that run on it. In principle, users could obtain the Linux kernel from the 'net or elsewhere, and compile it themselves. They could then obtain source code for many applications in the same way, compile the programs, then install them into their systems. For complicated programs, this process can be not only time-consuming but error-prone. To avoid it, users often choose to obtain the operating system and the application packages from one of the Linux distributors. What distinguishes the various Linux distributors are the software, protocols, and practices they use for packaging, installing, and tracking applications packages on users' systems, combined with installation and maintenance tools, documentation, and other services.

Debian GNU/Linux is the result of a volunteer effort to create a free, high-quality Unix-compatible operating system, complete with a suite of applications. The idea of a free Unix-like system originates from the GNU project, and many of the applications that make Debian GNU/Linux so useful were developed by the GNU project. Debian was created by Ian Murdock in 1993, initially under the sponsorship of the Free Software Foundations's GNU project. Today, Debian's developers think of it as a direct descendent of the GNU project.

Debian GNU/Linux is

- **full featured**: The Debian distribution includes nearly 500 software packages at present. Users can select which packages to install; Debian provides a tool for this purpose. You can find a list and descriptions of the packages currently available in the Debian distribution at any of the Debian mirror sites.
- **free to use and redistribute**: There is no consortium membership or payment required to participate in its distribution and development. All packages that are formally part of the Debian distribution are free to redistribute, usually under terms specified by the GNU General Public License. The Debian FTP archives also carry approximately 50 software packages (in the `non-free` and `contrib` directories at the FTP archives), which are distributable under specific terms included with each package.
- **dynamic**: With about 100 volunteers constantly contributing new and improved code, the Debian distribution is evolving rapidly. New releases are made about every three months, and the FTP archives are updated daily.

Though Debian itself is free software, it is a base upon which value-added GNU/Linux distributions can be built. By providing a reliable, full-featured base system, Debian provides GNU/Linux users with increased compatibility, and allows GNU/Linux distribution creators to eliminate duplication-of-effort and focus on the things that make their distribution special.

2.2 Ok, now I know what Debian is...what's Linux!?

In short, Linux is the kernel of a Unix-like operating system. It was originally designed for 386/486/Pentium PCs; now, ports to other systems, including multi-processor systems, are under development. Linux is written by Linus Torvalds and many computer scientists around the world.

Besides its kernel, a "Linux" system usually has

- a file system that follows the Linux Filesystem Hierarchy Standard `<ftp://tsx-11.mit.edu/pub/linux/docs/linux-standards/private/fsstnd/draft5.tar.gz>`.
- a wide range of Unix utilities, many of which have been developed by the GNU project and the Free Software Foundation.

The combination of the Linux kernel, the filesystem, the GNU and FSF utilities, and the other utilities are designed to achieve compliance with the POSIX (IEEE 1003.1) standard. **(The authors would very much like to give you a pointer to an on-line document that described that standard, but the IEEE is another one of those organizations that gets away with declaring standards and then requiring that people PAY to find out what they are. This makes about as much sense as having to find out the significance of various colored lights on traffic signals.)** At least one distribution (*Linux-FT by LaserMoon* `<http://www.lasermoon.co.uk/linux-ft/linux-ft.html>`) has already been certified POSIX compliant.

For more information about Linux, see Michael Johnson's *INFO-SHEET* `<ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/INFO-SHEET>` and *META-FAQ* `<ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/META-FAQ>`.

2.3 What's the difference between Debian and other Linux distributions?

Three key features distinguish Debian from other distributions of GNU/Linux:

The Debian package maintenance system:

The entire system, or any individual component of it, can be upgraded in place without reformatting, without losing custom configuration files, and (in most cases) without rebooting the system. Most GNU/Linux distributions available today have some kind of package maintenance system; the 6 is unique and particularly robust.

Open development:

Whereas other GNU/Linux distributions are developed by individuals, small, closed groups, or commercial vendors, Debian is the only GNU/Linux distribution that is being developed cooperatively by many individuals through the Internet, in the same spirit as GNU/Linux and other Free Software. More than 100 volunteer package maintainers are working on over 500 packages and improving Debian GNU/Linux. The Debian developers contribute to the project not by writing new applications (in most cases), but by packaging existing software according to the standards of the project, by communicating bug reports to upstream developers, and by providing user support. See also additional information on 14.1.

The Bug Tracking System:

The geographical dispersion of the Debian developers required sophisticated tools and quick communication of bugs and bug-fixes to accelerate the development of the system. Users are encouraged to post bugs in a formal style, which are quickly accessible by both FTP and WWW archives to developers and users alike. See additional information in this FAQ on the management of the 13.

2.4 How does the Debian project fit in or compare with the Free Software Foundation's GNU project?

The Debian system builds on the ideals of free software first championed by the FSF and in particular by Richard Stallman. FSF's powerful system development tools, utilities, and applications are also a key part of the Debian system.

At one time, there was a formal relationship between the Debian project and the FSF, in that the FSF employed Ian Murdock for a year while he was project leader. The project was then called the "Debian GNU/Linux" system. Now, however, the FSF will pursue its long-standing objective of developing a new operating system based on HURD (`<http://www.cs.pdx.edu/ trent/gnu/hurd/index.html>`), rather than the Linux kernel, while the Debian project will take exclusive control over its own technical direction.

The Debian project supports the goals of FSF and we like to think of Debian as "Son of GNU". We think the FSF considered Debian as a first step toward the completed HURD system, and still encourage FSF to derive from Debian. Though the Debian project is no longer sponsored by the FSF, both groups have decided that this should not keep us from working together, and that we should be partners. We all look forward to more support for Linux in GNU software, and more support for FSF's goals in Linux software.

2.5 How does one pronounce Debian and what does this word mean?

The project name is pronounced Deb'-ian, with a short e, and emphasis on the first syllable. This word is a contraction of the names of Debra and Ian Murdock, who founded the project. (Dictionaries seem to offer some ambiguity in the pronunciation of Ian (!), but Ian prefers ee'-an.)

3 Hardware Requirements

3.1 On what architectures/systems does Debian GNU/Linux run?

The Debian Linux distribution includes complete source-code for all of the included programs, so it should work on all systems which are supported by the Linux Kernel; see the *Linux FAQ* `<http://www.cl.cam.ac.uk/users/iwj10/linux-faq/>` for details.

The current Debian Linux distribution contains a complete, binary distribution for Intel-based PC's, including 80386, 80486, Pentium, and Pentium Pro's. The development of binary distributions of Debian 1.1 for 68k processors is currently underway, and ports to Alpha, Sparc, and MIPS processors are expected to follow.

3.2 What hardware is assumed by the stock Debian GNU/Linux boot disks?

The configuration file used to build the standard distribution Debian GNU/Linux kernel assumes an 80386 CPU, and includes support for PCMCIA cards, most SCSI cards for which there exist Linux drivers (with the notable exception of the NCR53c810), and most Ethernet cards for which there exist Linux drivers.

Linux kernels suited to a variety of special platforms are included in the directory `buzz-fixed/disks-i386/special-kernels` at any Debian FTP archive site, along with their configuration files. Briefly, these platforms include:

- Special Kernel 0: Small kernel (Standard Distribution, but without drivers for PCMCIA, QLlogic SCSI host adapters, Ethernet adapters, and Sound cards)
- Special Kernel 1: All Adaptec SCSI + AdvanSys + Always + AM5379c974 + BusLogic
- Special Kernel 2: DTC + eata_dma (DPT), eata_pio (OLD DPT) + Future Domain
- Special Kernel 3: NCR5380/53c400 + NCR53c406a + NCR53c7/8xx
- Special Kernel 4: Iomega Parallel + PAS16 + Qlogic + Seagate + Trantor + UltraStor
- Special Kernel 5: Special NCR53c400 only Kernel
- Special Kernel 6: Special NCR53C8xx only Kernel

3.3 What amount of disk space is recommended?

A generous installation, sufficient to accommodate a few users, X windows software, and several large applications, might require disk partitions at least as large as:

- 50 MBytes for the root directory (/)
- 500 MBytes for /usr
- 50 MBytes for swap space
- 50 MBytes for each home directories
- 100 MBytes for /tmp
- 100 MBytes for /var

The optimum disk space allocated for swap depends critically on the way the system will be used. Many people just choose to set aside twice as much disk space as they have RAM space. Systems with large RAM may not need so much swap space, especially if there are only a few users. The installation process supports systems with no swap space.

3.3.1 But how much RAM and disk space are absolutely essential?

These *minimal* requirements are sufficient for a system without X windows and only 1 or 2 users:

- 15 MBytes of disk space for the `base` system; this provides a minimally-functioning Unix system, but includes no application programs and no network support.
- 80 MBytes of disk space for a standard system running applications on the console (i.e., without an X11 interface).

- 120 MBytes of disk space for a standard system running applications with an X11 window system.
- 300 MBytes and more for all all the optional programs (You probably don't need all of these prorams.)

Debian Linux can be installed on systems with only 4 MBytes of RAM. The latest installation disks are especially organized to provide an easy installation path for machines with small memories. Some users report success at using Debian Linux to convert PC's having limited RAM (and disk space) into X terminals. An 80386-based system with only 4 MBytes of RAM and 40 MBytes disk space has been used to run Debian Linux in this way; i.e., both networking and basic X11 server functions operated satisfactorily. This mode of operation even works if 1 MByte of the RAM is used as a ramdisk when the machine is booted, implying that only 3 Mbytes of RAM is absolutely essential for using Debian Linux on a PC in order to use it as an X server. This mode of operation requires a swap partition; without it, the system won't even go into multi-user mode.

3.4 How should I partition my drive?

Partitioning a drive has the disadvantage that drive space can be used much less flexibly than an unpartitioned drive. Most users find, however, that this disadvantage is more than offset by the fact that damage to a filesystem on a partitioned disk is usually limited to a single partition. Furthermore, backups of a partitioned hard disk can be more easily managed because the files that change most frequently are likely to be localized to a single partition.

A user with a 1.6 GByte drive has concluded after a survey of Debian users that it's reasonable to design a partitioning scheme that closely follows the Linux standard *File System Structure* `<http://www.eg.bucknell.edu/ quinlan/fsstnd/>`. For his 1.6 GByte disk, he chose these partitions:

- 30 MBytes for the root directory (/)
- 450 MBytes for /usr
- 50 MBytes for swap space
- 1000 MBytes for home directories (some of this could be used for /usr/local)
- 0 MBytes for /tmp; make /tmp a symbolic link to /var/tmp
- 40 MBytes for /var

Note that for best performance one should definitely place the swap file in its own partition. Ideally, this partition would reside on a drive controlled by a different controller than that which contains the user partition. (But hard evidence that this approach really yields better performance is lacking at present.)

3.5 Are very large disks supported?

There is an upper limit on the size of the disk partition that's used for booting. This limit applies to all operating systems, not just Linux, and not just Debian Linux. Basically, the BIOS's typically available on PC's cannot access disk partitions larger than 1024 cylinders or tracks. Thus, *any* operating system used on a PC cannot be booted from a disk partition larger than 1 GByte. It's worth emphasizing that this restriction only applies to the partition from which Linux is booted. Other partitions can be larger. One solution to this limitation is to place the directory `/boot/` in its own (very small) partition, entirely within the first 1024 blocks of the disk.

Recent versions of the Linux kernel include the the Multi-Device disk driver ('md'), which provides striping support (sometimes called RAID 0) in software as well as a linear mode of joining 2 or more disks into a single logical device that can be formatted as a single file system.

3.6 (How) does Debian provide PCMCIA support?

The version of the kernel distributed with the Debian installation disks includes support for PCMCIA cards. That is, its `.config` file includes the line: `CONFIG_BLK_DEV_IDE_PCMCIA=y`. This configuration just supports modification of the IDE driver in the kernel to support IDE disks that are PCMCIA cards.

Utilities that actually provide pcmcia card services have been developed by David Hinds. These utilities are provided in the Debian distribution by the package `pcmcia-cs-KKK_VVV-RRR.i386.deb`, where the components 'VVV' and 'RRR' follow the usual conventions on 6.3, and the component 'KKK' refers to the kernel version for which the `pcmcia-cs` package was built.

4 Compatibility issues

4.1 How compatible is Debian with other distributions of Linux?

Debian developers communicate with other GNU/Linux distribution creators in an effort to maintain binary compatibility across Linux distributions. Most commercial GNU/Linux products run as well under Debian as they do on the system upon which they were built.

Debian GNU/Linux adheres strictly to the *Linux File System Structure* `<http://www.eg.bucknell.edu/ quinlan/fsstnd/>` (now known as the FHS). However, there is room for interpretation in some of the rules within this standard, so there may be differences between a Debian system and other Linux systems.

4.2 Can I use Debian packages (".deb" files) on my RedHat/Slackware/... Linux system? Can I use RedHat packages (".rpm" files) on my Debian Linux system?

Different Linux distributions use different package formats and different package management programs.

You probably can:

A program to unpack a Debian package onto a Linux host that's been built from a 'foreign' distribution is available, and will generally work, in the sense that files will be unpacked. The converse is probably also true, that is, a program to unpack a RedHat or Slackware package on a host that's based on

Debian Linux will probably succeed in unpacking the package and placing most files in their intended directories. This is largely a consequence of the existence (and broad adherence to) the Linux File System Standard.

You probably don't want to:

Most package managers write administrative files when they are used to unpack an archive. These administrative files are generally not standardized. Therefore, the effect of unpacking a Debian package on a 'foreign' host will have unpredictable (certainly not useful) effects on the package manager on that system. Likewise, utilities from other distributions might succeed in unpacking their archives on Debian systems, but will probably cause the Debian package management system to fail when the time comes to upgrade or remove some packages, or even simply to report exactly what packages are present on a system.

A better way:

The Linux File System Standard (and therefore Debian Linux) requires that subdirectories under `/usr/local` be entirely under the user's discretion. Therefore, users can unpack 'foreign' packages into this directory, and then manage their configuration, upgrade, removal individually.

4.3 Is Debian able to run my old "a.out" programs?

- To *execute* a program whose binary is in `a.out` (i.e., QMAGIC or ZMACI) format,
  - make sure your kernel has `a.out` support built into it, either directly (CONFIG_BINFMT_AOUT=y) or as a module (CONFIG_BINFMT_AOUT=m). (Debian is distributed with a kernel with support for `a.out` binaries built in directly.) If your kernel supports `a.out` binaries by a module, then be sure that the `binfmt_aout` module is loaded. You can do this at boot time by entering the line `binfmt_aout` into the file `/etc/modules`. You can do it from the command line by executing the command `insmod DIRNAME/binfmt_aout.o` where `DIRNAME` is the name of the directory where the modules that have been built for the version of the kernel now running are stored. On a system with the 2.0.0 version of the kernel, `DIRNAME` is likely to be `/lib/modules/2.0.0/fs/`.
  - install the package `libc4`.
- To *compile* programs in the `a.out` binary format,
  - install the Debian `a.out` developer's packages, which are
    - `libc4-dev`
    - `aout-binutils`
    - `aout-librl`
    - `aout-gcc`
  - place the `a.out` tools ahead of the `ELF` tools in your path. That is, execute the command `export PATH=/usr/i486-linuxaout/bin:`$PATH(Thisisn'tessential, justadvantageous.)Ifyou'reonlygoingtodothisonce, youcouldexecute:$ `PATH` = `/usr/i486 − linuxaout/bin` :`PATH make [target]`.
- To execute `a.out` X clients, install the `xcompat` package.

If you have a commercial application in `a.out` format, now would be a good time to ask them to send you an `ELF` upgrade.

4.4 How should I install a non-Debian package?

Files under the directory `/usr/local` are not under the control of the Debian package management system. Therefore, it is good practice to place the source code for your program in /usr/local/src/. For example, you might extract the files for a package named "foo.tar" into the directory `/usr/local/src/foo`. After you compile them, place the binaries in `/usr/local/bin/foo`, the libraries in `/usr/local/lib`, and the configuration files in `/usr/local/etc/`.

If your programs and/or files really must be placed in some other directory, you could still store them in `/usr/local`, and build the appropriate symbolic links from the required location to its location in `/usr/local`, e.g., you could make the link

```
ln -s /usr/local/bin/foo /usr/bin/foo
```

In any case, if you obtain a package whose copyright allows redistribution, you should consider making a Debian package of it, and uploading it for the Debian system. Guidelines for becoming a package developer are included in the Debian `doc-debian` package; or see the *on-line guide* `<http://www.debian.org/Documentation/Debian/guidelines_toc.html>` to the package development guidelines.

4.5 Why can't I compile programs that require libtermcap?

Debian uses the `terminfo` database and the `ncurses` library of terminal interface routes, rather than the `termcap` database and the `termcap` library. Users who are compiling programs that require some knowledge of the terminal interface should replace references to `libtermcap` with references to `libncurses`.

To support binaries that have already been linked with the `termcap` library, and for which you don't have the source, Debian provides a package called `termcap-compat`. This provides both `libtermcap.so.2` and `/etc/termcap`. Install this package if the program fails to run with the error message "can't load library 'libtermcap.so.2'", or complains about a missing `/etc/termcap` file.

4.6 Why can't I install AccelX? (termcap again)

AccelX uses the `termcap` library for installation. Debian uses the `terminfo` database and `ncurses` library instead. To run programs which use the `termcap` library, install the package `termcap-compat`; it can be found in the `misc` section in the Debian archives.

4.7 Why do I get "Can't find libX11.so.6" errors when I try to run `foo`?

Programs that require the `libX11.so.6` library are in `a.out` format. To execute them, you need to install the `xcompat` package.

4.8 Can I install and compile a kernel without some Debian-specific tweaking?

Yes. But you have to understand the Debian policy with respect to headers.

The Debian C libraries are built with the most recent *stable* releases of the `gcc` headers. For example, the Debian-1.1 release uses version 5.2.18 of the headers. This practice contrasts with the Linux kernel source packages distributed at all Linux FTP archive sites, which uses even more recent versions of the headers. The kernel headers distributed with the kernel source are located in `/usr/include/linux/include`.

If you need to compile a program with kernel headers that are newer than those provided by `libc5-dev`, then you must add `-I/usr/src/linux/include` to your command line when compiling. This came up at one point, for example, with the packaging of the automounter daemon (`amd`). When new kernels changed some internals dealing with NFS, `amd` needed to know about them. This required the inclusion of the latest kernel headers.

5 Software Available in the Debian system

5.1 What types of applications and development software are available for Debian GNU/Linux?

Like most Linux distributions, Debian GNU/Linux provides

- the major GNU applications for software development, file manipulation, and text processing, including gcc, libc, g++, make, texinfo, gnuplot, the Bash shell and numerous upgraded Unix utilities.
- Perl and its libraries
- TeX, LaTeX, and dvips
- the X Window system, which provides a networked graphical user interface for Linux, and numerous X applications,
- a full suite of other networking applications, including WWW servers, browsers, and development tools.

Nearly 500 packages, ranging from news servers and readers to sound support, FAX programs, database and spreadsheet programs, image processing programs, communications, net, and mail utilities, Web servers, and even ham-radio programs are included in the distribution. Another 50 software suites are available as Debian packages, but are not formally part of the Debian distribution due to license restrictions.

5.2 Who wrote all that software?

- *Authors* whose software is included in the Debian Linux system are credited in files listed in `/usr/doc/copyright`.
- *Maintainers* who package this software for the Debian Linux system are listed in the 6.4 that comes with each package.

5.3 How can I get a current list of programs that have been packaged for the Debian project?

A complete list is available from any of the Debian mirrors.

The file `indices/Packages-Master-i386.gz` provides a list, including short descriptions, of all packages that are available for computers with 80386 (or more advanced) chips. The file `indices/Packages-Master-m68k.gz` provides a similar list of packages that are available for computers with Motorola 68xxx CPUs.

The *WWW interface to the Debian packages* `<http://www.debian.org/FTP/>` conveniently summarizes the packages in each of about twenty "sections" of the Debian archive, as well as the 10 most recently uploaded packages.

5.4 What's missing from Debian GNU/Linux?

A list of packages which are most urgently needed for the Debian distribution is maintained by Sven Rudolph. For more details, see the section 14.1.

5.5 Where is 'which'?

This is a historical question, no longer relevant, but is included with this FAQ because it may still hold the record as the single most frequently asked question on the Debian mailing lists.

The functionality of the Berkeley utility `which` is provided in the Bash shell by the built-in command `type`. To learn the path name for an executable file "foo", use the command `type foo`. To learn all the paths where "foo" can be found, use `type -a foo`.

To accommodate users' lack of familiarity with this (in this writer's opinion, obscure) feature of Bash, the Debian maintainers have built a package called `debianutils`, which includes a Bash script called `which`. This script includes only these lines:

```
#!/bin/bash
for i in $*; do
  type -all -path $i | head -1
done
```

The `debianutils` package is normally stored in the `base` section of the FTP archive, and is therefore part of the initial installation of every Debian system.

5.6 (How) Does Debian support Java?

The Java Development Kit from Sun is currently available as a Debian package (`jdk_VVV-RRR_i386.deb`). The JDK will allow you to run Java programs and applets, and write your own. If your kernel is properly configured (see below), the JDK will allow you to *run* Java programs just like other executables. The JDK package also includes a number of demo programs.

Debian's kernel is configured with Java support built in as a module (i.e., CONFIG_BINFMT_JAVA=m). Users who wish to build their own 10.1 can of course omit this if they choose. Once module support is available in the kernel, you need to make sure the module is loaded. You can do it at boot time by inserting the line `binfmt_java` in the file `/etc/modules`. Alternatively, you can install the module from the command line by executing the command `insmod DIRNAME/binfmt_java.o` where `DIRNAME` is the name of the directory where the modules that have been built for the version of the kernel now running are stored. On a system with the 2.0.0 version of the kernel, `DIRNAME` is likely to be `/lib/modules/2.0.0/fs/`. You can check that a module is loaded using the command `lsmod`.

Running a Java applet requires a browser with the capability to recognize and execute them. The Netscape browser that can be installed as a Debian package will run Java applets. (The Netscape source code is not publicly available. The Debian netscape package provides a wrapper which aids the installation and management of Netscape on a Debian system. This is actually a good example of the integration of commercial packages with the Debian system.)

A final note of mixed blessings: Sun's licensing policies on the JDK are becoming more restricted with time, so this package may not be available as part of the Debian distribution soon. It is possible that it may be available in the same way that Netscape is available for Debian Linux. Better news is that a number of ports are currently being developed with may provide attractive publicly-available alternatives.

5.7 What are all those directories (buzz/non-free/contrib/project) at the Debian FTP archives?

The software that's been packaged for Debian GNU/Linux is available in one of several directory trees on each Debian mirror site.

5.7.1 Top Level Directories

The Major Package Trees

- Debian-1.1, a.k.a. "buzz": This directory contains the packages which formally constitute the most recent release of the Debian Linux system. The contents of this directory do not change.
- Debian-1.1-updates, a.k.a. "buzz-fixes": Updates to the current release are in this directory.
- Debian-1.1.x, a.k.a. "stable" and "buzz-fixed": A copy of the the current release with the updates applied. The minor number is changed whenever new updates are added.
- "non-free": This directory contains packages whose distribution is restricted in a way which requires that distributors take careful account of the specified copyright requirements. For example, some packages have licenses which prohibit commercial distribution. Others can be redistributed but are in fact shareware and not freeware. The licenses of each of these packages must be studied, and possibly negotiated, before the packages are included in any redistribution (e.g., in a CD-ROM).
- "contrib": This directory contains packages which are *freely distributable*, but do not meet the policy requirements for distribution by the Debian project for some reason, e.g., the packages have some unusual restriction on modification by other users, or are only available in binary form. For

such packages, the project cannot given the user any means to assure that they are free of Trojan horses, and cannot port them to other architectures. Binary-only packages which are not freely redistributable are placed in the `non-free` directory.

Supplementary directories:

- "tools": DOS utilities for creating boot disks, partitioning your disk drive, compressing/decompressing files, and booting Linux.
- "upgrades": Files needed for upgrading a debian-0.93 system (based on `a.out` binary format files) to Debian-1.1 (based on `ELF` binary files). Included are instructions for upgrading (either "manually" using `dpkg`, or more automatically, using `dselect`), along with a copy of the particular version of the package management tool, `dpkg` that must be used in `a.out` systems to start the upgrade, and a list of files that will be needed in order to do the upgrade.
- "doc": Documentation, instructions on how to submit bug reports.
- "debian-lists": Archives of the Debian mailing lists.

Of particular interest to developers:

- "project/experimental": This directory contains packages and tools which are being developed specifically for the Debian project, and are still in the testing stage. Users are welcome to use and test these packages, but warned about their state of readiness.
- "rex", a.k.a. "unstable", a.k.a. "development": A snapshot of the current development system. Users are welcome to use and test these packages, but are warned about their state of readiness.
- "private/project/Incoming": Packages that have been uploaded by developers, and which are awaiting placement in the directory hierarchy by the distribution maintainer.
- "debian-bugs/": Archive of the bug reports.
- "indices": Various lists, including the `Packages-Master` file.

5.7.2 Sub-directories

Within each of the major directory trees (`Debian-1.1`, `non-free`, and `contrib`, but not `project/experimental`), which is too small to subdivide), the binary packages reside in sub-directories whose names indicate the chip architecture for which they were compiled:

- binary, which is normally a link to binary-i386.
- binary-all, for packages which are architecture-independent. These include, for example, Perl scripts.
- binary-i386, for packages which execute on 80x86 machines.
- binary-m68k, for packages which execute on machines based on one of the Motorola 680x0 processors. Currently this is done mainly for Atari and Amiga computers, and also for some VME based industry standard boards. There is no port of Linux to the old m68k based Macintosh's, since Apple will not supply the needed hardware information.
- binary-sparc, for packages which execute on Sun Sparcstations.
- binary-alpha, for packages which execute on DEC Alpha machines.

5.8 Is source code included with the system?

Source code is included for everything in the Debian system. Most of the license terms of programs in the system *require* that source code be distributed along with the programs, or that an offer to provide the source code accompany the programs.

Source code may or may not be available for packages in the "contrib" "non-free" directories, which are not formally part of the Debian system.

5.9 How can I check that I'm using a Debian system?

Check for the existence of the file `/var/lib/dpkg/status`. This contains a stanza reporting the status of each of the installed Debian packages on a system.

5.10 How can I tell what "version" of the Debian system I am using?

There is a file, `/etc/debian_version`, which contains a single one-line entry giving the version number of the release, as defined by the package `base`.

Users should be aware, however, that the Debian system consists of many parts, each of which can be updated (almost) independently. Each Debian "release" contains well defined and unchanging contents. Updates are separately available. For a one-line description of the installation status of package `foo`, use the command `dpkg -l foo`. (With no arguments, this command prints out versions of all installed packages.) For a more verbose description, use `dpkg --status foo`.

5.11 How does the Debian distribution support non-English languages?

- Debian GNU/Linux is distributed with keymaps for nearly two dozen keyboards, and with utilities (in the `kbd` package) to install, view, and modify the tables. The installation prompts the user to specify the keyboad he'll use.
- Support for German-languange manpages is provided through the `manpages-de` package, and support for Spanish-language manpages is provided through the `manpages-es` package. To access an NLS manpage, the user must set LC_MESSAGES to the appropriate string. In the case of the German-language manpages, LC_MESSAGES must be set to 'de_DE'. The `man` program will then search for german manpages under `/usr/man/de_DE`.

6 Basics of the Debian Package Management System

6.1 What is a Debian package?

Packages generally contain all of the files necessary to implement a set of related commands or features. There are two types of Debian packages:

- *Binary packages*, which contain executables, configuration files, man/info pages, copyright information, and other documentation. These packages are distributed in a Debian-specific 6.2; they are distinguished by having a '.deb' file extension.
- *Source packages*, which contain all of the files necessary to create the comparable binary package. These packages are distributed as gzip-compressed tar files, and therefore are distinguished by a '.tar.gz' file extension.

Installation of software by the package system uses "dependencies" which are carefully designed by the package maintainers. These dependencies are documented in the `control` file associated with each package. For example, the package containing the GNU C compiler (`gcc`) "depends" on the package `binutils` which includes the linker and assembler. If a user attempts to install `gcc` without having first installed `binutils`, Debian's package system will send an error message that it also needs `binutils`, and will install `gcc` only if the user agrees to install `binutils` first. (However, this facility can be overridden by the insistent user.) More details are given 6.9 on package dependencies.

Debian's packaging tools can be used to:

- manipulate and manage packages or parts of packages,
- aid the user in the break-up of packages that must be transmitted through a limited-size medium such as floppy disks,
- aid developers in the construction of package archives, and
- aid users in the installation of packages which reside on a remote FTP site.

6.2 What is the format of a Debian package?

A Debian "package", or a Debian archive file, contains the executable files, libraries, and documentation associated with a particular suite of program or set of related programs. Normally, a Debian archive file has a filename that ends in `.deb`.

Debian archive files can be parsed and manipulated by the utility `ar`. The precise contents of Debian archive files changed since Debian 0.93. The new contents are understood by versions of the primary package tool, `dpkg`, later than 0.93.76, and is described in the "deb"(5) man page. The old format is described in "deb-old"(5).

Using the command `ar -t foo_VVV-RRR.deb`, one sees that a Debian archive file contains these members:

- **debian-binary:** Contains one or more lines; currently it contains only one line giving the version number (2.0) of the Debian package format.
- **control.tar.gz:** A compressed (`gzip'd`) `tar` file which contains the 6.6 for this package. (Confusingly, one of these files, and the only one which is required, is itself named `control`.)
- **data.tar.gz:** A compressed (`gzip'd`) `tar` file which contains the executables, libraries, documentation, etc., associated with this package. In other words, this component is the filesystem data part of a Debian package.

Additional members may be added in the future. Detailed requirements for adding them are given in the `deb` manual page.

6.3 Why are Debian package names so long?

The Debian package names conform to the following convention: <foo>_<VersionNumber>-<DebianRevisionNumber>_<Arch>.deb

Note that `foo` is supposed to be the package name. This naming convention is rather new, and some packages have slightly different name formats. As a check, one can learn the package name associated with a particular Debian archive file (.deb file) in one of these ways:

- inspect the "Packages" file in the directory where it was stored in a Debian FTP archive site. This file contains a stanza describing each package; the first field in each stanza is the formal Package Name.
- use the command `dpkg --info foo_VVV-RRR.deb`. This sends a message to STDOUT which gives, among other things, the Package Name corresponding to the archive file being unpacked.

The `VVV` component is the version number specified by the upstream developer. There are no standards in place here, so the version number may have formats as different as "960428" and "2.7.2.l.3".

The `RRR` component is the Debian revision number, and is specified by the Debian developer (or an individual user if he chooses to build the package himself). This number corresponds to the revision level of the Debian package (which includes the Debian-specific Makefile, called `debian.rules`, as well as the Debian control file, usually called `debian.control`). Thus, a new revision level usually signifies changes in the Debian Makefile, the Debian control file, the installation or removal scripts, or in the configuration files used with the package.

The `Arch` component identifies the processor for which the package was built. This is commonly `i386`, which refers to chips in the 80x86 family of vintage 80386 or later. 5.7 include `m68k` for processors in the Motorola 680x0 family, etc.

6.4 What's a Debian control file?

Specifics regarding the contents of a Debian control file are provided in the manual page `deb-control (5)`. Briefly, a sample control file is shown below for the Debian package libc5_5.2.18-9.deb:

```
PACKAGE: libc5
SECTION: base
SOURCE: libc5
DESCRIPTION: The Linux C library version 5 (run-time libraries).
 Includes shared libraries needed to run programs built with libc 5.
MAINTAINER: David Engel <david@ods.com>
VERSION: 5.2.18-9
PRE-DEPENDS: ldso (>=1.7.14-2)
CONFLICTS: elf-libc, pthreads1
REPLACES: elf-libc
```

```
PROVIDES: elf-libc, pthreads1
ARCHITECTURE: i386
```

The first line gives the package name. This is the name by which the package can be manipulated by the package tools, and usually similar to but not necessarily the same as, the first component string in the Debian archive file name.

The second line gives the "section" where this Debian package is stored at the Debian FTP sites. This is the name of a subdirectory (within one of the main directories, see 5.7) where the package is stored.

The `Source` field identifies the source package from which the binary package was made. Normally, this is the same as the name of the package itself. It can be different, however, when one source package actually provides more than one binary package. For example, three different binary packages related to the Java Developer's Kit are produced from the single `jdk` source package: one for systems with Motif libraries (`jdk-shared`), one for systems without Motif (`jdk-static`), and one package common to both (`jdk-common`).

The Description field gives a brief summary of the package's features, and the Maintainer field names the Debian package developer and his email address. The version field gives both the upstream developer's version number and (in the last component) the revision level of the Debian package of this program.

The Pre-Depends field gives a list of packages that have to be available in order to make dpkg even try to install a package. This feature is for expert use only.

The Conflicts field tells the user (and the Debian package maintenance tools) what other packages cannot co-exist with the programs in this package. The Replaces field tells what packages will be replaced when this one is installed. The Provides field tells what packages will be installed by this package; this is a mechanism by which multiple packages can be distributed as a single package, which is in some cases an aid to the package maintenance system.

The final field (Architecture) specifies the chip for which this particular binary was compiled.

6.5 What's a Debian Conffile?

Conffiles are listings of configuration files, usually placed in `/etc`, that the package management system will not overwrite when a package is upgraded. This ensures that local values for the contents of these files will be preserved, and is a critical feature enabling the in-place upgrade of packages on a running system.

To determine exactly which files are preserved during an upgrade, users can inspect the contents of `/var/lib/dpkg/info/package_name.conffiles` For example, the `netbase.conffiles` package contains these entries:

```
/etc/init.d/netbase
/etc/gateways
/etc/protocols
/etc/services
/etc/hosts.allow
/etc/hosts.deny
/etc/rpc
```

6.6 What's a Debian preinst, postinst, prerm, and postrm script?

These files are executable scripts which are automatically run before or after a package is installed. Along with a file named `control`, all of these files are part of the "control" section of a Debian archive file.

The individual files are:

preinst

This script executes the configuration of a package once that package has been unpacked from its Debian archive (".deb") file. Many 'preinst' scripts also stop services for packages which are being upgraded until their installation or upgrade is completed (following the successful execution of the 'postinst' script).

postinst

This script typically completes any required configuration of the package `foo` once `foo` has been unpacked from its Debian archive (".deb") file. Often, 'postinst' scripts ask the user for input, and/or warn the user that if he accepts default values, he should remember to go back and re-configure that package as the situation warrants. Many 'postinst' scripts then execute any commands necessary to start or restart a service once a new package has been installed or upgraded. *It is a good idea to check the contents of the 'postinst' script for any configuration advice, when trying out a package for the first time.*

prerm

This script typically stops any daemons which are associated with a package. It is executed before the removal of files associated with the package.

postrm

This script typically modifies links or other files associated with `foo`. (See notes on 6.8.)

All of the control files can be found in `/var/lib/dpkg/info`. The files relevant to package `foo` begin with the name "foo" and have file extensions of "preinst", "postinst", etc., as appropriate. The file `foo.list` in that directory lists all of the files that were installed with the package `foo`.

6.7 What's a Required/Important/Standard/Optional/Extra package?

Each Debian package is assigned a *priority* by the distribution maintainers, as an aid to the package management system. The priorities are:

- **Required** packages are absolutely essential. Required packages are necessary for the functioning and repairability of the system. You must not remove these packages or your system may become totally broken and you may probably not even be able to use dpkg to put things back. Systems with only the Required packages are probably unuseable, but they do have enough functionality to allow the sysadmin to boot and install more software.

- **Important** packages should be found on any Unix-like system. Other packages which the system will not run well or be usable without will be here. This does *NOT* include Emacs or X11 or TeX or any other large applications. These packages only constitute the bare infrastructure.
- **Standard** packages are standard on any Linux system, including a reasonably small but not too limited character-mode system. This is what will install by default if users don't select anything else. It doesn't include many large applications, but it does include Emacs (this is more of a piece of infrastructure than an application) and a reasonable subset of TeX and LaTeX (if this turns out to be possible without X).
- **Optional** packages include all those that you might reasonably want to install if you didn't know what it was or don't have specialized requirements. This includes X11, a full TeX distribution, and lots of applications.
- **Extra** packages conflict with others with higher priorities, or are only likely to be useful if you already know what they are or have specialised requirements.

6.8 What's a Virtual Package?

A virtual package is a generic name that applies to any one of a group of packages, all of which provide similar basic functionality. For example, both the `tin` and `trn` programs are both news readers, and should therefore satisfy any dependency of a program that required a news reader on a system in order to work. They are therefore both said to provide the "virtual package" called `news reader`.

Similary, `smail` and `sendmail` both provide the functionality of a mail transport agent. They are therefore said to provide the virtual package, "mail transport agent". If either one is installed, then any program depending on the installation of a `mail transport agent` will be satisfied by the existence of this virtual package.

Debian provides a mechanism so that, if more than one package which provide the same virtual package is installed on a system, then system administrators can set one as the preferred package. The relevant command is `update-alternatives`, and is described further in the section on 11.6.

6.9 What is meant by saying that a file Depends/Recommends/Suggests/Conflicts/Replaces/Provides another package ?

The Debian package system has a range of package "dependencies" which are designed to indicate (in a single flag) the level at which Program A can operate independently of the existence of Program B on a given system:

- Package A *depends* on Package B if B absolutely must be installed in order to run A. In some cases, A depends not only on B, but on a version of B. In this case, the version dependency is usually a lower limit, in the sense that A depends on any version of B more recent than some specified version.
- Package A *recommends* Package B, if the package maintainer judges that most users would not want A without also having the functionality provided by B.

- Package A *suggests* Package B if B contains files that are related to (and usually enhance) the functionality of A.
- Package A *conflicts* with Package B when A will not operate if B is installed on the system. Most often, conflicts are cases where A contains files which are an improvement over those in B. "Conflicts" are often combined with "replaces".
- Package A *replaces* with Package B when files installed by B are removed and (in some cases) overwritten by files in A.
- Package A *provides* Package B when all of the files and functionality of B are incorporated into A. This mechanism provides a way for users with constrained disk space to get only that part of package A which they really need.

6.10 What is meant by Pre-Depends?

"Pre-Depends" is a special dependency.

In the case of most packages, `dpkg` will unpack its archive file (i.e., its `.deb` file independently of whether or not the files on which it depends exist on the system. Simplistically, unpacking means that `dpkg` will extract the files from the archive file that were meant to be installed on your filesystem, and put them in place. If those packages *depend* on the existence of some other packages on your system, `dpkg` will refuse to complete the installation by executing its "configure" action until the other packages are installed.

However, for some packages, `dpkg` will refuse even to unpack them until certain dependencies are resolved. Such packages are said to "Pre-depend" on the presence of some other packages. The Debian project provided this mechanism to support the safe upgrading of systems from `a.out` format to `ELF` format, where the *order* in which packages were unpacked was critical.

6.11 What is meant by unknown/install/remove/purge/hold in the package status?

These "want" flags tell what the user wanted to do with a package (as indicated either by the user's actions in the "Select" section of `dselect`, or by the user's direct invocations of `dpkg`). Their meanings are:

- unknown - the user hasn't said whether they want the package
- install - the user wants the package installed or upgraded
- remove - the user wants the package removed (but doesn't want to remove any existing configuration)
- purge - the user wants the package to be removed completely
- hold - the user wants this package not to be processed, i.e., he wants to keep the current version with the current status whatever that is.

7 The Debian Package Management Tools

7.1 What program(s) does Debian provide for managing its packages?

7.1.1 `dselect`

This program is a menu-driven interface to the Debian package management system. It is particularly useful for first-time installations and large-scale upgrades. `dselect` can

- guide the user as he/she chooses among packages to install or remove, ensuring that no packages are installed that conflict with one another, and that all packages required to make each package work properly are installed;
- warn the user about inconsistencies or incompatibilities in their selections;
- determine the order in which the packages must be installed;
- automatically perform the installation or removal; and
- guide the user through whatever configuration process are required for each package.

`dselect` begins by presenting the user with a menu of 7 items, each of which is a specific action. The user can select one of the actions by using the arrow keys to move the highlighter bar, then pressing the ENTER key to select the highlighted (highlit?) action.

What the user sees next depends on the action he selected. If he selects any option but `Access` or `Select`, then `dselect` will simply proceed to execute the specified action: e.g., if the user selected the action `Remove`, then dselect would proceed to remove all of the files selected for removal when the user last chose the `Select` action.

Both the `Access` menu item and the `Select` menu item lead to additional menus. In both cases, the menus are presented as split screens; the top screen gives a scrollable list of choices, while the bottom screen gives a brief explanation ("info") for each choice.

Extensive on-line help is available: Use the '?' key to get to a help screen, then use the '.' key to see each of the several pages of help one screen at a time.

Some users find it easier to navigate `dselect` when it is colorized. To see color screens in `dselect`, be sure that you've executed: `export TERM=linux` before invoking `dselect`.

The order in which the actions are presented in the first `dselect` menu represents the order in which a user would normally choose `dselect` to install packages. However, a user can pick any of the main menu choices as often as needed (including not at all, depending on what one wants to do).

- Begin by choosing an "Access Method". This is the method by which the user plans on accessing Debian packages; e.g., some users have Debian packages available on CD-ROM, while others plan to fetch them using anonymous ftp. (To use the latter approach, a user will have had to fetch and install the package `dpkg-ftp` before running `dselect`.) The selected "Access Method" is stored in the file `/var/lib/dpkg/cmethopt` after `dselect` exits, so if it doesn't change, then this option need not be invoked again.

- Then "Update" the list of available packages. To do this, `dselect` reads the file "Packages.gz" which should be included in the top level of the directory where the Debian packages to be installed are stored. (But if it's not there, `dselect` will offer to make it for you.)
- `Select` specific packages for installation on his system. After choosing this menu item, the user is first presented with a full screen of help; he can exit it (and any help screen) by pressing the SPACEBAR. Better (for first time users) is to read *all* of the help screen, but repeatedly pressing the '.' key to fetch one page of help after another. Once the user exits the Help screen, he sees the split-screen menu for choosing packages to install (or remove). The top screen is a relatively narrow window into the list of Debian's nearly 500 packages; the bottom screen is a window into "info" about the package or group of packages which are highlighted in the top.

  First-time users are often confused by these aspects of the `Select` screen:

  – "Deselecting" a packages: One can specify which packages should be removed (previously called "deselecting" an item) by highlighting a package name or the label for a group of packages (e.g., "All") and then pressing either:

    ∗ the '-' key. This removes most of the files associated with the package, but *NOT* the configuration files, i.e., those listed in `/var/lib/dpkg/info/PACKAGENAME.conffiles`

    ∗ the '_' key. This removes *every* file that's part of the package.

    Note that if you "de-select" "All Packages" your system will be reduced to the initial installed base packages. This is probably what you wanted.

  – Putting a package "on hold" (by pressing '='): This effectively tells `dselect` not to upgrade a package even if the version currently installed on your system is not as recent as the version that's available in the Debian repository you are using (this was specified when you set the `Access Method`). (The version that's available in the repository is given in the file `Packages.gz` that's read when the "Update" menu choice is activated. Putting a package on "unhold" (by pressing ':'): This is the default, and means that the packages will be upgraded if a newer version is available.

  – The order in which the packages are presented: The default order is to present packages by Priority; within each priority, packages are presented in order of the directory (a.k.a. section) of the archive in which they are stored. Given this sort order, some packages in section A (say) may be presented first, followed by some packages in section B, followed by more packages (of lower priority) in section A. Users can select a different presentation order by using the 'o' key to cycle between various options for sorting the packages.

  – Meanings of the labels at the top of the screen: The labels at the top can be expanded by using the 'v' (verbose) key. This action pushes much of the text that formerly fit onto the display off to the right. To see it, press the right arrow; to scroll back to the left, press the left arrow.

  – What to do with a package conflict/dependency screen: If a user selects (for installation or removal) a package, e.g. `foo.deb` that depend on or recommend another package, e.g., `blurf.deb`, then `dselect` will place the user in a sub-screen of the main selection screen. This process begins by presenting the user with a full-screen Help file, which can be escaped by pressing SPACEBAR. Thereafter, the user can choose among the related packages, accepting the suggested actions (to install or not), or rejecting them. To do the latter, press SHIFT-D; to return to the former, press

SHIFT-U. In any case, the user can save his selections and return to the main selection screen by pressing SHIFT-Q.

- Users returning to the main menu can then select either the "Install" menu item, followed by the "Configure" menu item to unpack and then be guided through the configuration files for the selected packages. Alternatively, users wishing to remove files can choose the "Remove" menu item. At any point, users can choose "Quit" to exit dselect; `dselect` stores users' selections in the file `/var/lib/dpkg/status`.

7.1.2 `dpkg:`

This is the main package management program, `dpkg` can be invoked with many options. Some common uses are:

- Find out all the options: `dpkg --help`.
- Print out the control file (and other information) for a specified package: `dpkg --info foo_VVV-RRR.deb`
- Install a package (including unpacking and configuring) onto the file system of the hard disk: `dpkg --install foo_VVV-RRR.deb`.
- Unpack (but do not configure) a Debian archive into the file system of the hard disk: `dpkg --unpack foo_VVV-RRR.deb`. Note that this operation does *not* necessarily leave the package in a usable state; some files may need further customization to run properly. This command removes any already-installed version of the program and runs the 6.6 script associated with the package.
- Configure a package that's already been unpacked: `dpkg --configure foo`. Among other things, this action runs the 6.6 script associated with the package. It also updates the files listed in the `conffiles` for this package. Notice that the 'configure' operation takes as its argument a package name (e.g., foo), *not* the name of a Debian archive file (e.g., foo_VVV-RRR.deb).
- Extract a single file named "blurf" (or a group of files named "blurf*" from a Debian archive: `dpkg --fsys-tarfile foo_VVV-RRR.deb | tar -xf - blurf*`
- Remove a package (but not its configuration files): `dpkg --remove foo`.
- Remove a package (including its configuration files): `dpkg --purge foo`.
- List the installation status of packages containing the string 'foo*': `dpkg -l foo*`.

7.1.3 `dpkg-deb`

This tool manipulates Debian archive (`.deb`)files. Some command uses are:

- Find out all the options: `dpkg-deb --help`.
- Determine what files are contained in a Debian archive file: `dpkg-deb -c foo_VVV-RRR.deb`)

- Extract the files contained in a named Debian archive into a user specified directory: `dpkg-deb -x foo_VVV-RRR.deb tmp` extracts each of the files in `foo_VVV-RRR.deb` into the directory `./tmp`. This is convenient for examining the contents of a package in a localized directory, without installing the package into the root file system.

More information is given in the man page `dpkg-deb`(1).

7.1.4 `dpkg-split`

This Perl script splits large package into smaller files (e.g., for writing onto a set of floppy disks), and can also be used to merge a set of split files back into a single file. It can only be used on a Debian system, since it calls the program `dpkg-deb` to parse the debian package file into its component records. For example, to split a big .deb file into N parts,

- Execute the command `dpkg-split -s foo.deb`. This will produce N files each of approximately 460 KBytes long in the current directory.
- Copy those N files to floppy disks.
- Copy the contents of the floppy disks onto the hard disk of your choice on the other machine.
- Join those part-files together using `dpkg-split -j "foo*"`.

7.2 Debian claims to be able to update a running program; how is this accomplished?

Debian GNU/Linux provides a program called the `start-stop-daemon` which is used by installation scripts to start daemons at boot time or to stop daemons when the kernel runlevel is changed (e.g., from multi-user to single-user or to halt). The `start-stop-daemon` command is also used when a new package containing a daemon is installed, to stop running daemons, and restart them as necessary, e.g., when a package is being installed with an updated configuration script.

7.3 How can I tell what packages are already installed on a Debian system?

To learn the status of all the packages installed on a Debian system, execute the command: `dpkg -l`. This prints out a one-line summary for each package, giving a 2-letter status symbol, the package name, the version which is *installed*, and a very brief description.

To learn the status of packages whose names match the string any pattern beginning with 'foo' by executing the command: `dpkg -l "foo*"`

To get a more verbose report for a particular package, execute the command: `dpkg --status foo`.

7.4 How can I find out what package produced a particular file?

To identify the package that produced the file named `foo` execute either:

- `dpkg -S filename`. This searches through the lists of installed files. This is (currently) equivalent to searching all of the files having the file extension of `.list` in the directory `/var/lib/dpkg/info/`.
- `grep foo Contents`, or `zgrep foo Contents.gz`. This searches for files which contain the substring `foo` in their full path names. The files `Contents` and `Contents.gz` reside in the major package directories (Debian-1.1, non-free, contrib, etc.) at a Debian FTP site. A `Contents` file refers only to the packages in the subdirectory tree where it resides. Therefore, a user might have to search more than one `Contents` files to find the package containing the file `foo`.

8 Getting and Installing Debian GNU/Linux

8.1 Where/how can I get the Debian installation disks?

You can get the installation disks by downloading the appropriate files from any Debian FTP site: `<ftp://ftp.debian.org/debian/>` and its *mirrors* `<http://www.debian.org/ftplist.html>`. From the directory `Debian-1.1/disks-i386/current`, you'll need:

The Boot Disk.

If your system can boot from a 1.44 MByte floppy disk drive, download the file called `boot1440.bin`. On the other hand, if your system can boot only from a 1.22 MByte floppy disk drive, download the file called `boot1220.bin`.

The Root Disk.

Download the file called `root.bin`. It will fit on either a 1.44 MByte or a 1.2 MByte floppy disk.

The Base Disks.

These disks contain the software that's necessary to have a bare-bones system. Depending on your configuration, you may need 1, 3, or 4 of these disks. (See notes below.)

You may of course get these disks from a friend. You can make as many copies as you want. Debian is freely redistributable, provided you follow a few simple rules. For more information, see *The GNU General Public License* `<http://www.cygnus.com/library/misc/license.html>`.

If you cannot get copies of the installation disks from one of the Debian FTP sites or from a friend, then make a request to one of the Debian mailins lists. See 12.2. The likelihood is very high that someone will be willing to help.

8.2 OK, I've got the disks; how do I install the system?

Complete, detailed instructions are given in the same FTP sites as the installation disks. Fetch the *plain ASCII file* `<ftp://ftp.debian.org/Debian-1.1/disks-i386/current/install.txt>`, the *Postscript file* `<ftp://ftp.debian.org/Debian-1.1/disks-i386/current/install.ps>`, or view them directly *on the WWW* `<ftp://www.debian.org/Debian-1.1/disks-i386/current/install.html>`.

8.3 How (and on what media) can I get the Debian packages?

Once the base system has been installed, you can complete the installation of your Debian GNU/Linux system by accessing packages on any (or several) of these media:

- a 8.4 containing a Debian distribution.
- a remote 8.5 server.
- a partition of your 8.7.
- a collection of 8.6.
- a remote system accessible by 8.8.

In each case, you can use either

- 7.1.1, Debian's user-interface program to its package system. (This will help you mount the device containing the Debian package files (if necessary), and then guide you through the selection of packages and their installation on your system.) Or,
- use 7.1.2 directly to control the installation of individual packages. Debian gurus quizzically refer to this approach as installing packages "by hand".

8.4 How do I get and install the Debian distribution from CD-ROM?

Linux supports the ISO 9660 (CD-ROM) file system with Rock Ridge extensions (formerly known as "High Sierra"). Several vendors provide the Debian distribution in this format; these vendors have directly supported the Debian project:

- I-Connect Co.: I-Connect provides the master FTP system used by Debian developers, and its network connection and system administration. They provide our 100 developers with free accounts on a system that is exclusively for our use, and they run our DNS domain. They also maintain several Debian packages. Use the following contact info to order their CD-ROM:
  - WWW:*http://www.i-connect.net/i-connect/services/cdrom.html* `<http://www.i-connect.net/i-connect/services/cdrom.html>`
  - Email (sales): sales@i-connect.net
  - Email (help): help@i-connect.net

  - Phone from inside the US: 1-503-677-2900
  - Phone from outside the US: +1-503-677-2900
  - Postal address:

    ```
    iConnect Corp.
    14355 SW Allen Blvd., Suite 140
    Beaverton, OR 97008
    ```

- Flexible Software: This company maintains 8 Debian packages, and puts a lot of manpower into supporting users on the debian-user mailing list and refining the system design with our developers. They have carefully documented the development of one of the two methods used to upgrade from Debian 0.93 to Debian 1.1.
  - Email (sales): dwarf@polaris.net
  - Postal address:

    ```
    Flexible Software
    1000 McCrackin Road
    Tallahassee, FL 32308
    ```

To install packages provided on a CD-ROM, you must have CD-ROM support built into the kernel you have. This may require a special kernel (see the kernels in the directory Debian-1.1/disks-i386/special/), a custom modification of the file `/etc/conf.modules`, or (least likely nowadays) a special argument on the boot command line.

To mount a CD-ROM under the mount point `/cdrom` (a directory which should have been created during installation), use these commands:

- `mount -t iso9660 /dev/hdd /cdrom` if the CD-ROM is an IDE drive on the secondary IDE controller, or
- `mount -t iso9660 /dev/scd0 /cdrom` if the CD-ROM is a SCSI drive, or
- `mount -t iso9660 /dev/xxxxx /cdrom` where xxxxx refers to CD-ROm's which are sbpcd, cdu31a, or some other proprietary (non-IDE) drive.

8.5 How do I get and install the Debian distribution from an NFS-mounted drive?

Debian packages on remote hosts can be installed via the Network File System using a SLIP, PPP, PLIP or Ethernet connection. SLIP and PPP require that the the netbase and netstd packages be installed first. These packages are not part of the Debian base system, and must be obtained by one of the other methods described in this section.

No special kernel configuration (other than the network drivers for SLIP, PPP, PLIP, or one of the Ethernet cards) is necessary to access a remote filesystem using NFS. As with all network communication, the appropriate `ifconfig` and `route` commands will have to be issued before accessing a remote disk using NFS.

To mount a network-accessible file system using NFS (say, /var/spool/debian) under the local directory /mnt, issue the command

- mount -t nfs host@Fully-qualified.domain.name:/var/spool/debian -o ro,nosuid,soft /mnt

To ensure that the filesystem is always accessible, the appropriate string could be added to `/etc/fstab`:

- rulway.leidenuniv.nl:/var/spool/ftp.pub.linux.debian /mnt/debian nfs ro,nosuid,soft

The preceding example shows the appropriate entry for the NFS site at Leiden University, the Netherlands, which is the only FTP site known to export Debian via NFS officially at this time.

8.6 How can I get/install the Debian distribution from a set of floppy disks?

Copy the Debian packages onto formatted floppy disks. Either a DOS, the native Linux "ext2", or the "minix" format will do; one just has to use a mount command appropriate to the floppy that's being used.

Using floppy disks has these complications:

- Short MS-DOS filenames: If you are trying to place Debian package files onto MS-DOS formatted disks, you will find that their names are generally too long, and do not conform to the MS-DOS 8.3 filename limitation. To overcome this, Debian developers make all of their packages available by 8.3 filenames in separate "msdos" subdirectories (Debian-1.1/msdos-i386, non-free/msdos-i386, and contrib/msdos). The files in these subdirectories are merely symbolic links to the Debian archive files; they only differ from the files in the binary-i386, etc. directories by having shorter file names.
- Large file sizes: Some packages are larger than 1.44 MBytes, and won't fit onto a single floppy disk. To solve this problem, use the 7.1.4 tool, available in the `tools` directory at `<ftp://ftp.debian.org/debian/>` and its *mirrors* `<http://www.debian.org/ftplist.html>`.

You must have support in the kernel for floppy disks in order to read and write to floppy disk; most kernels come with floppy drive support included in them.

To mount a floppy disk under the mount point /floppy (a directory which should have been created during installation), use:

- `mount -t msdos /dev/fd0 /floppy` if the floppy disk is in drive A: and has an MS-DOS filesystem,
- `mount -t msdos /dev/fd1 /floppy` if the floppy disk is in drive B: and has an MS-DOS filesystem,
- `mount -t ext2 /dev/fd0 /floppy` if the floppy disk is in drive A: and has an ext2 (i.e., a normal Linux) filesystem.

8.7 How can I get/install the Debian distribution from my own hard disk?

The Debian packages can be installed from a hard disk on your system, either IDE, EIDE, SCSI (including `Jaz` drives), or a drive connected through a parallel port.

Kernel support must of course be available for the type of drive you have. The appropriate drivers are:

- `CONFIG_BLK_DEV_IDE` for IDE or EIDE drives; most kernels include this driver.
- `CONFIG_SCSI` for SCSI drives (including `Jaz` drives), along with one of the (many) drivers for your particular SCSI interface.
- `CONFIG_SCSI_PPA` for the parallel port version of `ZIP` drives.

To mount partition X (X =1,2,3, or 4) for a given drive under the directory `/mnt` (which should have been created during installation), use:

- `mount -t msdos /dev/hdaX /mnt` if the disk is drive C: and has an MS-DOS filesystem,
- `mount -t ext2 /dev/sdaX /mnt` if the disk is a SCSI drive and has the lowest SCSI ID of any SCSI device on your system, and has an ext2 filesystem,
- `mount -t ext2 /dev/sdbX /mnt` if the disk is a SCSI drive and has the second lowest SCSI ID of any SCSI device on your system, and has an ext2 filesystem,
- `mount -t ext2 /dev/fd0 /floppy` if the floppy disk is in drive A: and has an ext2 (i.e., a normal Linux) filesystem.
- `mount -t msdos /dev/ppa0 /mnt` if the disk is a Zip drive with an MS-DOS filesystem.
- `mount -t ext2 /dev/ppa0 /mnt` if the disk has an ext2 filesystem.

Note that files on a disk partition having an MS-DOS filesystem must conform to the DOS filename limitation of 8 characters plus a 3 character extension. Files in the `Debian-1.1/msdos-i386`, `non-free/msdos-i386`, and `contrib/msdos` directories conform to this limitation.

8.8 How can I get and install the Debian distribution directly from a remote ftp site?

Install the Debian tool 9.2. This package is currently installed in the directory `project/experimental` at `<ftp://ftp.debian.org/debian/>` and its *mirrors* `<http://www.debian.org/ftplist.html>`. To do install a package, see notes on 7.1.2.

Then invoke the program `dselect`, which will call `dpkg-ftp` for you, guide you through the selection of packages, then install the packages, without every downloading the packages themselves to your machine. This method is designed to save the user both disk space and time. Note that no special kernel configuration is needed to access and install Debian packages by this method.

To use this service of `dselect`, you will need to know:

- the fully qualified domain name of the anonymous ftp site you plan to use. For example, you might use `ftp.nluug.nl.gov` if you live in or near the Netherlands.

- the directory which contains the files you want to install, or the subdirectories which contain files you want to install. This directory must contain a file called "Packages" (or its compressed equivalent, "Packages.gz"). For example, if you want to install files from the most uptodate revisions of the current Debian distribution, then (for `ftp.nluug.nl`) you would specify the directory `pub/os/Linux/distr/Debian/Debian-1.1.1/`.

8.9 How can I get and install the Debian distribution from a tape??

At present, installing packages directly from tape is not supported. One can however, use `tar`, `cpio`, or `afio` to copy Debian archive files onto a tape, then copy them onto your local disk for installation. In the same vein, floppy disks containing "tar" files would have to be copied onto a local disk before they could be managed with the Debian package tools.

8.10 What's the latest version of Debian?

Currently there are two versions of the Debian distribution:

- "1.1": this is stable software, and will not change.
- the "development" version. This is updated continuously. You can retrieve packages from the "development" archive on our FTP sites and use them to upgrade your system at any time.

Approximately three months from today, the "development" software will have been stabilized and made into Debian 1.2. Further releases will follow at three-month intervals.

9 Keeping Your Debian System Up To Date

9.1 How can I upgrade my Debian 0.93 (or earlier) distribution, based on a.out format binaries, to 1.1, based on ELF binaries?

The directory /debian/upgrades contains files needed by users upgrading from Debian 0.93R6 to Debian 1.1. There are two ways to upgrade:

- the manual way, installing the packages by hand. Instructions are in `upgrade_manual.doc`. You will also need the files `UpGrade`, `base_list`, and `dpkg-*.i386.deb`.
- using dselect to upgrade all the packages *en masse*. Instructions are in `upgrade_dselect.doc`. You will also need `dpkg-*.i386.deb`.

Note that the version of `dpkg` in this directory has the a.out binary format. The versions of `dpkg` in the unstable and stable trees have the ELF format.

9.2 How can I keep my Debian system current?

One could simply execute an anonymous ftp call to a Debian archive, then peruse the directories until he finds the desired file, and then fetch it, and finally install it using `dpkg`. Note that `dpkg` will install upgrade files in place, even on a running system. Sometimes, a revised package will require the installation of a newly revised version of another package, in which case the installation will fail until/unless the other package is installed.

Many people find this approach much too time-consuming, since the Debian distribution evolves so quickly—typically, a dozen or more new packages are uploaded every week. This number is larger just before a new major release. To deal with this avalanche, many people prefer to use an automated programs. Three different packages are available for this purpose:

- `mirror`. This perl script, and its (optional) manager program called `mirror-master`, can be used to fetch user-specified parts of a directory tree from a specified host *via* anonymous ftp. `mirror` is particularly useful for downloading large volumes of software. After the first time files have been downloaded from a site, a file called ".mirrorinfo" is stored on the local host. Changes to the remote filesystem are tracked automatically by `mirror`, which compares this file to a similar file on the remote system. The mirror program is generally useful for updating local copies of remote directory trees. The files fetched need not be Debian files. (Since mirror is a Perl script, it can also run on non-Unix systems.) Though the mirror programs provides mechanisms for excluding files whose names match user-specified strings, this program is most useful when the objective is to download whole directory trees, rather than selected packages.

  The mirror program has been packaged for the Debian distribution, and is found in the directory `Debian-1.1/binary-all/net` in the Debian FTP archive.

- `dftp`. This C shell script can be used to fetch user-specified Debian packages from a specified host. After the first time files have been downloaded from a site, a file called "Packages.gz" is stored on the local host. Changes to the remote filesystem are tracked automatically by `dftp`, which compares this file to a similar file on the remote system. The dftp program can be used to call 7.1.1, thereby providing an integrated way to fetch and update the Debian packages on one's system.

  The dftp program is available as a script in the directory `contrib/tools` in the Debian FTP archive.

- `dpkg-ftp`. This is a Perl script that allows one to install Debian packages that are stored on a remote Debian FTP archive. For users with limited disk space and/or limited bandwidth access to the Internet, this approach is particularly beneficial. Alternatively, the `dpkg-ftp` program can be invoked from within `dselect`, thereby allowing a user the ability to download files and install them directly in one step. To do this, bring up the `dselect` program, choose option "0" ("Choose the access method to use"), highlight the option "ftp" then specify the remote hostname and directory. `dpkg-ftp` will then automatically download the files that are selected (either in this session of `dselect` or earlier ones).

  Note that, unlike the `mirror` program, `dpkg-ftp` does not grab everything at a mirror site. Rather, it downloads only those files which you have selected (when first starting up `dpkg-ftp`, and which need to be updated.

  The `dpkg-ftp` program is available both as a gzip-compressed tar archive and a Debian package in the directory project/experimental in the Debian FTP archive. `dpkg-ftp` is now part of the `base` Debian distribution; it is included in the first "base" disk used in the installation.

All of the above programs can be set up (using the `cron` facility) to be executed automatically at specific times.

9.3 Must I go into single user mode in order to upgrade a package?

No. Packages can be upgraded in place, even in running systems. The Debian distribution has a `start-stop-daemon` that is invoked to stop, then restart running process if necessary duing a package upgrade.

9.4 Do I have to keep all those .deb archive files on my disk?

No. If you've downloaded the files to your disk (which isn't absolutely necessary (see 9.2 or 9.2), then after you've installed the packages, you can simply `rm` them from your system.

9.5 How can I keep a log of the packages I added to the system?

`dpkg` keeps a record of the packages that have been unpacked, configured, removed, and/or purged, but does not (currently) keep a log of terminal activity that occured while a package was being so manipulated. Some users overcome this simply by using `tee`:

```
dpkg -iGROEB stable/binary non-free/binary contrib/binary | \
   tee -a > /root/dpkg.log
```

10 Debian and the kernel

10.1 What tools does Debian provide to build custom kernels?

Users who wish to (or must) build a custom kernel are encouraged to download the package `kernel-package_VVV_all.deb`, which contains the script to build the kernel package, and provides the capability to create a Debian kernel-image package just by running the command `make-kpkg kernel_image` in the top-level kernel source directory. Help is available by executing the command `make-kpkg --help`, and through the section 8 manual page for `make-kpkg`.

Users must separately download the source code for the most recent kernel (or the kernel of their choice) from their favorite Linux archive site.

To build a custom kernel, users must have these packages installed: `gcc`, `libc5-dev`, `bin86`, `binutils`, `gawk`, `make`, `gzip`, and `grep`.

Executing the command `dpkg --install kernel-package_VVV_all.deb` sets up the directory `/usr/src/linux`, and sets up the link `/usr/src/linux` to point to the directory containing the kernel sources appropriate to the kernel version contained in the package. In the case of Debian-1.1, this kernel version is 2.0.0.

Detailed instructions for using the package are given in the file `/usr/doc/kernel-package/README`. Briefly, one should:

- Unpack the kernel sources, and `cd` to the newly created directory.
- Modify the kernel configuration using one of these commands:
  - `make config` (for a tty one-line-at-a-time-interface).
  - `make menuconfig` (for an ncurses-based menu driven interface).
  - `make xmenuconfig` (for an X-windows interface).

  Any of the above steps generates a new `.config` in the top-level kernel source directory.
- Execute the command: `make-kpkg -r Custom.N kernel_image`, where N is a revision number assinged by the user. The new Debian archive thus formed would have revision Custom.1, e.g., `kernel-image-2.0.7-Custom.1_i386.deb` for the Linux kernel 2.0.7.
- Install the package created.
  - `dpkg -i /usr/src/kernel-image_VVV-Custom.N_i386.deb` to install the kernel itself. The installation script will:
    * run the boot loader, LILO (if it is installed),
    * install the custom kernel in /boot/vmlinuz_VVV-Custom.N, and set up appropriate symbolic links to the most recent kernel version.
    * prompt the user to make a boot floppy. This boot floppy will contain the raw kernel only. See additional notes for making a 10.2.
  - To employ a secondary boot loaders (e.g., `loadlin`), copy this image to other locations (e.g., an `MS-DOS` partition).

10.2 How can I make a custom boot floppy?

This task is greatly aided by the Debian package `boot-floppies_VVV-RRR.deb`, normally found in the binary/devel section of the Debian FTP archive. Shell scripts in this package produce boot floppies in the `SYSLINUX` format. These are `MS-DOS` formatted floppies whose master boot records have been altered so that they boot Linux directly (or whatever other operating system has been defined in the syslinux.cfg file on the floppy). Other shell scripts in this package produce emergency root disks and can even reproduce the base disks.

When the boot-floppies package is installed, it produces files which reside in `/usr/src/boot-floppies-VVV`, including the three scripts, called `bootdisk.sh`, `rootdisk.sh`, and `basedisk.sh`. Users of these scripts are encouraged to read them and customize as appropriate. In particular, one should include the kernel of your choice when invoking the `bootdisk.sh` script. *Be aware that this kernel cannot be used to load the root disk into a RAMDISK unless it was compiled with* `CONFIG_RAMDISK=y`*, and* `CONFIG_INITRD=y`.

The command

```
./bootdisk.sh /usr/src/kernel-image-2.0.0-2.i386.deb /dev/fd0 1440
```

writes the kernel-image-2.0.0 package to the 1.44 MByte disk in drive A:.

Be sure to include the editor and whatever other tools you prefer when building your custom root disk with `rootdisk.sh`.

10.3 What special provisions does Debian provide to deal with modules?

Debian's `modconf` package provides a shell script (`/usr/sbin/modconf`) which can be used to customize the configuration of modules. This script presents a menu-based interface, prompting the user for particulars on the loadable device drivers in his system. The responses are used to customize the file `/etc/conf.modules` (which lists aliases, and other arguments that must be used in conjuction with various modules), and `/etc/modules` (which lists the modules that must be loaded at boot time).

Like the (new) Configure.help files that are now available to support the construction of custom kernels, the modconf package comes with a series of help files (in `/usr/lib/modules_help`) which provide detailed information on appropriate arguments for each of the modules.

11 Customizing your installation of Debian GNU/Linux

11.1 Every distribution seems to have a different boot-up method. Tell me about Debian's.

Like all Unices, Debian Linux boots up by executing the program `init`. The configuration file for `init` (which is `/etc/inittab`) specifies that the first script to be executed should be `/etc/init.d/boot`. This script checks and mounts file systems, loads modules, starts the network services (by calling the script `/etc/init.d/network`), sets the clock, performs other initialization, and then runs all of the scripts (except those with a '.' in the filename) in `/etc/rc.boot`. The latter script sets the default keyboard, recovers lost editor files, and configures the serial ports.

After completing the boot process, `init` executes all start scripts in a directory specified by the default runlevel (this runlevel is given by the entry for `id` in `/etc/inittab`). Like most System V compatible Unices, Linux has 6 runlevels: 0 (halt the system), 1 (single-user mode), 2 through 5 (various multi-user modes), and 6 (reboot the system). Debian Linux systems come with id=2, which indicates that the default runlevel will be '2' when the multi-user state is entered, and the scripts in `/etc/rc2.d` will be run.

In fact, the scripts in any of the directories, `/etc/rcN.d` are just symbolic links back to scripts in `/etc/init.d`. However, the *names* of the files in each of the `/etc/rcN.d` directories are selected to indicate the *way* the scripts in `/etc/init.d` will be run. Specifically, before entering any runlevel, all the scripts beginning with 'K' are run; these scripts kill services. Then all the scripts beginning with 'S' are run; these scripts start services. The two-digit number following the 'K' or 'S' indicates the order in which the script is run. Lower numbered scripts are executed first.

This approach works because the scripts in `/etc/init.d` all take an argument which can be either 'start', 'stop', or 'restart', and will then do the task indicated by the argument. For example, with the argument 'restart' the command `/etc/init.d/sendmail restart` restarts the sendmail daemon. These scripts can be used even after a system has been booted to control various processes.

11.2 It looks as if Debian doesn't use `rc.local` to customize the boot process; what facilities are provided?

Suppose a system needs to execute script `foo` on start-up, or on entry to a particular (System V) runlevel. Then the system administrator should:

- Enter the script `foo` into the directory `/etc/init.d/`.
- Run the Debian command `update-rc.d` with appropriate arguments, to set up links between the (command-line-specified) directories rc?.d and `/etc/init.d/foo`. Here, '?' is a number from 0 through 6 and corresponds to each of the System V runlevels.
- Reboot the system.

The command `update-rc.d` will set up links between files in the directories rc?.d and the script in `/etc/init.d/`. Each link will begin with a 'S' or a 'K', followed by a number, followed by the name of the script. Scripts beginning with 'S' in `/etc/rcN.d/` are executed when runlevel `N` is entered. Scripts beginning with a 'K' are executed when leaving runlevel `N`.

One might, for example, cause the script `foo` to execute at bootup, by putting it in `/etc/init.d/` and installing the links with `update-rc.d foo defaults 19`. The argument 'defaults' refers to the default runlevels, which are 2 through 5. The argument '19' ensures that `foo` is called before any scripts containing numbers 20 or larger. the appropriate time.

11.3 How does the package management system deal with packages that contain configuration files for other packages?

Some users wish to create, for example, a new server by installing a group of Debian packages and a locally generated package consisting of configuration files. This is not generally a good idea, because `dpkg` won't know about those configuration files if they're in a different package, and may write conflicting configurations when one of the initial "group" of packages is upgraded.

Instead, create a local package that modifies the config files of the "group" of Debian packages of interest. Then `dpkg` and the rest of the package management system will see that the files have been modified by the local "sysadmin" and will not try to overwrite them when those packages are upgraded.

11.4 How do I override a file installed by a package so that a different version can be used instead?

Suppose a sysadmin or local user wishes to use a program "login-local" rather than the program "login" provided by the Debian `login` package. Do not:

- Overwrite `/bin/login` with `bin/login-local`.

The package management system won't know about this change, and will simply overwrite `/bin/login-local` whenever `login` (or any package that provides `login` is installed or updated.

Rather, do

- Execute `dpkg-override --divert /bin/login.debian /bin/login` in order to cause all future installations of the Debian `login` packages to write the file `/bin/login` to `/bin/login.debin/` instead.
- Then execute `cp login-local /bin/login` to move your own locally-built program into place.

More notes on this technique are given in the file `/usr/doc/debian/package-developer/diversions.txt.gz` which is installed as part of the `doc-debian` package.

11.5 How can I have my locally-built package included in the list of available packages that the package management system knows about?

You can do this in either of two ways:

- Use `dselect`, and select the option "Update list of available packages". Then execute the command `dpkg --avail foo.deb`, where foo.deb is the name of your locally-built Debian archive file.
- Execute the command `dpkg-scanpackages BIN_DIR OVERRIDE_FILE [PATHPREFIX] > Packages.new` where:
  - BIN-DIR is a directory where Debian archive files (which usually have an extension of ".deb") are stored.
  - OVERRIDE_FILE is a file that's edited by the distribution maintainers and is usually stored on a Debian FTP archive at `indices/override.main.gz` for the Debian packages in the "main" distribution.
  - PATHPREFIX is an optional string that can be prepended to the Packages.new file being produced.

  Once you've built the file `Packages.new`, tell the package management system about it by using the command `dpkg --update-avail Packages.new`.

11.6 Some users like mawk, others like gawk; some like vim, others like elvis; some like trn, others like tin; how does Debian support diversity?

There are several cases where two packages provide two different versions of a program, both of which provide the same core functionality. Users might prefer one over another out of habit, or because the user interface of one package is somehow more pleasing than the interface of another. Other users on the same system might make a different choice.

Debian uses a "virtual" package system to allow system administrators to choose (or let users choose) their favorite tools when there are two or more that provide the same basic functionality, yet satisfy package dependency requirements without specifying a particular package.

For example, there might exist two different versions of newsreaders on a system. The news server package might 'recommend' that there exist *some* news reader on the system, but the choice of `tin` or `trn` is left up to the individual user. This is satisfied by having both the `tin` and `trn` packages provide the virtual package `news-reader`. *Which* program is invoked is determined by a link pointing from a file with the virtual package name `/etc/alternatives/news-reader` to the selected file, e.g., `/usr/bin/trn`.

A single link is insufficient to support full use of an alternate program; normally, manual pages, and possibly other supporting files must be selected as well. The Perl script `update-alternatives` provides a way of ensuring that all the files associated with a specified package are selected as a system default.

12 Getting support for Debian Linux

12.1 What other documentation exists on and for a Debian system?

- Installation instructions for the current release: see `<http://www.debian.org/1.1/install>`.
- Debian-specific documentation: Avaiable in three ways:
  - on installed Debian systems. The latest versions of the `dpkg` package come with extensive new *programmers' manual* and *Debian policy manual* which are installed into the directory `/usr/doc/dpkg`. These documents are now available in the "unstable" distribution, but will be included in the next release.

    Users who install the package `doc-debian_VVV-RRR.deb` will find additional Debian-specific documentation in `/usr/doc/debian`. This package is stored in the `Debian-1.1/binary/doc` section in the Debian FTP archives.
  - on the WWW: See .
  - at Debian FTP mirror sites: See the directory `doc/`, or `doc/` (which is just a link to `doc`).
- Documentation on installed Debian packages: Most packages have files that are unpacked into `/usr/doc` and `/usr/doc/examples`.
- Documentation on the Linux project: The Debian package `doc-linux_VVV-RRR.deb` installs all of the most recent versions of the HOWTO's and mini-HOWTO's from the Linux Documentation Project. This package is in the Debian-1.1/binary/doc section in the FTP archives.
- Unix-style 'man' pages: Most commands have manual pages written in the style of the original Unix 'man' files. They are referenced by the section of the 'man' directory where they reside: e.g., foo(3) refers to a man page which resides in /usr/man/man3, and it can be called by executing the command: `man 3 foo`, or just `man foo` if section 3 is the only one containing a page on `foo`. One can learn which directory of `/usr/man` contains a certian man page by executing `man -w foo`. New Debian users should note that the 'man' pages of many general system commands are not available until they install these packages (both of which are normally stored in the `Debian-1.1/binary/doc` directory of the Debian FTP archive:

- `manpages_VVV-RRR.deb` (or `manpages-de_VVV-RRR.deb` for German text).
- `man_VVV-RRR.deb`, which contains the `man` program itself, and other programs for manipulating the manpages.
- GNU-style 'info' pages: User documentation for many commands, particularly GNU tools, is available not in 'man' pages, but in 'info' files which can be read by the GNU tool `info`. Its main advantage over the original 'man' pages are that it is a hypertext system. It does *not* require the WWW, however; `info` can be run from a plain text console. It was designed by Richard Stallman and preceded the WWW. The program that manipulates 'info' files is installed as a separate package `info_VVV-RRR.deb`, usually stored in the directory `Debian-1.1/binary/doc` on the FTP archive.

12.2 Are there any on-line resources for discussing Debian?

There are several Debian-related mailing lists:

- **debian-announce@lists.debian.org (moderated)**. (Usually) major system announcements. On average, fewer than one message per week are posted here.
- **debian-changes@lists.debian.org**. Announcements of new package uploads for the Debian system. This list may carry several announcements in a day.
- **debian-user@lists.debian.org** . A mailing list where users can ask for and receive advice on the use or configuration of Debian packages or other aspects of Debian Linux. This list is an invaluable learning resource; there can be 20 messages a day or more on this list.
- **debian-sparc@lists.debian.org** . This list is for those involved in porting Debian software to the Sparc platform.
- **debian-alpha@lists.debian.org** . This list is for those involved in porting Debian software to DEC alpha platforms.
- **debian-68k@lists.debian.org** . This list is for those involved in porting Debian software to Motorola 680x0 platforms; currently this means the Atari and Amiga only.
- **debian-talk@lists.debian.org** .

To subscribe to debian-X (for X in announce, changes, user), send mail to debian-X-REQUEST@lists.debian.org with the word "subscribe" in the Subject: header. If you have a forms-capable World Wide Web browser, you can subscribe to debian-announce and debian-user by using the *WWW form* `<http://www.rahul.net/perens/Debian/MailingLists.html>`. You can also un-subscribe using that form.

The lists are managed by Anders Chigstrom (ac@netg.se)

Archives of the Debian mailing lists are available on WWW at `<http://www.debian.org/Lists-Archives/>`.

Users can address questions to individual package maintainers, since their email addresses are provided in the Debian control file (see Section 6.4 that is included within each package. One can also learn the maintainers' names and email addresses by searching the "Packages" file, since this file is just a concatenation of all the available package control files in a particular directory tree. To extract a control file from a particular debian package, use the command,

```
dpkg --info packageName_VVV-RRR.deb
```

Another related mailing list, **debiangame**, is being run by Gary Moore (htmlurl url="mailto://gary@ssc.com" name="mailto://gary@ssc.com) at the University of Washington. As the name suggests, it is devoted to the discussion of games that have been (or might be) packaged for Debian. To subscribe, send mail to `listproc@u.washington.edu`, putting in the message body:

```
subscribe debiangame FirstName LastName
```

Both FirstName and LastName are required for ListProc.

Users should post non-Debian-specific questions to one of the Linux Usenet groups, which are named comp.os.linux.* or linux.*. Specialized Systems Consultants (a.k.a. SSC) maintains a *list of Linux, Unix, X, and networking newsgroups* `<http://www.ssc.com/linux/news.html>` on their WWW site.

12.3 What is the code of conduct for the mailing lists?

When using the Debian mailing lists, please follow these rules:

- Don't flame; it's not polite. Besides the people developing Debian are all volunteers, donating their time, energy and money in an attempt to bring the Debian project together.
- Don't use four-letter words, some people receive the lists via packet radio, where swear words are illegal.
- Make sure that you're using the proper list.
- Don't try posting unless you are subscribed. Posting by non-subscribers was turned off to make the mailing lists usable. Thus, only postings from email addresses matching those of a subscriber are accepted. If you often post from other addresses than the one you subscribe with, mail Anders Chigstrom (ac@netg.se) with the subject "please add subscriber alias".
- See section 13.2. for notes on reporting bugs.

12.4 Are there any other on-line resources dealing with Debian Linux?

Software in the Public Interest, Inc ., operates a WWW-server at `<http://www.debian.org>` which provides additional on-line resources. See also Section 14.3.1.

Generally, users should also exploit the WWW search engines for addressing specific problems. For example, the DejaNews news search service is an invaluable resource for exploring Usenet newsgroups. Try this search:

```
linux & WD
```

This tells DejaNews to report any postings containing either the string "linux" or the string "WD" When I used it, I discovered that my WD card (which I've only had for 6 months) has been declared outmoded by Adaptec, now that they've bought WD. So there are no drivers available. (Bless Adaptec's hearts.)

Or search the WWW using the Altavista Search Engine. For example, searching on the string "cgi-perl" gives a more detailed explanation of this package than the brief description field in its control file.

13 The Debian Bug Report System

13.1 Are there logs of known bugs?

The Debian Linux distribution has a bug tracking system which files details of bugs reported by users and developers. Each bug is given a number, and is kept on file until it is marked as having been dealt with.

Copies of this information are available at `<http://www.cl.cam.ac.uk/users/iwj10/debian-bugs/>` and `<http://www.debian.org/Bugs/>`, but the latter is not up-to-date at present.

The HTML files, in a form suitable for using as part of an HTTP server document tree, are in the WebPages subdirectory of the Debian FTP archive.

The bug reports are also available by anonymous FTP from any Debian mirror in the directory `debian-bugs/html/` for hypertext form, and in the directory `debian-bugs/text/` for plain text form.

13.2 How do I report a bug in Debian?

If you have found a bug in Debian, please read the instructions for reporting a bug in Debian. These instructions can be obtained in one of several ways:

- By anonymous ftp. Debian mirror sites contain the instructions in the file `debian-bugs/Reporting.html`.
- From the WWW. A copy of the instructions is shown at `<http://www.cl.cam.ac.uk/users/iwj10/debian-bugs/Reporting.html>`
- On any Debian system with the `doc-debian` package installed. The instructions are in the compressed (gzip'd) file `/usr/doc/debian/bug-reporting.txt.gz`.

(Note that at the time of this writing, the files included in the `doc-debian` package are not up to date with recent changes in the bug tracking system.)

Use these mailing lists for bugs:

- submit@bugs.debian.org: for general bug reports. Expect to get an automatic acknowledgement of your bug report. It will also be automatically given a bug tracking number and entered into the Bug log.

- maintonly@bugs.debian.org: to send bug reports to the maintainer only. (It would otherwise be copied to the debian-devel mailing list.) For example, if one were to identify a bug that was common to many programs, then rather than entering dozens of bug reports, one might prefer to send individual bugs to maintonly, then send a summary report to debian-devel.
- quiet@bugs.debian.org: to submit bug reports to the list only, without having them sent either to debian-devel or to the maintainer.

14 Contributing to the Debian project

Donations of time (to develop new packages, maintain existing packages, or provide user support), resources (to mirror the FTP and WWW archives), and money (to pay for new testbeds as well as hardware for the archives) can help the project.

14.1 How can I become a Debian software developer?

The development of the Debian distribution is open to all, and new users with the right skills and/or the willingness to learn are needed to maintain existing packages which have been "orphaned" by their previous maintainers, to develop new packages, and to provide user support.

To develop a Debian package, you should:

- (of course) download the Debian distribution and install it on your system.
- find a program you'd like to package that is not presently part of Debian. In this regard, the *Work Needing and Prospective Packages document* `<ftp://www.debian.org/debian/doc/package-developer/prospective-packages.txt>` should serve as a useful guide.
- Subscribe to debian-devel. To do this execute

  ```
  mail debian-devel-REQUEST@lists.debian.org -s Subscribe
  ```

  That is, send mail to the above address with the subject set to 'Subscribe'.
- Set up your pgp key. Install the PGP package; read /usr/doc/PGP; generate a public-secret key pair by typing `pgp -kg` and answering the questions. (Be prepared to write some arbitrary text when asked. PGP generates its true random numbers by measuring the intervals between your keystrokes.)
- Shake hands with the project administration, as it were: Send emails to:
  - The Project Leader. Bruce Perens
  - The Administrator of master.debian.org: Simon Shapiro.
  - The Mailing List Administrator: Anders Chigstrom.
  - The dpkg maintainer: Ian Jackson.

i.e., To: bruce@pixar.com, shimon@i-connect.net, ac@netg.se, ijackson@gnu.ai.mit.edu

with this content:

- what you've done,
- who you are,
- request for an account on master,
- request to be subscribed to debian-private (the developers-only mailing list).
- your PGP key, obtained by executing 'pgp -kxa'.

- Make the programs: .deb, .diff, .tar.gz, .changes. Use the `hello*deb` and `dpkg*deb` packages as guides, and read the *policy manual* and *dpkg programmers manual*.
- Upload the files to
  - `ftp://master.debian.org/home/Debian/ftp/private/project/Incoming` using your username and password, or
  - `ftp://chiark.chu.cam.uc.uk/` using anonymous ftp. For instructions to upload to chiark, read this file at chiark: `/pub/debian/private/project/README.how-to-upload`. *chiark* then sends the files it gets to master.

14.1.1 Are there additional information resources for developers?

Besides `debian-devel`, there are these mailing lists:

- The sparc port developers list: debian-sparc@lists.debian.org.
- The alpha port developers list: debian-alpha@lists.debian.org.
- The motorola 680xx port developers list: debian-68k@lists.debian.org.
- The dpkg developemnt list: debian-dpkg@lists.debian.org.
- Debian for Hams: debian-hams@lists.debian.org.
- Debian package maintainers only: debian-private@lists.debian.org.

To subscribe to debian-devel, or other developers' list, send a message to Anders Chigstrom (ac@netg.se) with a one-line explanation of what you would like to test or develop.

14.2 How can I contribute resources to the Debian project?

Since the project aims to make a substantial body of software rapidly and easily accessible throughout the globe, mirrors are urgently needed. It is desirable but not absolutely necessary to mirror all of the archive. The current components amount to just over 1.1 GBytes, broken down roughly as follows:

- binaries

  - Debian-1.1 (including updates): 200 MBytes.
  - non-free: 26 MBytes.
  - contrib: 6 MBytes.
  - rex: 216 MBytes.
- source code:
  - Debian-1.1 (including updates): 235 MBytes.
  - non-free: 23 MBytes.
  - contrib: 3 MBytes.
  - project/experimental: 13 MBytes.
  - rex: 242 MBytes.
- project/experimental: 13 MBytes.
- bug log: 18 MBytes.
- mailing list archives: 42 MBytes.
- private/project/Incoming: 150 MBytes.

Most of the mirroring is accomplished entirely automatically by scripts, without any human intervention. However, the occasional glitch or system change occurs which requires real personnel resources.

If you have a high-speed connection to the Internet, the resources to mirror all or part of the distribution, and are willing to take the time (or find someone) who can provide regular maintenance of the system, then please contact Bruce Perens (Email: bruce@pixar.com; Voice (work): 510-215-3502).

14.3 How can I contribute financially to the Debian project?

One can make individual donations to one of two organizations that are critical to the development of the Debian project.

14.3.1 Software in the Public Interest

Software in the Public Interest (SPI) is a non-profit organization formed when FSF withdrew their sponsorship of Debian. We are currently incorporating as an IRS 501(c)(3) non-profit organization. The purpose of the organization is to develop and distribute free software. Our goals are very much like those of FSF, and we encourage programmers to use the GNU General Public License on their programs. However, we have a slightly different focus in that we are building and distributing a Linux system that diverges in many technical details from the GNU system planned by FSF. We still communicate with FSF, and we cooperate in sending them changes to GNU software and in asking our users to donate to FSF and the GNU project.

SPI can be reached at:

- Email: bruce@pixar.com

- Postal address:

```
Software in the Public Interest
P.O. Box 70152
Pt. Richmond, CA 94807-0152
```

- Phone: 510-215-3502 (Bruce Perens at work)

14.3.2 Free Software Foundation

At this time there is no formal connection between Debian and the Free Software Foundation. However, the Free Software Foundation is responsible for some of the most important software components in Debian, including the GNU C compiler, GNU Emacs, and much of the C run-time library that is used by all programs on the system. FSF pioneered much of what free software is today: they wrote the the General Public License that is used on much of the Debian software, and they invented the "GNU" project to create an entirely free Unix system. Debian should be considered a descendent of the GNU system.

When you make a donation to FSF, please be sure to tell them that you are a Debian user. Please contact them at:

- E-mail: gnu@prep.ai.mit.edu
- Postal address:

```
Free Software Foundation
59 Temple Place - Suite 330
Boston, MA  02111-1307
USA
```

- Phone number: +1-617-542-5942
- Fax (including Japan): +1-617-542-2652
- Free Dial Fax (in Japan): 0031-13-2473 (KDD), 0066-3382-0158 (IDC)
- On the WWW (unofficial FSF page): http://www.cs.pdx.edu/ trent/gnu/

15 Redistributing Debian Linux in a commercial product

15.1 Can I make and sell Debian CDs?

Go ahead. You don't need permission to distribute anything we've *released*, so that you can master your CD as soon as the beta-test ends. You don't have to pay us anything. We will, however, publish a list of CD manufacturers who donate money, software, and time to the Debian project, and we'll encourage users to buy from manufacturers who donate, so it's good advertising to make donations. Of course all CD manufacturers must honor the licenses of the programs in Debian. For example, many of the programs are licensed under the GPL, which requires you to distribute their source code.

15.2 Can Debian be packaged with non-free software?

Yes. While all the main components of Debian are free software, we provide a non-free directory for programs that aren't freely redistributable. CD manufacturers *may* be able to distribute the programs we've placed in that directory, depending on the license terms or their private arrangements with the authors of those software packages. CD manufacturers can also distribute the non-free software they get from other sources on the same CD. This is nothing new: free and commercial software are distributed on the same CD by many manufacturers now. Of course we still encourage software authors to release the programs they write as free software.

15.3 I'm making a special Linux distribution for a "vertical market". Can I use Debian 1.1 for the guts of a Linux system and add my own applications on top of it?

Yes. For example, one person is building a "Linux for Hams" distribution, with specialized programs for Radio Amateurs. He's starting with Debian 1.1 as the "base system", and adding programs to control the transmitter, track satellites, etc. All of the programs he adds are packaged with the Debian package system so that his users will be able to upgrade easily when he releases subsequent CDs.

Debian also provides a mechanism to allow developers and system administrators to install local versions of selected files in such a way that they will not be overwritten when other packages are upgraded. This is discussed further in the question on 11.4; more details are given in the file `/usr/doc/debian/package-developers/diversions.txt.gz`, which is part of the `doc-debian` package.

15.4 Can I put my commercial program in a Debian "package" so that it installs effortlessly on any Debian system?

Go right ahead. The package tool is free software.

16 What kind of changes should I expect to see in the next major release of Debian Linux?

16.1 A New Approach to the Boot Floppies

In order to accomodate all of the hardware for which Linux drivers have been written, yet still provide boot-floppies that can fit on a 5.25-inch disk, the Debian 1.2 distribution will provide a mechanism to access all drivers that can be modularized as modules.

The default kernel has no IDE, no SCSI, little else. When the user sets up the system, it builds a RAM disk image that is loaded at boot time. The script on the RAM disk loads the modules for your local hardware configuration _before_ it mounts the root. Some things that still aren't modularized are in the default kernel.

The stripped-down kernel and the compressed installation root filesystem fit on one 1200K floppy and run fine that way. The floppy also contains a few of the most popular modules, and you get to feed it more floppies (or a CD) containing other modules depending on your configuration.

No-floppy bootstrap should be possible for systems with 8MB RAM and a CD or DOS hard disk. One-floppy NFS bootstrap should be possible.

See /usr/src/linux/Documentation/initrd.txt .

16.2 New Debian package tools and documentation

Programmers' and Policy Manuals that provide detailed explanations of the Debian package management system and approach are provided with the new version of `dpkg`.

In addition, the new version of `dpkg` will include new tools for automating the construction of Debian packages, and for supporting multiple architechtures.

16.3 Increased security

Debian 1.2 will be equipped with the Shadow password suite of programs and library routines which hide the plain text of the password file from all users except root and provide for terminal access control, and user and group administration. For details, see `<http://sunsite.unc.edu/mdw/HOWTO/Shadow-Password-HOWTO.html>`. This document is of course also included along with all the HOWTO's in the Debian `doc-debian` package.

Most of the relevant Debian utilities have been modified to make them support shadow passwords. This includes the default `login` utility, as well as `vipw` and `vigr`. The `xdm` X11 login utility and the `adduser` script will be made shadow aware as well. In addition, there will be minor changes to the shadow packages themselves.

In addition, the Linux library of Pluggable Authentication Modules (`<http://gluon.physics.ucla.edu/ morgan/pam>`; a.k.a. `libpam`) that allow sysadmins to choose authorization modes on an application-specific basis will be available, and initially set to authenticate via shadow password. However, release 1.2 will not include PAM-aware applications.