Ubuntu Installation Guide

Ubuntu Installation Guide

This document contains installation instructions for the Ubuntu 18.04 system (codename "'Bionic Beaver'"), for the 64-bit ARM ("arm64") architecture. It also contains pointers to more information and information on how to make the most of your new Ubuntu system.

Table of Contents

Installing Ubuntu 18.04 “Bionic Beaver” For arm64 ix
1. Welcome to Ubuntu 1
1.1. What is Ubuntu? 1
1.1.1. Sponsorship by Canonical 1
1.2. What is Debian? 1
1.2.1. Ubuntu and Debian 2
1.2.1.1. Package selection 2
1.2.1.2. Releases 3
1.2.1.3. Development community 3
1.2.1.4. Freedom and Philosophy 3
1.2.1.5. Ubuntu and other Debian derivatives 4
1.3. What is GNU/Linux? 4
1.4. Getting Ubuntu 5
1.5. Getting the Newest Version of This Document 5
1.6. Organization of This Document 5
1.7. Your Documentation Help is Welcome 6
1.8. About Copyrights and Software Licenses 6
2. System Requirements 8
2.1. Supported Hardware 8
2.1.1. Supported Architectures 8
2.1.2. Variations in ARM CPU designs and support complexity 8
2.1.3. Platforms supported by Ubuntu/arm64 9
2.1.3.1. Other platforms 9
2.1.4. Multiple Processors 9
2.1.5. Graphics Hardware Support 10
2.1.6. Network Connectivity Hardware 10
2.1.7. Peripherals and Other Hardware 10
2.2. Devices Requiring Firmware 10
2.3. Purchasing Hardware Specifically for GNU/Linux 11
2.3.1. Avoid Proprietary or Closed Hardware 11
2.4. Installation Media 12
2.4.1. CD-ROM/DVD-ROM/BD-ROM 12
2.4.2. Network 12
2.4.3. Hard Disk 12
2.4.4. Un*x or GNU system 13
2.4.5. Supported Storage Systems 13
2.5. Memory and Disk Space Requirements 13
3. Before Installing Ubuntu 14
3.1. Overview of the Installation Process 14
3.2. Back Up Your Existing Data! 15
3.3. Information You Will Need 15
3.3.1. Documentation 15
3.3.1.1. Installation Manual 15
3.3.1.2. Hardware documentation 15
3.3.2. Finding Sources of Hardware Information 15
3.3.3. Hardware Compatibility 16
3.3.3.1. Testing hardware compatibility with a Live-System 17
3.3.4. Network Settings 18

3.4. Meeting Minimum Hardware Requirements 18
3.5. Pre-Partitioning for Multi-Boot Systems 19
3.6. Pre-Installation Hardware and Operating System Setup 20
3.6.1. Systems with U-Boot firmware 20
3.6.1.1. Setting the ethernet MAC address in U-Boot 20
3.6.1.2. Kernel/Initrd/Device-Tree relocation issues in U-Boot 21
3.6.2. Systems with UEFI firmware 21
4. Obtaining System Installation Media 23
4.1. Official Ubuntu CD-ROMs 23
4.2. Downloading Files from Ubuntu Mirrors 23
4.2.1. Where to Find Installation Images 23
4.3. Preparing Files for TFTP Net Booting 23
4.3.1. Setting up RARP server 24
4.3.2. Setting up a DHCP server 24
4.3.3. Setting up a BOOTP server 25
4.3.4. Enabling the TFTP Server 25
4.3.5. Move TFTP Images Into Place 26
4.4. Automatic Installation 26
4.4.1. Automatic Installation Using the Ubuntu Installer 26
4.4.2. Automatic Installation Using Kickstart 26
4.4.2.1. Additions 27
4.4.2.2. Missing features 28
4.4.2.3. Example 28
5. Booting the Installation System 31
5.1. Booting the Installer on 64-bit ARM 31
5.1.1. Console configuration 31
5.1.2. Juno Installation 31
5.1.3. Applied Micro Mustang Installation 31
5.1.4. Booting by TFTP 32
5.1.4.1. TFTP-booting in U-Boot 32
5.2. Accessibility 33
5.2.1. Installer front-end 34
5.2.2. Board Devices 34
5.2.3. High-Contrast Theme 34
5.2.4. Zoom 34
5.2.5. Preseeding 34
5.2.6. Accessibility of the installed system 34
5.3. Boot Parameters 34
5.3.1. Boot console 35
5.3.2. Ubuntu Installer Parameters 35
5.3.3. Using boot parameters to answer questions 38
5.3.4. Passing parameters to kernel modules 39
5.3.5. Blacklisting kernel modules 40
5.4. Troubleshooting the Installation Process 40
5.4.1. CD-ROM Reliability 40
5.4.1.1. Common issues 40
5.4.1.2. How to investigate and maybe solve issues 40
5.4.2. Boot Configuration 42
5.4.3. Interpreting the Kernel Startup Messages 42
5.4.4. Reporting Installation Problems 42
5.4.5. Submitting Installation Reports 42

6. Using the Ubuntu Installer 44
6.1. How the Installer Works 44
6.2. Components Introduction 45
6.3. Using Individual Components 47
6.3.1. Setting up Ubuntu Installer and Hardware Configuration 47
6.3.1.1. Check available memory / low memory mode 47
6.3.1.2. Selecting Localization Options 48
6.3.1.3. Choosing a Keyboard 49
6.3.1.4. Looking for the Ubuntu Installer ISO Image 49
6.3.1.5. Configuring the Network 49
6.3.1.5.1. Automatic network configuration 50
6.3.1.5.2. Manual network configuration 50
6.3.1.5.3. IPv4 and IPv6 50
6.3.1.6. Configuring the Clock and Time Zone 50
6.3.2. Setting Up Users And Passwords 51
6.3.2.1. Create an Ordinary User 51
6.3.3. Partitioning and Mount Point Selection 51
6.3.3.1. Supported partitioning options 52
6.3.3.2. Guided Partitioning 53
6.3.3.3. Manual Partitioning 54
6.3.3.4. Configuring Multidisk Devices (Software RAID) 55
6.3.3.5. Configuring the Logical Volume Manager (LVM) 58
6.3.3.6. Configuring Encrypted Volumes 59
6.3.4. Installing the Base System 62
6.3.5. Installing Additional Software 62
6.3.5.1. Configuring apt 62
6.3.5.1.1. Installing from more than one CD or DVD 63
6.3.5.1.2. Using a network mirror 63
6.3.5.1.3. Choosing a network mirror 64
6.3.5.2. Selecting and Installing Software 64
6.3.6. Making Your System Bootable 65
6.3.6.1. Detecting other operating systems 65
6.3.6.2. Making the system bootable with flash-kernel 65
6.3.6.3. Continue Without Boot Loader 66
6.3.7. Finishing the Installation 66
6.3.7.1. Setting the System Clock 66
6.3.7.2. Reboot the System 66
6.3.8. Troubleshooting 66
6.3.8.1. Saving the installation logs 66
6.3.8.2. Using the Shell and Viewing the Logs 67
6.3.9. Installation Over the Network 67
6.4. Loading Missing Firmware 69
6.4.1. Preparing a medium 69
6.4.2. Firmware and the Installed System 70
7. Booting Into Your New Ubuntu System 71
7.1. The Moment of Truth 71
7.2. Mounting encrypted volumes 71
7.2.1. Troubleshooting 71
7.3. Log In 72

8. Next Steps and Where to Go From Here 74
8.1. Shutting down the system 74
8.2. If You Are New to Unix 74
8.3. Orienting Yourself to Ubuntu 74
8.3.1. Ubuntu Packaging System 74
8.3.2. Additional Software Available for Ubuntu 75
8.3.3. Application Version Management 75
8.3.4. Cron Job Management 75
8.4. Further Reading and Information 75
8.5. Setting Up Your System To Use E-Mail 76
8.5.1. Default E-Mail Configuration 76
8.5.2. Sending E-Mails Outside The System 76
8.5.3. Configuring the Exim4 Mail Transport Agent 77
8.6. Compiling a New Kernel 78
8.6.1. Kernel Image Management 78
8.7. Recovering a Broken System 79
A. Installation Howto 82
A.1. Booting the installer 82
A.1.1. Booting from CDROM 82
A.1.2. Booting from network 82
A.2. Installation 82
A.3. And finally... 83
B. Automating the installation using preseeding 84
B.1. Introduction 84
B.1.1. Preseeding methods 84
B.1.2. Limitations 85
B.1.3. Debconf basics 85
B.2. Using preseeding 85
B.2.1. Loading the preconfiguration file 86
B.2.2. Using boot parameters to preseed questions 86
B.2.3. Auto mode 87
B.2.4. Aliases useful with preseeding 89
B.2.5. Using a DHCP server to specify preconfiguration files 89
B.3. Creating a preconfiguration file 90
B.4. Contents of the preconfiguration file (for bionic) 91
B.4.1. Localization 91
B.4.2. Network configuration 92
B.4.3. Network console 94
B.4.4. Mirror settings 94
B.4.5. Account setup 95
B.4.6. Clock and time zone setup 96
B.4.7. 64-bit ARM specific disk storage 96
B.4.8. Partitioning 96
B.4.8.1. Partitioning example 97
B.4.8.2. Partitioning using RAID 98
B.4.8.3. Controlling how partitions are mounted 99
B.4.9. Base system installation 100
B.4.10. Apt setup 100
B.4.11. Package selection 101
B.4.12. Finishing up the installation 102
B.4.13. Preseeding other packages 102

B.5. Advanced options 103
B.5.1. Running custom commands during the installation 103
B.5.2. Using preseeding to change default values 103
B.5.3. Chainloading preconfiguration files 104
C. Partitioning for Ubuntu 105
C.1. Deciding on Ubuntu Partitions and Sizes 105
C.2. The Directory Tree 105
C.3. Recommended Partitioning Scheme 106
C.4. Device Names in Linux 107
C.5. Ubuntu Partitioning Programs 108
D. Random Bits 109
D.1. Linux Devices 109
D.1.1. Setting Up Your Mouse 109
D.2. Disk Space Needed for Tasks 110
D.3. Disk Space Needed 110
D.4. Installing Ubuntu from a Unix/Linux System 111
D.4.1. Getting Started 111
D.4.2. Install **debootstrap** 112
D.4.3. Run **debootstrap** 112
D.4.4. Configure The Base System 113
D.4.4.1. Configure Apt 113
D.4.4.2. Install additional packages 113
D.4.4.3. Create device files 114
D.4.4.4. Mount Partitions 114
D.4.4.5. Setting Timezone 115
D.4.4.6. Configure Networking 116
D.4.4.7. Configure Locales and Keyboard 117
D.4.5. Install a Kernel 117
D.4.6. Set up the Boot Loader 117
D.4.7. Remote access: Installing SSH and setting up access 118
D.4.8. Finishing touches 118
D.4.9. Create a User 119
D.5. Installing Ubuntu using PPP over Ethernet (PPPoE) 119
E. Administrivia 121
E.1. About This Document 121
E.2. Contributing to This Document 121
E.3. Major Contributions 121
E.4. Trademark Acknowledgement 122
F. GNU General Public License 123
F.1. Preamble 123
F.2. GNU GENERAL PUBLIC LICENSE 123
F.3. How to Apply These Terms to Your New Programs 127

List of Tables

3-1. Hardware Information Helpful for an Install 16
3-2. Recommended Minimum System Requirements 19

Installing Ubuntu 18.04 “Bionic Beaver” For arm64

We are delighted that you have decided to try Ubuntu, and are sure that you will find that Ubuntu’s GNU/Linux distribution is unique. Ubuntu brings together high-quality free software from around the world, integrating it into a coherent whole. We believe that you will find that the result is truly more than the sum of the parts.

We understand that many of you want to install Ubuntu without reading this manual, and the Ubuntu installer is designed to make this possible. If you don’t have time to read the whole Installation Guide right now, we recommend that you read the Installation Howto, which will walk you through the basic installation process, and links to the manual for more advanced topics or for when things go wrong. The Installation Howto can be found in Appendix A.

With that said, we hope that you have the time to read most of this manual, and doing so will lead to a more informed and likely more successful installation experience.

Chapter 1. Welcome to Ubuntu

This chapter provides an overview of the Ubuntu Project, and the Debian Project upon which it is based. If you already know about the Ubuntu Project's history and the Ubuntu distribution, feel free to skip to the next chapter.

1.1. What is Ubuntu?

Ubuntu is a complete Linux operating system, freely available with both community and professional support. The Ubuntu community is built on the ideas enshrined in the Ubuntu Manifesto: that software should be available free of charge, that software tools should be usable by people in their local language and despite any disabilities, and that people should have the freedom to customize and alter their software in whatever way they see fit.

- *Ubuntu will always be free of charge*, and there is no extra fee for the "enterprise edition", we make our very best work available to everyone on the same Free terms.
- Ubuntu includes the *very best in translations and accessibility infrastructure* that the Free Software community has to offer, to make Ubuntu usable by as many people as possible.
- Ubuntu is shipped in stable and regular release cycles; *a new release will be shipped every six months*. Every two even years an Ubuntu long term support (LTS) release will become available, that is supported for 5 years. The Ubuntu releases in between (known as development or non-LTS releases) are supported for 9 month each.
- Ubuntu is entirely committed to the principles of open source software development; we encourage people to use open source software, improve it and pass it on.

Ubuntu is suitable for both desktop and server use. The current Ubuntu release supports Intel x86 (IBM-compatible PC), AMD64 (x86-64), ARMv7, ARMv8 (ARM64), IBM POWER8/POWER9 (ppc64el), IBM Z zEC12/zEC13/z14 and IBM LinuxONE Rockhopper I+II/Emporer I+II (s390x).

Ubuntu includes thousands of pieces of software, starting with the Linux kernel version 4.15 and GNOME 3.28, and covering every standard desktop application from word processing and spreadsheet applications to internet access applications, web server software, email software, programming languages and tools and of course several games.

1.1.1. Sponsorship by Canonical

The Ubuntu Project is sponsored by Canonical Ltd (http://www.canonical.com/). Canonical will not charge licence fees for Ubuntu, now or at any stage in the future. Canonical's business model is to provide technical support and professional services related to Ubuntu. We encourage more companies also to offer support for Ubuntu, and will list those that do on the Support pages of this web site.

1.2. What is Debian?

Debian is an all-volunteer organization dedicated to developing free software and promoting the ideals of the Free Software community. The Debian Project began in 1993, when Ian Murdock issued

an open invitation to software developers to contribute to a complete and coherent software distribution based on the relatively new Linux kernel. That relatively small band of dedicated enthusiasts, originally funded by the Free Software Foundation (http://www.fsf.org/) and influenced by the GNU (http://www.gnu.org/gnu/the-gnu-project.html) philosophy, has grown over the years into an organization of around 1026 *Debian Developers*.

Debian Developers are involved in a variety of activities, including Web (http://www.debian.org/) and FTP (ftp://ftp.debian.org/) site administration, graphic design, legal analysis of software licenses, writing documentation, and, of course, maintaining software packages.

In the interest of communicating our philosophy and attracting developers who believe in the principles that Debian stands for, the Debian Project has published a number of documents that outline our values and serve as guides to what it means to be a Debian Developer:

- The Debian Social Contract (http://www.debian.org/social_contract) is a statement of Debian's commitments to the Free Software Community. Anyone who agrees to abide to the Social Contract may become a maintainer (http://www.debian.org/doc/maint-guide/). Any maintainer can introduce new software into Debian — provided that the software meets our criteria for being free, and the package follows our quality standards.
- The Debian Free Software Guidelines (http://www.debian.org/social_contract#guidelines) are a clear and concise statement of Debian's criteria for free software. The DFSG is a very influential document in the Free Software Movement, and was the foundation of the The Open Source Definition (http://opensource.org/osd).
- The Debian Policy Manual (http://www.debian.org/doc/debian-policy/) is an extensive specification of the Debian Project's standards of quality.

Debian developers are also involved in a number of other projects; some specific to Debian, others involving some or all of the Linux community. Some examples include:

- The Linux Standard Base (http://www.linuxbase.org/) (LSB) is a project aimed at standardizing the basic GNU/Linux system, which will enable third-party software and hardware developers to easily design programs and device drivers for Linux-in-general, rather than for a specific GNU/Linux distribution.
- The Filesystem Hierarchy Standard (http://www.pathname.com/fhs/) (FHS) is an effort to standardize the layout of the Linux file system. The FHS will allow software developers to concentrate their efforts on designing programs, without having to worry about how the package will be installed in different GNU/Linux distributions.
- Debian Jr. (http://www.debian.org/devel/debian-jr/) is an internal project, aimed at making sure Debian has something to offer to our youngest users.

For more general information about Debian, see the Debian FAQ (http://www.debian.org/doc/FAQ/).

1.2.1. Ubuntu and Debian

Ubuntu and Debian are distinct but parallel and closely linked systems. The Ubuntu project seeks to complement the Debian project in the following areas:

1.2.1.1. Package selection

Ubuntu does not provide security updates and professional support for every package available in the open source world, but selects a complete set of packages making up a solid and comprehensive system and provides support for that set of packages.

For users that want access to every known package, Ubuntu provides a "universe" component (set of packages) where users of Ubuntu systems install the latest version of any package that is not in the supported set. Most of the packages in Ubuntu universe are also in Debian, although there are other sources for universe too. See the Ubuntu Components page for more detail on the structure of the Ubuntu web distribution.

1.2.1.2. Releases

Ubuntu makes a release every six months, and supports those releases for 18 months with daily security fixes and patches to critical bugs.

As Ubuntu prepares for release, we “freeze” a snapshot of Debian’s development archive (“sid”). We start from “sid” in order to give ourselves the freedom to make our own decisions with regard to release management, independent of Debian’s release-in-preparation. This is necessary because our release criteria are very different from Debian’s.

As a simple example, a package might be excluded from Debian “testing” due to a build failure on any of the 11 architectures supported by Debian “sarge”, but it is still suitable for Ubuntu if it builds and works on only three of them. A package will also be prevented from entering Debian “testing” if it has release-critical bugs according to Debian criteria, but a bug which is release-critical for Debian may not be as important for Ubuntu.

As a community, we choose places to diverge from Debian in ways that minimize the difference between Debian and Ubuntu. For example, we usually choose to update to the very latest version of Gnome rather than the older version in Debian, and we might do the same for key other pieces of infrastructure such as X or GCC. Those decisions are listed as Feature Goals for that release, and we work as a community to make sure that they are in place before the release happens.

1.2.1.3. Development community

Many Ubuntu developers are also recognized members of the Debian community. They continue to stay active in contributing to Debian both in the course of their work on Ubuntu and directly in Debian.

When Ubuntu developers fix bugs that are also present in Debian packages -- and since the projects are linked, this happens often -- they send their bugfixes to the Debian developers responsible for that package in Debian and record the patch URL in the Debian bug system. The long term goal of that work is to ensure that patches made by the full-time Ubuntu team members are immediately also included in Debian packages where the Debian maintainer likes the work.

In Ubuntu, team members can make a change to any package, even if it is one maintained by someone else. Once you are an Ubuntu maintainer it’s encouraged that you fix problems you encounter, although we also encourage polite discussions between people with an interest in a given package to improve cooperation and reduce friction between maintainers.

1.2.1.4. Freedom and Philosophy

Debian and Ubuntu are grounded on the same free software philosophy. Both groups are explicitly committed to building an operating system of free software.

Differences between the groups lie in their treatment of non-computer applications (like documentation, fonts and binary firmware) and non-free software. Debian distributes a small amount of non-free software from their Internet servers. Ubuntu will also distribute binary drivers in the "restricted" component on its Internet servers but will not distribute any other software applications that do not meet its own Ubuntu Licensing Guidelines.

1.2.1.5. Ubuntu and other Debian derivatives

There are many other distributions that also share the same basic infrastructure (package and archive format). Ubuntu is distinguished from them in a number of ways.

First, Ubuntu contributes patches directly to Debian as bugs are fixed during the Ubuntu release process, not just when the release is actually made. With other Debian-style distributions, the source code and patches are made available in a "big bang" at release time, which makes them difficult to integrate into the upstream HEAD.

Second, Ubuntu includes a number of full-time contributors who are also Debian developers. Many of the other distributions that use Debian-style packaging do not include any active Debian contributors.

Third, Ubuntu makes much more frequent and fresher releases. Our release policy of releasing every six months is (at the time of writing :-) unique in the Linux distribution world. Ubuntu aims to provide you with a regular stable and security-supported snapshot of the best of the open source world.

1.3. What is GNU/Linux?

Linux is an operating system: a series of programs that let you interact with your computer and run other programs.

An operating system consists of various fundamental programs which are needed by your computer so that it can communicate and receive instructions from users; read and write data to hard disks, tapes, and printers; control the use of memory; and run other software. The most important part of an operating system is the kernel. In a GNU/Linux system, Linux is the kernel component. The rest of the system consists of other programs, many of which were written by or for the GNU Project. Because the Linux kernel alone does not form a working operating system, we prefer to use the term “GNU/Linux” to refer to systems that many people casually refer to as “Linux”.

Linux is modelled on the Unix operating system. From the start, Linux was designed to be a multitasking, multi-user system. These facts are enough to make Linux different from other well-known operating systems. However, Linux is even more different than you might imagine. In contrast to other operating systems, nobody owns Linux. Much of its development is done by unpaid volunteers.

Development of what later became GNU/Linux began in 1984, when the Free Software Foundation (http://www.fsf.org/) began development of a free Unix-like operating system called GNU.

The GNU Project (http://www.gnu.org/) has developed a comprehensive set of free software tools for use with Unix™ and Unix-like operating systems such as Linux. These tools enable users to perform tasks ranging from the mundane (such as copying or removing files from the system) to the arcane (such as writing and compiling programs or doing sophisticated editing in a variety of document formats).

While many groups and individuals have contributed to Linux, the largest single contributor is still the Free Software Foundation, which created not only most of the tools used in Linux, but also the philosophy and the community that made Linux possible.

The Linux kernel (http://www.kernel.org/) first appeared in 1991, when a Finnish computing science student named Linus Torvalds announced an early version of a replacement kernel for Minix to the Usenet newsgroup **comp.os.minix**. See Linux International's Linux History Page (http://www.cs.cmu.edu/~awb/linux.history.html).

Linus Torvalds continues to coordinate the work of several hundred developers with the help of a number of subsystem maintainers. There is an official website (http://www.kernel.org/) for the Linux kernel. Information about the **linux-kernel** mailing list can be found on the linux-kernel mailing list FAQ (http://www.tux.org/lkml/).

Linux users have immense freedom of choice in their software. For example, Linux users can choose from a dozen different command line shells and several graphical desktops. This selection is often bewildering to users of other operating systems, who are not used to thinking of the command line or desktop as something that they can change.

Linux is also less likely to crash, better able to run more than one program at the same time, and more secure than many operating systems. With these advantages, Linux is the fastest growing operating system in the server market. More recently, Linux has begun to be popular among home and business users as well.

1.4. Getting Ubuntu

For information on how to download Ubuntu from the Internet, see the download web page (http://www.ubuntu.com/download/). The list of Ubuntu mirrors (http://wiki.ubuntu.com/Archive) contains a full set of official Ubuntu mirrors, so you can easily find the nearest one.

Ubuntu can be upgraded after installation very easily. The installation procedure will help set up the system so that you can make those upgrades once installation is complete, if need be.

1.5. Getting the Newest Version of This Document

This document is constantly being revised. Updated versions of this installation manual are available from the official Install Manual pages (http://help.ubuntu.com/18.04/installation-guide/arm64/).

1.6. Organization of This Document

This document is meant to serve as a manual for first-time Ubuntu users. It tries to make as few assumptions as possible about your level of expertise. However, we do assume that you have a general understanding of how the hardware in your computer works.

Expert users may also find interesting reference information in this document, including minimum installation sizes, details about the hardware supported by the Ubuntu installation system, and so on. We encourage expert users to jump around in the document.

In general, this manual is arranged in a linear fashion, walking you through the installation process from start to finish. Here are the steps in installing Ubuntu, and the sections of this document which correlate with each step:

1. Determine whether your hardware meets the requirements for using the installation system, in Chapter 2.

2. Backup your system, perform any necessary planning and hardware configuration prior to installing Ubuntu, in Chapter 3. If you are preparing a multi-boot system, you may need to create partitionable space on your hard disk for Ubuntu to use.
3. In Chapter 4, you will obtain the necessary installation files for your method of installation.
4. Chapter 5 describes booting into the installation system. This chapter also discusses troubleshooting procedures in case you have problems with this step.
5. Perform the actual installation according to Chapter 6. This involves choosing your language, configuring peripheral driver modules, configuring your network connection, so that remaining installation files can be obtained directly from an Ubuntu server (if you are not installing from a CD), partitioning your hard drives and installation of a base system, then selection and installation of tasks. (Some background about setting up the partitions for your Ubuntu system is explained in Appendix C.)
6. Boot into your newly installed base system, from Chapter 7.

Once you've got your system installed, you can read Chapter 8. That chapter explains where to look to find more information about Unix and Ubuntu, and how to replace your kernel.

Finally, information about this document and how to contribute to it may be found in Appendix E.

1.7. Your Documentation Help is Welcome

Any help, suggestions, and especially, patches, are greatly appreciated. Working versions of this document can be found at http://d-i.alioth.debian.org/manual/. There you will find a list of all the different architectures and languages for which this document is available.

Source is also available publicly; look in Appendix E for more information concerning how to contribute. We welcome suggestions, comments, patches, and bug reports (use the package `installation-guide` for bugs, but check first to see if the problem is already reported).

1.8. About Copyrights and Software Licenses

We're sure that you've read some of the licenses that come with most commercial software — they usually say that you can only use one copy of the software on a single computer. This system's license isn't like that at all. We encourage you to put a copy of Ubuntu on every computer in your school or place of business. Lend your installation media to your friends and help them install it on their computers! You can even make thousands of copies and *sell* them — albeit with a few restrictions. Your freedom to install and use the system comes directly from Ubuntu being based on *free software*.

Calling software *free* doesn't mean that the software isn't copyrighted, and it doesn't mean that CDs/DVDs containing that software must be distributed at no charge. Free software, in part, means that the licenses of individual programs do not require you to pay for the privilege of distributing or using those programs. Free software also means that not only may anyone extend, adapt, and modify the software, but that they may distribute the results of their work as well.

> **Note:** The Ubuntu project, as a pragmatic concession to its users, does make some packages available that do not meet our criteria for being free. These packages are not part of the official distribution, however, and are only available from the **`multiverse`** area of Ubuntu mirrors; see the Ubuntu web site (http://www.ubuntu.com/ubuntu/components) for more information about the layout and contents of the archives.

Many of the programs in the system are licensed under the *GNU General Public License*, often simply referred to as "the GPL". The GPL requires you to make the *source code* of the programs available whenever you distribute a binary copy of the program; that provision of the license ensures that any user will be able to modify the software. Because of this provision, the source code[1] for all such programs is available in the Ubuntu system.

There are several other forms of copyright statements and software licenses used on the programs in Ubuntu. You can find the copyrights and licenses for every package installed on your system by looking in the file `/usr/share/doc/`*`package-name`*`/copyright` once you've installed a package on your system.

For more information about licenses and how Ubuntu determines whether software is free enough to be included in the main distribution, see the Ubuntu License Policy (http://www.ubuntu.com/ubuntu/licensing).

The most important legal notice is that this software comes with *no warranties*. The programmers who have created this software have done so for the benefit of the community. No guarantee is made as to the suitability of the software for any given purpose. However, since the software is free, you are empowered to modify that software to suit your needs — and to enjoy the benefits of the changes made by others who have extended the software in this way.

1. For information on how to locate, unpack, and build binaries from Ubuntu source packages, see the Debian FAQ (http://www.debian.org/doc/FAQ/), under "Basics of the Debian Package Management System".

Chapter 2. System Requirements

This section contains information about what hardware you need to get started with Ubuntu. You will also find links to further information about hardware supported by GNU and Linux.

2.1. Supported Hardware

Ubuntu does not impose hardware requirements beyond the requirements of the Linux kernel and the GNU tool-sets. Therefore, any architecture or platform to which the Linux kernel, libc, **gcc**, etc. have been ported, and for which an Ubuntu port exists, can run Ubuntu.

Rather than attempting to describe all the different hardware configurations which are supported for 64-bit ARM, this section contains general information and pointers to where additional information can be found.

2.1.1. Supported Architectures

Ubuntu 18.04 supports six major architectures and several variations of each architecture known as "flavors". One other architecture (IBM/Motorola PowerPC) has an unofficial port.

Architecture	Ubuntu Designation	Subarchitecture	Flavor
Intel x86-based	i386		
AMD64 & Intel 64	amd64		
ARM with hardware FPU	armhf	multiplatform	generic
		multiplatform for LPAE-capable systems	generic-lpae
64bit ARM	arm64		
IBM POWER Systems	ppc64el	IBM POWER8 and newer machines	
IBM z/Architecture	arm64	IBM Z and IBM LinuxONE, no s390 (31-bit mode) support	zEC12 and newer machines

2.1.2. Variations in ARM CPU designs and support complexity

ARM systems are much more heterogeneous than those based on the i386/amd64-based PC architecture, so the support situation can be much more complicated.

The ARM architecture is used mainly in so-called "system-on-chip" (SoC) designs. These SoCs are designed by many different companies, often with vastly varying hardware components even for the very basic functionality required to bring the system up. Older versions of the ARM architecture have seen massive differences from one SoC to the next, but ARMv8 (arm64) is much more standardised and so is easier for the Linux kernel and other software to support.

Server versions of ARMv8 hardware are typically configured using the Unified Extensible Firmware Interface (UEFI) and Advanced Configuration and Power Interface (ACPI) standards. These two provide common, device-independent ways to boot and configure computer hardware. They are also common in the x86 PC world.

2.1.3. Platforms supported by Ubuntu/arm64

The following platforms are known to be supported by Ubuntu/arm64 in this release. There is only one kernel image, which supports all the listed platforms.

Applied Micro (APM) Mustang/X-Gene

> The APM Mustang was the first Linux-capable ARMv8 system available. It uses the X-gene SoC, which has since also been used in other machines. It is an 8-core CPU, with ethernet, USB and serial. A common form-factor looks just like a desktop PC box, but many other versions are expected in the future. Most of the hardware is supported in the mainline kernel, but at this point USB support is lacking in the Bionic kernel.

ARM Juno Development Platform

> Juno is a capable development board with a 6-core (2xA57, 4xA53) ARMv8-A 800Mhz CPU, Mali (T624) graphics, 8GB DDR3 RAM, Ethernet, USB, Serial. It was designed for system bring-up and power testing so is neither small nor cheap, but was one of the first boards available. All the on-board hardware is supported in the mainline kernel and in Bionic.

Cavium ThunderX Reference Platforms CN8800-1S-CDK & CN8800-2S-CDK

> The ThunderX 1S Reference Platform is a 1U rack mount chassis with a Micro-ATX single socket motherboard supporting 10GE & 40GE networking, x8 PCIe, 4 HDD drive slots and 8 DDR3 memory slots. The ThunderX 2S Reference Platform is a 2U rack mount chassis with a ½ SSI dual socket motherboard supporting 10GE & 40GE networking, x8 PCIe, 6 HDD drive slots and 8 DDR4 memory slots. The 2U chassis can accommodate 4 sleds and each of the sleds has a dual 48 core socket design (96 cores per sled). The Bionic kernel has support for nearly all on-board hardware. Certain exceptions, such as the IPMI system interface (`/dev/ipmi`), may be supported in a future Bionic update.

When using `debian-installer` on non-UEFI systems, you may have to manually make the system bootable at the end of the installation, e.g. by running the required commands in a shell started from within `debian-installer`. flash-kernel knows how to set up an X-Gene system booting with U-Boot.

2.1.3.1. Other platforms

The multiplatform support in the arm64 Linux kernel may also allow running `debian-installer` on arm64 systems not explicitly listed above. So long as the kernel used by `debian-installer` has support for the target system's components, and a device-tree file for that target is available, a new target system may work just fine. In these cases, the installer can usually provide a working installation, and so long as UEFI is in use, it should be able to make the system bootable as well. If UEFI is not used you may also need to perform some manual configuration steps to make the system bootable.

2.1.4. Multiple Processors

Multiprocessor support — also called “symmetric multiprocessing” or SMP — is available for this architecture. Having multiple processors in a computer was originally only an issue for high-end server systems but has become common in recent years nearly everywhere with the introduction of so called “multi-core” processors. These contain two or more processor units, called “cores”, in one physical chip.

The standard Ubuntu 18.04 kernel image has been compiled with SMP support. It is also usable on non-SMP systems without problems.

2.1.5. Graphics Hardware Support

Ubuntu’s support for graphical interfaces is determined by the underlying support found in X.Org’s X11 system, and the kernel. Basic framebuffer graphics is provided by the kernel, whilst desktop environments use X11. Whether advanced graphics card features such as 3D-hardware acceleration or hardware-accelerated video are available, depends on the actual graphics hardware used in the system and in some cases on the installation of additional “firmware” images (see Section 2.2).

Nearly all ARM machines have the graphics hardware built-in, rather than being on a plug-in card. Some machines do have expansion slots which will take graphics cards, but that is a rarity. Hardware designed to be headless with no graphics at all is quite common. Whilst basic framebuffer video provided by the kernel should work on all devices that have graphics, fast 3D graphics invariably needs binary drivers to work. The situation is changing quickly but at the time of the bionic release free drivers for nouveau (Nvidia Tegra K1 SoC) and freedreno (Qualcomm Snapdragon SoCs) are available in the release. Other hardware needs non-free drivers from 3rd parties.

Details on supported graphics hardware and pointing devices can be found at http://xorg.freedesktop.org/. Ubuntu 18.04 ships with X.Org version 7.7.

2.1.6. Network Connectivity Hardware

Almost any network interface card (NIC) supported by the Linux kernel should also be supported by the installation system; drivers should normally be loaded automatically.

On 64-bit ARM, most built-in Ethernet devices are supported and modules for additional PCI and USB devices are provided.

2.1.7. Peripherals and Other Hardware

Linux supports a large variety of hardware devices such as mice, printers, scanners, PCMCIA/CardBus/ExpressCard and USB devices. However, most of these devices are not required while installing the system.

2.2. Devices Requiring Firmware

Besides the availability of a device driver, some hardware also requires so-called *firmware* or *microcode* to be loaded into the device before it can become operational. This is most common for network interface cards (especially wireless NICs), but for example some USB devices and even some

hard disk controllers also require firmware. With many graphics cards, basic functionality is available without additional firmware, but the use of advanced features requires an appropriate firmware file to be installed in the system.

On many older devices which require firmware to work, the firmware file was permanently placed in an EEPROM/Flash chip on the device itself by the manufacturer. Nowadays most new devices do not have the firmware embedded this way anymore, so the firmware file must be uploaded into the device by the host operating system every time the system boots.

In most cases firmware is non-free according to the criteria used by the Ubuntu project and thus cannot be included in the main distribution or in the installation system. If the device driver itself is included in the distribution and if Ubuntu legally can distribute the firmware, it will often be available as a separate package from the non-free section of the archive.

However, this does not mean that such hardware cannot be used during an installation. The `debian-installer` supports loading firmware files or packages containing firmware from a removable medium, such as a USB stick. See Section 6.4 for detailed information on how to load firmware files or packages during the installation.

If the `debian-installer` prompts for a firmware file and you do not have this firmware file available or do not want to install a non-free firmware file on your system, you can try to proceed without loading the firmware. There are several cases where a driver prompts for additional firmware because it may be needed under certain circumstances, but the device does work without it on most systems (this e.g. happens with certain network cards using the tg3 driver).

2.3. Purchasing Hardware Specifically for GNU/Linux

There are several vendors, who ship systems with Ubuntu or other distributions of GNU/Linux pre-installed (http://www.debian.org/distrib/pre-installed). You might pay more for the privilege, but it does buy a level of peace of mind, since you can be sure that the hardware is well-supported by GNU/Linux.

Whether or not you are purchasing a system with Linux bundled, or even a used system, it is still important to check that your hardware is supported by the Linux kernel. Check if your hardware is listed in the references found above. Let your salesperson (if any) know that you're shopping for a Linux system. Support Linux-friendly hardware vendors.

2.3.1. Avoid Proprietary or Closed Hardware

Some hardware manufacturers simply won't tell us how to write drivers for their hardware. Others won't allow us access to the documentation without a non-disclosure agreement that would prevent us from releasing the driver's source code, which is one of the central elements of free software. Since we haven't been granted access to usable documentation on these devices, they simply won't work under Linux.

In many cases there are standards (or at least some de-facto standards) describing how an operating system and its device drivers communicate with a certain class of devices. All devices which comply to such a (de-facto-)standard can be used with a single generic device driver and no device-specific drivers are required. With some kinds of hardware (e.g. USB "Human Interface Devices", i.e. keyboards, mice, etc., and USB mass storage devices like USB flash disks and memory card readers) this works very well and practically every device sold in the market is standards-compliant.

In other fields, among them e.g. printers, this is unfortunately not the case. While there are many printers which can be addressed via a small set of (de-facto-)standard control languages and therefore

can be made to work without problems in any operating system, there are quite a few models which only understand proprietary control commands for which no usable documentation is available and therefore either cannot be used at all on free operating systems or can only be used with a vendor-supplied closed-source driver.

Even if there is a vendor-provided closed-source driver for such hardware when purchasing the device, the practical lifespan of the device is limited by driver availability. Nowadays product cycles have become short and it is not uncommon that a short time after a consumer device has ceased production, no driver updates get made available any more by the manufacturer. If the old closed-source driver does not work anymore after a system update, an otherwise perfectly working device becomes unusable due to lacking driver support and there is nothing that can be done in this case. You should therefore avoid buying closed hardware in the first place, regardless of the operating system you want to use it with.

You can help improve this situation by encouraging manufacturers of closed hardware to release the documentation and other resources necessary for us to provide free drivers for their hardware.

2.4. Installation Media

This section will help you determine which different media types you can use to install Ubuntu. There is a whole chapter devoted to media, Chapter 4, which lists the advantages and disadvantages of each media type. You may want to refer back to this page once you reach that section.

2.4.1. CD-ROM/DVD-ROM/BD-ROM

> **Note:** Whenever you see "CD-ROM" in this manual, it applies to all of CD-ROMs, DVD-ROMs and BD-ROMs, because all these technologies are really the same from the operating system's point of view.

CD-ROM based installation is supported for most architectures.

2.4.2. Network

The network can be used during the installation to retrieve files needed for the installation. Whether the network is used or not depends on the installation method you choose and your answers to certain questions that will be asked during the installation. The installation system supports most types of network connections (including PPPoE, but not ISDN or PPP), via either HTTP or FTP. After the installation is completed, you can also configure your system to use ISDN and PPP.

You can also *boot* the installation system over the network without needing any local media like CDs/DVDs or USB sticks. If you already have a netboot-infrastructure available (i.e. you are already running DHCP and TFTP services in your network), this allows an easy and fast deployment of a large number of machines. Setting up the necessary infrastructure requires a certain level of technical experience, so this is not recommended for novice users.

Diskless installation, using network booting from a local area network and NFS-mounting of all local filesystems, is another option.

2.4.3. Hard Disk

Booting the installation system directly from a hard disk is another option for many architectures. This will require some other operating system to load the installer onto the hard disk. This method is only recommended for special cases when no other installation method is available.

2.4.4. Un*x or GNU system

If you are running another Unix-like system, you could use it to install Ubuntu without using the `debian-installer` described in the rest of this manual. This kind of install may be useful for users with otherwise unsupported hardware or on hosts which can't afford downtime. If you are interested in this technique, skip to the Section D.4. This installation method is only recommended for advanced users when no other installation method is available.

2.4.5. Supported Storage Systems

The Ubuntu installer contains a kernel which is built to maximize the number of systems it runs on.

2.5. Memory and Disk Space Requirements

You must have at least 31MB of memory and 680MB of hard disk space to perform a normal installation. Note that these are fairly minimal numbers. For more realistic figures, see Section 3.4.

Installation on systems with less memory or disk space available may be possible but is only advised for experienced users.

Chapter 3. Before Installing Ubuntu

This chapter deals with the preparation for installing Ubuntu before you even boot the installer. This includes backing up your data, gathering information about your hardware, and locating any necessary information.

3.1. Overview of the Installation Process

First, just a note about re-installations. With Ubuntu, a circumstance that will require a complete re-installation of your system is very rare; perhaps mechanical failure of the hard disk would be the most common case.

Many common operating systems may require a complete installation to be performed when critical failures take place or for upgrades to new OS versions. Even if a completely new installation isn't required, often the programs you use must be re-installed to operate properly in the new OS.

Under Ubuntu, it is much more likely that your OS can be repaired rather than replaced if things go wrong. Upgrades never require a wholesale installation; you can always upgrade in-place. And the programs are almost always compatible with successive OS releases. If a new program version requires newer supporting software, the Ubuntu packaging system ensures that all the necessary software is automatically identified and installed. The point is, much effort has been put into avoiding the need for re-installation, so think of it as your very last option. The installer is *not* designed to re-install over an existing system.

Here's a road map for the steps you will take during the installation process.

1. Back up any existing data or documents on the hard disk where you plan to install.
2. Gather information about your computer and any needed documentation, before starting the installation.
3. Create partitionable space for Ubuntu on your hard disk.
4. Locate and/or download the installer software and any specialized driver or firmware files your machine requires.
5. Set up boot media such as CDs/DVDs/USB sticks or provide a network boot infrastructure from which the installer can be booted.
6. Boot the installation system.
7. Select the installation language.
8. Activate the ethernet network connection, if available.
9. Create and mount the partitions on which Ubuntu will be installed.
10. Watch the automatic download/install/setup of the *base system*.
11. Install a *boot loader* which can start up Ubuntu and/or your existing system.
12. Load the newly installed system for the first time.

If you have problems during the installation, it helps to know which packages are involved in which steps. Introducing the leading software actors in this installation drama:

The installer software, `debian-installer`, is the primary concern of this manual. It detects hardware and loads appropriate drivers, uses `dhcp-client` to set up the network connection, runs `debootstrap` to install the base system packages, and runs `tasksel` to allow you to install certain

additional software. Many more actors play smaller parts in this process, but `debian-installer` has completed its task when you load the new system for the first time.

To tune the system to your needs, `tasksel` allows you to choose to install various predefined bundles of software like a Web server or a Desktop environment.

Just be aware that the X Window System is completely separate from `debian-installer`, and in fact is much more complicated. Troubleshooting of the X Window System is not within the scope of this manual.

3.2. Back Up Your Existing Data!

Before you start, make sure to back up every file that is now on your system. If this is the first time a non-native operating system is going to be installed on your computer, it is quite likely you will need to re-partition your disk to make room for Ubuntu. Anytime you partition your disk, you run a risk of losing everything on the disk, no matter what program you use to do it. The programs used in the installation are quite reliable and most have seen years of use; but they are also quite powerful and a false move can cost you. Even after backing up, be careful and think about your answers and actions. Two minutes of thinking can save hours of unnecessary work.

If you are creating a multi-boot system, make sure that you have the distribution media of any other present operating systems on hand. Even though this is normally not necessary, there might be situations in which you could be required to reinstall your operating system's boot loader to make the system boot or in a worst case even have to reinstall the complete operating system and restore your previously made backup.

3.3. Information You Will Need

3.3.1. Documentation

3.3.1.1. Installation Manual

This document you are now reading, in plain ASCII, HTML or PDF format.

- install.en.txt
- install.en.html
- install.en.pdf

3.3.1.2. Hardware documentation

Often contains useful information on configuring or using your hardware.

3.3.2. Finding Sources of Hardware Information

In many cases, the installer will be able to automatically detect your hardware. But to be prepared, we do recommend familiarizing yourself with your hardware before the install.

Hardware information can be gathered from:

- The manuals that come with each piece of hardware.
- The BIOS setup screens of your computer. You can view these screens when you start your computer by pressing a combination of keys. Check your manual for the combination. Often, it is the **Delete** or the **F2** key, but some manufacturers use other keys or key combinations. Usually upon starting the computer there will be a message stating which key to press to enter the setup screen.
- The cases and boxes for each piece of hardware.
- System commands or tools in another operating system, including file manager displays. This source is especially useful for information about RAM and hard drive memory.
- Your system administrator, network administrator or Internet Service Provider. These sources can tell you the settings you need to set up your networking and e-mail.

Table 3-1. Hardware Information Helpful for an Install

Hardware	Information You Might Need
Hard Drives	How many you have.
	Their order on the system.
	Whether IDE (also known as PATA), SATA or SCSI.
	Available free space.
	Partitions.
	Partitions where other operating systems are installed.
Network interfaces	Type/model of available network interfaces.
Printer	Model and manufacturer.
Video Card	Type/model and manufacturer.

3.3.3. Hardware Compatibility

Many products work without trouble on Linux. Moreover, hardware support in Linux is improving daily. However, Linux still does not run as many different types of hardware as some operating systems.

Drivers in Linux in most cases are not written for a certain “product” or “brand” from a specific manufacturer, but for a certain hardware/chipset. Many seemingly different products/brands are based on the same hardware design; it is not uncommon that chip manufacturers provide so-called “reference designs” for products based on their chips which are then used by several different device manufacturers and sold under lots of different product or brand names.

This has advantages and disadvantages. An advantage is that a driver for one chipset works with lots of different products from different manufacturers, as long as their product is based on the same chipset. The disadvantage is that it is not always easy to see which actual chipset is used in a certain product/brand. Unfortunately sometimes device manufacturers change the hardware base of their product without changing the product name or at least the product version number, so that when having two

items of the same brand/product name bought at different times, they can sometimes be based on two different chipsets and therefore use two different drivers or there might be no driver at all for one of them.

For USB and PCI/PCI-Express/ExpressCard devices, a good way to find out on which chipset they are based is to look at their device IDs. All USB/PCI/PCI-Express/ExpressCard devices have so called "vendor" and "product" IDs, and the combination of these two is usually the same for any product based on the same chipset.

On Linux systems, the devices and their IDs can be read using:

- **lsusb** command for USB devices
- **lspci -nn** command for PCI-Express/PCIe devices

The vendor and product IDs are usually given in the form of two hexadecimal numbers, seperated by a colon, such as "1d6b:0001".

An example for the output of **lsusb**:

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Whereby 1d6b is the vendor ID and 0002 is the product ID.

An example for the output of **lspci -nn** for an Ethernet card:

```
03:00.0 Ethernet controller [0200]: Realtek Semiconductor Co., Ltd.
RTL8111/8168B PCI Express Gigabit Ethernet controller [10ec:8168] (rev 06).
```

The IDs are given inside the rightmost square brackets, i.e. here 10ec is the vendor- and 8168 is the product ID.

As another example, a graphics card could give the following output:

```
04:00.0 VGA compatible controller [0300]: Advanced Micro Devices [AMD] nee
ATI RV710 [Radeon HD 4350] [1002:954f].
```

On Windows systems, the IDs for a device can be found in the Windows device manager on the tab "details", where the vendor ID is prefixed with VEN_ and the product ID is prefixed with DEV_. On Windows 7 systems, you have to select the property "Hardware IDs" in the device manager's details tab to actually see the IDs, as they are not displayed by default.

Searching on the internet with the vendor/product ID, "Linux" and "driver" as the search terms often results in information regarding the driver support status for a certain chipset. If a search for the vendor/product ID does not yield usable results, a search for the chip code names, which are also often provided by lsusb and lspci ("RTL8111"/"RTL8168B" in the network card example and "RV710" in the graphics card example), can help.

3.3.3.1. Testing hardware compatibility with a Live-System

Ubuntu is also available as a so-called "live system" for certain architectures. A live system is a preconfigured ready-to-use system in a compressed format that can be booted and used from a read-only medium like a CD, DVD or USB drive. Using it by default does not create any permanent changes on your computer. You can change user settings and install additional programs from within the live system, but all this only happens in the computer's RAM, i.e. if you turn off the computer and boot the live system again or if you restart the system, everything is reset to its defaults. If you want to see whether your hardware is supported by Ubuntu, the easiest way is to run a Ubuntu live system on it and try it out.

There are a few limitations in using a live system. The first is that as all changes you do within the live system must be held in your computer's RAM, this only works on systems with enough RAM

to do that, so installing additional large software packages may fail due to memory constraints. Another limitation with regards to hardware compatibility testing is that the official Ubuntu live system contains only free components, i.e. there are no non-free firmware files included in it. Such non-free packages can of course be installed manually within the system, but there is no automatic detection of required firmware files like in the `debian-installer`, so installation of non-free components must be done manually if needed.

Information about the available variants of the Ubuntu live images can be found at the download web page (http://www.ubuntu.com/download/).

3.3.4. Network Settings

If your computer is connected to a fixed network (i.e. an Ethernet or equivalent connection— not a dialup/PPP connection) which is administered by somebody else, you should ask your network's system administrator for this information:

- Your host name (you may be able to decide this on your own).
- Your domain name.
- Your computer's IP address.
- The netmask to use with your network.
- The IP address of the default gateway system you should route to, if your network *has* a gateway.
- The system on your network that you should use as a DNS (Domain Name Service) server.
- If the system is connected to a VLAN trunk port the VLAN Id is needed.

  To display the VLAN configuration settings, the debian-installer's debconf priority needs to be switched from high (default) to medium.

If the network you are connected to uses DHCP (Dynamic Host Configuration Protocol) for configuring network settings, you don't need this information because the DHCP server will provide it directly to your computer during the installation process.

If you have internet access via DSL or cable modem (i.e. over a cable tv network) and have a router (often provided preconfigured by your phone or catv provider) which handles your network connectivity, DHCP is usually available by default.

If you use a WLAN/WiFi network, you should find out:

- The ESSID ("network name") of your wireless network.
- The WEP or WPA/WPA2 security key to access the network (if applicable).

3.4. Meeting Minimum Hardware Requirements

Once you have gathered information about your computer's hardware, check that your hardware will let you do the type of installation that you want to do.

Depending on your needs, you might manage with less than some of the recommended hardware listed in the table below. However, most users risk being frustrated if they ignore these suggestions.

Table 3-2. Recommended Minimum System Requirements

Install Type	RAM (minimum)	RAM (recommended)	Hard Drive
No desktop	128 megabytes	512 megabytes	2 gigabytes
With Desktop	256 megabytes	1 gigabyte	10 gigabytes

The actual minimum memory requirements may be less than the numbers listed in this table. Depending on the architecture, it is possible to install Ubuntu with as little as 60MB (for amd64). The same goes for the disk space requirements, especially if you pick and choose which applications to install; see Section D.3 for additional information on disk space requirements.

It is possible to run a graphical desktop environment on older or low-end systems, but in that case it is recommended to install a window manager that is less resource-hungry than those of the GNOME or KDE desktop environments; alternatives include `xfce4`, `icewm` and `wmaker`, but there are others to choose from.

It is practically impossible to give general memory or disk space requirements for server installations as those very much depend on what the server is to be used for.

Remember that these sizes don't include all the other materials which are usually to be found, such as user files, mail, and data. It is always best to be generous when considering the space for your own files and data.

Disk space required for the smooth operation of the Ubuntu system itself is taken into account in these recommended system requirements. Notably, the `/var` partition contains a lot of state information specific to Ubuntu in addition to its regular contents, like logfiles. The **dpkg** files (with information on all installed packages) can easily consume 40MB. Also, **apt-get** puts downloaded packages here before they are installed. You should usually allocate at least 200MB for `/var`.

3.5. Pre-Partitioning for Multi-Boot Systems

Partitioning your disk simply refers to the act of breaking up your disk into sections. Each section is then independent of the others. It's roughly equivalent to putting up walls inside a house; if you add furniture to one room it doesn't affect any other room.

If you already have an operating system on your system which uses the whole disk and you want to stick Ubuntu on the same disk, you will need to repartition it. Ubuntu requires its own hard disk partitions. It cannot be installed on Windows or Mac OS X partitions. It may be able to share some partitions with other Unix systems, but that's not covered here. At the very least you will need a dedicated partition for the Ubuntu root filesystem.

You can find information about your current partition setup by using a partitioning tool for your current operating system. Partitioning tools always provide a way to show existing partitions without making changes.

In general, changing a partition with a file system already on it will destroy any information there. Thus you should always make backups before doing any repartitioning. Using the analogy of the house, you would probably want to move all the furniture out of the way before moving a wall or you risk destroying it.

Several modern operating systems offer the ability to move and resize certain existing partitions without destroying their contents. This allows making space for additional partitions without losing existing data. Even though this works quite well in most cases, making changes to the partitioning of a disk is an inherently dangerous action and should only be done after having made a full backup of all data.

Creating and deleting partitions can be done from within `debian-installer` as well as from an existing operating system. As a rule of thumb, partitions should be created by the system for which they are to be used, i.e. partitions to be used by Ubuntu should be created from within `debian-installer` and partitions to be used from another operating system should be created from there. `debian-installer` is capable of creating non-Linux partitions, and partitions created this way usually work without problems when used in other operating systems, but there are a few rare corner cases in which this could cause problems, so if you want to be sure, use the native partitioning tools to create partitions for use by other operating systems.

If you are going to install more than one operating system on the same machine, you should install all other system(s) before proceeding with the Ubuntu installation.Windows and other OS installations may destroy your ability to start Ubuntu, or encourage you to reformat non-native partitions.

You can recover from these actions or avoid them, but installing the native system first saves you trouble.

3.6. Pre-Installation Hardware and Operating System Setup

This section will walk you through pre-installation hardware setup, if any, that you will need to do prior to installing Ubuntu. Generally, this involves checking and possibly changing BIOS/system firmware settings for your system. The “BIOS” or “system firmware” is the core software used by the hardware; it is most critically invoked during the bootstrap process (after power-up).

3.6.1. Systems with U-Boot firmware

As already mentioned before, there is unfortunately no standard for system firmware on ARM systems. Even the behaviour of different systems which use nominally the same firmware can be quite different. This results from the fact that a large part of the devices using the ARM architecture are embedded systems, for which the manufacturers usually build heavily customized firmware versions and include device-specific patches. Unfortunately the manufacturers often do not submit their changes and extensions back to the mainline firmware developers, so their changes are not integrated into newer versions of the original firmware.

As a result even newly sold systems often use a firmware that is based on a years-old manufacturer-modified version of a firmware whose mainline codebase has evolved a lot further in the meantime and offers additional features or shows different behaviour in certain aspects. In addition to that, the naming of onboard devices is not consistent between different manufacturer-modified versions of the same firmware, therefore it is nearly impossible to provide usable product-independend instructions for ARM-based systems.

3.6.1.1. Setting the ethernet MAC address in U-Boot

The MAC address of every ethernet interface should normally be globally unique, and it technically has to be unique within its ethernet broadcast domain. To achieve this, the manufacturer usually

allocates a block of MAC addresses from a centrally-administered pool (for which a fee has to be paid) and preconfigures one of these addresses on each item sold.

In the case of development boards, sometimes the manufacturer wants to avoid paying these fees and therefore provides no globally unique addresses. In these cases the users themselves have to define MAC addresses for their systems. When no MAC address is defined for an ethernet interface, some network drivers generate a random MAC address that can change on every boot, and if this happens, network access would be possible even when the user has not manually set an address, but e.g. assigning semi-static IP addresses by DHCP based on the MAC address of the requesting client would obviously not work reliably.

To avoid conflicts with existing officially-assigned MAC addresses, there is an address pool which is reserved for so-called "locally administered" addresses. It is defined by the value of two specific bits in the first byte of the address (the article "MAC address" in the English language Wikipedia gives a good explanation). In practice this means that e.g. any address starting with hexadecimal ca (such as ca:ff:ee:12:34:56) can be used as a locally administered address.

On systems using U-Boot as system firmware, the ethernet MAC address is placed in the "ethaddr" environment variable. It can be checked at the U-Boot command prompt with the command "printenv ethaddr" and can be set with the command "setenv ethaddr ca:ff:ee:12:34:56". After setting the value, the command "saveenv" makes the assignment permanent.

3.6.1.2. Kernel/Initrd/Device-Tree relocation issues in U-Boot

On some systems with older U-Boot versions there can be problems with properly relocating the Linux kernel, the initial ramdisk and the device-tree blob in memory during the boot process. In this case, U-Boot shows the message "Starting kernel ...", but the system freezes afterwards without further output. These issues have been solved with newer U-Boot versions from v2014.07 onwards.

If the system has originally used a U-Boot version older than v2014.07 and has been upgraded to a newer version later, the problem might still occur even after upgrading U-Boot. Upgrading U-Boot usually does not modify the existing U-Boot environment variables and the fix requires an additional environment variable (bootm_size) to be set, which U-Boot does automatically only on fresh installations without existing environment data. It is possible to manually set bootm_size to the new U-Boot's default value by running the command "env default bootm_size; saveenv" at the U-Boot prompt.

Another possibility to circumvent relocation-related problems is to run the command "setenv fdt_high ffffffff; setenv initrd_high 0xffffffff; saveenv" at the U-Boot prompt to completely disable the relocation of the initial ramdisk and the device-tree blob.

3.6.2. Systems with UEFI firmware

UEFI ("Unified Extensible Firmware Interface") is a new kind of system firmware that is used on many modern systems and is - among other uses - intended to replace the classic PC BIOS.

Currently most PC systems that use UEFI also have a so-called "Compatibility Support Module" (CSM) in the firmware, which provides excatly the same interfaces to an operating system as a classic PC BIOS, so that software written for the classic PC BIOS can be used unchanged. Nonetheless UEFI is intended to one day completely replace the old PC BIOS without being fully backwards-compatible and there are already a lot of systems with UEFI but without CSM.

On systems with UEFI there are a few things to take into consideration when installing an operating system. The way the firmware loads an operating system is fundamentally different between the classic BIOS (or UEFI in CSM mode) and native UEFI. One major difference is the way the harddisk

partitions are recorded on the harddisk. While the classic BIOS and UEFI in CSM mode use a DOS partition table, native UEFI uses a different partitioning scheme called “GUID Partition Table” (GPT). On a single disk, for all practical purposes only one of the two can be used and in case of a multi-boot setup with different operating systems on one disk, all of them must therefore use the same type of partition table. Booting from a disk with GPT is only possible in native UEFI mode, but using GPT becomes more and more common as hard disk sizes grow, because the classic DOS partition table cannot address disks larger than about 2 Terabytes while GPT allows for far larger disks. The other major difference between BIOS (or UEFI in CSM mode) and native UEFI is the location where boot code is stored and in which format it has to be. This means that different bootloaders are needed for each system.

The latter becomes important when booting `debian-installer` on a UEFI system with CSM because `debian-installer` checks whether it was started on a BIOS- or on a native UEFI system and installs the corresponding bootloader. Normally this simply works but there can be a problem in multi-boot environments. On some UEFI systems with CSM the default boot mode for removable devices can be different from what is actually used when booting from hard disk, so when booting the installer from a USB stick in a different mode from what is used when booting another already installed operating system from the hard disk, the wrong bootloader might be installed and the system might be unbootable after finishing the installation. When choosing the boot device from a firmware boot menu, some systems offer two seperate choices for each device, so that the user can select whether booting shall happen in CSM or in native UEFI mode.

Chapter 4. Obtaining System Installation Media

4.1. Official Ubuntu CD-ROMs

By far the easiest way to install Ubuntu is from an Official Ubuntu CD-ROM (http://releases.ubuntu.com/bionic/) . You may download the CD-ROM image from an Ubuntu mirror and make your own CD, if you have a fast network connection and a CD burner. If you have an Ubuntu CD and CDs are bootable on your machine , you can skip right to Chapter 5; much effort has been expended to ensure the files most people need are there on the CD.

If your machine doesn't support CD booting, but you do have a CD or an ISO image, you can use an alternative strategy such as net boot, or manually loading the kernel from the CD to initially boot the system installer. The files you need for booting by another means are also on the CD; the Ubuntu network archive and CD folder organization are identical. So when archive file paths are given below for particular files you need for booting, look for those files in the same directories and subdirectories on your CD.

Once the installer is booted, it will be able to obtain all the other files it needs from the CD.

If you don't have a CD, then you will need to download the installer system files and place them on a a connected computer, so they can be used to find and boot the installer.

4.2. Downloading Files from Ubuntu Mirrors

To find the nearest (and thus probably the fastest) mirror, see the list of Ubuntu mirrors (http://wiki.ubuntu.com/Archive).

When downloading files from an Ubuntu mirror using FTP, be sure to download the files in *binary* mode, not text or automatic mode.

4.2.1. Where to Find Installation Images

The installation images are located on each Ubuntu mirror in the directory ubuntu/dists/bionic/main/installer-arm64/current/images/ (http://ports.ubuntu.com/ubuntu-ports/dists/bionic/main/installer-arm64/current/images)
— the MANIFEST (http://ports.ubuntu.com/ubuntu-ports/dists/bionic/main/installer-arm64/current/images/MANIFEST) lists each image and its purpose.

4.3. Preparing Files for TFTP Net Booting

If your machine is connected to a local area network, you may be able to boot it over the network from another machine, using TFTP. If you intend to boot the installation system from another machine, the boot files will need to be placed in specific locations on that machine, and the machine configured to support booting of your specific machine.

You need to set up a TFTP server, and for many machines a DHCP server, or RARP server, or BOOTP server.

The Reverse Address Resolution Protocol (RARP) is one way to tell your client what IP address to use for itself. Another way is to use the BOOTP protocol. BOOTP is an IP protocol that informs a computer of its IP address and where on the network to obtain a boot image. The DHCP (Dynamic Host Configuration Protocol) is a more flexible, backwards-compatible extension of BOOTP. Some systems can only be configured via DHCP.

The Trivial File Transfer Protocol (TFTP) is used to serve the boot image to the client. Theoretically, any server, on any platform, which implements these protocols, may be used. In the examples in this section, we shall provide commands for SunOS 4.x, SunOS 5.x (a.k.a. Solaris), and GNU/Linux.

4.3.1. Setting up RARP server

To set up RARP, you need to know the Ethernet address (a.k.a. the MAC address) of the client computers to be installed. If you don't know this information, you can boot into "Rescue" mode (e.g., from the rescue floppy) and use the command **`ip addr show dev eth0`**.

On a RARP server system using a Linux kernel or Solaris/SunOS, you use the **rarpd** program. You need to ensure that the Ethernet hardware address for the client is listed in the "ethers" database (either in the `/etc/ethers` file, or via NIS/NIS+) and in the "hosts" database. Then you need to start the RARP daemon. Issue the command (as root): **`/usr/sbin/rarpd -a`** on most Linux systems and SunOS 5 (Solaris 2), **`/usr/sbin/in.rarpd -a`** on some other Linux systems, or **`/usr/etc/rarpd -a`** in SunOS 4 (Solaris 1).

4.3.2. Setting up a DHCP server

One free software DHCP server is ISC **dhcpd**. For Ubuntu, the `isc-dhcp-server` package is recommended. Here is a sample configuration file for it (see `/etc/dhcp/dhcpd.conf`):

```
option domain-name "example.com";
option domain-name-servers ns1.example.com;
option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;
server-name "servername";

subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.200 192.168.1.253;
  option routers 192.168.1.1;
}

host clientname {
  filename "/tftpboot.img";
  server-name "servername";
  next-server servername;
  hardware ethernet 01:23:45:67:89:AB;
  fixed-address 192.168.1.90;
}
```

In this example, there is one server *servername* which performs all of the work of DHCP server, TFTP server, and network gateway. You will almost certainly need to change the domain-name options, as well as the server name and client hardware address. The *filename* option should be the name of the file which will be retrieved via TFTP.

After you have edited the **dhcpd** configuration file, restart it with **/etc/init.d/isc-dhcp-server restart**.

4.3.3. Setting up a BOOTP server

There are two BOOTP servers available for GNU/Linux. The first is CMU **bootpd**. The other is actually a DHCP server: ISC **dhcpd**. In Ubuntu these are contained in the `bootp` and `isc-dhcp-server` packages respectively.

To use CMU **bootpd**, you must first uncomment (or add) the relevant line in `/etc/inetd.conf`. On Debian or Ubuntu, you can run **update-inetd --enable bootps**, then **/etc/init.d/inetd reload** to do so. Just in case your BOOTP server does not run Debian or Ubuntu, the line in question should look like:

```
bootps  dgram  udp  wait  root  /usr/sbin/bootpd  bootpd -i -t 120
```

Now, you must create an `/etc/bootptab` file. This has the same sort of familiar and cryptic format as the good old BSD `printcap`, `termcap`, and `disktab` files. See the `bootptab` manual page for more information. For CMU **bootpd**, you will need to know the hardware (MAC) address of the client. Here is an example `/etc/bootptab`:

```
client:\
  hd=/tftpboot:\
  bf=tftpboot.img:\
  ip=192.168.1.90:\
  sm=255.255.255.0:\
  sa=192.168.1.1:\
  ha=0123456789AB:
```

You will need to change at least the “ha” option, which specifies the hardware address of the client. The “bf” option specifies the file a client should retrieve via TFTP; see Section 4.3.5 for more details.

By contrast, setting up BOOTP with ISC **dhcpd** is really easy, because it treats BOOTP clients as a moderately special case of DHCP clients. Some architectures require a complex configuration for booting clients via BOOTP. If yours is one of those, read the section Section 4.3.2. Otherwise you will probably be able to get away with simply adding the **allow bootp** directive to the configuration block for the subnet containing the client in `/etc/dhcp/dhcpd.conf`, and restart **dhcpd** with **/etc/init.d/isc-dhcp-server restart**.

4.3.4. Enabling the TFTP Server

To get the TFTP server ready to go, you should first make sure that **tftpd** is enabled.

In the case of `tftpd-hpa` there are two ways the service can be run. It can be started on demand by the system’s `inetd` daemon, or it can be set up to run as an independent daemon. Which of these methods is used is selected when the package is installed and can be changed by reconfiguring the package.

Note: Historically, TFTP servers used `/tftpboot` as directory to serve images from. However, Ubuntu packages may use other directories to comply with the Filesystem Hierarchy Standard (http://www.pathname.com/fhs/). For example, `tftpd-hpa` by default uses `/srv/tftp`. You may have to adjust the configuration examples in this section accordingly.

All **in.tftpd** alternatives available in Ubuntu should log TFTP requests to the system logs by default. Some of them support a **-v** argument to increase verbosity. It is recommended to check these log messages in case of boot problems as they are a good starting point for diagnosing the cause of errors.

4.3.5. Move TFTP Images Into Place

Next, place the TFTP boot image you need, as found in Section 4.2.1, in the **tftpd** boot image directory. You may have to make a link from that file to the file which **tftpd** will use for booting a particular client. Unfortunately, the file name is determined by the TFTP client, and there are no strong standards.

4.4. Automatic Installation

For unattended installs on multiple computers it's possible to do fully automatic installations using the Ubuntu Installer itself.

4.4.1. Automatic Installation Using the Ubuntu Installer

The Ubuntu Installer supports automating installs via preconfiguration files. A preconfiguration file can be loaded from the network or from removable media, and used to fill in answers to questions asked during the installation process.

Full documentation on preseeding including a working example that you can edit is in Appendix B.

4.4.2. Automatic Installation Using Kickstart

The Ubuntu installer supports automating installs using Kickstart files, as designed by Red Hat for use in their Anaconda installer. This method is not as flexible as the preconfiguration file method above, but it requires less knowledge of how the installer works.

This section documents only the basics, and differences between Anaconda and the Ubuntu installer. Refer to the Red Hat documentation (http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/ch-kickstart2.html) for detailed instructions.

To generate a Kickstart file, install the `system-config-kickstart` package and run `system-config-kickstart`. This offers you a graphical user interface to the various options available.

Once you have a Kickstart file, you can edit it if necessary, and place it on a web, FTP, or NFS server, or copy it onto the installer's boot media. Wherever you place the file, you need to pass a parameter to the installer at boot time to tell it to use the file.

To make the installer use a Kickstart file downloaded from a web or FTP server, add ks=http://url/to/ks.cfg or ks=ftp://url/to/ks.cfg respectively to the kernel boot parameters. This requires the installer to be able to set up the network via DHCP on the first connected interface without asking any questions; you may also need to add ksdevice=eth1 or similar if the installer fails to determine the correct interface automatically.

Similarly, to make the installer use a Kickstart file on an NFS server, add ks=nfs:server:/path/to/ks.cfg to the kernel boot parameters. The method supported by Anaconda of adding a plain "ks" boot parameter to work out the location of the Kickstart file from a DHCP response is not yet supported by the Ubuntu installer.

To place a Kickstart file on a CD, you would need to remaster the ISO image to include your Kickstart file, and add ks=cdrom:/path/to/ks.cfg to the kernel boot parameters. See the manual page for mkisofs for details.

4.4.2.1. Additions

The Ubuntu installer supports a few extensions to Kickstart that were needed to support automatic installations of Ubuntu:

- The **rootpw** command now takes the **--disabled** option to disable the root password. If this is used, the initial user will be given root privileges via `sudo`.
- A new **user** command has been added to control the creation of the initial user:

  ```
  user joe --fullname "Joe User" --password iamjoe
  ```

  The **--disabled** option prevents any non-root users from being created. The **--fullname** option specifies the user's full name, as opposed to the Unix username. The **--password** option supplies the user's password, by default in the clear (in which case make sure your Kickstart file is kept confidential!); the **--iscrypted** option may be used to state that the password is already MD5-hashed.
- A new **preseed** command has been added to provide a convenient way to preseed additional items in the debconf database that are not directly accessible using the ordinary Kickstart syntax:

  ```
  preseed --owner gdm shared/default-x-display-manager select gdm
  ```

  Note that if the value contains any special characters, then the value must be quoted, as follows:

  ```
  preseed preseed/late_command string "sed -i 's/foo/bar/g' /target/etc/hosts"
  ```

  The **--owner** option sets the name of the package that owns the question; if omitted, it defaults to d-i, which is generally appropriate for items affecting the first stage of the installer. The three mandatory arguments are the question name, question type, and answer, in that order, just as would be supplied as input to the `debconf-set-selections` command.
- As of Ubuntu 6.10, the **keyboard** option takes X layout names. To use an X keyboard variant, set this option to **layout_variant**, with appropriate values of **layout** and **variant**. For example, **in_guj** selects the Gujarati variant of the Indian layout.
- You may use the **apt-install** command to install packages in **%post --nochroot** scripts (although you might also choose to generate a **%packages** section in a **%pre** script and include it using **%include**). Note that this does not work if the post-installation script is run in the chroot environment.

4.4.2.2. Missing features

As yet, the Ubuntu installer only supports a subset of Kickstart's features. The following is a brief summary of features that are known to be missing:

- LDAP, Kerberos 5, Hesiod, and Samba authentication.
- The **`auth --enablecache`** command to enable `nscd`.
- Upgrades. To upgrade from one Ubuntu release to another, use the facilities provided by `apt` and its frontends.
- Partitioning of multiple drives. Due to current limitations in the partition manager, it is only possible to partition a single drive.
- Using the **`device`** command to install extra kernel modules.
- Driver disks.
- Firewall configuration.
- Installation from an archive on a local hard disk or from an NFS archive.
- The **`logvol --percent`**, **`--bytes-per-inode`**, and **`--fsoptions`** options for certain kinds of detailed Logical Volume Management (LVM) configuration. (LVM configuration in general is experimentally supported as of Ubuntu 9.04; please let us know about your experiences with it.)
- Restrictions of a partition to a particular disk or device, and specifications of the starting or ending cylinder for a partition.
- Checking a partition for bad sectors.
- RAID configuration.
- Exclusions in %packages sections are no longer supported as of Ubuntu 6.10, as a casualty of other improvements. You may need to use a %post script instead to remove unnecessary packages.
- Pre-installation scripts and non-chrooted post-installation scripts may only be shell scripts; other interpreters are not available at this point in the installation.

4.4.2.3. Example

Here is an example Kickstart file that can be used as a starting point:

```
#
#Generic Kickstart template for Ubuntu
#Platform: x86 and x86-64
#

#System language
lang en_US

#Language modules to install
langsupport en_US

#System keyboard
```

```
keyboard us

#System mouse
mouse

#System timezone
timezone America/New_York

#Root password
rootpw --disabled

#Initial user (user with sudo capabilities)
user ubuntu --fullname "Ubuntu User" --password root4me2

#Reboot after installation
reboot

#Use text mode install
text

#Install OS instead of upgrade
install

#Installation media
cdrom
#nfs --server=server.com --dir=/path/to/ubuntu/
#url --url http://server.com/path/to/ubuntu/
#url --url ftp://server.com/path/to/ubuntu/

#System bootloader configuration
bootloader --location=mbr

#Clear the Master Boot Record
zerombr yes

#Partition clearing information
clearpart --all --initlabel

#Basic disk partition
part / --fstype ext4 --size 1 --grow --asprimary
part swap --size 1024
part /boot --fstype ext4 --size 256 --asprimary

#Advanced partition
#part /boot --fstype=ext4 --size=500 --asprimary
#part pv.aQcByA-UM0N-siuB-Y96L-rmd3-n6vz-NMo8Vr --grow --size=1
#volgroup vg_mygroup --pesize=4096 pv.aQcByA-UM0N-siuB-Y96L-rmd3-n6vz-NMo8Vr
#logvol / --fstype=ext4 --name=lv_root --vgname=vg_mygroup --grow --size=10240 \
--maxsize=20480
#logvol swap --name=lv_swap --vgname=vg_mygroup --grow --size=1024 --maxsize=8192

#System authorization infomation
auth  --useshadow  --enablemd5

#Network information
network --bootproto=dhcp --device=eth0
```

```
#Firewall configuration
firewall --disabled --trust=eth0 --ssh

#Do not configure the X Window System
skipx
```

Chapter 5. Booting the Installation System

5.1. Booting the Installer on 64-bit ARM

5.1.1. Console configuration

The graphical installer is not enabled on the arm64 `debian-installer` images for 18.04 so the serial console is used. The console device should be detected automatically from the firmware, but if it is not then after you boot linux from the GRUB menu you will see a "Booting Linux" message, then nothing more.

If you hit this issue you will need to set a specific console config on the kernel command line. Hit **e** for "Edit Kernel command-line" at the GRUB menu, and change

```
--- quiet
```

to

```
console=<device>,<speed>
```

e.g.

```
console=ttyAMA0,115200n8
```

. When finished hit **Control**-**x** to continue booting with new setting.

5.1.2. Juno Installation

Juno has UEFI so the install is straightforward. The most practical method is installing from USB stick. You need up to date firmware for USB-booting to work. Builds from http://releases.linaro.org/latest/members/arm/ after March 2015 tested OK. Consult Juno documentation on firmware updating.

Prepare a standard arm64 CD image on a USB stick. Insert it in one of the USB ports on the back. Plug a serial cable into the upper 9-pin serial port on the back. If you need networking (netboot image) plug the ethernet cable into the socket on the front of the machine.

Run a serial console at 115200, 8bit no parity, and boot the Juno. It should boot from the USB stick to a GRUB menu. The console config is not correctly detected on Juno so just hitting return will show no kernel output. Set the console to

```
console=ttyAMA0,115200n8
```

as described in (Section 5.1.1). **Control**-**x** to boot should show you the `debian-installer` screens, and allow you to proceed with a standard installation.

5.1.3. Applied Micro Mustang Installation

UEFI is available for this machine but it is normally shipped with U-Boot so you will need to either install UEFI firmware first then use standard boot/install methods, or use U-Boot boot methods. Also USB is not supported in the jessie kernel so installing from a USB stick does not work. You must use a serial console to control the installation because the graphical installer is not enabled on the arm64 architecture.

The recommended install method is to copy the `debian-installer` kernel and initrd onto the hard drive, using the openembedded system supplied with the machine, then boot from that to run the installer. Alternatively use TFTP to get the kernel/dtb/initrd copied over and booted (Section 5.1.4.1). After installation, manual changes to boot from the installed image are needed.

Run a serial console at 115200, 8bit no parity, and boot the machine. Reboot the machine and when you see “Hit any key to stop autoboot:” hit a key to get a Mustang# prompt. Then use U-Boot commands to load and boot the kernel, dtb and initrd.

5.1.4. Booting by TFTP

Booting from the network requires that you have a network connection and a TFTP network boot server (and probably also a DHCP, RARP, or BOOTP server for automatic network configuration).

The server-side setup to support network booting is described in Section 4.3.

5.1.4.1. TFTP-booting in U-Boot

Network booting on systems using the U-Boot firmware consists of three steps: a) configuring the network, b) loading the images (kernel/initial ramdisk/dtb) into memory and c) actually executing the previosly loaded code.

First you have to configure the network, either automatically via DHCP by running

```
setenv autoload no
dhcp
```

or manually by setting several environment variables

```
setenv ipaddr <ip address of the client>
setenv netmask <netmask>
setenv serverip <ip address of the tftp server>
setenv dnsip <ip address of the nameserver>
setenv gatewayip <ip address of the default gateway>
```

If you prefer, you can make these settings permanent by running

```
saveenv
```

Afterwards you need to load the images (kernel/initial ramdisk/dtb) into memory. This is done with the tftpboot command, which has to be provided with the address at which the image shall be stored in memory. Unfortunately the memory map can vary from system to system, so there is no general rule which addresses can be used for this.

On some systems, U-Boot predefines a set of environment variables with suitable load addresses: kernel_addr_r, ramdisk_addr_r and fdt_addr_r. You can check whether they are defined by running

```
printenv kernel_addr_r ramdisk_addr_r fdt_addr_r
```

If they are not defined, you have to check your system's documentation for appropriate values and set them manually. For systems based on Allwinner SunXi SOCs (e.g. the Allwinner A10, architecture name "sun4i" or the Allwinner A20, architecture name "sun7i"), you can e.g. use the follwing values:

```
setenv kernel_addr_r 0x46000000
setenv fdt_addr_r 0x47000000
setenv ramdisk_addr_r 0x48000000
```

When the load addresses are defined, you can load the images into memory from the previously defined tftp server with

```
tftpboot ${kernel_addr_r} <filename of the kernel image>
tftpboot ${fdt_addr_r} <filename of the dtb>
tftpboot ${ramdisk_addr_r} <filename of the initial ramdisk image>
```

The third part is setting the kernel commandline and actually executing the loaded code. U-Boot passes the content of the "bootargs" environment variable as commandline to the kernel, so any parameters for the kernel and the installer - such as the console device (see Section 5.3.1) or preseeding options (see Section 5.3.2 and Appendix B) - can be set with a command like

```
setenv bootargs console=ttyS0,115200 rootwait panic=10
```

The exact command to execute the previously loaded code depends on the image format used. With uImage/uInitrd, the command is

```
bootm ${kernel_addr_r} ${ramdisk_addr_r} ${fdt_addr_r}
```

and with native Linux images it is

```
bootz ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Note: When booting standard linux images, it is important to load the initial ramdisk image after the kernel and the dtb as U-Boot sets the filesize variable to the size of the last file loaded and the bootz command requires the size of the ramdisk image to work correctly. In case of booting a platform-specific kernel, i.e. a kernel without device-tree, simply omit the ${fdt_addr_r} parameter.

5.2. Accessibility

Some users may need specific support because of e.g. some visual impairment. Most accessibility features have to be enabled manually. Some boot parameters can be appended to enable accessibility features. Note that on most architectures the boot loader interprets your keyboard as a QWERTY keyboard.

5.2.1. Installer front-end

The Ubuntu installer supports several front-ends for asking questions, with varying convenience for accessibility: notably, **`text`** uses plain text while **`newt`** uses text-based dialog boxes. The choice can be made at the boot prompt, see the documentation for **`DEBIAN_FRONTEND`** in Section 5.3.2.

5.2.2. Board Devices

Some accessibility devices are actual boards that are plugged inside the machine and that read text directly from the video memory. To get them to work framebuffer support must be disabled by using the **`fb=false`** boot parameter. This will however reduce the number of available languages.

5.2.3. High-Contrast Theme

For users with low vision, the installer can use a high-contrast color theme that makes it more readable. To enable it, append the **`theme=dark`** boot parameter.

5.2.4. Zoom

For users with low vision, the graphical installer has a very basic zoom support: the **Control**-**+** and **Control**-**-** shortcuts increase and decrease the font size.

5.2.5. Preseeding

Alternatively, Ubuntu can be installed completely automatically by using preseeding. This is documented in Appendix B.

5.2.6. Accessibility of the installed system

Documentation on accessibility of the installed system is available on the Debian Accessibility wiki page (http://wiki.debian.org/accessibility).

5.3. Boot Parameters

Boot parameters are Linux kernel parameters which are generally used to make sure that peripherals are dealt with properly. For the most part, the kernel can auto-detect information about your peripherals. However, in some cases you'll have to help the kernel a bit.

If this is the first time you're booting the system, try the default boot parameters (i.e., don't try setting parameters) and see if it works correctly. It probably will. If not, you can reboot later and look for any special parameters that inform the system about your hardware.

Information on many boot parameters can be found in the Linux BootPrompt HOWTO (http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html), including tips for obscure hardware. This section contains only a sketch of the most salient parameters. Some common gotchas are included below in Section 5.4.

5.3.1. Boot console

If you are booting with a serial console, generally the kernel will autodetect this. If you have a video-card (framebuffer) and a keyboard also attached to the computer which you wish to boot via serial console, you may have to pass the **console=*device*** argument to the kernel, where *device* is your serial device, which is usually something like `ttyS0`.

You may need to specify parameters for the serial port, such as speed and parity, for instance **console=ttyS0,9600n8**; other typical speeds may be 57600 or 115200. Be sure to specify this option after "---", so that it is copied into the bootloader configuration for the installed system (if supported by the installer for the bootloader).

In order to ensure the terminal type used by the installer matches your terminal emulator, the parameter **TERM=*type*** can be added. Note that the installer only supports the following terminal types: `linux`, `bterm`, `ansi`, `vt102` and `dumb`. The default for serial console in `debian-installer` is **vt102**. If you are using a virtualization tool which does not provide conversion into such terminals types itself, e.g. QEMU/KVM, you can start it inside a **screen** session. That will indeed perform translation into the `screen` terminal type, which is very close to `vt102`.

5.3.2. Ubuntu Installer Parameters

The installation system recognizes a few additional boot parameters[1] which may be useful.

A number of parameters have a "short form" (or alias) that helps avoid the limitations of the kernel command line options and makes entering the parameters easier. If a parameter has a short form, it will be listed in brackets behind the (normal) long form. Examples in this manual will normally use the short form too.

debconf/priority (priority)

This parameter sets the lowest priority of messages to be displayed.

The default installation uses **priority=high**. This means that both high and critical priority messages are shown, but medium and low priority messages are skipped. If problems are encountered, the installer adjusts the priority as needed.

If you add **priority=medium** as boot parameter, you will be shown the installation menu and gain more control over the installation. When **priority=low** is used, all messages are shown (this is equivalent to the *expert* boot method). With **priority=critical**, the installation system will display only critical messages and try to do the right thing without fuss.

Note: In order to get asked for a VLAN configuration during the network setup a priority of **medium** or **low** is needed.

1. With current kernels (2.6.9 or newer) you can use 32 command line options and 32 environment options. If these numbers are exceeded, the kernel will panic.

DEBIAN_FRONTEND

This boot parameter controls the type of user interface used for the installer. The current possible parameter settings are:

- **DEBIAN_FRONTEND=noninteractive**
- **DEBIAN_FRONTEND=text**
- **DEBIAN_FRONTEND=newt**
- **DEBIAN_FRONTEND=gtk**

The default frontend is **DEBIAN_FRONTEND=newt**. **DEBIAN_FRONTEND=text** may be preferable for serial console installs . Some specialized types of install media may only offer a limited selection of frontends, but the **newt** and **text** frontends are available on most default install media. On architectures that support it, the graphical installer uses the **gtk** frontend.

BOOT_DEBUG

Setting this boot parameter to 2 will cause the installer's boot process to be verbosely logged. Setting it to 3 makes debug shells available at strategic points in the boot process. (Exit the shells to continue the boot process.)

BOOT_DEBUG=0

This is the default.

BOOT_DEBUG=1

More verbose than usual.

BOOT_DEBUG=2

Lots of debugging information.

BOOT_DEBUG=3

Shells are run at various points in the boot process to allow detailed debugging. Exit the shell to continue the boot.

INSTALL_MEDIA_DEV

The value of the parameter is the path to the device to load the Ubuntu installer from. For example,

log_host
log_port

Causes the installer to send log messages to a remote syslog on the specified host and port as well as to a local file. If not specified, the port defaults to the standard syslog port 514.

lowmem

Can be used to force the installer to a lowmem level higher than the one the installer sets by default based on available memory. Possible values are 1 and 2. See also Section 6.3.1.1.

noshell

Prevents the installer from offering interactive shells on tty2 and tty3. Useful for unattended installations where physical security is limited.

debian-installer/framebuffer (fb)

Some architectures use the kernel framebuffer to offer installation in a number of languages. If framebuffer causes a problem on your system you can disable the feature using the parameter **fb=false**. Problem symptoms are error messages about bterm or bogl, a blank screen, or a freeze within a few minutes after starting the install.

debian-installer/theme (theme)

A theme determines how the user interface of the installer looks (colors, icons, etc.). What themes are available differs per frontend. Currently both the newt and gtk frontends only have a "dark" theme that was designed for visually impaired users. Set the theme by booting with **theme=*dark***.

netcfg/disable_autoconfig

By default, the `debian-installer` automatically probes for network configuration via IPv6 autoconfiguration and DHCP. If the probe succeeds, you won't have a chance to review and change the obtained settings. You can get to the manual network setup only in case the automatic configuration fails.

If you have an IPv6 router or a DHCP server on your local network, but want to avoid them because e.g. they give wrong answers, you can use the parameter **netcfg/disable_autoconfig=true** to prevent any automatic configuration of the network (neither v4 nor v6) and to enter the information manually.

disk-detect/dmraid/enable (dmraid)

Set to **true** to enable support for Serial ATA RAID (also called ATA RAID, BIOS RAID or fake RAID) disks in the installer. Note that this support is currently experimental. Additional information can be found on the Debian Installer Wiki (http://wiki.debian.org/DebianInstaller/).

preseed/url (url)

Specify the url to a preconfiguration file to download and use for automating the install. See Section 4.4.

preseed/file (file)

Specify the path to a preconfiguration file to load for automating the install. See Section 4.4.

preseed/interactive

Set to **true** to display questions even if they have been preseeded. Can be useful for testing or debugging a preconfiguration file. Note that this will have no effect on parameters that are passed as boot parameters, but for those a special syntax can be used. See Section B.5.2 for details.

auto-install/enable (auto)

Delay questions that are normally asked before preseeding is possible until after the network is configured. See Section B.2.3 for details about using this to automate installs.

finish-install/keep-consoles

During installations from serial or management console, the regular virtual consoles (VT1 to VT6) are normally disabled in `/etc/inittab`. Set to **true** to prevent this.

cdrom-detect/eject

By default, before rebooting, `debian-installer` automatically ejects the optical media used during the installation. This can be unnecessary if the system does not automatically boot off the CD. In some cases it may even be undesirable, for example if the optical drive cannot reinsert the media itself and the user is not there (since working from remote) to do it manually. Many slot loading, slim-line, and caddy style drives cannot reload media automatically.

Set to **`false`** to disable automatic ejection, and be aware that you may need to ensure that the system does not automatically boot from the optical drive after the initial installation.

base-installer/install-recommends (recommends)

By setting this option to **`false`**, the package management system will be configured to not automatically install "Recommends", both during the installation and for the installed system. See also Section 6.3.4.

Note that this option allows to have a leaner system, but can also result in features being missing that you might normally expect to be available. You may have to manually install some of the recommended packages to obtain the full functionality you want. This option should therefore only be used by very experienced users.

debian-installer/allow_unauthenticated

By default the installer requires that repositories be authenticated using a known gpg key. Set to **`true`** to disable that authentication. **Warning: insecure, not recommended.**

rescue/enable

Set to **`true`** to enter rescue mode rather than performing a normal installation. See Section 8.7.

5.3.3. Using boot parameters to answer questions

With some exceptions, a value can be set at the boot prompt for any question asked during the installation, though this is only really useful in specific cases. General instructions how to do this can be found in Section B.2.2. Some specific examples are listed below.

debian-installer/language (language)
debian-installer/country (country)
debian-installer/locale (locale)

There are two ways to specify the language, country and locale to use for the installation and the installed system.

The first and easiest is to pass only the parameter `locale`. Language and country will then be derived from its value. You can for example use **`locale=de_CH`** to select German as language and Switzerland as country (`de_CH.UTF-8` will be set as default locale for the installed system). Limitation is that not all possible combinations of language, country and locale can be achieved this way.

The second, more flexible option is to specify `language` and `country` separately. In this case `locale` can optionally be added to specify a specific default locale for the installed system. Example: **`language=en country=DE locale=en_GB.UTF-8`**.

anna/choose_modules (modules)

Can be used to automatically load installer components that are not loaded by default. Examples of optional components that may be useful are `openssh-client-udeb` (so you can use **scp** during the installation) and `ppp-udeb` (see Section D.5).

netcfg/disable_autoconfig

Set to **`true`** if you want to disable IPv6 autoconfiguration and DHCP and instead force static network configuration.

mirror/protocol (protocol)

By default the installer will use the http protocol to download files from Ubuntu mirrors and changing that to ftp is not possible during installations at normal priority. By setting this parameter to **`ftp`**, you can force the installer to use that protocol instead. Note that you cannot select an ftp mirror from a list, you have to enter the hostname manually.

tasksel:tasksel/first (tasks)

Can be used to select tasks that are not available from the interactive task list, such as the `kde-desktop` task. See Section 6.3.5.2 for additional information.

apt-setup/restricted=false apt-setup/multiverse=false

This is the equivalent of enabling the “Free Software Only” boot mode. This will disable the restricted and multiverse repositories at the earliest stages of the `debian-installer`. This can be helpful is you don’t need them at all or don’t mirror those repositoriesat your local mirror.’

5.3.4. Passing parameters to kernel modules

If drivers are compiled into the kernel, you can pass parameters to them as described in the kernel documentation. However, if drivers are compiled as modules and because kernel modules are loaded a bit differently during an installation than when booting an installed system, it is not possible to pass parameters to modules as you would normally do. Instead, you need to use a special syntax recognized by the installer which will then make sure that the parameters are saved in the proper configuration files and will thus be used when the modules are actually loaded. The parameters will also be propagated automatically to the configuration for the installed system.

Note: It became quite rare these days that parameters need to be passed to modules. In most cases the kernel will be able to probe the hardware present in a system and set good defaults that way. However, in some situations it may still be needed to set parameters manually.

The syntax to use to set parameters for modules is:

```
module_name.parameter_name=value
```

If you need to pass multiple parameters to the same or different modules, just repeat this. For example, to set an old 3Com network interface card to use the BNC (coax) connector and IRQ 10, you would pass:

```
3c509.xcvr=3 3c509.irq=10
```

5.3.5. Blacklisting kernel modules

Sometimes it may be necessary to blacklist a module to prevent it from being loaded automatically by the kernel and udev. One reason could be that a particular module causes problems with your hardware. The kernel also sometimes lists two different drivers for the same device. This can cause the device to not work correctly if the drivers conflict or if the wrong driver is loaded first.

You can blacklist a module using the following syntax: ***module_name*.blacklist=yes**. This will cause the module to be blacklisted in `/etc/modprobe.d/blacklist.local` both during the installation and for the installed system.

Note that a module may still be loaded by the installation system itself. You can prevent that from happening by running the installation in expert mode and unselecting the module from the list of modules displayed during the hardware detection phases.

5.4. Troubleshooting the Installation Process

5.4.1. CD-ROM Reliability

Sometimes, especially with older CD-ROM drives, the installer may fail to boot from a CD-ROM. The installer may also — even after booting successfully from CD-ROM — fail to recognize the CD-ROM or return errors while reading from it during the installation.

There are many different possible causes for these problems. We can only list some common issues and provide general suggestions on how to deal with them. The rest is up to you.

There are two very simple things that you should try first.

- If the CD-ROM does not boot, check that it was inserted correctly and that it is not dirty.
- If the installer fails to recognize a CD-ROM, try just running the option Detect and mount CD-ROM a second time. Some DMA related issues with very old CD-ROM drives are known to be resolved in this way.

If this does not work, then try the suggestions in the subsections below. Most, but not all, suggestions discussed there are valid for both CD-ROM and DVD, but we'll use the term CD-ROM for simplicity.

If you cannot get the installation working from CD-ROM, try one of the other installation methods that are available.

5.4.1.1. Common issues

- Some older CD-ROM drives do not support reading from discs that were burned at high speeds using a modern CD writer.
- Some very old CD-ROM drives do not work correctly if "direct memory access" (DMA) is enabled for them.

5.4.1.2. How to investigate and maybe solve issues

If the CD-ROM fails to boot, try the suggestions listed below.

- Check that your BIOS actually supports booting from CD-ROM (only an issue for very old systems) and that CD booting is enabled in the BIOS.
- If you downloaded an iso image, check that the md5sum of that image matches the one listed for the image in the `MD5SUMS` file that should be present in the same location as where you downloaded the image from.

  ```
  $ md5sum debian-testing-i386-netinst.iso
  a20391b12f7ff22ef705cee4059c6b92  debian-testing-i386-netinst.iso
  ```

  Next, check that the md5sum of the burned CD-ROM matches as well. The following command should work. It uses the size of the image to read the correct number of bytes from the CD-ROM.

  ```
  $ dd if=/dev/cdrom | \
  > head -c `stat --format=%s debian-testing-i386-netinst.iso` | \
  > md5sum
  a20391b12f7ff22ef705cee4059c6b92  -
  262668+0 records in
  262668+0 records out
  134486016 bytes (134 MB) copied, 97.474 seconds, 1.4 MB/s
  ```

If, after the installer has been booted successfully, the CD-ROM is not detected, sometimes simply trying again may solve the problem. If you have more than one CD-ROM drive, try changing the CD-ROM to the other drive. If that does not work or if the CD-ROM is recognized but there are errors when reading from it, try the suggestions listed below. Some basic knowledge of Linux is required for this. To execute any of the commands, you should first switch to the second virtual console (VT2) and activate the shell there.

- Switch to VT4 or view the contents of `/var/log/syslog` (use **nano** as editor) to check for any specific error messages. After that, also check the output of **dmesg**.
- Check in the output of **dmesg** if your CD-ROM drive was recognized. You should see something like (the lines do not necessarily have to be consecutive):

  ```
  Probing IDE interface ide1...
  hdc: TOSHIBA DVD-ROM SD-R6112, ATAPI CD/DVD-ROM drive
  ide1 at 0x170-0x177,0x376 on irq 15
  hdc: ATAPI 24X DVD-ROM DVD-R CD-R/RW drive, 2048kB Cache, UDMA(33)
  Uniform CD-ROM driver Revision: 3.20
  ```

  If you don't see something like that, chances are the controller your CD-ROM is connected to was not recognized or may be not supported at all. If you know what driver is needed for the controller, you can try loading it manually using **modprobe**.
- Check that there is a device node for your CD-ROM drive under `/dev/`. In the example above, this would be `/dev/hdc`. There should also be a `/dev/cdrom`.
- Use the **mount** command to check if the CD-ROM is already mounted; if not, try mounting it manually:

  ```
  $ mount /dev/hdc /cdrom
  ```

Check if there are any error messages after that command.

- Check if DMA is currently enabled:

```
$ cd /proc/ide/hdc
$ grep using_dma settings
using_dma       1       0       1       rw
```

A "1" in the first column after `using_dma` means it is enabled. If it is, try disabling it:

```
$ echo -n "using_dma:0" >settings
```

Make sure that you are in the directory for the device that corresponds to your CD-ROM drive.

- If there are any problems during the installation, try checking the integrity of the CD-ROM using the option near the bottom of the installer's main menu. This option can also be used as a general test if the CD-ROM can be read reliably.

5.4.2. Boot Configuration

If you have problems and the kernel hangs during the boot process, doesn't recognize peripherals you actually have, or drives are not recognized properly, the first thing to check is the boot parameters, as discussed in Section 5.3.

In some cases, malfunctions can be caused by missing device firmware (see Section 2.2 and Section 6.4).

5.4.3. Interpreting the Kernel Startup Messages

During the boot sequence, you may see many messages in the form `can't find` *`something`*, or *`something`* `not present`, `can't initialize` *`something`*, or even `this driver release depends on` *`something`*. Most of these messages are harmless. You see them because the kernel for the installation system is built to run on computers with many different peripheral devices. Obviously, no one computer will have every possible peripheral device, so the operating system may emit a few complaints while it looks for peripherals you don't own. You may also see the system pause for a while. This happens when it is waiting for a device to respond, and that device is not present on your system. If you find the time it takes to boot the system unacceptably long, you can create a custom kernel later (see Section 8.6).

5.4.4. Reporting Installation Problems

If you get through the initial boot phase but cannot complete the install, the menu option Save debug logs may be helpful. It lets you store system error logs and configuration information from the installer to a disk or download them using a web browser. This information may provide clues as to what went wrong and how to fix it. If you are submitting a bug report, you may want to attach this information to the bug report.

Other pertinent installation messages may be found in `/var/log/` during the installation, and `/var/log/installer/` after the computer has been booted into the installed system.

5.4.5. Submitting Installation Reports

If you still have problems, please submit an installation report. We also encourage installation reports to be sent even if the installation is successful, so that we can get as much information as possible on the largest number of hardware configurations.

If you have a working Ubuntu system, the easiest way to send an installation report is to install the `installation-report` and `reportbug` packages (**apt install installation-report reportbug**), configure `reportbug` as explained in Section 8.5.2, and run the command **reportbug installation-reports**.

Alternatively you can use this template when filling out installation reports, and send the report to <`ubuntu-users@lists.ubuntu.com`>.

```
Package: installation-reports

Boot method: <How did you boot the installer? CD? floppy? network?>
Image version: <Full URL to image you downloaded is best>
Date: <Date and time of the install>

Machine: <Description of machine (eg, IBM Thinkpad W520)>
Processor:
Memory:
Partitions: <df -Tl will do; the raw partition table is preferred>

Output of lspci -knn (or lspci -nn):

Base System Installation Checklist:
[O] = OK, [E] = Error (please elaborate below), [ ] = didn't try it

Initial boot:           [ ]
Detect network card:    [ ]
Configure network:      [ ]
Detect CD:              [ ]
Load installer modules: [ ]
Detect hard drives:     [ ]
Partition hard drives:  [ ]
Install base system:    [ ]
Clock/timezone setup:   [ ]
User/password setup:    [ ]
Install tasks:          [ ]
Install boot loader:    [ ]
Overall install:        [ ]

Comments/Problems:

<Description of the install, in prose, and any thoughts, comments
      and ideas you had during the initial install.>
```

In the bug report, describe what the problem is, including the last visible kernel messages in the event of a kernel hang. Describe the steps that you did which brought the system into the problem state.

Chapter 6. Using the Ubuntu Installer

6.1. How the Installer Works

The Ubuntu Installer (based on the Debian Installer, and so often called simply `debian-installer` or just `d-i`) consists of a number of special-purpose components to perform each installation task. Each component performs its task, asking the user questions as necessary to do its job. The questions themselves are given priorities, and the priority of questions to be asked is set when the installer is started.

When a default installation is performed, only essential (high priority) questions will be asked. This results in a highly automated installation process with little user interaction. Components are automatically run in sequence; which components are run depends mainly on the installation method you use and on your hardware. The installer will use default values for questions that are not asked.

If there is a problem, the user will see an error screen, and the installer menu may be shown in order to select some alternative action. If there are no problems, the user will never see the installer menu, but will simply answer questions for each component in turn. Serious error notifications are set to priority "critical" so the user will always be notified.

Some of the defaults that the installer uses can be influenced by passing boot arguments when `debian-installer` is started. If, for example, you wish to force static network configuration (IPv6 autoconfiguration and DHCP are used by default if available), you could add the boot parameter **`netcfg/disable_autoconfig=true`**. See Section 5.3.2 for available options.

Power users may be more comfortable with a menu-driven interface, where each step is controlled by the user rather than the installer performing each step automatically in sequence. To use the installer in a manual, menu-driven way, add the boot argument **`priority=medium`**.

If your hardware requires you to pass options to kernel modules as they are installed, you will need to start the installer in "expert" mode. This can be done by either using the **expert** command to start the installer or by adding the boot argument **`priority=low`**. Expert mode gives you full control over `debian-installer`.

For this architecture the installer uses a character-based user interface. A graphical user interface is currently not available.

In the character-based environment the use of a mouse is not supported. Here are the keys you can use to navigate within the various dialogs. The **Tab** or **right** arrow keys move "forward", and the **Shift**-**Tab** or **left** arrow keys move "backward" between displayed buttons and selections. The **up** and **down** arrow select different items within a scrollable list, and also scroll the list itself. In addition, in long lists, you can type a letter to cause the list to scroll directly to the section with items starting with the letter you typed and use **Pg-Up** and **Pg-Down** to scroll the list in sections. The **space bar** selects an item such as a checkbox. Use **Enter** to activate choices.

Some dialogs may offer additional help information. If help is available this will be indicated on the bottom line of the screen by displaying that help information can be accessed by pressing the **F1** key.

Error messages and logs are redirected to the fourth console. You can access this console by pressing **Left Alt**-**F4** (hold the left **Alt** key while pressing the **F4** function key); get back to the main installer process with **Left Alt**-**F1**.

The error messages are logged in `/var/log/syslog`. After installation, this log is copied to `/var/log/installer/syslog` on your new system. Other installation messages may be found in `/var/log/` during the installation, and `/var/log/installer/` after the computer has been booted into the installed system.

6.2. Components Introduction

Here is a list of installer components with a brief description of each component's purpose. Details you might need to know about using a particular component are in Section 6.3.

main-menu

Shows the list of components to the user during installer operation, and starts a component when it is selected. Main-menu's questions are set to priority medium, so if your priority is set to high or critical (high is the default), you will not see the menu. On the other hand, if there is an error which requires your intervention, the question priority may be downgraded temporarily to allow you to resolve the problem, and in that case the menu may appear.

You can get to the main menu by selecting the Go Back button repeatedly to back all the way out of the currently running component. .

localechooser

Allows the user to select localization options for the installation and the installed system: language, country and locales. The installer will display messages in the selected language, unless the translation for that language is not complete in which case some messages may be shown in English.

console-setup

Shows a list of keyboards, from which the user chooses the model which matches his own.

hw-detect

Automatically detects most of the system's hardware, including network cards, PCMCIA and disk drives.

cdrom-detect

Looks for and mounts an Ubuntu installation CD.

netcfg

Configures the computer's network connections so it can communicate over the internet.

iso-scan

Searches for ISO images (`.iso` files) on hard drives.

choose-mirror

Presents a list of Ubuntu archive mirrors. The user may choose the source of his installation packages.

cdrom-checker

Checks integrity of a CD-ROM. This way, the user may assure him/herself that the installation CD-ROM was not corrupted.

lowmem

Lowmem tries to detect systems with low memory and then does various tricks to remove unnecessary parts of `debian-installer` from the memory (at the cost of some features).

anna

Anna's Not Nearly APT. Installs packages which have been retrieved from the chosen mirror or CD.

user-setup

Sets up the root password, and adds a non-root user.

clock-setup

Updates the system clock and determines whether the clock is set to UTC or not.

tzsetup

Selects the time zone, based on the location selected earlier.

partman

Allows the user to partition disks attached to the system, create file systems on the selected partitions, and attach them to the mountpoints. Included are also interesting features like a fully automatic mode or LVM support. This is the preferred partitioning tool in Ubuntu.

lvmcfg

Helps the user with the configuration of the *LVM* (Logical Volume Manager).

mdcfg

Allows the user to set up Software *RAID* (Redundant Array of Inexpensive Disks). This Software RAID is usually superior to the cheap IDE (pseudo hardware) RAID controllers found on newer motherboards.

base-installer

Installs the most basic set of packages which would allow the computer to operate under Ubuntu when rebooted.

apt-setup

Configures apt, mostly automatically, based on what media the installer is running from.

pkgsel

Uses `tasksel` to select and install additional software.

os-prober

Detects currently installed operating systems on the computer and passes this information to the bootloader-installer, which may offer you an ability to add discovered operating systems to the bootloader's start menu. This way the user could easily choose at the boot time which operating system to start.

bootloader-installer

The various bootloader installers each install a boot loader program on the hard disk, which is necessary for the computer to start up using Linux without requiring further (removable) media, like floppy or CD-ROM. Many boot loaders allow the user to choose an alternate operating system each time the computer boots.

shell

Allows the user to execute a shell from the menu, or in the second console.

save-logs

Provides a way for the user to record information on a hard disk, floppy disk, network, or other (removable) media when trouble is encountered, in order to accurately report installer software problems to Ubuntu developers later.

6.3. Using Individual Components

In this section we will describe each installer component in detail. The components have been grouped into stages that should be recognizable for users. They are presented in the order they appear during the install. Note that not all modules will be used for every installation; which modules are actually used depends on the installation method you use and on your hardware.

6.3.1. Setting up Ubuntu Installer and Hardware Configuration

Let's assume the Ubuntu Installer has booted and you are facing its first screen. At this time, the capabilities of `debian-installer` are still quite limited. It doesn't know much about your hardware, preferred language, or even the task it should perform. Don't worry. Because `debian-installer` is quite clever, it can automatically probe your hardware, locate the rest of its components and upgrade itself to a capable installation system. However, you still need to help `debian-installer` with some information it can't determine automatically (like selecting your preferred language, keyboard layout or desired network mirror).

You will notice that `debian-installer` performs *hardware detection* several times during this stage. The first time is targeted specifically at the hardware needed to load installer components (e.g. your CD-ROM or network card). As not all drivers may be available during this first run, hardware detection needs to be repeated later in the process.

During hardware detection `debian-installer` checks if any of the drivers for the hardware devices in your system require firmware to be loaded. If any firmware is requested but unavailable, a dialog will be displayed that allows the missing firmware to be loaded from a removable medium. See Section 6.4 for further details.

6.3.1.1. Check available memory / low memory mode

One of the first things `debian-installer` does, is to check available memory. If the available memory is limited, this component will make some changes in the installation process which hopefully will allow you to install Ubuntu on your system.

The first measure taken to reduce memory consumption by the installer is to disable translations, which means that the installation can only be done in English. Of course, you can still localize the installed system after the installation has completed.

If that is not sufficient, the installer will further reduce memory consumption by loading only those components essential to complete a basic installation. This reduces the functionality of the installation system. You will be given the opportunity to load additional components manually, but you should be aware that each component you select will use additional memory and thus may cause the installation to fail.

If the installer runs in low memory mode, it is recommended to create a relatively large swap partition (64–128MB). The swap partition will be used as virtual memory and thus increases the amount of memory available to the system. The installer will activate the swap partition as early as possible in the installation process. Note that heavy use of swap will reduce performance of your system and may lead to high disk activity.

Despite these measures, it is still possible that your system freezes, that unexpected errors occur or that processes are killed by the kernel because the system runs out of memory (which will result in "Out of memory" messages on VT4 and in the syslog).

For example, it has been reported that creating a big ext3 file system fails in low memory mode when there is insufficient swap space. If a larger swap doesn't help, try creating the file system as ext2 (which is an essential component of the installer) instead. It is possible to change an ext2 partition to ext3 after the installation.

It is possible to force the installer to use a higher lowmem level than the one based on available memory by using the boot parameter "lowmem" as described in Section 5.3.2.

6.3.1.2. Selecting Localization Options

In most cases the first questions you will be asked concern the selection of localization options to be used both for the installation and for the installed system. The localization options consist of language, location and locales.

The language you choose will be used for the rest of the installation process, provided a translation of the different dialogs is available. If no valid translation is available for the selected language, the installer will default to English.

The selected geographic location (in most cases a country) will be used later in the installation process to select the correct time zone and an Ubuntu mirror appropriate for that country. Language and country together will help determine the default locale for your system and select the correct keyboard layout.

You will first be asked to select your preferred language. The language names are listed both in English (left side) and in the language itself (right side); the names on the right side are also shown in the proper script for the language. The list is sorted on the English names. At the top of the list is an extra option that allows you to select the "C" locale instead of a language. Choosing the "C" locale will result in the installation proceding in English; the installed system will have no localization support as the `locales` package will not be installed.

Next you will be asked to select your geographic location. If you selected a language that is recognized as an official language for more than one country[1], you will be shown a list of only those countries. To select a country that is not in that list, choose other (the last option). You will then be presented with a list of continents; selecting a continent will lead to a list of relevant countries on that continent.

If the language has only one country associated with it, a list of countries will be displayed for the continent or region the country belongs to, with that country selected as the default. Use the Go Back option to select countries on a different continent.

> **Note:** It is important to select the country where you live or where you are located as it determines the time zone that will be configured for the installed system.

1. In technical terms: where multiple locales exist for that language with differing country codes.

If you selected a combination of language and country for which no locale is defined and there exist multiple locales for the language, then the installer will allow you to choose which of those locales you prefer as the default locale for the installed system[2]. In all other cases a default locale will be selected based on the selected language and country.

Any default locale selected as described in the previous paragraph will use *UTF-8* as character encoding.

If you are installing at low priority, you will have the option of selecting additional locales, including so-called "legacy" locales[3], to be generated for the installed system; if you do, you will be asked which of the selected locales should be the default for the installed system.

6.3.1.3. Choosing a Keyboard

Keyboards are often tailored to the characters used in a language. Select a layout that conforms to the keyboard you are using, or select something close if the keyboard layout you want isn't represented. Once the system installation is complete, you'll be able to select a keyboard layout from a wider range of choices (run **dpkg-reconfigure keyboard-configuration** as root after you have completed the installation).

Move the highlight to the keyboard selection you desire and press **Enter**. Use the arrow keys to move the highlight — they are in the same place in all national language keyboard layouts, so they are independent of the keyboard configuration.

6.3.1.4. Looking for the Ubuntu Installer ISO Image

When installing via the *hd-media* method, there will be a moment where you need to find and mount the Ubuntu Installer iso image in order to get the rest of the installation files. The component **iso-scan** does exactly this.

At first, **iso-scan** automatically mounts all block devices (e.g. partitions) which have some known filesystem on them and sequentially searches for filenames ending with `.iso` (or `.ISO` for that matter). Beware that the first attempt scans only files in the root directory and in the first level of subdirectories (i.e. it finds `/`*`whatever`*`.iso`, `/data/`*`whatever`*`.iso`, but not `/data/tmp/`*`whatever`*`.iso`). After an iso image has been found, **iso-scan** checks its content to determine if the image is a valid Ubuntu iso image or not. In the former case we are done, in the latter **iso-scan** seeks for another image.

In case the previous attempt to find an installer iso image fails, **iso-scan** will ask you whether you would like to perform a more thorough search. This pass doesn't just look into the topmost directories, but really traverses whole filesystem.

If **iso-scan** does not discover your installer iso image, reboot back to your original operating system and check if the image is named correctly (ending in `.iso`), if it is placed on a filesystem recognizable by `debian-installer`, and if it is not corrupted (verify the checksum). Experienced Unix users could do this without rebooting on the second console.

2. At medium and low priority you can always select your preferred locale from those available for the selected language (if there's more than one).
3. Legacy locales are locales which do not use UTF-8, but one of the older standards for character encoding such as ISO 8859-1 (used by West European languages) or EUC-JP (used by Japanese).

6.3.1.5. Configuring the Network

As you enter this step, if the system detects that you have more than one network device, you'll be asked to choose which device will be your *primary* network interface, i.e. the one which you want to use for installation. The other interfaces won't be configured at this time. You may configure additional interfaces after installation is complete; see the interfaces(5) man page.

6.3.1.5.1. Automatic network configuration

By default, `debian-installer` tries to configure your computer's network automatically as far as possible. If the automatic configuration fails, that may be caused by many factors ranging from an unplugged network cable to missing infrastructure for automatic configuration. For further explanation in case of errors, check the error messages on the fourth console. In any case, you will be asked if you want to retry, or if you want to perform a manual setup. Sometimes the network services used for autoconfiguration can be slow in their responses, so if you are sure everything is in place, simply start the autoconfiguration attempt again. If autoconfiguration fails repeatedly, you can instead choose the manual network setup.

6.3.1.5.2. Manual network configuration

The manual network setup in turn asks you a number of questions about your network, notably `IP address`, `Netmask`, `Gateway`, `Name server addresses`, `Hostname`, and a `Domain name`. Moreover, if you have a wireless network interface, you will be asked to provide your `Wireless ESSID` ("wireless network name") and a `WEP key` or `WPA/WPA2 passphrase`. Fill in the answers from Section 3.3.

> **Note:** Some technical details you might, or might not, find handy: the program assumes the network IP address is the bitwise-AND of your system's IP address and your netmask. The default broadcast address is calculated as the bitwise OR of your system's IP address with the bitwise negation of the netmask. It will also guess your gateway. If you can't find any of these answers, use the offered defaults — if necessary, you can change them by editing `/etc/netplan/01-netcfg.yaml` (or `/etc/network/interfaces` - in case you switched from netplan to ifupdown) once the system has been installed.

6.3.1.5.3. IPv4 and IPv6

Ubuntu supports IPv6 as well as the "classic" IPv4. All combinations of IPv4 and IPv6 (IPv4-only, IPv6-only and dual-stack configurations) are supported.

Autoconfiguration for IPv4 is done via DHCP (Dynamic Host Configuration Protocol). Autoconfiguration for IPv6 supports stateless autoconfiguration using NDP (Neighbor Discovery Protocol, including recursive DNS server (RDNSS) assignment), stateful autoconfiguration via DHCPv6 and mixed stateless/stateful autoconfiguration (address configuration via NDP, additional parameters via DHCPv6).

6.3.1.6. Configuring the Clock and Time Zone

The installer will first attempt to connect to a time server on the Internet (using the *NTP* protocol) in order to correctly set the system time. If this does not succeed, the installer will assume the time

and date obtained from the system clock when the installation system was booted are correct. It is not possible to manually set the system time during the installation process.

Depending on the location selected earlier in the installation process, you may be shown a list of time zones relevant for that location. If your location has only one time zone and you are doing a default installation, you will not be asked anything and the system will assume that time zone.

In expert mode or when installing at medium priority, you will have the additional option to select "Coordinated Universal Time" (UTC) as time zone.

If for some reason you wish to set a time zone for the installed system that does *not* match the selected location, there are two options.

1. The simplest option is to just select a different time zone after the installation has been completed and you've booted into the new system. The command to do this is:

   ```
   # dpkg-reconfigure tzdata
   ```

2. Alternatively, the time zone can be set at the very start of the installation by passing the parameter **time/zone=*value*** when you boot the installation system. The value should of course be a valid time zone, for example **Europe/London** or **UTC**.

For automated installations the time zone can be set to any desired value using preseeding.

6.3.2. Setting Up Users And Passwords

Just before configuring the clock, the installer will allow you to set up the "root" account and/or an account for the first user. Other user accounts can be created after the installation has been completed.

6.3.2.1. Create an Ordinary User

The system will ask you whether you wish to create an ordinary user account at this point. This account should be your main personal log-in.

The account you create at this point will be given *root* privileges by means of the **sudo** command, and the root account itself will have login disabled. If you wish, you can enable the root account later by setting a password for it with the command **sudo passwd root**.

You should *not* use the root account for daily use or as your personal login, nor should you use **sudo** except when root privileges are really required.

Why not? Well, one reason to avoid using root's privileges is that it is very easy to do irreparable damage as root. Another reason is that you might be tricked into running a *Trojan-horse* program — that is a program that takes advantage of your super-user powers to compromise the security of your system behind your back. Any good book on Unix system administration will cover this topic in more detail — consider reading one if it is new to you.

You will first be prompted for the user's full name. Then you'll be asked for a name for the user account; generally your first name or something similar will suffice and indeed will be the default. Finally, you will be prompted for a password for this account.

If at any point after installation you would like to create another account, use the **adduser** command.

6.3.3. Partitioning and Mount Point Selection

At this time, after hardware detection has been executed a final time, `debian-installer` should be at its full strength, customized for the user's needs and ready to do some real work. As the title of this section indicates, the main task of the next few components lies in partitioning your disks, creating filesystems, assigning mountpoints and optionally configuring closely related options like RAID, LVM or encrypted devices.

DASDs (Direct Attached Storage Devices) are Enhanced Count Key Data (ECKD) encoded, FICON-attached devices and belong to the CCW (channel command word) IO-layer that is unique to arm64. They are available in different types, like the common types 3390-3 (or Model 3), 3390-9 (or Model 9), 3390-27 (or Model 27), 3390-54 (or Model 54), or others. The DASD block size is 4096 bytes (4KB) and they support up to 3 partitions per volume.

If you are uncomfortable with partitioning, or just want to know more details, see Appendix C.

First you will be given the opportunity to automatically partition either an entire drive, or available free space on a drive. This is also called "guided" partitioning. If you do not want to autopartition, choose Manual from the menu.

6.3.3.1. Supported partitioning options

The partitioner used in `debian-installer` is fairly versatile. It allows to create many different partitioning schemes, using various partition tables, file systems and advanced block devices.

Exactly which options are available depends mainly on the architecture, but also on other factors. For example, on systems with limited internal memory some options may not be available. Defaults may vary as well. The type of partition table used by default can for example be different for large capacity hard disks than for smaller hard disks. Some options can only be changed when installing at medium or low debconf priority; at higher priorities sensible defaults will be used.

The installer supports various forms of advanced partitioning and use of storage devices, which in many cases can be used in combination.

- *Logical Volume Management (LVM)*
- *Software RAID*

  Supported are RAID levels 0, 1, 4, 5, 6 and 10.
- *Encryption*
- *Multipath*

  See our Wiki (https://help.ubuntu.com/lts/serverguide/dm-multipath-chapter.html) for information.

  Support for multipath is currently only available if enabled when the installer is booted.

The following file systems are supported.

- *ext2r0*, *ext2*, *ext3*, *ext4*

  The default file system selected in most cases is ext4; for `/boot` partitions ext2 will be selected by default when guided partitioning is used.
- *jfs* (not available on all architectures)
- *xfs* (not available on all architectures)
- *reiserfs* (optional; not available on all architectures)

Support for the Reiser file system is no longer available by default. When the installer is running at medium or low debconf priority it can be enabled by selecting the `partman-reiserfs` component. Only version 3 of the file system is supported.

- *jffs2*

  Used on some systems to read flash memory. It is not possible to create new jffs2 partitions.

- *FAT16*, *FAT32*

6.3.3.2. Guided Partitioning

If you choose guided partitioning, you may have three options: to create partitions directly on the hard disk (classic method), or to create them using Logical Volume Management (LVM), or to create them using encrypted LVM[4].

> **Note:** The option to use (encrypted) LVM may not be available on all architectures.

When using LVM or encrypted LVM, the installer will create most partitions inside one big partition; the advantage of this method is that partitions inside this big partition can be resized relatively easily later. In the case of encrypted LVM the big partition will not be readable without knowing a special key phrase, thus providing extra security of your (personal) data.

When using encrypted LVM, the installer will also automatically erase the disk by writing random data to it. This further improves security (as it makes it impossible to tell which parts of the disk are in use and also makes sure that any traces of previous installations are erased), but may take some time depending on the size of your disk.

> **Note:** If you choose guided partitioning using LVM or encrypted LVM, some changes in the partition table will need to be written to the selected disk while LVM is being set up. These changes effectively erase all data that is currently on the selected hard disk and you will not be able to undo them later. However, the installer will ask you to confirm these changes before they are written to disk.

If you choose guided partitioning (either classic or using (encrypted) LVM) for a whole disk, you will first be asked to select the disk you want to use. Check that all your disks are listed and, if you have several disks, make sure you select the correct one. The order they are listed in may differ from what you are used to. The size of the disks may help to identify them.

Any data on the disk you select will eventually be lost, but you will always be asked to confirm any changes before they are written to the disk. If you have selected the classic method of partitioning, you will be able to undo any changes right until the end; when using (encrypted) LVM this is not possible.

Next, you will be able to choose from the schemes listed in the table below. All schemes have their pros and cons, some of which are discussed in Appendix C. If you are unsure, choose the first one. Bear in mind that guided partitioning needs a certain minimal amount of free space to operate with. If you don't give it at least about 1GB of space (depends on chosen scheme), guided partitioning will fail.

4. The installer will encrypt the LVM volume group using a 256 bit AES key and makes use of the kernel's "dm-crypt" support.

Partitioning scheme	Minimum space	Created partitions
All files in one partition	600MB	/, swap
Separate /home partition	500MB	/, /home, swap
Separate /home, /var and /tmp partitions	1GB	/, /home, /var, /tmp, swap

If you choose guided partitioning using (encrypted) LVM, the installer will also create a separate /boot partition. The other partitions, including the swap partition, will be created inside the LVM partition.

If you have booted in EFI mode then within the guided partitioning setup there will be an additional partition, formatted as a FAT32 bootable filesystem, for the EFI boot loader. This partition is known as an EFI System Partition (ESP). There is also an additional menu item in the formatting menu to manually set up a partition as an ESP.

After selecting a scheme, the next screen will show your new partition table, including information on whether and how partitions will be formatted and where they will be mounted.

The list of partitions might look like this:

```
SCSI1 (0,0,0) (sda) - 6.4 GB WDC AC36400L
      #1 primary   16.4 MB  B f ext2       /boot
      #2 primary  551.0 MB      swap       swap
      #3 primary    5.8 GB      ntfs
         pri/log    8.2 MB      FREE SPACE

SCSI2 (1,0,0) (sdb) - 80.0 GB ST380021A
      #1 primary   15.9 MB      ext3
      #2 primary  996.0 MB      fat16
      #3 primary    3.9 GB      xfs        /home
      #5 logical    6.0 GB    f ext4       /
      #6 logical    1.0 GB    f ext3       /var
      #7 logical  498.8 MB      ext3
```

This example shows two hard drives divided into several partitions; the first disk has some free space. Each partition line consists of the partition number, its type, size, optional flags, file system, and mountpoint (if any).

> **Note:** This particular setup cannot be created using guided partitioning but it does show possible variation that can be achieved using manual partitioning.

This concludes the guided partitioning. If you are satisfied with the generated partition table, you can choose Finish partitioning and write changes to disk from the menu to implement the new partition table (as described at the end of this section). If you are not happy, you can choose to Undo changes to partitions and run guided partitioning again, or modify the proposed changes as described below for Manual Partitioning.

6.3.3.3. Manual Partitioning

A similar screen to the one shown just above will be displayed if you choose manual partitioning except that your existing partition table will be shown and without the mount points. How to manually set up your partition table and the usage of partitions by your new Ubuntu system will be covered in the remainder of this section.

If you select a pristine disk which has neither partitions nor free space on it, you will be asked if a new partition table should be created (this is needed so you can create new partitions). After this, a new line entitled "FREE SPACE" should appear in the table under the selected disk.

If you select some free space, you will have the opportunity to create a new partition. You will have to answer a quick series of questions about its size, type (primary or logical), and location (beginning or end of the free space). After this, you will be presented with a detailed overview of your new partition. The main setting is Use as:, which determines if the partition will have a file system on it, or be used for swap, software RAID, LVM, an encrypted file system, or not be used at all. Other settings include mountpoint, mount options, and bootable flag; which settings are shown depends on how the partition is to be used. If you don't like the preselected defaults, feel free to change them to your liking. E.g. by selecting the option Use as:, you can choose a different filesystem for this partition, including options to use the partition for swap, software RAID, LVM, or not use it at all. When you are satisfied with your new partition, select Done setting up the partition and you will return to **partman**'s main screen.

If you decide you want to change something about your partition, simply select the partition, which will bring you to the partition configuration menu. This is the same screen as is used when creating a new partition, so you can change the same settings. One thing that may not be very obvious at a first glance is that you can resize the partition by selecting the item displaying the size of the partition. Filesystems known to work are at least fat16, fat32, ext2, ext3, ext4 and swap. This menu also allows you to delete a partition.

Be sure to create at least one partition for the *root* filesystem (which must be mounted as `/`) and possibly another one for *swap* - in case you dislike going with a swap file. If you forget to mount the root filesystem, **partman** won't let you continue until you correct this issue.

If you boot in EFI mode but forget to select and format an EFI System Partition, **partman** will detect this and will not let you continue until you allocate one.

Capabilities of **partman** can be extended with installer modules, but are dependent on your system's architecture. So if you can't see all promised goodies, check if you have loaded all required modules (e.g. `partman-ext3`, `partman-xfs`, or `partman-lvm`).

After you are satisfied with partitioning, select Finish partitioning and write changes to disk from the partitioning menu. You will be presented with a summary of changes made to the disks and asked to confirm that the filesystems should be created as requested.

6.3.3.4. Configuring Multidisk Devices (Software RAID)

If you have more than one harddrive[5] in your computer, you can use **mdcfg** to set up your drives for increased performance and/or better reliability of your data. The result is called *Multidisk Device* (or after its most famous variant *software RAID*).

MD is basically a bunch of partitions located on different disks and combined together to form a *logical* device. This device can then be used like an ordinary partition (i.e. in **partman** you can format it, assign a mountpoint, etc.).

What benefits this brings depends on the type of MD device you are creating. Currently supported are:

5. To be honest, you can construct an MD device even from partitions residing on single physical drive, but that won't give any benefits.

RAID0

Is mainly aimed at performance. RAID0 splits all incoming data into *stripes* and distributes them equally over each disk in the array. This can increase the speed of read/write operations, but when one of the disks fails, you will lose *everything* (part of the information is still on the healthy disk(s), the other part *was* on the failed disk).

The typical use for RAID0 is a partition for video editing.

RAID1

Is suitable for setups where reliability is the first concern. It consists of several (usually two) equally-sized partitions where every partition contains exactly the same data. This essentially means three things. First, if one of your disks fails, you still have the data mirrored on the remaining disks. Second, you can use only a fraction of the available capacity (more precisely, it is the size of the smallest partition in the RAID). Third, file-reads are load-balanced among the disks, which can improve performance on a server, such as a file server, that tends to be loaded with more disk reads than writes.

Optionally you can have a spare disk in the array which will take the place of the failed disk in the case of failure.

RAID5

Is a good compromise between speed, reliability and data redundancy. RAID5 splits all incoming data into stripes and distributes them equally on all but one disk (similar to RAID0). Unlike RAID0, RAID5 also computes *parity* information, which gets written on the remaining disk. The parity disk is not static (that would be called RAID4), but is changing periodically, so the parity information is distributed equally on all disks. When one of the disks fails, the missing part of information can be computed from remaining data and its parity. RAID5 must consist of at least three active partitions. Optionally you can have a spare disk in the array which will take the place of the failed disk in the case of failure.

As you can see, RAID5 has a similar degree of reliability to RAID1 while achieving less redundancy. On the other hand, it might be a bit slower on write operations than RAID0 due to computation of parity information.

RAID6

Is similar to RAID5 except that it uses two parity devices instead of one.

A RAID6 array can survive up to two disk failures.

RAID10

RAID10 combines striping (as in RAID0) and mirroring (as in RAID1). It creates `n` copies of incoming data and distributes them across the partitions so that none of the copies of the same data are on the same device. The default value of `n` is 2, but it can be set to something else in expert mode. The number of partitions used must be at least `n`. RAID10 has different layouts for distributing the copies. The default is near copies. Near copies have all of the copies at about the same offset on all of the disks. Far copies have the copies at different offsets on the disks. Offset copies copy the stripe, not the individual copies.

RAID10 can be used to achieve reliability and redundancy without the drawback of having to calculate parity.

To sum it up:

Type	Minimum Devices	Spare Device	Survives disk failure?	Available Space
RAID0	2	no	no	Size of the smallest partition multiplied by number of devices in RAID
RAID1	2	optional	yes	Size of the smallest partition in RAID
RAID5	3	optional	yes	Size of the smallest partition multiplied by (number of devices in RAID minus one)
RAID6	4	optional	yes	Size of the smallest partition multiplied by (number of devices in RAID minus two)
RAID10	2	optional	yes	Total of all partitions divided by the number of chunk copies (defaults to two)

If you want to know more about Software RAID, have a look at Software RAID HOWTO (http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html).

To create an MD device, you need to have the desired partitions it should consist of marked for use in a RAID. (This is done in **partman** in the Partition settings menu where you should select Use as:⟶physical volume for RAID.)

> **Note:** Make sure that the system can be booted with the partitioning scheme you are planning. In general it will be necessary to create a separate file system for `/boot` when using RAID for the root (`/`) file system. Most boot loaders do support mirrored (not striped!) RAID1, so using for example RAID5 for `/` and RAID1 for `/boot` can be an option.

Next, you should choose Configure software RAID from the main **partman** menu. (The menu will only appear after you mark at least one partition for use as physical volume for RAID.) On the first screen of **mdcfg** simply select Create MD device. You will be presented with a list of supported types of MD devices, from which you should choose one (e.g. RAID1). What follows depends on the type of MD you selected.

- RAID0 is simple — you will be issued with the list of available RAID partitions and your only task is to select the partitions which will form the MD.

- RAID1 is a bit more tricky. First, you will be asked to enter the number of active devices and the number of spare devices which will form the MD. Next, you need to select from the list of available RAID partitions those that will be active and then those that will be spare. The count of selected partitions must be equal to the number provided earlier. Don't worry. If you make a mistake and select a different number of partitions, `debian-installer` won't let you continue until you correct the issue.
- RAID5 has a setup procedure similar to RAID1 with the exception that you need to use at least *three* active partitions.
- RAID6 also has a setup procedure similar to RAID1 except that at least *four* active partitions are required.
- RAID10 again has a setup procedure similar to RAID1 except in expert mode. In expert mode, `debian-installer` will ask you for the layout. The layout has two parts. The first part is the layout type. It is either `n` (for near copies), `f` (for far copies), or `o` (for offset copies). The second part is the number of copies to make of the data. There must be at least that many active devices so that all of the copies can be distributed onto different disks.

It is perfectly possible to have several types of MD at once. For example, if you have three 200 GB hard drives dedicated to MD, each containing two 100 GB partitions, you can combine the first partitions on all three disks into the RAID0 (fast 300 GB video editing partition) and use the other three partitions (2 active and 1 spare) for RAID1 (quite reliable 100 GB partition for `/home`).

After you set up MD devices to your liking, you can Finish **mdcfg** to return back to the **partman** to create filesystems on your new MD devices and assign them the usual attributes like mountpoints.

6.3.3.5. Configuring the Logical Volume Manager (LVM)

If you are working with computers at the level of system administrator or "advanced" user, you have surely seen the situation where some disk partition (usually the most important one) was short on space, while some other partition was grossly underused and you had to manage this situation by moving stuff around, symlinking, etc.

To avoid the described situation you can use Logical Volume Manager (LVM). Simply said, with LVM you can combine your partitions (*physical volumes* in LVM lingo) to form a virtual disk (so called *volume group*), which can then be divided into virtual partitions (*logical volumes*). The point is that logical volumes (and of course underlying volume groups) can span across several physical disks.

Now when you realize you need more space for your old 160GB `/home` partition, you can simply add a new 300GB disk to the computer, join it with your existing volume group and then resize the logical volume which holds your `/home` filesystem and voila — your users have some room again on their renewed 460GB partition. This example is of course a bit oversimplified. If you haven't read it yet, you should consult the LVM HOWTO (http://www.tldp.org/HOWTO/LVM-HOWTO.html).

LVM setup in `debian-installer` is quite simple and completely supported inside **partman**. First, you have to mark the partition(s) to be used as physical volumes for LVM. This is done in the Partition settings menu where you should select Use as:⟶physical volume for LVM.

When you return to the main **partman** screen, you will see a new option Configure the Logical Volume Manager. When you select that, you will first be asked to confirm pending changes to the partition table (if any) and after that the LVM configuration menu will be shown. Above the menu a summary of the LVM configuration is shown. The menu itself is context sensitive and only shows valid actions. The possible actions are:

- Display configuration details: shows LVM device structure, names and sizes of logical volumes and more
- Create volume group
- Create logical volume
- Delete volume group
- Delete logical volume
- Extend volume group
- Reduce volume group
- Finish: return to the main **partman** screen

Use the options in that menu to first create a volume group and then create your logical volumes inside it.

After you return to the main **partman** screen, any created logical volumes will be displayed in the same way as ordinary partitions (and you should treat them as such).

6.3.3.6. Configuring Encrypted Volumes

`debian-installer` allows you to set up encrypted partitions. Every file you write to such a partition is immediately saved to the device in encrypted form. Access to the encrypted data is granted only after entering the *passphrase* used when the encrypted partition was originally created. This feature is useful to protect sensitive data in case your laptop or hard drive gets stolen. The thief might get physical access to the hard drive, but without knowing the right passphrase, the data on the hard drive will look like random characters.

The two most important partitions to encrypt are: the home partition, where your private data resides, and the swap partition, where sensitive data might be stored temporarily during operation. Of course, nothing prevents you from encrypting any other partitions that might be of interest. For example `/var` where database servers, mail servers or print servers store their data, or `/tmp` which is used by various programs to store potentially interesting temporary files. Some people may even want to encrypt their whole system. The only exception is the `/boot` partition which must remain unencrypted, because currently there is no way to load the kernel from an encrypted partition.

> **Note:** Please note that the performance of encrypted partitions will be less than that of unencrypted ones because the data needs to be decrypted or encrypted for every read or write. The performance impact depends on your CPU speed, chosen cipher, the key length and whether you use hardware assisted cryptography operations or not.

To use encryption, you have to create a new partition by selecting some free space in the main partitioning menu. Another option is to choose an existing partition (e.g. a regular partition, an LVM logical volume or a RAID volume). In the Partition settings menu, you need to select physical volume for encryption at the Use as: option. The menu will then change to include several cryptographic options for the partition.

The encryption method supported by `debian-installer` is *dm-crypt* (included in newer Linux kernels, able to host LVM physical volumes).

Let's have a look at the options available when you select encryption via **Device-mapper (dm-crypt)**. As always: when in doubt, use the defaults, because they have been carefully chosen with security in mind.

Encryption: **aes**

This option lets you select the encryption algorithm (*cipher*) which will be used to encrypt the data on the partition. `debian-installer` currently supports the following block ciphers: *aes*, *blowfish*, *serpent*, and *twofish*. It is beyond the scope of this document to discuss the qualities of these different algorithms, however, it might help your decision to know that in 2000, *AES* was chosen by the American National Institute of Standards and Technology as the standard encryption algorithm for protecting sensitive information in the 21st century.

Key size: **256**

Here you can specify the length of the encryption key. With a larger key size, the strength of the encryption is generally improved. On the other hand, increasing the length of the key usually has a negative impact on performance. Available key sizes vary depending on the cipher.

IV algorithm: **xts-plain64**

The *Initialization Vector* or *IV* algorithm is used in cryptography to ensure that applying the cipher on the same *clear text* data with the same key always produces a unique *cipher text*. The idea is to prevent the attacker from deducing information from repeated patterns in the encrypted data.

From the provided alternatives, the default **xts-plain64** is currently the least vulnerable to known attacks. Use the other alternatives only when you need to ensure compatibility with some previously installed system that is not able to use newer algorithms.

Encryption key: **Passphrase**

Here you can choose the type of the encryption key for this partition.

Passphrase

The encryption key will be computed[6] on the basis of a passphrase which you will be able to enter later in the process.

Random key

A new encryption key will be generated from random data each time you try to bring up the encrypted partition. In other words: on every shutdown the content of the partition will be lost as the key is deleted from memory. (Of course, you could try to guess the key with a brute force attack, but unless there is an unknown weakness in the cipher algorithm, it is not achievable in our lifetime.)

Random keys are useful for swap partitions because you do not need to bother yourself with remembering the passphrase or wiping sensitive information from the swap partition before shutting down your computer. However, it also means that you will *not* be able to use the "suspend-to-disk" functionality offered by newer Linux kernels as it will be impossible (during a subsequent boot) to recover the suspended data written to the swap partition.

6. Using a passphrase as the key currently means that the partition will be set up using LUKS (https://gitlab.com/cryptsetup/cryptsetup).

Erase data: **yes**

Determines whether the content of this partition should be overwritten with random data before setting up the encryption. This is recommended because it might otherwise be possible for an attacker to discern which parts of the partition are in use and which are not. In addition, this will make it harder to recover any leftover data from previous installations[7].

After you have selected the desired parameters for your encrypted partitions, return back to the main partitioning menu. There should now be a new menu item called Configure encrypted volumes. After you select it, you will be asked to confirm the deletion of data on partitions marked to be erased and possibly other actions such as writing a new partition table. For large partitions this might take some time.

Next you will be asked to enter a passphrase for partitions configured to use one. Good passphrases should be longer than 8 characters, should be a mixture of letters, numbers and other characters and should not contain common dictionary words or information easily associable with you (such as birthdates, hobbies, pet names, names of family members or relatives, etc.).

Warning

Before you input any passphrases, you should have made sure that your keyboard is configured correctly and generates the expected characters. If you are unsure, you can switch to the second virtual console and type some text at the prompt. This ensures that you won't be surprised later, e.g. by trying to input a passphrase using a qwerty keyboard layout when you used an azerty layout during the installation. This situation can have several causes. Maybe you switched to another keyboard layout during the installation, or the selected keyboard layout might not have been set up yet when entering the passphrase for the root file system.

If you selected to use methods other than a passphrase to create encryption keys, they will be generated now. Because the kernel may not have gathered a sufficient amount of entropy at this early stage of the installation, the process may take a long time. You can help speed up the process by generating entropy: e.g. by pressing random keys, or by switching to the shell on the second virtual console and generating some network and disk traffic (downloading some files, feeding big files into `/dev/null`, etc.). This will be repeated for each partition to be encrypted.

After returning to the main partitioning menu, you will see all encrypted volumes as additional partitions which can be configured in the same way as ordinary partitions. The following example shows a volume encrypted via dm-crypt.

```
Encrypted volume (sda2_crypt) - 115.1 GB Linux device-mapper
     #1 115.1 GB  F ext3
```

Now is the time to assign mount points to the volumes and optionally change the file system types if the defaults do not suit you.

Pay attention to the identifiers in parentheses (*sda2_crypt* in this case) and the mount points you assigned to each encrypted volume. You will need this information later when booting the new system. The differences between the ordinary boot process and the boot process with encryption involved will be covered later in Section 7.2.

Once you are satisfied with the partitioning scheme, continue with the installation.

7. It is believed that the guys from three-letter agencies can restore the data even after several rewrites of the magnetooptical media, though.

6.3.4. Installing the Base System

Although this stage is the least problematic, it consumes a significant fraction of the install because it downloads, verifies and unpacks the whole base system. If you have a slow computer or network connection, this could take some time.

During installation of the base system, package unpacking and setup messages are redirected to **tty4**. You can access this terminal by pressing **Left Alt**-**F4**; get back to the main installer process with **Left Alt**-**F1**.

The unpack/setup messages generated during this phase are also saved in `/var/log/syslog`. You can check them there if the installation is performed over a serial console.

As part of the installation, a Linux kernel will be installed. At the default priority, the installer will choose one for you that best matches your hardware. In lower priority modes, you will be able to choose from a list of available kernels.

When packages are installed using the package management system, it will by default also install packages that are recommended by those packages. Recommended packages are not strictly required for the core functionality of the selected software, but they do enhance that software and should, in the view of the package maintainers, normally be installed together with that software.

> **Note:** For technical reasons packages installed during the installation of the base system are installed without their “Recommends”. The rule described above only takes effect after this point in the installation process.

6.3.5. Installing Additional Software

At this point you have a usable but limited system. Most users will want to install additional software on the system to tune it to their needs, and the installer allows you to do so. This step can take even longer than installing the base system if you have a slow computer or network connection.

6.3.5.1. Configuring apt

Some of the tools used to install packages on a Ubuntu system are the programs called **apt-get** or just **apt**, from the `apt` package[8]. Other front-ends for package management, like **aptitude** and **synaptic**, are also in use. These front-ends are recommended for new users, since they integrate some additional features (package searching and status checks) in a nice user interface.

The **apt** and **apt-get** front-end must be configured so that it knows from where to retrieve packages. The results of this configuration are written to the file `/etc/apt/sources.list`. You can examine and edit this file to your liking after the installation is complete.

If you are installing at default priority, the installer will largely take care of the configuration automatically, based on the installation method you are using and possibly using choices made earlier in the installation. In most cases the installer will automatically add a security mirror and, if you are installing the stable distribution, a mirror for the “stable-updates” service.

8. Note that the program which actually installs the packages is called **dpkg**. However, this program is more of a low-level tool. **apt-get** and **apt** are higher-level tools, which will invoke **dpkg** as appropriate. They know how to retrieve packages from your CD, the network, or wherever. They are also able to automatically install other packages which are required to make the package you’re trying to install work correctly.

If you are installing at a lower priority (e.g. in expert mode), you will be able to make more decisions yourself. You can choose whether or not to use the security and/or stable-updates services, and you can choose to add packages from the "contrib" and "non-free" sections of the archive.

6.3.5.1.1. Installing from more than one CD or DVD

If you are installing from a CD or a DVD that is part of a larger set, the installer will ask if you want to scan additional CDs or DVDs. If you have additional CDs or DVDs available, you probably want to do this so the installer can use the packages included on them.

If you do not have any additional CDs or DVDs, that is no problem: using them is not required. If you also do not use a network mirror (as explained in the next section), it can mean that not all packages belonging to the tasks you select in the next step of the installation can be installed.

> **Note:** Packages are included on CDs (and DVDs) in the order of their popularity. This means that for most uses only the first CDs in a set are needed and that only very few people actually use any of the packages included on the last CDs in a set.
>
> It also means that buying or downloading and burning a full CD set is just a waste of money as you'll never use most of them. In most cases you are better off getting only the first 3 to 8 CDs and installing any additional packages you may need from the Internet by using a mirror. The same goes for DVD sets: the first DVD, or maybe the first two DVDs will cover most needs.

If you do scan multiple CDs or DVDs, the installer will prompt you to exchange them when it needs packages from another CD/DVD than the one currently in the drive. Note that only CDs or DVDs that belong to the same set should be scanned. The order in which they are scanned does not really matter, but scanning them in ascending order will reduce the chance of mistakes.

6.3.5.1.2. Using a network mirror

One question that will be asked during most installs is whether or not to use a network mirror as a source for packages. In most cases the default answer should be fine, but there are some exceptions.

If you are *not* installing from a full CD or DVD or using a full CD/DVD image, you really should use a network mirror as otherwise you will end up with only a very minimal system. However, if you have a limited Internet connection it is best *not* to select the `desktop` task in the next step of the installation.

If you are installing from a single full CD or using a full CD image, using a network mirror is not required, but is still strongly recommended because a single CD or image contains only a fairly limited number of packages. If you have a limited Internet connection it may still be best to *not* select a network mirror here, but to finish the installation using only what's available on the CD and selectively install additional packages after the installation (i.e. after you have rebooted into the new system).

If you are installing from a DVD or using a DVD image, any packages needed during the installation should be present on the first DVD. The same is true if you have scanned multiple CDs as explained in the previous section. Use of a network mirror is optional.

One advantage of adding a network mirror is that updates that have occurred since the CD/DVD set was created and have been included in a point release, will become available for installation, thus extending the life of your CD/DVD set without compromising the security or stability of the installed system.

In summary: selecting a network mirror is generally a good idea, except if you do not have a good Internet connection. If the current version of a package is available from CD/DVD, the installer will always use that. The amount of data that will be downloaded if you do select a mirror thus depends on

1. the tasks you select in the next step of the installation,
2. which packages are needed for those tasks,
3. which of those packages are present on the CDs or DVDs you have scanned, and
4. whether any updated versions of packages included on the CDs or DVDs are available from a mirror (either a regular package mirror, or a mirror for security or stable-updates).

Note that the last point means that, even if you choose not to use a network mirror, some packages may still be downloaded from the Internet if there is a security or stable-updates update available for them and those services have been configured.

6.3.5.1.3. Choosing a network mirror

If you have selected to use a network mirror during the installation (optional for CD/DVD installs, required for netboot images), you will be presented with a list of geographically nearby (and therefore hopefully fast) network mirrors, based upon your country selection earlier in the installation process. Choosing the offered default is usually fine.

A mirror can also be specified by hand by choosing "enter information manually" . You can then specify a mirror host name and an optional port number.

6.3.5.2. Selecting and Installing Software

During the installation process, you may be given the opportunity to select additional software to install. Rather than picking individual software packages from the 58805 available packages, this stage of the installation process focuses on selecting and installing predefined collections of software to quickly set up your computer to perform various tasks.

So, you have the ability to choose *tasks* first, and then add on more individual packages later. These tasks loosely represent a number of different jobs or things you want to do with your computer, such as "Desktop environment", "Mail server", "Web server", or "Print server"[9]. Section D.2 lists the space requirements for the available tasks.

Some tasks may be pre-selected based on the characteristics of the computer you are installing. If you disagree with these selections you can deselect them. You can even opt to install no tasks at all at this point.

> **Tip:** In the standard user interface of the installer, you can use the space bar to toggle selection of a task.

9. You should know that to present this list, the installer is merely invoking the **tasksel** program. It can be run at any time after installation to install more packages (or remove them), or you can use more fine-grained tools like **aptitude**, **apt-get** or **apt**. If you are looking for a specific single package, after installation is complete, simply run **`apt install`** ***`package`***, where *`package`* is the name of the package you are looking for.

The various server tasks will install software roughly as follows. Web server: `apache2`; Print server: `cups`;

The "Standard system" task will install any package that has a priority "standard". This includes a lot of common utilities that are normally available on any Linux or Unix system. You should leave this task selected unless you know what you are doing and want a really minimal system.

If during language selection a default locale other than the "C" locale was selected, **tasksel** will check if any localization tasks are defined for that locale and will automatically try to install relevant localization packages. This includes for example packages containing word lists or special fonts for your language. If a desktop environment was selected, it will also install appropriate localization packages for that (if available).

Once you've selected your tasks, select Continue. At this point, **aptitude** will install the packages that are part of the selected tasks. If a particular program needs more information from the user, it will prompt you during this process.

You should be aware that especially the Desktop task is very large. Especially when installing from a normal CD-ROM in combination with a mirror for packages not on the CD-ROM, the installer may want to retrieve a lot of packages over the network. If you have a relatively slow Internet connection, this can take a long time. There is no option to cancel the installation of packages once it has started.

Even when packages are included on the CD-ROM, the installer may still retrieve them from the mirror if the version available on the mirror is more recent than the one included on the CD-ROM. If you are installing the stable distribution, this can happen after a point release (an update of the original stable release); if you are installing the testing distribution this will happen if you are using an older image.

6.3.6. Making Your System Bootable

If you are installing a diskless workstation, obviously, booting off the local disk isn't a meaningful option, and this step will be skipped.

6.3.6.1. Detecting other operating systems

Before a boot loader is installed, the installer will attempt to probe for other operating systems which are installed on the machine. If it finds a supported operating system, you will be informed of this during the boot loader installation step, and the computer will be configured to boot this other operating system in addition to Ubuntu.

Note that multiple operating systems booting on a single machine is still something of a black art. The automatic support for detecting and setting up boot loaders to boot other operating systems varies by architecture and even by subarchitecture. If it does not work you should consult your boot manager's documentation for more information.

6.3.6.2. Making the system bootable with flash-kernel

As there is no common firmware interface on all ARM platforms, the steps required to make the system bootable on ARM devices are highly device-dependent. Ubuntu uses a tool called **flash-kernel** to take care of this. Flash-kernel contains a database which describes the particular operations that are required to make the system bootable on various devices. It detects whether the current device is supported, and if yes, performs the necessary operations.

On devices which boot from internal NOR- or NAND-flash memory, flash-kernel writes the kernel and the initial ramdisk to this internal memory. This method is particularly common on older armel devices. Please note that most of these devices do not allow having multiple kernels and ramdisks in their internal flash memory, i.e. running flash-kernel on them usually overwrites the previous contents of the flash memory!

For ARM systems that use U-Boot as their system firmware and boot the kernel and the initial ramdisk from external storage media (such as MMC/SD-cards, USB mass storage devices or IDE/SATA hard-disks), flash-kernel generates an appropriate boot script to allow autobooting without user interaction.

6.3.6.3. Continue Without Boot Loader

This option can be used to complete the installation even when no boot loader is to be installed, either because the arch/subarch doesn't provide one, or because none is desired (e.g. you will use existing boot loader).

If you plan to manually configure your bootloader, you should check the name of the installed kernel in `/target/boot`. You should also check that directory for the presence of an *initrd*; if one is present, you will probably have to instruct your bootloader to use it. Other information you will need are the disk and partition you selected for your `/` filesystem and, if you chose to install `/boot` on a separate partition, also your `/boot` filesystem.

6.3.7. Finishing the Installation

This is the last step in the Ubuntu installation process during which the installer will do any last minute tasks. It mostly consists of tidying up after the `debian-installer`.

6.3.7.1. Setting the System Clock

The installer may ask you if the computer's clock is set to UTC. Normally this question is avoided if possible and the installer tries to work out whether the clock is set to UTC based on things like what other operating systems are installed.

In expert mode you will always be able to choose whether or not the clock is set to UTC.

At this point `debian-installer` will also attempt to save the current time to the system's hardware clock. This will be done either in UTC or local time, depending on the selection that was just made.

6.3.7.2. Reboot the System

You will be prompted to remove the boot media (CD, floppy, etc) that you used to boot the installer. After that the system will be rebooted into your new Ubuntu system.

6.3.8. Troubleshooting

The components listed in this section are usually not involved in the installation process, but are waiting in the background to help the user in case something goes wrong.

6.3.8.1. Saving the installation logs

If the installation is successful, the logfiles created during the installation process will be automatically saved to `/var/log/installer/` on your new Ubuntu system.

Choosing Save debug logs from the main menu allows you to save the log files to a floppy disk, network, hard disk, or other (removable) media. This can be useful if you encounter fatal problems during the installation and wish to study the logs on another system or attach them to an installation report.

6.3.8.2. Using the Shell and Viewing the Logs

There are several methods you can use to get a shell while running an installation. On most systems, and if you are not installing over serial console, the easiest method is to switch to the second *virtual console* by pressing **Left Alt**-**F2**[10] (on a Mac keyboard, **Option**-**F2**). Use **Left Alt**-**F1** to switch back to the installer itself.

In case switching consoles is not an option, there is also an Execute a Shell item on the main menu that can be used to start a shell. You can get to the main menu from most dialogs by using the Go Back button one or more times. Alternatively you can also open an additional ssh installer session and execute the Execute a Shell there - see next chapter. Type **`exit`** to close the shell and return to the installer.

At this point you are booted from the RAM disk, and there is a limited set of Unix utilities available for your use. You can see what programs are available with the command **ls /bin /sbin /usr/bin /usr/sbin** and by typing **help**. The shell is a Bourne shell clone called **ash** and has some nice features like autocompletion and history.

To edit and view files, use the text editor **nano**. Log files for the installation system can be found in the `/var/log` directory.

> **Note:** Although you can do basically anything in a shell that the available commands allow you to do, the option to use a shell is really only there in case something goes wrong and for debugging.
>
> Doing things manually from the shell may interfere with the installation process and result in errors or an incomplete installation. In particular, you should always use let the installer activate your swap partition and not do this yourself from a shell.

6.3.9. Installation Over the Network

One of the more interesting components is *network-console*. It allows you to do a large part of the installation over the network via SSH. The use of the network implies you will have to perform the first steps of the installation from the console, at least to the point of setting up the networking. (Although you can automate that part with Section 4.4.)

This component is not loaded into the main installation menu by default, so you have to explicitly ask for it. If you are installing from CD, you need to boot with medium priority or otherwise invoke the main installation menu and choose Load installer components from CD and from the list of additional components select network-console: Continue installation remotely using SSH. Successful load is indicated by a new menu entry called Continue installation remotely using SSH.

10. That is: press the **Alt** key on the left-hand side of the **space bar** and the **F2** function key at the same time.

After selecting this new entry, you will be asked for a temporary password to be used for connecting to the installation system and for its confirmation. That's all. Now you should see a screen which instructs you to login remotely as the user *installer* with the password you just provided. Another important detail to notice on this screen is the fingerprint of this system. You need to transfer the fingerprint securely to the person who will continue the installation remotely.

Should you decide to continue with the installation locally, you can always press **Enter**, which will bring you back to the main menu, where you can select another component.

Now let's switch to the other side of the wire. As a prerequisite, you need to configure your terminal for UTF-8 encoding, because that is what the installation system uses. If you do not, remote installation will be still possible, but you may encounter strange display artefacts like destroyed dialog borders or unreadable non-ascii characters. Establishing a connection with the installation system is as simple as typing:

```
$ ssh -l installer install_host
```

Where *install_host* is either the name or IP address of the computer being installed. Before the actual login the fingerprint of the remote system will be displayed and you will have to confirm that it is correct.

> **Note:** The **ssh** server in the installer uses a default configuration that does not send keep-alive packets. In principle, a connection to the system being installed should be kept open indefinitely. However, in some situations — depending on your local network setup — the connection may be lost after some period of inactivity. One common case where this can happen is when there is some form of Network Address Translation (NAT) somewhere between the client and the system being installed. Depending on at which point of the installation the connection was lost, you may or may not be able to resume the installation after reconnecting.
>
> You may be able to avoid the connection being dropped by adding the option **-o ServerAliveInterval=*value*** when starting the **ssh** connection, or by adding that option in your **ssh** configuration file. Note however that in some cases adding this option may also *cause* a connection to be dropped (for example if keep-alive packets are sent during a brief network outage, from which **ssh** would otherwise have recovered), so it should only be used when needed.

> **Note:** If you install several computers in turn and they happen to have the same IP address or hostname, **ssh** will refuse to connect to such host. The reason is that it will have different fingerprint, which is usually a sign of a spoofing attack. If you are sure this is not the case, you will need to delete the relevant line from `~/.ssh/known_hosts` [11] and try again.

After the login you will be presented with an initial screen where you have two possibilities called Start menu and Start shell. The former brings you to the main installer menu, where you can continue with the installation as usual. The latter starts a shell from which you can examine and possibly fix the remote system. You should only start one SSH session for the installation menu, but may start multiple sessions for shells.

11. The following command will remove an existing entry for a host: **ssh-keygen -R <*hostname*|*IP address*>**.

Warning

After you have started the installation remotely over SSH, you should not go back to the installation session running on the local console. Doing so may corrupt the database that holds the configuration of the new system. This in turn may result in a failed installation or problems with the installed system.

6.4. Loading Missing Firmware

As described in Section 2.2, some devices require firmware to be loaded. In most cases the device will not work at all if the firmware is not available; sometimes basic functionality is not impaired if it is missing and the firmware is only needed to enable additional features.

If a device driver requests firmware that is not available, `debian-installer` will display a dialog offering to load the missing firmware. If this option is selected, `debian-installer` will scan available devices for either loose firmware files or packages containing firmware. If found, the firmware will be copied to the correct location (`/lib/firmware`) and the driver module will be reloaded.

> **Note:** Which devices are scanned and which file systems are supported depends on the architecture, the installation method and the stage of the installation. Especially during the early stages of the installation, loading the firmware is most likely to succeed from a FAT-formatted floppy disk or USB stick.

Note that it is possible to skip loading the firmware if you know the device will also function without it, or if the device is not needed during the installation.

`debian-installer` only prompts for firmware needed by kernel modules loaded during the installation. Not all drivers are included in `debian-installer`, in particular radeon is not, so this implies that the capabilities of some devices may be no different at the end of the installation from what they were at the beginning. Consequently, some of your hardware may not be being used to its full potential. If you suspect this is the case, or are just curious, it is not a bad idea to check the output of the **dmesg** command on the newly booted system and search for “firmware”.

6.4.1. Preparing a medium

To prepare a USB stick (or other medium like a hard drive partition, or floppy disk), the firmware files or packages must be placed in either the root directory or a directory named `/firmware` of the file system on the medium. The recommended file system to use is FAT as that is most certain to be supported during the early stages of the installation.

Tarballs and zip files containing current packages for the most common firmware are available from:

- http://cdimage.debian.org/cdimage/unofficial/non-free/firmware/

Just download the tarball or zip file for the correct release and unpack it to the file system on the medium.

If the firmware you need is not included in the tarball, you can also download specific firmware packages from the (non-free section of the) archive. The following overview should list most available

firmware packages but is not guaranteed to be complete and may also contain non-firmware packages:

- http://packages.debian.org/search?keywords=firmware

It is also possible to copy individual firmware files to the medium. Loose firmware could be obtained for example from an already installed system or from a hardware vendor.

6.4.2. Firmware and the Installed System

Any firmware loaded during the installation will be copied automatically to the installed system. In most cases this will ensure that the device that requires the firmware will also work correctly after the system is rebooted into the installed system. However, if the installed system runs a different kernel version from the installer there is a slight chance that the firmware cannot be loaded due to version skew.

If the firmware was loaded from a firmware package, `debian-installer` will also install this package for the installed system and will automatically add the non-free section of the package archive in APT's `sources.list`. This has the advantage that the firmware should be updated automatically if a new version becomes available.

If loading the firmware was skipped during the installation, the relevant device will probably not work with the installed system until the firmware (package) is installed manually.

> **Note:** If the firmware was loaded from loose firmware files, the firmware copied to the installed system will *not* be automatically updated unless the corresponding firmware package (if available) is installed after the installation is completed.

Chapter 7. Booting Into Your New Ubuntu System

7.1. The Moment of Truth

Your system's first boot on its own power is what electrical engineers call the "smoke test".

If the system fails to start up correctly, don't panic. If the installation was successful, chances are good that there is only a relatively minor problem that is preventing the system from booting Ubuntu. In most cases such problems can be fixed without having to repeat the installation. One available option to fix boot problems is to use the installer's built-in rescue mode (see Section 8.7).

7.2. Mounting encrypted volumes

If you created encrypted volumes during the installation and assigned them mount points, you will be asked to enter the passphrase for each of these volumes during the boot.

For partitions encrypted using dm-crypt you will be shown the following prompt during the boot:

```
Starting early crypto disks... part_crypt(starting)
Enter LUKS passphrase:
```

In the first line of the prompt, *part* is the name of the underlying partition, e.g. sda2 or md0. You are now probably wondering *for which volume* you are actually entering the passphrase. Does it relate to your `/home`? Or to `/var`? Of course, if you have just one encrypted volume, this is easy and you can just enter the passphrase you used when setting up this volume. If you set up more than one encrypted volume during the installation, the notes you wrote down as the last step in Section 6.3.3.6 come in handy. If you did not make a note of the mapping between *part*`_crypt` and the mount points before, you can still find it in `/etc/crypttab` and `/etc/fstab` of your new system.

The prompt may look somewhat different when an encrypted root file system is mounted. This depends on which initramfs generator was used to generate the initrd used to boot the system. The example below is for an initrd generated using `initramfs-tools`:

```
Begin: Mounting root file system... ...
Begin: Running /scripts/local-top ...
Enter LUKS passphrase:
```

No characters (even asterisks) will be shown while entering the passphrase. If you enter the wrong passphrase, you have two more tries to correct it. After the third try the boot process will skip this volume and continue to mount the next filesystem. Please see Section 7.2.1 for further information.

After entering all passphrases the boot should continue as usual.

7.2.1. Troubleshooting

If some of the encrypted volumes could not be mounted because a wrong passphrase was entered, you will have to mount them manually after the boot. There are several cases.

- The first case concerns the root partition. When it is not mounted correctly, the boot process will halt and you will have to reboot the computer to try again.
- The easiest case is for encrypted volumes holding data like `/home` or `/srv`. You can simply mount them manually after the boot.

  However for dm-crypt this is a bit tricky. First you need to register the volumes with device mapper by running:

  ```
  # /etc/init.d/cryptdisks start
  ```

  This will scan all volumes mentioned in `/etc/crypttab` and will create appropriate devices under the `/dev` directory after entering the correct passphrases. (Already registered volumes will be skipped, so you can repeat this command several times without worrying.) After successful registration you can simply mount the volumes the usual way:

  ```
  # mount /mount_point
  ```

- If any volume holding noncritical system files could not be mounted (`/usr` or `/var`), the system should still boot and you should be able to mount the volumes manually like in the previous case. However, you will also need to (re)start any services usually running in your default runlevel because it is very likely that they were not started. The easiest way is to just reboot the computer.

7.3. Log In

Once your system boots, you'll be presented with the login prompt. Log in using the personal login and password you selected during the installation process. Your system is now ready for use.

If you are a new user, you may want to explore the documentation which is already installed on your system as you start to use it. There are currently several documentation systems, work is proceeding on integrating the different types of documentation. Here are a few starting points.

Documentation accompanying programs you have installed can be found in `/usr/share/doc/`, under a subdirectory named after the program (or, more precise, the Ubuntu package that contains the program). However, more extensive documentation is often packaged separately in special documentation packages that are mostly not installed by default. For example, documentation about the package management tool **apt** can be found in the packages `apt-doc` or `apt-howto`.

In addition, there are some special folders within the `/usr/share/doc/` hierarchy. Linux HOWTOs are installed in *.gz* (compressed) format, in `/usr/share/doc/HOWTO/en-txt/`. After installing `dhelp`, you will find a browsable index of documentation in `/usr/share/doc/HTML/index.html`.

One easy way to view these documents using a text based browser is to enter the following commands:

```
$ cd /usr/share/doc/
$ w3m .
```

The dot after the **w3m** command tells it to show the contents of the current directory.

You can also type **info** *command* or **man** *command* to see documentation on most commands available at the command prompt. Typing **help** will display help on shell commands. And typing a command followed by **--help** will usually display a short summary of the command's usage. If a command's

results scroll past the top of the screen, type **| more** after the command to cause the results to pause before scrolling past the top of the screen. To see a list of all commands available which begin with a certain letter, type the letter and then two tabs.

Even the usual login screen shows lot's of useful information regarding documentation, support, basic system information and last login data:

```
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-22-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Jun  5 17:17:17 EDT 2018

  System load:    0.01              Processes:              180
  Usage of /home: 0.2% of 20.18GB   Users logged in:        1
  Memory usage:   4%                IP address for encc000: 169.254.232.169
  Swap usage:     0%                IP address for virbr0:  192.168.122.1


0 packages can be updated.
0 updates are security updates.

Last login: Tue Jun  5 17:46:34 2018 from 10.177.178.179
```

Chapter 8. Next Steps and Where to Go From Here

8.1. Shutting down the system

To shut down a running Ubuntu system, you must not reboot with the reset switch on the front or back of your computer, or just turn off the computer. Ubuntu should be shut down in a controlled manner, otherwise files might get lost and/or disk damage might occur.

Alternatively you can press the key combination **Ctrl**-**Alt**-**Del** . If the key combinations do not work, a last option is to log in as root and type the necessary commands.

Just use **reboot** to reboot the system and **halt** to halt the system without powering it off [1]. To power off the machine, use **poweroff** or **shutdown -h now**. The systemd init system provides additional commands that perform the same functions; for example **systemctl reboot** or **systemctl poweroff**.

8.2. If You Are New to Unix

If you are new to Unix, you probably should go out and buy some books and do some reading. A lot of valuable information can also be found in the Debian Reference (http://www.debian.org/doc/user-manuals#quick-reference). This list of Unix FAQs (http://www.faqs.org/faqs/unix-faq/) contains a number of UseNet documents which provide a nice historical reference.

Linux is an implementation of Unix. The Linux Documentation Project (LDP) (http://www.tldp.org/) collects a number of HOWTOs and online books relating to Linux. Most of these documents can be installed locally; just install the `doc-linux-html` package (HTML versions) or the `doc-linux-text` package (ASCII versions), then look in `/usr/share/doc/HOWTO`. International versions of the LDP HOWTOs are also available as Ubuntu packages.

8.3. Orienting Yourself to Ubuntu

Ubuntu is a little different from other distributions. Even if you're familiar with Linux in other distributions, there are things you should know about Ubuntu to help you to keep your system in a good, clean state. This chapter contains material to help you get oriented; it is not intended to be a tutorial for how to use Ubuntu, but just a very brief glimpse of the system for the very rushed.

8.3.1. Ubuntu Packaging System

The most important concept to grasp is the Ubuntu packaging system, which may be familiar to those who have already used Debian. In essence, large parts of your system should be considered under the control of the packaging system. These include:

- `/usr` (excluding `/usr/local`)
- `/var` (you could make `/var/local` and be safe in there)

1. Under the SysV init system **halt** had the same effect as **poweroff**, but with systemd as init system their effects are different.

- /bin
- /sbin
- /lib

For instance, if you replace /usr/bin/perl, that will work, but then if you upgrade your perl package, the file you put there will be replaced. Experts can get around this by putting packages on "hold" using **aptitude** or **apt-mark**.

One of the best installation methods is apt. You can use the command line versions **apt** and **apt-get**, the full-screen text version aptitude, or the graphical version synaptic.

> **Note:** apt will also let you merge main, contrib, and non-free so you can have export-restricted packages as well as standard versions.

8.3.2. Additional Software Available for Ubuntu

There are official and unofficial software repositories that are not enabled in the default Ubuntu install. These contain software which many find important and expect to have. Information on these additional repositories can be found on the Ubuntu Wiki page titled Ubuntu documentation - Repositories (https://help.ubuntu.com/community/Repositories).

8.3.3. Application Version Management

Alternative versions of applications are managed by update-alternatives. If you are maintaining multiple versions of your applications, read the update-alternatives man page.

8.3.4. Cron Job Management

Any jobs under the purview of the system administrator should be in /etc, since they are configuration files. If you have a root cron job for daily, weekly, or monthly runs, put them in /etc/cron.{daily,weekly,monthly}. These are invoked from /etc/crontab, and will run in alphabetic order, which serializes them.

On the other hand, if you have a cron job that (a) needs to run as a special user, or (b) needs to run at a special time or frequency, you can use either /etc/crontab, or, better yet, /etc/cron.d/whatever. These particular files also have an extra field that allows you to stipulate the user account under which the cron job runs.

In either case, you just edit the files and cron will notice them automatically. There is no need to run a special command. For more information see cron(8), crontab(5), and /usr/share/doc/cron/README.Debian.

8.4. Further Reading and Information

If you need information about a particular program, you should first try **man *program***, or **info *program***.

There is lots of useful documentation in `/usr/share/doc` as well. In particular, `/usr/share/doc/HOWTO` and `/usr/share/doc/FAQ` contain lots of interesting information. To submit bugs, look at `/usr/share/doc/debian/bug*`. To read about Debian/Ubuntu-specific issues for particular programs, look at `/usr/share/doc/(package name)/README.Debian`.

Help on Ubuntu can be found on the Ubuntu web site (http://www.ubuntu.com/). In particular, see the Ubuntu documentation (http://help.ubuntu.com/) pages for information on a wide variety of topics. The Ubuntu mailing lists (http://lists.ubuntu.com/) and the Ubuntu Forums (http://www.ubuntuforums.org/) can be invaluable sources of help from the Ubuntu community.

A general source of information on GNU/Linux is the Linux Documentation Project (http://www.tldp.org/). There you will find the HOWTOs and pointers to other very valuable information on parts of a GNU/Linux system.

8.5. Setting Up Your System To Use E-Mail

Today, email is an important part of many people's life. As there are many options as to how to set it up, and as having it set up correctly is important for some Ubuntu utilities, we will try to cover the basics in this section.

There are three main functions that make up an e-mail system. First there is the *Mail User Agent* (MUA) which is the program a user actually uses to compose and read mails. Then there is the *Mail Transfer Agent* (MTA) that takes care of transferring messages from one computer to another. And last there is the *Mail Delivery Agent* (MDA) that takes care of delivering incoming mail to the user's inbox.

These three functions can be performed by separate programs, but they can also be combined in one or two programs. It is also possible to have different programs handle these functions for different types of mail.

On Linux and Unix systems **mutt** is historically a very popular MUA. Like most traditional Linux programs it is text based. It is often used in combination with **exim** or **sendmail** as MTA and **procmail** as MDA.

8.5.1. Default E-Mail Configuration

Even if you are planning to use a graphical mail program, it is important that a traditional MTA/MDA is also installed and correctly set up on your Ubuntu system. Reason is that various utilities running on the system[2] can send important notices by e-mail to inform the system administrator of (potential) problems or changes.

For this reason the packages `exim4` and `mutt` will be installed by default (provided you did not unselect the "standard" task during the installation). `exim4` is a combination MTA/MDA that is relatively small but very flexible. By default it will be configured to only handle e-mail local to the system itself and e-mails addressed to the system administrator (root account) will be delivered to the regular user account created during the installation[3].

When system e-mails are delivered they are added to a file in `/var/mail/account_name`. The e-mails can be read using **mutt**.

2. Examples are: **cron**, **quota**, **logcheck**, **aide**, ...
3. The forwarding of mail for root to the regular user account is configured in `/etc/aliases`. If no regular user account was created, the mail will of course be delivered to the root account itself.

8.5.2. Sending E-Mails Outside The System

As mentioned earlier, the installed Ubuntu system is only set up to handle e-mail local to the system, not for sending mail to others nor for receiving mail from others.

If you would like `exim4` to handle external e-mail, please refer to the next subsection for the basic available configuration options. Make sure to test that mail can be sent and received correctly.

If you intend to use a graphical mail program and use a mail server of your Internet Service Provider (ISP) or your company, there is not really any need to configure `exim4` for handling external e-mail. Just configure your favorite graphical mail program to use the correct servers to send and receive e-mail (how is outside the scope of this manual).

However, in that case you may need to configure individual utilities to correctly send e-mails. One such utility is **reportbug**, a program that facilitates submitting bug reports against Ubuntu packages. By default it expects to be able to use `exim4` to submit bug reports.

To correctly set up **reportbug** to use an external mail server, please run the command **reportbug --configure** and answer "no" to the question if an MTA is available. You will then be asked for the SMTP server to be used for submitting bug reports.

8.5.3. Configuring the Exim4 Mail Transport Agent

If you would like your system to also handle external e-mail, you will need to reconfigure the `exim4` package[4]:

```
# dpkg-reconfigure exim4-config
```

After entering that command (as root), you will be asked if you want split the configuration into small files. If you are unsure, select the default option.

Next you will be presented with several common mail scenarios. Choose the one that most closely resembles your needs.

internet site

> Your system is connected to a network and your mail is sent and received directly using SMTP. On the following screens you will be asked a few basic questions, like your machine's mail name, or a list of domains for which you accept or relay mail.

mail sent by smarthost

> In this scenario your outgoing mail is forwarded to another machine, called a "smarthost", which takes care of sending the message on to its destination. The smarthost also usually stores incoming mail addressed to your computer, so you don't need to be permanently online. That also means you have to download your mail from the smarthost via programs like fetchmail.
>
> In a lot of cases the smarthost will be your ISP's mail server, which makes this option very suitable for dial-up users. It can also be a company mail server, or even another system on your own network.

4. You can of course also remove `exim4` and replace it with an alternative MTA/MDA.

mail sent by smarthost; no local mail

This option is basically the same as the previous one except that the system will not be set up to handle mail for a local e-mail domain. Mail on the system itself (e.g. for the system administrator) will still be handled.

local delivery only

This is the option your system is configured for by default.

no configuration at this time

Choose this if you are absolutely convinced you know what you are doing. This will leave you with an unconfigured mail system — until you configure it, you won't be able to send or receive any mail and you may miss some important messages from your system utilities.

If none of these scenarios suits your needs, or if you need a finer grained setup, you will need to edit configuration files under the `/etc/exim4` directory after the installation is complete. More information about `exim4` may be found under `/usr/share/doc/exim4`; the file `README.Debian.gz` has further details about configuring `exim4` and explains where to find additional documentation.

Note that sending mail directly to the Internet when you don't have an official domain name, can result in your mail being rejected because of anti-spam measures on receiving servers. Using your ISP's mail server is preferred. If you still do want to send out mail directly, you may want to use a different e-mail address than is generated by default. If you use `exim4` as your MTA, this is possible by adding an entry in `/etc/email-addresses`.

8.6. Compiling a New Kernel

Why would someone want to compile a new kernel? It is often not necessary since the default kernel shipped with Ubuntu handles most configurations. Also, Ubuntu often offers several alternative kernels. So you may want to check first if there is an alternative kernel image package that better corresponds to your hardware. However, it can be useful to compile a new kernel in order to:

- handle special hardware needs, or hardware conflicts with the pre-supplied kernels
- use options of the kernel which are not supported in the pre-supplied kernels (such as high memory support)
- optimize the kernel by removing useless drivers to speed up boot time
- create a monolithic instead of a modularized kernel
- run an updated or development kernel
- learn more about linux kernels

8.6.1. Kernel Image Management

Don't be afraid to try compiling the kernel. It's fun and profitable.

To compile a kernel the Debian/Ubuntu way, you need some packages: `fakeroot`, `kernel-package`, `linux-source` and a few others which are probably already installed (see `/usr/share/doc/kernel-package/README.gz` for the complete list).

This method will make a .deb of your kernel source, and, if you have non-standard modules, make a synchronized dependent .deb of those too. It's a better way to manage kernel images; `/boot` will hold the kernel, the System.map, and a log of the active config file for the build.

Note that you don't *have* to compile your kernel the "Debian/Ubuntu way"; but we find that using the packaging system to manage your kernel is actually safer and easier. In fact, you can get your kernel sources right from Linus instead of `linux-source`, yet still use the `kernel-package` compilation method.

Note that you'll find complete documentation on using `kernel-package` under `/usr/share/doc/kernel-package`. This section just contains a brief tutorial.

Hereafter, we'll assume you have free rein over your machine and will extract your kernel source to somewhere in your home directory[5]. We'll also assume that your kernel version is 4.15. Make sure you are in the directory to where you want to unpack the kernel sources, extract them using **`tar xf /usr/src/linux-source-4.15.tar.xz`** and change to the directory `linux-source-4.15` that will have been created.

Now, you can configure your kernel. Run **`make xconfig`** if X11 is installed, configured and being run; run **`make menuconfig`** otherwise (you'll need `libncurses5-dev` installed). Take the time to read the online help and choose carefully. When in doubt, it is typically better to include the device driver (the software which manages hardware peripherals, such as Ethernet cards, SCSI controllers, and so on) you are unsure about. Be careful: other options, not related to a specific hardware, should be left at the default value if you do not understand them. Do not forget to select "Kernel module loader" in "Loadable module support" (it is not selected by default). If not included, your Ubuntu installation will experience problems.

Clean the source tree and reset the `kernel-package` parameters. To do that, do **`make-kpkg clean`**.

Now, compile the kernel: **`fakeroot make-kpkg --initrd --revision=1.0.custom kernel_image`**. The version number of "1.0" can be changed at will; this is just a version number that you will use to track your kernel builds. Likewise, you can put any word you like in place of "custom" (e.g., a host name). Kernel compilation may take quite a while, depending on the power of your machine.

Once the compilation is complete, you can install your custom kernel like any package. As root, do **`dpkg -i ../linux-image-4.15-`***`subarchitecture`***`_1.0.custom_arm64.deb`**. The *`subarchitecture`* part is an optional sub-architecture, depending on what kernel options you set. **`dpkg -i`** will install the kernel, along with some other nice supporting files. For instance, the `System.map` will be properly installed (helpful for debugging kernel problems), and `/boot/config-4.15` will be installed, containing your current configuration set. Your new kernel package is also clever enough to automatically update your boot loader to use the new kernel. If you have created a modules package, you'll need to install that package as well.

It is time to reboot the system: read carefully any warning that the above step may have produced, then **`shutdown -r now`**.

For more information on Debian/Ubuntu kernels and kernel compilation, see the Debian Linux Kernel Handbook (http://kernel-handbook.alioth.debian.org/). For more information on `kernel-package`, read the fine documentation in `/usr/share/doc/kernel-package`.

5. There are other locations where you can extract kernel sources and build your custom kernel, but this is easiest as it does not require special permissions.

8.7. Recovering a Broken System

Sometimes, things go wrong, and the system you've carefully installed is no longer bootable. Perhaps the boot loader configuration broke while trying out a change, or perhaps a new kernel you installed won't boot, or perhaps cosmic rays hit your disk and flipped a bit in `/sbin/init`. Regardless of the cause, you'll need to have a system to work from while you fix it, and rescue mode can be useful for this.

There are several options to rescue a broken Ubuntu system on arm64:

- To access rescue mode, select **`rescue`** from the boot menu (if available) or append the boot parameter **`rescue`** or **`systemd.unit=emergency.target`** to the linux kernel boot entry of the boot loader.

  While booting the system enter the boot loader menu:

```
                             GNU GRUB  version 2.02

 *Ubuntu
  Advanced options for Ubuntu


      Use the ↑ and ↓ keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the commands
      before booting or 'c' for a command-line.
```

  Then type **e** to edit the boot loader configuration and entries, navigate to your prefered Linux kernel line and append either **`rescue`** or **`systemd.unit=emergency.target`**.

```
                             GNU GRUB  version 2.02

                insmod ext2
                set root='hd0,gpt2'or Ubuntu
                if [ x$feature_platform_search_hint = xy ]; then
                  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,g\
pt2 --hint-efi=hd0,gpt2 --hint-baremetal=ahci0,gpt2  ea967ae0-7519-11eB-85b\
d-5254008bdef4
                else
                  search --no-floppy --fs-uuid --set=root ea967ae0-7519-11e\
8-85bd-5254008bdef4
                fi
                echo        'Loading Linux 4.15.0-23-generic ...'
                linux        /boot/vmlinuz-4.15.0-23-generic root=UUID=ea96\
7ae0-7519-11eB-85bd-5254008bdef4 ro  maybe-ubiquity rescue

      Minimum Emacs-like screen editing is supported. TAB lists
      completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for
      a command-line or ESC to discard edits and return to the GRUB menu.
```

  Then press either Ctrl-x or F10 to boot with the modified entry and the system will enter the rescue mode.

```
...
[  OK  ] Started Update UTMP about System Runlevel Changes.
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
```

```
to boot into default mode.
Press Enter for maintenance
(or press Control-D to continue):
root@ubuntu:~#
```

- Alternatively the installer can be booted with the **rescue=true** boot parameter. You'll be shown the first few screens of the installer, with a note in the corner of the display to indicate that this is rescue mode, not a full installation. Don't worry, your system is not about to be overwritten! Rescue mode simply takes advantage of the hardware detection facilities available in the installer to ensure that your disks, network devices, and so on are available to you while repairing your system.

- A trivial option is to just boot the standard installation kernel and initrd without any additional kernel parameter, and select from the inital screen the Start shell entry.

  ```
                              [!!] Configuring d-i

  This is the network console for the Debian installer. From here, you
  may start the Debian installer, or execute an interactive shell.

  To return to this menu, you will need to log in again.

  Network console option:

                      Start installer
                      Start installer (expert mode)
                      Start shell
  ```

  The functionality in the `debian-installer` shell is limited, however, it can still act as a rescue system to fix a broken installation. While not using the rescue mode, be careful to not accidentially repartition or format any disk which may cause data loss.

In either case, after you exit the shell, the system will reboot.

Finally, note that repairing broken systems can be difficult, and this manual does not attempt to go into all the things that might have gone wrong or how to fix them. If you have problems, consult an expert.

Appendix A. Installation Howto

This document describes how to install Ubuntu 18.04 “Bionic Beaver” for the 64-bit ARM (“arm64”). It is a quick walkthrough of the installation process which should contain all the information you will need for most installs. When more information can be useful, we will link to more detailed explanations in other parts of this document.

A.1. Booting the installer

For more information on where to get CDs, see Section 4.1.

Some installation methods require other images than CD images. The `debian-installer` home page (http://www.debian.org/devel/debian-installer/) has links to other images. Section 4.2.1 explains how to find images on Ubuntu mirrors.

The easiest way is probably to download the (CDROM) image, because the files most people need are there on the image.

The subsections below will give the details about which image(s) you should get for each possible means of installation.

A.1.1. Booting from CDROM

First of all download the image for your architecture and burn it to a CD/DVD.

A.1.2. Booting from network

It’s also possible to boot the Ubuntu installer completely from the net. The various methods to netboot depend on your architecture and netboot setup. The files in `netboot/` can be used to netboot the Ubuntu installer.

A.2. Installation

Once the installer starts, you will be greeted with an initial screen. Press **Enter** to boot, or read the instructions for other boot methods and parameters (see Section 5.3).

After a while you will be asked to select your language. Use the arrow keys to pick a language and press **Enter** to continue. Next you’ll be asked to select your country, with the choices including countries where your language is spoken. If it’s not on the short list, a list of all the countries in the world is available.

You may be asked to confirm your keyboard layout. Choose the default unless you know better.

Now sit back while the installer detects some of your hardware, and loads the rest of itself from CD, USB, etc.

Next the installer will try to detect your network hardware and set up networking by DHCP. If you are not on a network or do not have DHCP, you will be given the opportunity to configure the network manually.

The next step is setting up your clock and time zone. The installer will try to contact a time server on the Internet to ensure the clock is set correctly. The time zone is based on the country selected earlier and the installer will only ask to select one if a country has multiple zones.

Setting up your clock and time zone is followed by the creation of user accounts. By default you are asked to provide a password for the "root" (administrator) account and information necessary to create one regular user account. If you choose "No" on the Allow login as root? question or if you do not specify a password for the "root" user this account will be disabled, but the **sudo** package will be installed later to enable administrative tasks to be carried out on the new system.

Now it is time to partition your disks. First you will be given the opportunity to automatically partition either an entire drive, or available free space on a drive (see Section 6.3.3.2). This is recommended for new users or anyone in a hurry. If you do not want to autopartition, choose Manual from the menu.

If you want to customize the partition layout, choose Manually edit partition table from the menu, and the next screen will show you your partition table, how the partitions will be formatted, and where they will be mounted. Select a partition to modify or delete it. Remember that since 18.04 a swap file is the default, rather than a swap partition. But you are still able to assign a partition for swap space. Remember to mount one partition on / (root). For more detailed information on how to use the partitioner, please refer to Section 6.3.3; the appendix Appendix C has more general information about partitioning.

Now the installer formats your partitions and starts to install the base system, which can take a while. That is followed by installing a kernel.

The base system that was installed earlier is a working, but very minimal installation. To make the system more functional the next step allows you to install additional packages by selecting tasks. Before packages can be installed `apt` needs to be configured as that defines from where the packages will be retrieved. The "Standard system" task will be selected by default and should normally be installed. See Section 6.3.5.2 for additional information about this step.

The last step is to install a boot loader. If the installer detects other operating systems on your computer, it will add them to the boot menu and let you know.

The installer will now tell you that the first stage of installation has finished. Remove the CD (if needed) and hit **Enter** to reboot your machine. It should boot up into the newly installed system and allow you to log in. This is explained in Chapter 7.

If you need more information on the install process, see Chapter 6.

A.3. And finally. . .

We hope that your Ubuntu installation is pleasant and that you find Ubuntu useful. You might want to read Chapter 8.

Appendix B. Automating the installation using preseeding

This appendix explains how to preseed answers to questions in `debian-installer` to automate your installation.

The configuration fragments used in this appendix are also available as an example preconfiguration file from ../example-preseed.txt.

B.1. Introduction

Preseeding provides a way to set answers to questions asked during the installation process, without having to manually enter the answers while the installation is running. This makes it possible to fully automate most types of installation and even offers some features not available during normal installations.

Preseeding is not required. If you use an empty preseed file, the installer will behave just the same way as in a normal manual installation. Each question you preseed will (if you got it right!) modify the installation in some way from that baseline.

B.1.1. Preseeding methods

There are three methods that can be used for preseeding: *initrd*, *file* and *network*. Initrd preseeding will work with any installation method and supports preseeding of more things, but it requires the most preparation. File and network preseeding each can be used with different installation methods.

The following table shows which preseeding methods can be used with which installation methods.

Installation method	initrd	file	network
CD/DVD	yes	yes	yes[a]
netboot	yes	no	yes
hd-media	yes	yes	yes[a]

Notes:
a. but only if you have network access, and set `preseed/url` appropriately

An important difference between the preseeding methods is the point at which the preconfiguration file is loaded and processed. For initrd preseeding this is right at the start of the installation, before the first question is even asked. Preseeding from the kernel command line happens just after. It is thus possible to override configuration set in the initrd by editing the kernel command line (either in the bootloader configuration or manually at boot time for bootloaders that allow it). For file preseeding this is after the CD or CD image has been loaded. For network preseeding it is only after the network has been configured.

> **Important:** Obviously, any questions that have been processed before the preconfiguration file is loaded cannot be preseeded (this will include questions that are only displayed at medium or low priority, like the first hardware detection run). A not so convenient way to avoid these questions from being asked is to preseed them through the boot parameters, as described in Section B.2.2.

Important: In order to easily avoid the questions that would normally appear before the preseeding occurs, you can start the installer in "auto" mode. This delays questions that would normally be asked too early for preseeding (i.e. language, country and keyboard selection) until after the network comes up, thus allowing them to be preseeded. It also runs the installation at critical priority, which avoids many unimportant questions. See Section B.2.3 for details.

B.1.2. Limitations

Although most questions used by `debian-installer` can be preseeded using this method, there are some notable exceptions. You must (re)partition an entire disk or use available free space on a disk; it is not possible to use existing partitions.

B.1.3. Debconf basics

Preseeding makes use of the `debconf` framework. This framework is the preferred mechanism used in Ubuntu to interact with the user when configuring packages and also forms the heart of `debian-installer`. In the `debconf` framework questions or dialogs are based on *templates*. There are different types of templates for different types of questions. The actual questions are "generated" from templates at runtime; multiple questions can use the same template.

The following types of templates are relevant for preseeding.

- string: allows the user to type any value
- password: similar to string but the value typed is not displayed
- boolean: for yes/no or true/false type of questions
- select: allows the user to select one option from a list
- multiselect: allows the user to select zero, one or more options from a list
- note: used to display a message

In `debian-installer` templates are stored in a readable file `/var/cache/debconf/templates.dat`. This file contains all fixed text and all translations. It can also contain a default value for the template. The fixed text can include variables that will be replaced at runtime.

Another readable file `/var/cache/debconf/questions.dat` is used to store the values for variables and the answers given to questions. A question always refers to the template used to ask it. For obvious security reasons the values for questions of type "password" are stored in a separate, non-readable file in the same directory.

B.2. Using preseeding

You will first need to create a preconfiguration file and place it in the location from where you want to use it. Creating the preconfiguration file is covered later in this appendix. Putting it in the correct location is fairly straightforward for network preseeding or if you want to read the file off a floppy or usb-stick. If you want to include the file on a CD or DVD, you will have to remaster the ISO image. How to get the preconfiguration file included in the initrd is outside the scope of this document; please consult the developers' documentation for `debian-installer`.

An example preconfiguration file that you can use as basis for your own preconfiguration file is available from ../example-preseed.txt. This file is based on the configuration fragments included in this appendix.

B.2.1. Loading the preconfiguration file

If you are using initrd preseeding, you only have to make sure a file named `preseed.cfg` is included in the root directory of the initrd. The installer will automatically check if this file is present and load it.

For the other preseeding methods you need to tell the installer what file to use when you boot it. This is normally done by passing the kernel a boot parameter, either manually at boot time or by editing the bootloader configuration file

If you do specify the preconfiguration file in the bootloader configuration, you might change the configuration so you don't need to hit enter to boot the installer. For syslinux this means setting the timeout to `1` in `syslinux.cfg`.

To make sure the installer gets the right preconfiguration file, you can optionally specify a checksum for the file. Currently this needs to be a md5sum, and if specified it must match the preconfiguration file or the installer will refuse to use it.

Boot parameters to specify:

```
- if you're netbooting:
  preseed/url=http://host/path/to/preseed.cfg
  preseed/url/checksum=5da499872becccfeda2c4872f9171c3d
- or
  preseed/url=ftp://host/path/to/preseed.cfg
  preseed/url/checksum=5da499872becccfeda2c4872f9171c3d
- or
  preseed/url=tftp://host/path/to/preseed.cfg
  preseed/url/checksum=5da499872becccfeda2c4872f9171c3d


- if you're booting a remastered CD or image:
  preseed/file=/cdrom/preseed.cfg
  preseed/file/checksum=5da499872becccfeda2c4872f9171c3d


- if you're installing from USB media (put the preconfiguration file in the
  toplevel directory of the USB stick):
  preseed/file=/hd-media/preseed.cfg
  preseed/file/checksum=5da499872becccfeda2c4872f9171c3d
```

Note that `preseed/url` can be shortened to just `url`, `preseed/file` to just `file` and `preseed/file/checksum` to just `preseed-md5` when they are passed as boot parameters.

B.2.2. Using boot parameters to preseed questions

If a preconfiguration file cannot be used to preseed some steps, the install can still be fully automated, since you can pass preseed values on the command line when booting the installer.

Boot parameters can also be used if you do not really want to use preseeding, but just want to provide an answer for a specific question. Some examples where this can be useful are documented elsewhere in this manual.

To set a value to be used inside `debian-installer`, just pass ***`path/to/variable=value`*** for any of the preseed variables listed in the examples in this appendix. If a value is to be used to configure packages for the target system, you will need to prepend the *owner*[1] of the variable as in ***`owner:path/to/variable=value`***. If you don't specify the owner, the value for the variable will not be copied to the debconf database in the target system and thus remain unused during the configuration of the relevant package.

Normally, preseeding a question in this way will mean that the question will not be asked. To set a specific default value for a question, but still have the question asked, use "?=" instead of "=" as operator. See also Section B.5.2.

Note that some variables that are frequently set at the boot prompt have a shorter alias. If an alias is available, it is used in the examples in this appendix instead of the full variable. The `preseed/url` variable for example has been aliased as `url`. Another example is the `tasks` alias, which translates to `tasksel:tasksel/first`.

A "---" in the boot options has special meaning. Kernel parameters that appear after the last "---" may be copied into the bootloader configuration for the installed system (if supported by the installer for the bootloader). The installer will automatically filter out any options (like preconfiguration options) that it recognizes.

> **Note:** Current linux kernels (2.6.9 and later) accept a maximum of 32 command line options and 32 environment options, including any options added by default for the installer. If these numbers are exceeded, the kernel will panic (crash). (For earlier kernels, these numbers were lower.)

For most installations some of the default options in your bootloader configuration file, like `vga=normal`, may be safely removed which may allow you to add more options for preseeding.

> **Note:** It may not always be possible to specify values with spaces for boot parameters, even if you delimit them with quotes.

B.2.3. Auto mode

There are several features of the Ubuntu Installer that combine to allow fairly simple command lines at the boot prompt to result in arbitrarily complex customized automatic installs.

This is enabled by using the `Automated install` boot choice, also called `auto` for some architectures or boot methods. In this section, `auto` is thus not a parameter, it means selecting that boot choice, and appending the following boot parameters on the boot prompt.

To illustrate this, here are some examples that can be used at the boot prompt:

```
auto url=autoserver
```

This relies on there being a DHCP server that will get the machine to the point where `autoserver` can be resolved by DNS, perhaps after adding the local domain if that was provided by DHCP. If this was done at a site where the domain is `example.com`, and they

1. The owner of a debconf variable (or template) is normally the name of the package that contains the corresponding debconf template. For variables used in the installer itself the owner is "d-i". Templates and variables can have more than one owner which helps to determine whether they can be removed from the debconf database if the package is purged.

have a reasonably sane DHCP setup, it would result in the preseed file being retrieved from `http://autoserver.example.com/d-i/bionic/./preseed.cfg`.

The last part of that url (`d-i/bionic/./preseed.cfg`) is taken from `auto-install/defaultroot`. By default this includes the directory `bionic` to allow future versions to specify their own codename and let people migrate forwards in a controlled manner. The `/./` bit is used to indicate a root, relative to which subsequent paths can be anchored (for use in preseed/include and preseed/run). This allows files to be specified either as full URLs, paths starting with / that are thus anchored, or even paths relative to the location where the last preseed file was found. This can be used to construct more portable scripts where an entire hierarchy of scripts can be moved to a new location without breaking it, for example copying the files onto a USB stick when they started out on a web server. In this example, if the preseed file sets `preseed/run` to `/scripts/late_command.sh` then the file will be fetched from `http://autoserver.example.com/d-i/bionic/./scripts/late_command.sh`.

If there is no local DHCP or DNS infrastructure, or if you do not want to use the default path to `preseed.cfg`, you can still use an explicit url, and if you don't use the `/./` element it will be anchored to the start of the path (i.e. the third `/` in the URL). Here is an example that requires minimal support from the local network infrastructure:

```
auto url=http://192.168.1.2/path/to/mypreseed.file
```

The way this works is that:

- if the URL is missing a protocol, http is assumed,
- if the hostname section contains no periods, it has the domain derived from DHCP appended to it, and
- if there's no `/`'s after the hostname, then the default path is added.

In addition to specifying the url, you can also specify settings that do not directly affect the behavior of `debian-installer` itself, but can be passed through to scripts specified using `preseed/run` in the loaded preseed file. At present, the only example of this is `auto-install/classes`, which has an alias `classes`. This can be used thus:

```
auto url=example.com classes=class_A;class_B
```

The classes could for example denote the type of system to be installed, or the localization to be used.

It is of course possible to extend this concept, and if you do, it is reasonable to use the auto-install namespace for this. So one might have something like `auto-install/style` which is then used in your scripts. If you feel the need to do this, please mention it on the <`debian-boot@lists.debian.org`> mailing list so that we can avoid namespace conflicts, and perhaps add an alias for the parameter for you.

The `auto` boot choice is not yet defined on all arches, but the same effect may be achieved by simply adding the two parameters `auto=true priority=critical` to the kernel command line. The `auto` kernel parameter is an alias for `auto-install/enable` and setting it to `true` delays the locale and keyboard questions until after there has been a chance to preseed them, while `priority` is an alias for `debconf/priority` and setting it to `critical` stops any questions with a lower priority from being asked.

Additional options that may be of interest while attempting to automate an install while using DHCP are: `interface=auto netcfg/dhcp_timeout=60` which makes the machine choose the first viable NIC and be more patient about getting a reply to its DHCP query.

> **Tip:** An extensive example of how to use this framework, including example scripts and classes, can be found on the website of its developer (http://hands.com/d-i/). The examples available there also show many other nice effects that can be achieved by creative use of preconfiguration.

B.2.4. Aliases useful with preseeding

The following aliases can be useful when using (auto mode) preseeding. Note that these are simply short aliases for question names, and you always need to specify a value as well: for example, `auto=true` or `interface=eth0`.

priority	debconf/priority
fb	debian-installer/framebuffer
language	debian-installer/language
country	debian-installer/country
locale	debian-installer/locale
theme	debian-installer/theme
auto	auto-install/enable
classes	auto-install/classes
file	preseed/file
url	preseed/url
domain	netcfg/get_domain
hostname	netcfg/get_hostname
interface	netcfg/choose_interface
protocol	mirror/protocol
suite	mirror/suite
modules	anna/choose_modules
recommends	base-installer/install-recommends
tasks	tasksel:tasksel/first
desktop	tasksel:tasksel/desktop
dmraid	disk-detect/dmraid/enable
keymap	keyboard-configuration/xkb-keymap
preseed-md5	preseed/file/checksum

B.2.5. Using a DHCP server to specify preconfiguration files

It's also possible to use DHCP to specify a preconfiguration file to download from the network. DHCP allows specifying a filename. Normally this is a file to netboot, but if it appears to be an URL then installation media that support network preseeding will download the file from the URL and use it as a preconfiguration file. Here is an example of how to set it up in the dhcpd.conf for version 3 of the

ISC DHCP server (the isc-dhcp-server Ubuntu package).

```
if substring (option vendor-class-identifier, 0, 3) = "d-i" {
    filename "http://host/preseed.cfg";
}
```

Note: The above example limits this filename to DHCP clients that identify themselves as "d-i", so it will not affect regular DHCP clients, but only the installer. You can also put the text in a stanza for only one particular host to avoid preseeding all installs on your network.

A good way to use the DHCP preseeding is to only preseed values specific to your network, such as the Ubuntu mirror to use. This way installs on your network will automatically get a good mirror selected, but the rest of the installation can be performed interactively. Using DHCP preseeding to fully automate Ubuntu installs should only be done with care.

B.3. Creating a preconfiguration file

The preconfiguration file is in the format used by the **debconf-set-selections** command. The general format of a line in a preconfiguration file is:

```
<owner> <question name> <question type> <value>
```

There are a few rules to keep in mind when writing a preconfiguration file.

- Put only a single space or tab between type and value: any additional whitespace will be interpreted as belonging to the value.
- A line can be split into multiple lines by appending a backslash (“\”) as the line continuation character. A good place to split a line is after the question name; a bad place is between type and value. Split lines will be joined into a single line with all leading/trailing whitespace condensed to a single space.
- For debconf variables (templates) used only in the installer itself, the owner should be set to “d-i”; to preseed variables used in the installed system, the name of the package that contains the corresponding debconf template should be used. Only variables that have their owner set to something other than “d-i” will be propagated to the debconf database for the installed system.
- Most questions need to be preseeded using the values valid in English and not the translated values. However, there are some questions (for example in `partman`) where the translated values need to be used.
- Some questions take a code as value instead of the English text that is shown during installation.

The easiest way to create a preconfiguration file is to use the example file linked in Section B.4 as basis and work from there.

An alternative method is to do a manual installation and then, after rebooting, use the **debconf-get-selections** from the `debconf-utils` package to dump both the debconf database and the installer’s cdebconf database to a single file:

```
$ debconf-get-selections --installer > file
$ debconf-get-selections >> file
```

However, a file generated in this manner will have some items that should not be preseeded, and the example file is a better starting place for most users.

> **Note:** This method relies on the fact that, at the end of the installation, the installer's cdebconf database is saved to the installed system in `/var/log/installer/cdebconf`. However, because the database may contain sensitive information, by default the files are only readable by root.
>
> The directory `/var/log/installer` and all files in it will be deleted from your system if you purge the package `installation-report`.

To check possible values for questions, you can use **nano** to examine the files in `/var/lib/cdebconf` while an installation is in progress. View `templates.dat` for the raw templates and `questions.dat` for the current values and for the values assigned to variables.

To check if the format of your preconfiguration file is valid before performing an install, you can use the command **debconf-set-selections -c *preseed.cfg***.

B.4. Contents of the preconfiguration file (for bionic)

The configuration fragments used in this appendix are also available as an example preconfiguration file from ../example-preseed.txt.

Note that this example is based on an installation for the Intel x86 architecture. If you are installing a different architecture, some of the examples (like keyboard selection and bootloader installation) may not be relevant and will need to be replaced by debconf settings appropriate for your architecture.

Details on how the different Debian Installer components actually work can be found in Section 6.3.

B.4.1. Localization

During a normal install the questions about localization are asked first, so these values can only be preseeded via the initrd or kernel boot parameter methods. Auto mode (Section B.2.3) includes the setting of `auto-install/enable=true` (normally via the `auto` preseed alias). This delays the asking of the localisation questions, so that they can be preseeded by any method.

The locale can be used to specify both language and country and can be any combination of a language supported by `debian-installer` and a recognized country. If the combination does not form a valid locale, the installer will automatically select a locale that is valid for the selected language. To specify the locale as a boot parameter, use **`locale=en_US`**.

Although this method is very easy to use, it does not allow preseeding of all possible combinations of language, country and locale[2]. So alternatively the values can be preseeded individually. Language and country can also be specified as boot parameters.

```
# Preseeding only locale sets language, country and locale.
d-i debian-installer/locale string en_US
```

2. Preseeding `locale` to **`en_NL`** would for example result in `en_US.UTF-8` as default locale for the installed system. If e.g. `en_GB.UTF-8` is preferred instead, the values will need to be preseeded individually.

```
# The values can also be preseeded individually for greater flexibility.
#d-i debian-installer/language string en
#d-i debian-installer/country string NL
#d-i debian-installer/locale string en_GB.UTF-8
# Optionally specify additional locales to be generated.
#d-i localechooser/supported-locales multiselect en_US.UTF-8, nl_NL.UTF-8
```

Keyboard configuration consists of selecting a keymap and (for non-latin keymaps) a toggle key to switch between the non-latin keymap and the US keymap. Only basic keymap variants are available during installation. Advanced variants are available only in the installed system, through **dpkg-reconfigure keyboard-configuration**.

To specify the keymap as a boot parameter, use **`console-setup/ask_detect=false keyboard-configuration/xkb-keymap=`*`us`**. The keymap is an X layout name, as would be used in the **`XkbLayout`** option in `/etc/X11/xorg.conf`.

```
# Keyboard selection.
# Disable automatic (interactive) keymap detection.
d-i console-setup/ask_detect boolean false
d-i keyboard-configuration/xkb-keymap select us
# To select a variant of the selected layout:
#d-i keyboard-configuration/xkb-keymap select us(dvorak)
# d-i keyboard-configuration/toggle select No toggling
```

To skip keyboard configuration, preseed `keymap` with **`SKIP`**. This will result in the kernel keymap remaining active.

B.4.2. Network configuration

Of course, preseeding the network configuration won't work if you're loading your preconfiguration file from the network. But it's great when you're booting from CD or USB stick. If you are loading preconfiguration files from the network, you can pass network config parameters by using kernel boot parameters.

If you need to pick a particular interface when netbooting before loading a preconfiguration file from the network, use a boot parameter such as **`interface=`*`eth1`**.

Although preseeding the network configuration is normally not possible when using network preseeding (using "preseed/url"), you can use the following hack to work around that, for example if you'd like to set a static address for the network interface. The hack is to force the network configuration to run again after the preconfiguration file has been loaded by creating a "preseed/run" script containing the following commands:

```
kill-all-dhcp; netcfg
```

The following debconf variables are relevant for network configuration.

```
# Disable network configuration entirely. This is useful for cdrom
# installations on non-networked devices where the network questions,
```

```
# warning and long timeouts are a nuisance.
#d-i netcfg/enable boolean false

# netcfg will choose an interface that has link if possible. This makes it
# skip displaying a list if there is more than one interface.
d-i netcfg/choose_interface select auto
# To pick a particular interface instead:
#d-i netcfg/choose_interface select eth1


# To set a different link detection timeout (default is 3 seconds).
# Values are interpreted as seconds.
#d-i netcfg/link_wait_timeout string 10

# If you have a slow dhcp server and the installer times out waiting for
# it, this might be useful.
#d-i netcfg/dhcp_timeout string 60
#d-i netcfg/dhcpv6_timeout string 60

# If you prefer to configure the network manually, uncomment this line and
# the static network configuration below.
#d-i netcfg/disable_autoconfig boolean true

# If you want the preconfiguration file to work on systems both with and
# without a dhcp server, uncomment these lines and the static network
# configuration below.
#d-i netcfg/dhcp_failed note
#d-i netcfg/dhcp_options select Configure network manually

# Static network configuration.
#
# IPv4 example
#d-i netcfg/get_ipaddress string 192.168.1.42
#d-i netcfg/get_netmask string 255.255.255.0
#d-i netcfg/get_gateway string 192.168.1.1
#d-i netcfg/get_nameservers string 192.168.1.1
#d-i netcfg/confirm_static boolean true
#
# IPv6 example
#d-i netcfg/get_ipaddress string fc00::2
#d-i netcfg/get_netmask string ffff:ffff:ffff:ffff::
#d-i netcfg/get_gateway string fc00::1
#d-i netcfg/get_nameservers string fc00::1
#d-i netcfg/confirm_static boolean true

# Any hostname and domain names assigned from dhcp take precedence over
# values set here. However, setting the values still prevents the questions
# from being shown, even if values come from dhcp.
d-i netcfg/get_hostname string unassigned-hostname
d-i netcfg/get_domain string unassigned-domain

# If you want to force a hostname, regardless of what either the DHCP
# server returns or what the reverse DNS entry for the IP is, uncomment
# and adjust the following line.
#d-i netcfg/hostname string somehost

# Disable that annoying WEP key dialog.
```

```
d-i netcfg/wireless_wep string
# The wacky dhcp hostname that some ISPs use as a password of sorts.
#d-i netcfg/dhcp_hostname string radish

# If non-free firmware is needed for the network or other hardware, you can
# configure the installer to always try to load it, without prompting. Or
# change to false to disable asking.
#d-i hw-detect/load_firmware boolean true
```

Please note that **netcfg** will automatically determine the netmask if `netcfg/get_netmask` is not preseeded. In this case, the variable has to be marked as `seen` for automatic installations. Similarly, **netcfg** will choose an appropriate address if `netcfg/get_gateway` is not set. As a special case, you can set `netcfg/get_gateway` to "none" to specify that no gateway should be used.

B.4.3. Network console

```
# Use the following settings if you wish to make use of the network-console
# component for remote installation over SSH. This only makes sense if you
# intend to perform the remainder of the installation manually.
#d-i anna/choose_modules string network-console
#d-i network-console/authorized_keys_url string http://10.0.0.1/openssh-key
#d-i network-console/password password r00tme
#d-i network-console/password-again password r00tme
# Use this instead if you prefer to use key-based authentication
#d-i network-console/authorized_keys_url http://host/authorized_keys
```

B.4.4. Mirror settings

Depending on the installation method you use, a mirror may be used to download additional components of the installer, to install the base system, and to set up the `/etc/apt/sources.list` for the installed system.

The parameter `mirror/suite` determines the suite for the installed system.

The parameter `mirror/udeb/suite` determines the suite for additional components for the installer. It is only useful to set this if components are actually downloaded over the network and should match the suite that was used to build the initrd for the installation method used for the installation. Normally the installer will automatically use the correct value and there should be no need to set this.

The parameter `mirror/udeb/components` determines the archive components from which additional installer components are fetched. It is only useful to set this if components are actually downloaded over the network. The default components are main and restricted.

```
# If you select ftp, the mirror/country string does not need to be set.
#d-i mirror/protocol string ftp
d-i mirror/country string manual
d-i mirror/http/hostname string ports.ubuntu.com
d-i mirror/http/directory string /ubuntu
d-i mirror/http/proxy string

# Alternatively: by default, the installer uses CC.archive.ubuntu.com where
# CC is the ISO-3166-2 code for the selected country. You can preseed this
# so that it does so without asking.
```

```
#d-i mirror/http/mirror select CC.archive.ubuntu.com

# Suite to install.
#d-i mirror/suite string bionic
# Suite to use for loading installer components (optional).
#d-i mirror/udeb/suite string bionic
# Components to use for loading installer components (optional).
#d-i mirror/udeb/components multiselect main, restricted
```

B.4.5. Account setup

The password for the root account and name and password for a first regular user's account can be preseeded. For the passwords you can use either clear text values or crypt(3) *hashes*.

Warning

Be aware that preseeding passwords is not completely secure as everyone with access to the preconfiguration file will have the knowledge of these passwords. Storing hashed passwords is considered secure unless a weak hashing algorithm like DES or MD5 is used which allow for bruteforce attacks. Recommended password hashing algorithms are SHA-256 and SHA512.

```
# Skip creation of a root account (normal user account will be able to
# use sudo). The default is false; preseed this to true if you want to set
# a root password.
#d-i passwd/root-login boolean false
# Alternatively, to skip creation of a normal user account.
#d-i passwd/make-user boolean false

# Root password, either in clear text
#d-i passwd/root-password password r00tme
#d-i passwd/root-password-again password r00tme
# or encrypted using a crypt(3)  hash.
#d-i passwd/root-password-crypted password [crypt(3) hash]

# To create a normal user account.
#d-i passwd/user-fullname string Ubuntu User
#d-i passwd/username string ubuntu
# Normal user's password, either in clear text
#d-i passwd/user-password password insecure
#d-i passwd/user-password-again password insecure
# or encrypted using a crypt(3) hash.
#d-i passwd/user-password-crypted password [crypt(3) hash]
# Create the first user with the specified UID instead of the default.
#d-i passwd/user-uid string 1010
# The installer will warn about weak passwords. If you are sure you know
# what you're doing and want to override it, uncomment this.
#d-i user-setup/allow-password-weak boolean true

# The user account will be added to some standard initial groups. To
# override that, use this.
#d-i passwd/user-default-groups string audio cdrom video

# Set to true if you want to encrypt the first user's home directory.
```

```
d-i user-setup/encrypt-home boolean false
```

The `passwd/root-password-crypted` and `passwd/user-password-crypted` variables can also be preseeded with "!" as their value. In that case, the corresponding account is disabled. This may be convenient for the root account, provided of course that an alternative method is set up to allow administrative activities or root login (for instance by using SSH key authentication or **sudo**).

The following command (available from the `whois` package) can be used to generate a SHA-512 based crypt(3) hash for a password:

```
mkpasswd -m sha-512
```

B.4.6. Clock and time zone setup

```
# Controls whether or not the hardware clock is set to UTC.
d-i clock-setup/utc boolean true

# You may set this to any valid setting for $TZ; see the contents of
# /usr/share/zoneinfo/ for valid values.
d-i time/zone string US/Eastern

# Controls whether to use NTP to set the clock during the install
d-i clock-setup/ntp boolean true
# NTP server to use. The default is almost always fine here.
#d-i clock-setup/ntp-server string ntp.example.com
```

B.4.7. 64-bit ARM specific disk storage

The arm64 offers two unique kinds of disk storage - DASD and FCP (or zFCP).

```
# Activate DASD disks
#d-i s390-dasd/dasd string 0.0.0200,0.0.0300,0.0.0400

# DASD configuration; by default dasdfmt (low-level format) if needed
#d-i s390-dasd/auto-format boolean true
#d-i s390-dasd/force-format boolean true

# zFCP activation and configuration
# d-i s390-zfcp/zfcp string 0.0.1b34:0x400870075678a1b2:0x201480c800000000, \
#                           0.0.1b34:0x400870075679a1b2:0x201480c800000000
```

B.4.8. Partitioning

Using preseeding to partition the harddisk is limited to what is supported by `partman-auto`. You can choose to partition either existing free space on a disk or a whole disk. The layout of the disk can be determined by using a predefined recipe, a custom recipe from a recipe file or a recipe included in the preconfiguration file.

Preseeding of advanced partition setups using RAID, LVM and encryption is supported, but not with the full flexibility possible when partitioning during a non-preseeded install.

The examples below only provide basic information on the use of recipes. For detailed information see the files `partman-auto-recipe.txt` and `partman-auto-raid-recipe.txt` included in the `debian-installer` package. Both files are also available from the `debian-installer` source repository (http://anonscm.debian.org/gitweb/?p=d-i/debian-installer.git;a=tree;f=doc/devel). Note that the supported functionality may change between releases.

Warning

The identification of disks is dependent on the order in which their drivers are loaded. If there are multiple disks in the system, make very sure the correct one will be selected before using preseeding.

B.4.8.1. Partitioning example

```
# If the system has free space you can choose to only partition that space.
# This is only honoured if partman-auto/method (below) is not set.
# Alternatives: custom, some_device, some_device_crypto, some_device_lvm.
#d-i partman-auto/init_automatically_partition select biggest_free

# Alternatively, you may specify a disk to partition. If the system has only
# one disk the installer will default to using that, but otherwise the device
# name must be given in traditional, non-devfs format (so e.g. /dev/sda
# and not e.g. /dev/discs/disc0/disc).
# For example, to use the first SCSI/SATA hard disk:
#d-i partman-auto/disk string /dev/sda
# In addition, you'll need to specify the method to use.
# The presently available methods are:
# - regular: use the usual partition types for your architecture
# - lvm:     use LVM to partition the disk
# - crypto:  use LVM within an encrypted partition
d-i partman-auto/method string lvm

# If one of the disks that are going to be automatically partitioned
# contains an old LVM configuration, the user will normally receive a
# warning. This can be preseeded away...
d-i partman-lvm/device_remove_lvm boolean true
# The same applies to pre-existing software RAID array:
d-i partman-md/device_remove_md boolean true
# And the same goes for the confirmation to write the lvm partitions.
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true

# For LVM partitioning, you can select how much of the volume group to use
# for logical volumes.
#d-i partman-auto-lvm/guided_size string max
#d-i partman-auto-lvm/guided_size string 10GB
#d-i partman-auto-lvm/guided_size string 50%

# You can choose one of the three predefined partitioning recipes:
# - atomic: all files in one partition
# - home:   separate /home partition
# - multi:  separate /home, /var, and /tmp partitions
```

```
d-i partman-auto/choose_recipe select atomic

# Or provide a recipe of your own...
# If you have a way to get a recipe file into the d-i environment, you can
# just point at it.
#d-i partman-auto/expert_recipe_file string /hd-media/recipe

# If not, you can put an entire recipe into the preconfiguration file in one
# (logical) line. This example creates a small /boot partition, suitable
# swap, and uses the rest of the space for the root partition:
#d-i partman-auto/expert_recipe string                         \
#      boot-root ::                                            \
#              40 50 100 ext3                                  \
#                      $primary{ } $bootable{ }                \
#                      method{ format } format{ }              \
#                      use_filesystem{ } filesystem{ ext3 }    \
#                      mountpoint{ /boot }                     \
#              .                                               \
#              500 10000 1000000000 ext3                       \
#                      method{ format } format{ }              \
#                      use_filesystem{ } filesystem{ ext3 }    \
#                      mountpoint{ / }                         \
#              .                                               \
#              64 512 300% linux-swap                          \
#                      method{ swap } format{ }                \
#              .

# If you just want to change the default filesystem from ext3 to something
# else, you can do that without providing a full recipe.
#d-i partman/default_filesystem string ext4

# The full recipe format is documented in the file partman-auto-recipe.txt
# included in the 'debian-installer' package or available from D-I source
# repository. This also documents how to specify settings such as file
# system labels, volume group names and which physical devices to include
# in a volume group.

# This makes partman automatically partition without confirmation, provided
# that you told it what to do using one of the methods above.
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
```

B.4.8.2. Partitioning using RAID

You can also use preseeding to set up partitions on software RAID arrays. Supported are RAID levels 0, 1, 5, 6 and 10, creating degraded arrays and specifying spare devices.

Warning

This type of automated partitioning is easy to get wrong. It is also functionality that receives relatively little testing from the developers of `debian-installer`. The responsibility to get the various recipes right (so they make sense and don't conflict) lies with the user. Check `/var/log/syslog` if you run into problems.

```
# The method should be set to "raid".
#d-i partman-auto/method string raid
# Specify the disks to be partitioned. They will all get the same layout,
# so this will only work if the disks are the same size.
#d-i partman-auto/disk string /dev/sda /dev/sdb

# Next you need to specify the physical partitions that will be used.
#d-i partman-auto/expert_recipe string \
#      multiraid ::                                          \
#              1000 5000 4000 raid                           \
#                      $primary{ } method{ raid }            \
#              .                                             \
#              64 512 300% raid                              \
#                      method{ raid }                        \
#              .                                             \
#              500 10000 1000000000 raid                     \
#                      method{ raid }                        \
#              .

# Last you need to specify how the previously defined partitions will be
# used in the RAID setup. Remember to use the correct partition numbers
# for logical partitions. RAID levels 0, 1, 5, 6 and 10 are supported;
# devices are separated using "#".
# Parameters are:
# <raidtype> <devcount> <sparecount> <fstype> <mountpoint> \
#          <devices> <sparedevices>

#d-i partman-auto-raid/recipe string \
#    1 2 0 ext3 /                    \
#          /dev/sda1#/dev/sdb1       \
#    .                               \
#    1 2 0 swap -                    \
#          /dev/sda5#/dev/sdb5       \
#    .                               \
#    0 2 0 ext3 /home                \
#          /dev/sda6#/dev/sdb6       \
#    .

# For additional information see the file partman-auto-raid-recipe.txt
# included in the 'debian-installer' package or available from D-I source
# repository.

# This makes partman automatically partition without confirmation.
d-i partman-md/confirm boolean true
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
```

B.4.8.3. Controlling how partitions are mounted

Normally, filesystems are mounted using a universally unique identifier (UUID) as a key; this allows them to be mounted properly even if their device name changes. UUIDs are long and difficult to read, so, if you prefer, the installer can mount filesystems based on the traditional device names, or based on a label you assign. If you ask the installer to mount by label, any filesystems without a label will be mounted using a UUID instead.

Devices with stable names, such as LVM logical volumes, will continue to use their traditional names rather than UUIDs.

Warning

Traditional device names may change based on the order in which the kernel discovers devices at boot, which may cause the wrong filesystem to be mounted. Similarly, labels are likely to clash if you plug in a new disk or a USB drive, and if that happens your system's behaviour when started will be random.

```
# The default is to mount by UUID, but you can also choose "traditional" to
# use traditional device names, or "label" to try filesystem labels before
# falling back to UUIDs.
#d-i partman/mount_style select uuid
```

B.4.9. Base system installation

There is actually not very much that can be preseeded for this stage of the installation. The only questions asked concern the installation of the kernel and the location of the base pre-configured filesystem for the installation.

```
# Configure a path to the preconfigured base filesystem. This can be used to
# specify a path for the installer to retrieve the filesystem image that will
# be deployed to disk and used as a base system for the installation.
#d-i live-installer/net-image string /install/filesystem.squashfs

# Configure APT to not install recommended packages by default. Use of this
# option can result in an incomplete system and should only be used by very
# experienced users.
#d-i base-installer/install-recommends boolean false

# The kernel image (meta) package to be installed; "none" can be used if no
# kernel is to be installed.
#d-i base-installer/kernel/image string linux-generic
```

B.4.10. Apt setup

Setup of the `/etc/apt/sources.list` and basic configuration options is fully automated based on your installation method and answers to earlier questions. You can optionally add other (local) repositories.

```
# You can choose to install restricted and universe software, or to install
# software from the backports repository.
#d-i apt-setup/restricted boolean true
#d-i apt-setup/universe boolean true
#d-i apt-setup/backports boolean true
# Uncomment this if you don't want to use a network mirror.
#d-i apt-setup/use_mirror boolean false
# Select which update services to use; define the mirrors to be used.
# Values shown below are the normal defaults.
#d-i apt-setup/services-select multiselect security
#d-i apt-setup/security_host string security.ubuntu.com
#d-i apt-setup/security_path string /ubuntu

# Additional repositories, local[0-9] available
#d-i apt-setup/local0/repository string \
#       http://local.server/ubuntu bionic main
#d-i apt-setup/local0/comment string local server
# Enable deb-src lines
#d-i apt-setup/local0/source boolean true
# URL to the public key of the local repository; you must provide a key or
# apt will complain about the unauthenticated repository and so the
# sources.list line will be left commented out
#d-i apt-setup/local0/key string http://local.server/key

# By default the installer requires that repositories be authenticated
# using a known gpg key. This setting can be used to disable that
# authentication. Warning: Insecure, not recommended.
#d-i debian-installer/allow_unauthenticated boolean true

# Uncomment this to add multiarch configuration for i386
#d-i apt-setup/multiarch string i386
```

B.4.11. Package selection

You can choose to install any combination of tasks that are available. Available tasks as of this writing include:

- **`standard`** (standard tools)
- **`lamp-server`**
- **`print-server`** (print server)

You can also choose to install no tasks, and force the installation of a set of packages in some other way. We recommend always including the **`standard`** task.

If you want to install some individual packages in addition to packages installed by tasks, you can use the parameter `pkgsel/include`. The value of this parameter can be a list of packages separated by either commas or spaces, which allows it to be used easily on the kernel command line as well. By default, recommended packages will not be installed; to change this, preseed `pkgsel/install-recommends` to true.

To install a different set of language packs, you can use the parameter `pkgsel/language-packs`. The value of this parameter should be a list of ISO-639 language codes. If not set, the language packs matching the language selected in the installer will be installed.

```
tasksel tasksel/first multiselect ubuntu-desktop
#tasksel tasksel/first multiselect lamp-server, print-server
#tasksel tasksel/first multiselect kubuntu-desktop

# Individual additional packages to install
#d-i pkgsel/include string openssh-server build-essential
# Whether to upgrade packages after debootstrap.
# Allowed values: none, safe-upgrade, full-upgrade
#d-i pkgsel/upgrade select none

# Language pack selection
#d-i pkgsel/language-packs multiselect de, en, zh

# Policy for applying updates. May be "none" (no automatic updates),
# "unattended-upgrades" (install security updates automatically), or
# "landscape" (manage system with Landscape).
#d-i pkgsel/update-policy select none

# Some versions of the installer can report back on what software you have
# installed, and what software you use. The default is not to report back,
# but sending reports helps the project determine what software is most
# popular and include it on CDs.
#popularity-contest popularity-contest/participate boolean false

# By default, the system's locate database will be updated after the
# installer has finished installing most packages. This may take a while, so
# if you don't want it, you can set this to "false" to turn it off.
#d-i pkgsel/updatedb boolean true
```

B.4.12. Finishing up the installation

```
# During installations from serial console, the regular virtual consoles
# (VT1-VT6) are normally disabled in /etc/inittab. Uncomment the next
# line to prevent this.
#d-i finish-install/keep-consoles boolean true

# Avoid that last message about the install being complete.
d-i finish-install/reboot_in_progress note

# This will prevent the installer from ejecting the CD during the reboot,
# which is useful in some situations.
#d-i cdrom-detect/eject boolean false

# This is how to make the installer shutdown when finished, but not
# reboot into the installed system.
#d-i debian-installer/exit/halt boolean true
# This will power off the machine instead of just halting it.
#d-i debian-installer/exit/poweroff boolean true
```

B.4.13. Preseeding other packages

```
# Depending on what software you choose to install, or if things go wrong
# during the installation process, it's possible that other questions may
# be asked. You can preseed those too, of course. To get a list of every
# possible question that could be asked during an install, do an
# installation, and then run these commands:
#   debconf-get-selections --installer > file
#   debconf-get-selections >> file
```

B.5. Advanced options

B.5.1. Running custom commands during the installation

A very powerful and flexible option offered by the preconfiguration tools is the ability to run commands or scripts at certain points in the installation.

When the filesystem of the target system is mounted, it is available in `/target`. If an installation CD is used, when it is mounted it is available in `/cdrom`.

```
# d-i preseeding is inherently not secure. Nothing in the installer checks
# for attempts at buffer overflows or other exploits of the values of a
# preconfiguration file like this one. Only use preconfiguration files from
# trusted locations! To drive that home, and because it's generally useful,
# here's a way to run any shell command you'd like inside the installer,
# automatically.

# This first command is run as early as possible, just after
# preseeding is read.
#d-i preseed/early_command string anna-install some-udeb
# This command is run immediately before the partitioner starts. It may be
# useful to apply dynamic partitioner preseeding that depends on the state
# of the disks (which may not be visible when preseed/early_command runs).
#d-i partman/early_command \
#       string debconf-set partman-auto/disk "$(list-devices disk | head -n1)"
# This command is run just before the install finishes, but when there is
# still a usable /target directory. You can chroot to /target and use it
# directly, or use the apt-install and in-target commands to easily install
# packages and run commands in the target system.
#d-i preseed/late_command string apt-install zsh; in-target chsh -s /bin/zsh
```

B.5.2. Using preseeding to change default values

It is possible to use preseeding to change the default answer for a question, but still have the question asked. To do this the *seen* flag must be reset to "false" after setting the value for a question.

```
d-i foo/bar string value
d-i foo/bar seen false
```

The same effect can be achieved for *all* questions by setting the parameter `preseed/interactive=true` at the boot prompt. This can also be useful for testing or debugging your preconfiguration file.

Note that the "d-i" owner should only be used for variables used in the installer itself. For variables belonging to packages installed on the target system, you should use the name of that package instead. See the footnote to Section B.2.2.

If you are preseeding using boot parameters, you can make the installer ask the corresponding question by using the "?=" operator, i.e. ***`foo/bar?=value`*** (or ***`owner:foo/bar?=value`***). This will of course only have effect for parameters that correspond to questions that are actually displayed during an installation and not for "internal" parameters.

For more debugging information, use the boot parameter `DEBCONF_DEBUG=5`. This will cause `debconf` to print much more detail about the current settings of each variable and about its progress through each package's installation scripts.

B.5.3. Chainloading preconfiguration files

It is possible to include other preconfiguration files from a preconfiguration file. Any settings in those files will override pre-existing settings from files loaded earlier. This makes it possible to put, for example, general networking settings for your location in one file and more specific settings for certain configurations in other files.

```
# More than one file can be listed, separated by spaces; all will be
# loaded. The included files can have preseed/include directives of their
# own as well. Note that if the filenames are relative, they are taken from
# the same directory as the preconfiguration file that includes them.
#d-i preseed/include string x.cfg

# The installer can optionally verify checksums of preconfiguration files
# before using them. Currently only md5sums are supported, list the md5sums
# in the same order as the list of files to include.
#d-i preseed/include/checksum string 5da499872becccfeda2c4872f9171c3d

# More flexibly, this runs a shell command and if it outputs the names of
# preconfiguration files, includes those files.
#d-i preseed/include_command \
#      string if [ "`hostname`" = bob ]; then echo bob.cfg; fi

# Most flexibly of all, this downloads a program and runs it. The program
# can use commands such as debconf-set to manipulate the debconf database.
# More than one script can be listed, separated by spaces.
# Note that if the filenames are relative, they are taken from the same
# directory as the preconfiguration file that runs them.
#d-i preseed/run string foo.sh
```

It is also possible to chainload from the initrd or file preseeding phase, into network preseeding by setting preseed/url in the earlier files. This will cause network preseeding to be performed when the network comes up. You need to be careful when doing this, since there will be two distinct runs at preseeding, meaning for example that you get another chance to run the preseed/early command, the second one happening after the network comes up.

Appendix C. Partitioning for Ubuntu

C.1. Deciding on Ubuntu Partitions and Sizes

At a bare minimum, GNU/Linux needs one partition for itself. You can have a single partition containing the entire operating system, applications, and your personal files. Most people feel that a separate swap partition is also a necessity, although it's not strictly true. In fact, since Ubuntu 18.04 a swap file rather than a swap partition is now used by default. "Swap" is scratch space for an operating system, which allows the system to use disk storage as "virtual memory". Linux may use a separate swap partition more efficiently compared to a swap file, but on the other hand side a swap file wastes less disk space compared to a permanently allocated swap partition.

Most people choose to give GNU/Linux more than the minimum number of partitions, however. There are two reasons you might want to break up the file system into a number of smaller partitions. The first is for safety. If something happens to corrupt the file system, generally only one partition is affected. Thus, you only have to replace (from the backups you've been carefully keeping) a portion of your system. At a bare minimum, you should consider creating what is commonly called a "root partition". This contains the most essential components of the system. If any other partitions get corrupted, you can still boot into GNU/Linux to fix the system. This can save you the trouble of having to reinstall the system from scratch.

The second reason is generally more important in a business setting, but it really depends on your use of the machine. For example, a mail server getting spammed with e-mail can easily fill a partition. If you made `/var/mail` a separate partition on the mail server, most of the system will remain working even if you get spammed.

The only real drawback to using more partitions is that it is often difficult to know in advance what your needs will be. If you make a partition too small then you will either have to reinstall the system or you will be constantly moving things around to make room in the undersized partition. On the other hand, if you make the partition too big, you will be wasting space that could be used elsewhere. Disk space is cheap nowadays, but why throw your money away?

C.2. The Directory Tree

Ubuntu adheres to the Filesystem Hierarchy Standard (http://www.pathname.com/fhs/) for directory and file naming. This standard allows users and software programs to predict the location of files and directories. The root level directory is represented simply by the slash `/`. At the root level, all Ubuntu systems include these directories:

Directory	Content
`bin`	Essential command binaries
`boot`	Static files of the boot loader
`dev`	Device files
`etc`	Host-specific system configuration
`home`	User home directories
`lib`	Essential shared libraries and kernel modules
`media`	Contains mount points for replaceable media

Directory	Content
mnt	Mount point for mounting a file system temporarily
proc	Virtual directory for system information
root	Home directory for the root user
run	Run-time variable data
sbin	Essential system binaries
sys	Virtual directory for system information
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data
srv	Data for services provided by the system
opt	Add-on application software packages

The following is a list of important considerations regarding directories and partitions. Note that disk usage varies widely given system configuration and specific usage patterns. The recommendations here are general guidelines and provide a starting point for partitioning.

- The root partition / must always physically contain /etc, /bin, /sbin, /lib and /dev, otherwise you won't be able to boot. Typically 150–310MB is needed for the root partition.
- /usr: contains all user programs (/usr/bin), libraries (/usr/lib), documentation (/usr/share/doc), etc. This is the part of the file system that generally takes up most space. You should provide at least 500MB of disk space. This amount should be increased depending on the number and type of packages you plan to install. A generous server installation should allow 4–6GB.
- It is now recommended to have /usr on the root partition /, otherwise it could cause some trouble at boot time. This means that you should provide at least 600–750MB of disk space for the root partition including /usr, or 5–6GB for a workstation or a server installation.
- /var: variable data like news articles, e-mails, web sites, databases, the packaging system cache, etc. will be placed under this directory. The size of this directory depends greatly on the usage of your system, but for most people will be dictated by the package management tool's overhead. If you are going to do a full installation of just about everything Ubuntu has to offer, all in one session, setting aside 2 or 3 GB of space for /var should be sufficient. If you are going to install in pieces (that is to say, install services and utilities, followed by text stuff, then X, ...), you can get away with 300–500 MB. If hard drive space is at a premium and you don't plan on doing major system updates, you can get by with as little as 30 or 40 MB.
- /tmp: temporary data created by programs will most likely go in this directory. 40–100MB should usually be enough. Some applications — including archive manipulators, CD/DVD authoring tools, and multimedia software — may use /tmp to temporarily store image files. If you plan to use such applications, you should adjust the space available in /tmp accordingly.
- /home: every user will put his personal data into a subdirectory of this directory. Its size depends on how many users will be using the system and what files are to be stored in their directories. Depending on your planned usage you should reserve about 100MB for each user, but adapt this value to your needs. Reserve a lot more space if you plan to save a lot of multimedia files (pictures, MP3, movies) in your home directory.

C.3. Recommended Partitioning Scheme

For new users, personal Ubuntu boxes, home systems, and other single-user setups, a single `/` partition (possibly plus a separate swap) is probably the easiest, simplest way to go. However, if your partition is larger than around 6GB, choose ext3 as your partition type. Ext2 partitions need periodic file system integrity checking, and this can cause delays during booting when the partition is large.

For multi-user systems or systems with lots of disk space, it's best to put `/var`, `/tmp`, and `/home` each on their own partitions separate from the `/` partition.

You might need a separate `/usr/local` partition if you plan to install many programs that are not part of the Ubuntu distribution. If your machine will be a mail server, you might need to make `/var/mail` a separate partition. Often, putting `/tmp` on its own partition, for instance 20–50MB, is a good idea. If you are setting up a server with lots of user accounts, it's generally good to have a separate, large `/home` partition. In general, the partitioning situation varies from computer to computer depending on its uses.

For very complex systems, you should see the Multi Disk HOWTO (http://www.tldp.org/HOWTO/Multi-Disk-HOWTO.html). This contains in-depth information, mostly of interest to ISPs and people setting up servers.

With respect to the issue of swap partition size, there are many views. One rule of thumb which works well is to use as much swap as you have system memory. It also shouldn't be smaller than 16MB, in most cases. Of course, there are exceptions to these rules. If you are trying to solve 10000 simultaneous equations on a machine with 256MB of memory, you may need a gigabyte (or more) of swap.

On some 32-bit architectures (m68k and PowerPC), the maximum size of a swap partition is 2GB. That should be enough for nearly any installation. However, if your swap requirements are this high, you should probably try to spread the swap across different disks (also called "spindles") and, if possible, different SCSI or IDE channels. The kernel will balance swap usage between multiple swap partitions, giving better performance.

As an example, an older home machine might have 32MB of RAM and a 1.7GB IDE drive on `/dev/sda`. There might be a 500MB partition for another operating system on `/dev/sda1`, a 32MB swap partition on `/dev/sda3` and about 1.2GB on `/dev/sda2` as the Linux partition.

For an idea of the space required by Ubuntu, check Section D.3.

C.4. Device Names in Linux

Linux disks and partition names may be different from other operating systems. You need to know the names that Linux uses when you create and mount partitions. Here's the basic naming scheme:

- The first floppy drive is named `/dev/fd0`.
- The second floppy drive is named `/dev/fd1`.
- The first hard disk detected is named `/dev/sda`.
- The second hard disk detected is named `/dev/sdb`, and so on.
- The first SCSI CD-ROM is named `/dev/scd0`, also known as `/dev/sr0`.

The partitions on each SCSI disk are represented by appending a decimal number to the disk name: `sda1` and `sda2` represent the first and second partitions of the first SCSI disk drive in your system.

Here is a real-life example. Let's assume you have a system with 2 SCSI disks, one at SCSI address 2 and the other at SCSI address 4. The first disk (at address 2) is then named `sda`, and the second `sdb`. If the `sda` drive has 3 partitions on it, these will be named `sda1`, `sda2`, and `sda3`. The same applies to the `sdb` disk and its partitions.

Note that if you have two SCSI host bus adapters (i.e., controllers), the order of the drives can get confusing. The best solution in this case is to watch the boot messages, assuming you know the drive models and/or capacities.

C.5. Ubuntu Partitioning Programs

Several varieties of partitioning programs have been adapted by Debian and Ubuntu developers to work on various types of hard disks and computer architectures. Following is a list of the program(s) applicable for your architecture.

partman

Recommended partitioning tool in Ubuntu. This Swiss army knife can also resize partitions, create filesystems and assign them to the mountpoints.

fdisk

The original Linux disk partitioner, good for gurus.

Be careful if you have existing FreeBSD partitions on your machine. The installation kernels include support for these partitions, but the way that **fdisk** represents them (or not) can make the device names differ. See the Linux+FreeBSD HOWTO (http://www.tldp.org/HOWTO/Linux+FreeBSD-2.html).

cfdisk

A simple-to-use, full-screen disk partitioner for the rest of us.

Note that **cfdisk** doesn't understand FreeBSD partitions at all, and, again, device names may differ as a result.

One of these programs will be run by default when you select Partition disks (or similar). It may be possible to use a different partitioning tool from the command line on VT2, but this is not recommended.

Appendix D. Random Bits

D.1. Linux Devices

In Linux various special files can be found under the directory `/dev`. These files are called device files and behave unlike ordinary files. The most common types of device files are for block devices and character devices. These files are an interface to the actual driver (part of the Linux kernel) which in turn accesses the hardware. Another, less common, type of device file is the named *pipe*. The most important device files are listed in the tables below.

`fd0`	First Floppy Drive
`fd1`	Second Floppy Drive

`sda`	First hard disk
`sdb`	Second hard disk
`sda1`	First partition of the first hard disk
`sdb7`	Seventh partition of the second hard disk

`sr0`	First CD-ROM
`sr1`	Second CD-ROM

`ttyS0`	Serial port 0, COM1 under MS-DOS
`ttyS1`	Serial port 1, COM2 under MS-DOS
`psaux`	PS/2 mouse device
`gpmdata`	Pseudo device, repeater data from GPM (mouse) daemon

`cdrom`	Symbolic link to the CD-ROM drive
`mouse`	Symbolic link to the mouse device file

`null`	Anything written to this device will disappear
`zero`	One can endlessly read zeros out of this device

D.1.1. Setting Up Your Mouse

The mouse can be used in both the Linux console (with gpm) and the X window environment. Normally, this is a simple matter of installing `gpm` and the X server itself. Both should be configured to use `/dev/input/mice` as the mouse device. The correct mouse protocol is named **`exps2`** in gpm, and **`ExplorerPS/2`** in X. The respective configuration files are `/etc/gpm.conf` and `/etc/X11/xorg.conf`.

Certain kernel modules must be loaded in order for your mouse to work. In most cases the correct modules are autodetected, but not always for old-style serial and bus mice[1], which are quite rare except on very old computers. Summary of Linux kernel modules needed for different mouse types:

Module	Description
psmouse	PS/2 mice (should be autodetected)
usbhid	USB mice (should be autodetected)
sermouse	Most serial mice
logibm	Bus mouse connected to Logitech adapter card
inport	Bus mouse connected to ATI or Microsoft InPort card

To load a mouse driver module, you can use the **modconf** command (from the package with the same name) and look in the category **`kernel/drivers/input/mouse`**.

D.2. Disk Space Needed for Tasks

A standard installation for the amd64 architecture, including all standard packages and using the default kernel, takes up 822MB of disk space. A minimal base installation, without the "Basic Ubuntu server" task selected, will take 506MB.

> **Important:** In both cases this is the actual disk space used *after* the installation is finished and any temporary files deleted. It also does not take into account overhead used by the file system, for example for journal files. This means that significantly more disk space is needed both *during* the installation and for normal system use.

The following table lists sizes reported by aptitude for the tasks listed in tasksel. Note that some tasks have overlapping constituents, so the total installed size for two tasks together may be less than the total obtained by adding up the numbers.

Note that you will need to add the sizes listed in the table to the size of the standard installation when determining the size of partitions. Most of the size listed as "Installed size" will end up in `/usr` and in `/lib`; the size listed as "Download size" is (temporarily) required in `/var`.

Task	Installed size (MB)	Download size (MB)	Space needed to install (MB)
Web server	31	7	38
Print server	234	73	307

If you install in a language other than English, **tasksel** may automatically install a *localization task*, if one is available for your language. Space requirements differ per language; you should allow up to 350MB in total for download and installation.

1. Serial mice usually have a 9-hole D-shaped connector; bus mice have an 8-pin round connector, not to be confused with the 6-pin round connector of a PS/2 mouse or the 4-pin round connector of an ADB mouse.

D.3. Disk Space Needed

A minimal server installation of bionic requires 400MB of disk space.

D.4. Installing Ubuntu from a Unix/Linux System

This section explains how to install Ubuntu from an existing Unix or Linux system, without using the menu-driven installer as explained in the rest of the manual. This "cross-install" HOWTO has been requested by users switching to Ubuntu from Debian, Red Hat, Mandriva, and SUSE. In this section some familiarity with entering *nix commands and navigating the file system is assumed. In this section, `$` symbolizes a command to be entered in the user's current system, while `#` refers to a command entered in the Ubuntu chroot.

Once you've got the new Ubuntu system configured to your preference, you can migrate your existing user data (if any) to it, and keep on rolling. This is therefore a "zero downtime" Ubuntu install. It's also a clever way for dealing with hardware that otherwise doesn't play friendly with various boot or installation media.

> **Note:** As this is a mostly manual procedure, you should bear in mind that you will need to do a lot of basic configuration of the system yourself, which will also require more knowledge of Ubuntu and of Linux in general than performing a regular installation. You cannot expect this procedure to result in a system that is identical to a system from a regular installation. You should also keep in mind that this procedure only gives the basic steps to set up a system. Additional installation and/or configuration steps may be needed. In general, this method of installation is *not* recommended for casual or first time users.

D.4.1. Getting Started

With your current *nix partitioning tools, repartition the hard drive as needed, creating at least one filesystem plus swap. You need around 506MB of space available for a console only install.

Next, create file systems on the partitions. For example, to create an ext3 file system on partition `/dev/sda6` (that's our example root partition):

```
# mke2fs -j /dev/sda6
```

To create an ext2 file system instead, omit **`-j`**.

Initialize and activate swap (substitute the partition number for your intended Ubuntu swap partition):

```
# mkswap /dev/sda5
# sync
# swapon /dev/sda5
```

> **Note:** Instead of using a dedicated swap partition, you may omit the swap partition setup here and later just use a swap file instead.

Mount one partition as `/mnt/ubuntu` (the installation point, to be the root (`/`) filesystem on your new system). The mount point name is strictly arbitrary, it is referenced later below.

```
# mkdir /mnt/ubuntu
# mount /dev/sda6 /mnt/ubuntu
```

Note: If you want to have parts of the filesystem (e.g. /usr) mounted on separate partitions, you will need to create and mount these directories manually before proceding with the next stage.

D.4.2. Install debootstrap

The utility used by the Ubuntu installer, and recognized as the official way to install an Ubuntu base system, is **debootstrap**. It uses **wget** and **ar**, but otherwise depends only on `/bin/sh` and basic Unix/Linux tools[2]. Install **wget** and **ar** if they aren't already on your current system, then download and install **debootstrap**. If these steps are executed under Ubuntu you can simply do this by **`apt install debootstrap`**.

If you have an RPM (Red Hat Package Manager) based system, you can use alien, which is available in the Debian repositories, to convert the .deb file to a useable .rpm file.

Or, you can use the following procedure to install it manually. Make a work folder for extracting the .deb into:

```
# mkdir work
# cd work
```

The **debootstrap** binary is located in the Ubuntu archive (be sure to select the proper file for your architecture). Download the **debootstrap** .deb from the pool (http://ports.ubuntu.com/ubuntu-ports/pool/main/d/debootstrap/), copy the package to the work folder, and extract the files from it. You will need to have root privileges to install the files.

```
# ar -x debootstrap_0.X.X_all.deb
# cd /
# zcat /full-path-to-work/work/data.tar.gz | tar xv
```

D.4.3. Run debootstrap

debootstrap can download the needed files directly from the archive when you run it. You can substitute any Ubuntu archive mirror for **`ports.ubuntu.com/ubuntu-ports`** in the command example below, preferably a mirror close to you network-wise. Mirrors are listed at http://wiki.ubuntu.com/Archive.

If you have an Ubuntu bionic CD mounted at `/cdrom`, you could substitute a file URL instead of the http URL: **`file:/cdrom/ubuntu/`**

Substitute one of the following for *ARCH* in the **debootstrap** command: **`amd64`**, **`arm64`**, **`armhf`**, **`i386`**, **`powerpc`**, **`ppc64el`**, or **`s390x`**.

2. These include the GNU core utilities and commands like **sed**, **grep**, **tar** and **gzip**.

```
# /usr/sbin/debootstrap --arch ARCH bionic /mnt/ubuntu
```

D.4.4. Configure The Base System

Now you've got a real Ubuntu system, though rather lean, on disk. **chroot** into it:

```
# LANG=C.UTF-8 chroot /mnt/ubuntu /bin/bash
```

After chrooting you may need to set the terminal definition to be compatible with the Ubuntu base system, for example:

```
# export TERM=xterm-color
```

Depending on the value of TERM, you may have to install the `ncurses-term` package to get support for it.

> **Note:** If warnings occur like:
>
> ```
> bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
> ```
>
> The required localization files need to be generated:
>
> ```
> # sudo locale-gen en_US.UTF-8
> Generating locales (this might take a while)...
>   en_US.UTF-8... done
> Generation complete.
> ```

D.4.4.1. Configure Apt

Debootstrap will have created a very basic `/etc/apt/sources.list` that will allow installing additional packages. However, it is suggested that you add some additional sources, for example for source packages and security updates:

```
deb-src http://archive.ubuntu.com/ubuntu bionic main

deb http://security.ubuntu.com/ubuntu bionic-security main
deb-src http://security.ubuntu.com/ubuntu bionic-security main
```

Make sure to run **`apt update`** after you have made changes to the sources list.

D.4.4.2. Install additional packages

Now it's required to install some additionally required packages, like makedev (needed for the next section): **`apt install makedev`**

D.4.4.3. Create device files

At this point `/dev/` only contains very basic device files. For the next steps of the installation additional device files may be needed. There are different ways to go about this and which method you should use depends on the host system you are using for the installation, on whether you intend to use a modular kernel or not, and on whether you intend to use dynamic (e.g. using `udev`) or static device files for the new system.

A few of the available options are:

- create a default set of static device files using (after chrooting)

```
# mount none /proc -t proc
# cd /dev
# MAKEDEV generic
or depending on your specific architecture:
# MAKEDEV std
# cd ..
```

- manually create only specific device files using **MAKEDEV**
- bind mount /dev from your host system on top of /dev in the target system, like:

```
mount --bind dev /dev
```

  Note that the postinst scripts of some packages may try to create device files, so this option should only be used with care.

D.4.4.4. Mount Partitions

You need to create `/etc/fstab`.

```
# editor /etc/fstab
```

Here is a sample you can modify to suit:

```
# /etc/fstab: static file system information.
#
# file system    mount point   type    options                 dump pass
/dev/XXX         /             ext3    defaults                0    1
/dev/XXX         /boot         ext3    ro,nosuid,nodev         0    2

/dev/XXX         none          swap    sw                      0    0
proc             /proc         proc    defaults                0    0
sys              /sys          sysfs   defaults                0    0

/dev/fd0         /media/floppy auto    noauto,rw,sync,user,exec 0    0
```

```
/dev/cdrom       /media/cdrom  iso9660 noauto,ro,user,exec      0    0

/dev/XXX         /tmp          ext3    rw,nosuid,nodev          0    2
/dev/XXX         /var          ext3    rw,nosuid,nodev          0    2
/dev/XXX         /usr          ext3    rw,nodev                 0    2
/dev/XXX         /home         ext3    rw,nosuid,nodev          0    2
```

Use **`mount -a`** to mount all the file systems you have specified in your `/etc/fstab`, or, to mount file systems individually, use:

```
# mount /path   # e.g.: mount /usr
```

Current Ubuntu systems have mountpoints for removable media under `/media`, but keep compatibility symlinks in `/`. Create these as as needed, for example:

```
# cd /media
# mkdir cdrom0
# ln -s cdrom0 cdrom
# cd /
# ln -s media/cdrom
```

You can mount the proc and sysfs file systems multiple times and to arbitrary locations, though `/proc` and `/sys` respectively are customary. If you didn't use **`mount -a`**, be sure to mount proc and sysfs before continuing:

```
# mount -t proc proc /proc
# mount -t sysfs sysfs /sys
```

The command **`ls /proc`** should now show a non-empty directory. Should this fail, you may be able to mount proc from outside the chroot:

```
# mount -t proc proc /mnt/ubuntu/proc
```

D.4.4.5. Setting Timezone

Setting the third line of the file `/etc/adjtime` to "UTC" or "LOCAL" determines whether the system will interpret the hardware clock as being set to UTC respective local time. The following command allows you to set that.

```
# editor /etc/adjtime
```

Here is a sample:

```
0.0 0 0.0
0
UTC
```

The following command allows you to choose your timezone.

```
# dpkg-reconfigure tzdata
```

D.4.4.6. Configure Networking

To configure networking, edit `/etc/network/interfaces`, `/etc/resolv.conf`, `/etc/hostname` and `/etc/hosts`.

```
# editor /etc/network/interfaces
```

Here are some simple examples from `/usr/share/doc/ifupdown/examples`:

```
###################################################################
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)
# See the interfaces(5) manpage for information on what options are
# available.
###################################################################

# We always want the loopback interface.
#
auto lo
iface lo inet loopback

# To use dhcp:
#
# auto eth0
# iface eth0 inet dhcp

# An example static IP setup: (broadcast and gateway are optional)
#
# auto eth0
# iface eth0 inet static
#     address 192.168.0.42
#     network 192.168.0.0
#     netmask 255.255.255.0
#     broadcast 192.168.0.255
#     gateway 192.168.0.1
```

Enter your nameserver(s) and search directives in `/etc/resolv.conf`:

```
# editor /etc/resolv.conf
```

A simple example `/etc/resolv.conf`:

```
search hqdom.local
nameserver 10.1.1.36
nameserver 192.168.9.100
```

Enter your system's host name (2 to 63 characters):

```
# echo UbuntuHostName > /etc/hostname
```

And a basic `/etc/hosts` with IPv6 support:

```
127.0.0.1 localhost
127.0.1.1 UbuntuHostName

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

If you have multiple network cards, you should arrange the names of driver modules in the `/etc/modules` file into the desired order. Then during boot, each card will be associated with the interface name (eth0, eth1, etc.) that you expect.

D.4.4.7. Configure Locales and Keyboard

To configure your locale settings to use a language other than English, install the appropriate language packs and configure them. Currently the use of UTF-8 locales is recommended.

```
# apt install language-pack-de language-pack-gnome-de
```

To configure your keyboard (if needed):

```
# apt install console-setup
# dpkg-reconfigure keyboard-configuration
```

Note that the keyboard cannot be set while in the chroot, but will be configured for the next reboot.

D.4.5. Install a Kernel

If you intend to boot this system, you probably want a Linux kernel and a boot loader. Identify available pre-packaged kernels with:

```
# apt-cache search linux-image
```

Then install the kernel package of your choice using its package name.

```
# apt install linux-image-arch-etc
```

(You may want install `linux-image-generic`, too.)

D.4.6. Set up the Boot Loader

To make your Ubuntu system bootable, set up your boot loader to load the installed kernel with your new root partition. Note that **debootstrap** does not install a boot loader, though you can use **apt** inside your Ubuntu chroot to do so.

Note that this assumes that a `/dev/sda` device file has been created. There are alternative methods to install **grub2**, but those are outside the scope of this appendix.

D.4.7. Remote access: Installing SSH and setting up access

In case you can login to the system via console, you can skip this section. If the system should be accessible via the network later on, you need to install SSH and set up access.

```
# apt install openssh-server
```

Root login with password is disabled by default, so setting up access can be done by setting a password and re-enable root login with password:

```
# passwd
# editor /etc/ssh/sshd_config
```

This is the option to be enabled:

```
PermitRootLogin yes
```

Access can also be set up by adding an ssh key to the root account:

```
# mkdir /root/.ssh
# cat << EOF > /root/.ssh/authorized_keys
ssh-rsa ....
EOF
```

Lastly, access can be set up by adding a non-root user and setting a password:

```
# adduser joe
# passwd joe
```

D.4.8. Finishing touches

As mentioned earlier, the installed system will be very basic. If you would like to make the system a bit more mature, there is an easy method to install all packages with “standard” priority:

```
# apt install tasksel
# tasksel install standard
```

Of course, you can also just use **apt** to install packages individually.

After the installation there will be a lot of downloaded packages in `/var/cache/apt/archives/`. You can free up some diskspace by running:

```
# apt clean
```

D.4.9. Create a User

Use the **adduser** command to create a new user account:

```
# adduser myusername
```

You will be prompted for a full name and a password.

The normal Ubuntu configuration is to allow this new user to administer the system using **sudo**. To set this up, first create an `admin` group and add your new user to it:

```
# addgroup --system admin
# adduser myusername admin
```

You can now use the **visudo** command to add these lines to the end of `/etc/sudoers`, so that any user in the `admin` group can administer the system:

```
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

If you don't want to follow this configuration, then remember to set a root password:

```
# passwd root
```

D.5. Installing Ubuntu using PPP over Ethernet (PPPoE)

In some countries PPP over Ethernet (PPPoE) is a common protocol for broadband (ADSL or cable) connections to an Internet Service Provider. Setting up a network connection using PPPoE is not supported by default in the installer, but can be made to work very simply. This section explains how.

The PPPoE connection set up during the installation will also be available after the reboot into the installed system (see Chapter 7).

To have the option of setting up and using PPPoE during the installation, you will need to install using one of the CD-ROM/DVD images that are available. It is not supported for other installation methods (e.g. netboot).

Installing over PPPoE is mostly the same as any other installation. The following steps explain the differences.

- Boot the installer with the boot parameter **`modules=ppp-udeb`**. This will ensure the component responsible for the setup of PPPoE (`ppp-udeb`) will be loaded and run automatically.
- Follow the regular initial steps of the installation (language, country and keyboard selection; the loading of additional installer components[3]).
- The next step is the detection of network hardware, in order to identify any Ethernet cards present in the system.
- After this the actual setup of PPPoE is started. The installer will probe all the detected Ethernet interfaces in an attempt to find a PPPoE concentrator (a type of server which handles PPPoE connections).

  It is possible that the concentrator will not to be found at the first attempt. This can happen occasionally on slow or loaded networks or with faulty servers. In most cases a second attempt to detect the concentrator will be successful; to retry, select Configure and start a PPPoE connection from the main menu of the installer.
- After a concentrator is found, the user will be prompted to type the login information (the PPPoE username and password).
- At this point the installer will use the provided information to establish the PPPoE connection. If the correct information was provided, the PPPoE connection should be configured and the installer should be able to use it to connect to the Internet and retrieve packages over it (if needed). If the login information is not correct or some error appears, the installer will stop, but the configuration can be attempted again by selecting the menu entry Configure and start a PPPoE connection.

3. The `ppp-udeb` component is loaded as one of the additional components in this step. If you want to install at medium or low priority (expert mode), you can also manually select the `ppp-udeb` instead of entering the "modules" parameter at the boot prompt.

Appendix E. Administrivia

E.1. About This Document

This manual was created for Sarge's debian-installer, based on the Woody installation manual for boot-floppies, which was based on earlier Debian installation manuals, and on the Progeny distribution manual which was released under GPL in 2003. It was subsequently modified for use in Ubuntu.

This document is written in DocBook XML. Output formats are generated by various programs using information from the `docbook-xml` and `docbook-xsl` packages.

In order to increase the maintainability of this document, we use a number of XML features, such as entities and profiling attributes. These play a role akin to variables and conditionals in programming languages. The XML source to this document contains information for each different architecture — profiling attributes are used to isolate certain bits of text as architecture-specific.

E.2. Contributing to This Document

If you have problems or suggestions regarding this document, please mail them to <`ubuntu-users@lists.ubuntu.com`>.

Please do *not* contact the authors of this document directly. There is also a discussion list for `debian-installer`, which includes discussions of this manual. The mailing list is <`debian-boot@lists.debian.org`>. Instructions for subscribing to this list can be found at the Debian Mailing List Subscription (http://www.debian.org/MailingLists/subscribe) page; or you can browse the Debian Mailing List Archives (http://lists.debian.org/) online. Please do not contact debian-boot about issues specific to Ubuntu.

E.3. Major Contributions

This document was originally written by Bruce Perens, Sven Rudolph, Igor Grobman, James Treacy, and Adam Di Carlo. Sebastian Ley wrote the Installation Howto.

Miroslav Kuře has documented a lot of the new functionality in Sarge's debian-installer. Frans Pop was the main editor and release manager during the Etch, Lenny and Squeeze releases.

Many, many Debian users and developers contributed to this document. Particular note must be made of Michael Schmitz (m68k support), Frank Neumann (original author of the Amiga install manual (http://www.informatik.uni-oldenburg.de/~amigo/debian_inst.html)), Arto Astala, Eric Delaunay/Ben Collins (SPARC information), Tapio Lehtonen, and Stéphane Bortzmeyer for numerous edits and text. We have to thank Pascal Le Bail for useful information about booting from USB memory sticks. Colin Watson and others made the modifications for Ubuntu.

Extremely helpful text and information was found in Jim Mintha's HOWTO for network booting (no URL available), the Debian FAQ (http://www.debian.org/doc/FAQ/), the Linux/m68k FAQ (http://www.linux-m68k.org/faq/faq.html), the Linux for SPARC Processors FAQ (http://www.ultralinux.org/faq.html), the Linux/Alpha FAQ (http://linux.iol.unh.edu/linux/alpha/faq/), amongst others. The maintainers of these freely available and rich sources of information must be recognized.

The section on chrooted installations in this manual (Section D.4) was derived in part from documents copyright Karsten M. Self.

E.4. Trademark Acknowledgement

All trademarks are property of their respective trademark owners.

Appendix F. GNU General Public License

Version 2, June 1991

F.1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the gnu General Public License is intended to guarantee your freedom to share and change free software — to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the gnu Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

F.2. GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by

all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL AND COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO

OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

F.3. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with absolutely no warranty; for details
type 'show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items — whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program 'Gnomovision' (which makes passes at compilers) written
by James Hacker.
```

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.