LINUX

STEP BY STEP
BEGINNER'S
GUIDE

INSTALLATION, CONFIGURATION & COMMAND LINE BASICS

NATHAN CLARK

Linux

Installation, Configuration and Command Line Basics

Nathan Clark

Table of Contents

About This Book

This book has been created to guide you through your very first steps in the Linux environment. If you are a complete novice, or need a refresher in Linux, you've chosen the right book.

In the upcoming chapters we will cover the Linux diversity and history and then continue on with setting up a Linux system from scratch for the end user. Here we will guide you through the setup and configuration process step by step.

We will take a detailed look at the infamous command line by covering numerous essential terminal commands. We will also address specific topics such as choosing a distribution, adding a graphical user interface, package management, navigating the filesystem and directories, partitioning, software selection, and using the help system.

By the end of this book you will have set up and configured Linux from start to finish, and be able to use Linux at a proficient level.

1. What is Linux?

Linux is the name for the kernel of an operating system that is based on the UNIX principles. The name is derived from the first name of its Finnish inventor, Linus Torvalds, and follows the methodology used by other UNIX-based systems (the last letter is an x). Today, Linux is developed and maintained by thousands of people around the world.

The kernel of an operating system is its heart. It is required for communication between the hardware of your computer and you, the user. An operating system is a collection of different software components: a kernel, various tools and the accompanying libraries. It is a software that extends the basic operating system of your computer, known as the BIOS.

1.1 From UNIX to Linux

The history of Linux can be traced back to the 1990s. In order to understand the story behind Linux, we also have to look back briefly at the early days of computing after the 2nd World War.

At that time computing machines filled entire buildings and the transformation from mechanical to electronical components, like microprocessors and the usage of multi-layer circuits, was underway. Moreover, in the 1960s and 1970s hardware and software components were quite expensive and not standardized. Various vendor-specific platforms existed and each of them had their own interface, protocols to transfer and exchange data, as well as operating system. The communication between these single computing devices required specific knowledge and the understanding of its protocols. The development of UNIX was an aim to circumvent these obstacles and to simplify the usage of computing devices on a larger scale.

UNIX

At the beginning of 1965 the development of the Multiplexed Information and Computing Service (Multics) started. Multics was the result of a collaboration between the Massachusetts Institute of Technology (MIT), General Electric (GE)

and Bell Labs/AT&T. Led by the developers Ken Thompson and Dennis Ritchie, the main product they developed was Unics. Later on it was renamed to UNIX. The UNIX operating system was mainly in use at the University of California in Berkeley.

UNIX Variants

The concept of UNIX became licensed to several companies that developed and maintained their own variant of UNIX. This included Solaris/SUN OS (SUN Microsystems, nowadays owned by Oracle), AIX (IBM), Scenix (Siemens), SCO UNIX, Xenix (Microsoft), as well as HP-UX (Hewlett-Packard), NeXTSTEP, Mac OS (Apple) and Android (Google).

Open-source implementations comprised of the Berkeley System Distribution (BSD) with its variants: NetBSD, OpenBSD, and FreeBSD. Today, Linux is the most popular free software among open source developers. There is also a strong commercial support for the systems mentioned above.

The UNIX Philosophy

UNIX is designed with a number of strict principles in mind. These principles cover portability, multi-tasking and multi-user orientation in combination with a time-sharing approach. Furthermore, it is based on network connectivity following the TCP/IP scheme.

The original development was done in the C programming language that resulted in independence from a hardware platform. Delivered with a selection of development tools and libraries, it allows you to easily extend it to your specific needs. It is simple, but has a powerful ability to automate tasks that supports complex but maintainable scripts.

Similar to a toolbox, UNIX consists of a variety of tools. Each of them having a specific purpose and being designed exactly for that task. The idea is to use one tool per task. In order to achieve more complex goals, you would combine several tools into a chain. The following example combines the two commands ‘ls’ and ‘wc’ by means of a pipe to be able to count the number of Python files in the current directory.

```
$ ls *.py | wc -l
6
$
```

We will explain these commands and their usage in more detail later on in the guide.

1.2 A Brief History of Linux

Similar to UNIX, the Linux operating system has different roots and is based on the work of quite a few masterminds. Among others, this includes Richard M. Stallman, Andrew S. Tanenbaum and Linus Torvalds.

Richard M. Stallman, a hacker and developer at MIT, is the first president of the Free Software Foundation (FSF), and the father of the GNU project. GNU abbreviates the slogan GNU is Not UNIX. The goal of the project was to develop a free UNIX operating system. Until the beginning of the 1990s a collection of tools were available, but the kernel was still missing. The entire software was published under the GNU Public License (GPL) around 1983.

The next step for Linux came from Andrew S. Tanenbaum. At that time he was a professor at the University of Amsterdam. For his students he developed Minix, an operating system for educational purposes to demonstrate and understand the UNIX principles. As he pointed out, Minix was not intended to be used in practice.

Linus Torvalds, a Finish student at the University of Helsinki, was a user of Minix and quite unhappy with its boundaries. In 1990 he began to develop a new operating system based on the ideas of Minix, the UNIX principles, and the POSIX standard. His motivation was to have his own system that was understandable, and maximized to the boundaries of the hardware. He also wanted to have fun, and had no commercial intent in mind. The entire story behind Linux is described in his autobiographical book titled *Just for Fun.* Today, Linus Torvalds oversees the development of the Linux kernel.

To make Linux attractive to the outside world it needed a nice logo. Based on a competition for mascots, a large number of proposals were handed in. Larry Ewing sent in his idea for a penguin as seen on the cover of this book, and his proposal won. Designed with a cheeky smile and a well-fed body this penguin, named Tux, represents the image of a happy and satisfied user.

1.3 Linux Range of Use

Originally designed for Intel-based systems, Linux runs on a variety of platforms today. Among others this includes the ARM architectures (named arm and arm64), Motorola/Freescale's 68k architecture (m68k), Intel x86 (i386 and amd64), IBM s390 (s390), PowerPC (powerpc) and SPARC (sparc).

Right from the beginning Linux focused on server systems. It is in constant use as a web server, file server, mail and news server, internet gateway, wireless router and firewall. Used as a computing unit, it helped to render video sequences and entire films such as Titanic, Shrek and Toy Story. Furthermore, Linux is in use in automotive products, astronautics, military, logistics and the engineering environment. Since 2006, Linux servers run all the world's stock exchanges. It also runs almost all internet search engines.

Over the last decade Linux also conquered the desktop. Due to its high flexibility and stability, it works as a reliable setup for text processing, graphic design, desktop publishing, calculations in spreadsheets, communication (email, chat, audio, and video) as well as user interfaces for your phone and television.

1.4 Linux Certifications

The widespread use of Linux has increased the demand for engineers and users who know exactly what they are doing. At this point a certification for Linux becomes advantageous. These certifications can be divided into programs that are general (not specific to a distribution) and focused (specific to a Linux distribution). The lists below give an overview of the primary certifications that currently exist.

Non-specific Certifications

- Linux Essentials
- LPIC-1: Linux Server Professional Certification
- LPIC-2: Linux Engineer
- Linux Foundation Certified System Administrator (LFCS)
- Linux Foundation Certified Engineer (LFCE)
- CompTIA A+
- CompTIA Network+

Distribution-specific Certifications

- RedHat Certified Engineer (RHCE)
- RedHat Certified System Architect (RHCSA)
- RedHat Certified Architect (RCA)
- SUSE Certified Administrator (SCA)
- SUSE Certified Engineer (SCE)
- SUSE Enterprise Architect (SEA)

2. Software Licenses

As with most products available on the market, software is also packaged with an according license. A software license describes the usage of the software and allows or limits its usage. For commercial software, changes and copies are allowed within rather strict boundaries only. As an example, the license restricts you to use it for 5 users in parallel, and requires you to obtain another license block to add a 6th or 7th user. Common licenses are Shared Source from Microsoft and the Apple License.

For open source software, licenses are much less restrictive. Changes and copies are explicitly allowed, and are even desired in some cases to make improvements for every user and purpose. The goal is to keep the software available for everyone from now and into the future. This so-called *Copyleft* principle ensures that everyone has access to do adaptations if needed, and the restrictions on the use and redistribution of covered software are as minimal as possible. Common licenses that are in use for the single Linux components and its tools are the GNU Public License (GPL), BSD Licenses and the Apache License.

The licenses for open source software follow a number of freedom rules as follows:

- To use the program for every purpose (right of unlimited use - freedom 0)
- To understand how the program works and how to change it according to your needs (right to read the source code of the program - freedom 1)
- To make copies of the software to help your neighbor (right of redistribution - freedom 2)
- To improve the software and to publish your changes so that all other users can also benefit from your improvements (freedom 3)

This ensures that the quality of available software constantly improves and everyone has access to these improvements. Using the GPL, the changes have to be published using the same license. Other open source licenses like BSD do not have this strong requirement.

3. Linux in Day-to-Day Life

There are a few terms that may confuse Linux beginners. The first thing is its name, Linux vs GNU/Linux. As described earlier in Chapter 1, the term Linux refers to the Linux kernel only. In reality many users refer to Linux as the operating system as a whole, the kernel plus libraries and tools. Also the term Linux is used to include all the programs that run on Linux, or that are available for this great operating system.

Furthermore, the description GNU/Linux needs understanding. Linux distributions with this name prefix are fleshed out with GNU implementations of the system tools and programs. One such example is Debian GNU/Linux. As already pointed out in Chapter 1, the GNU project goes back to the initiative of Richard M. Stallman and his dream to develop a free UNIX system. Based on his experiences at MIT and the collaboration with other colleagues he choose to use free software that was already available to rewrite the tools he needed. This included the TeX typesetting system as well as X11 window system. He published the rewritten tools under the GPL license whenever possible to make his work available freely to everyone who was interested in it.

Next we will have a closer look at the different Linux distributions.

3.1 What is a Linux Distribution?

A Linux distribution is a collection of software packages that fit together. A distribution is maintained by a team of software developers. Each member of the team focuses on a different package of the distribution. Together as a team they ensure that the single software packages are up-to-date and do not conflict with the other packages of the same release of the distribution.

As of 2018 for Debian GNU/Linux 9, the official repositories contain more than 51,000 different packages. A repository is a directory of packages with a certain purpose. Debian GNU/Linux sorts its packages according to the development state. The official repository is named *stable* and reflects the current release of stable packages. The other repositories are named *testing* and unstable, and work in the same way but do not count as official packages.

Typically a Linux distribution comprises of packages for a Linux kernel, a boot loader, GNU tools and libraries, a graphical desktop environment with a windows environment, as well as additional software like a web browser, an

email client, databases and documentation. The software is provided in two ways; as the source code and as the compiled binary packages. This allows you to understand how the software is designed, to study it and to adjust it according to your personal needs. This step is described as freedom 1 on the list shown in Chapter 2.

Depending on the focus of the Linux distribution, it also contains packages for a specific purpose like network or forensic tools, scientific software for educational purposes, and multimedia applications. More details are given below.

3.2 Which Linux Distributions Exist?

According to Distrowatch, more than 600 different Linux distributions exist. Major distributions are Debian GNU/Linux, Ubuntu, Linux Mint, Red Hat Enterprise Linux (RHEL), Fedora, CentOS, openSUSE Linux, Arch Linux, Gentoo and Slackware. One of the major questions is: which Linux distribution to use? Based on our experience these are the recommendations:

- For beginners: Ubuntu, Xubuntu, openSUSE, Linux Mint
- For advanced users with experience: Debian GNU/Linux, Red Hat Enterprise Linux (RHEL), Fedora, CentOS
- For developers: Arch Linux, Gentoo, Slackware

For the examples in this book we use Debian GNU/Linux. Even though this distribution is recommended for advanced users it is still very beginner friendly, which we will show later in the guide. But the most important reason for this selection is its stability and the trust in this Linux distribution that was built up during the last 20 years of permanent use as a server and desktop system. Other Linux distributions fluctuate too much for comfort.

You are more than welcome to choose a different Linux distribution. The majority of this book applies to most distributions. But if you are a complete novice, we highly recommend sticking to Debian GNU/Linux at least for the duration of this guide, as we will go step by step through its setup and configuration.

In general, choosing a Linux distribution can depend on several criteria as stated below:

- By its availability: free or commercial use

- By its purpose: desktop, server, Wi-Fi router/network appliance
- By the intended audience: end user, network engineer, system administrator, developer
- By the package format: .deb, .rpm, .tar.gz
- By the time updates are available: every Linux distribution follows its own update cycle
- By the support that is provided: support can be free (community-based) or with costs (based on a support contract)

When selecting a distribution, we recommend one that is stable, that is updated regularly and fits into the purpose you need the computer for. Below you will find a short description for each of the Linux distributions mentioned above.

Debian GNU/Linux

Established in 1993, Debian GNU/Linux (Debian for short) is an entirely free and community-based operating system that follows the GNU principles. More than 1,000 developers continuously work on it based on their own free will. Behind Debian is no company and there are no business interests involved.

One design goal is to have a stable and reliable operating system for computers that are actively delivering services. It is targeted to users who know what they want and have experience. The Debian developers maintain and use their own software. The packages are made available in .deb format, and are divided into categories according to the following licenses:

- Main: free software
- Contrib: free software that depends on non-free software
- Non-free: packages that have a non-free license

Debian works excellent on both servers and desktop systems. A range of architectures are supported like ARM EABI (arm), IA-64 (Itanium), mips, MIPSel, powerpc, s390 (32 and 64 bit), as well as sparc, i386 (32 bit) and amd64 (64 bit). The code name of each release is based on the name of a character from the film Toy Story, such as *Stretch* for Debian GNU/Linux 9.

Ubuntu

Ubuntu is a free Linux distribution that is financed by the company Canonical Ltd. It is based on Debian but focuses on beginners instead. That's why it contains just one tool per task. Also, the Ubuntu team tries to incorporate brand new elements that lack stability. The packages are made available in .deb format, and are divided into categories according to their support from Canonical:

- Main: free software, supported by Canonical
- Restricted: non-free software, supported by Canonical
- Universe: free software, unsupported
- Multiverse: non-free software, unsupported

Ubuntu is available in three official editions: Ubuntu Desktop, Ubuntu Server, and Ubuntu Core (for the Internet of Things). Supported are a range of architectures like i386, IA-32, amd64, ARMhf (ARMv7 VFPv3-D16), ARM64, powerpc (64 bit) and s390x.

Initially published in 2004, there are two releases per year: one in April and another in October. The release is reflected by the version number: 18.04 refers to the April release of the year 2018. The code name of a release is based on an adjective and an animal, such as Utopic Unicorn for Ubuntu 14.10.

Linux Mint

Linux Mint is a non-commercial distribution that is based on Ubuntu and follows its release scheme. The initial publication dates back to the year 2006. As of 2014 there have been two releases per year following the release from Ubuntu by one month. The code name for the release is a female name that ends with an *a*, such as Felicia for version 6. Linux Mint supports the two architectures IA-32 and amd64. The target of the distribution is desktop users that can use it easily.

Red Hat Enterprise Linux (RHEL) RHEL is a commercial Linux distribution. It is based on the combination of Red Hat Linux (available between 1995 and 2004) plus Fedora 19 and 20. Its original release dates back to the year 2000. Its focus on business customers includes long-term support, training, and a certification program (see Chapter 1). Red Hat's community project is called Fedora (see below).

The packages are made available in .rpm format (Red Hat Package Manager). RHEL supports the architectures arm (64 bit), i386, amd64, powerpc, as well as s390 and zSeries. The distribution targets both servers and desktops. The code name for the release looks rather random, as it does not follow a similar scheme as used for Debian or Ubuntu.

Fedora

Fedora is a community Linux distribution, aimed mainly at desktop usage. It is based on Red Hat Enterprise Linux (RHEL) and sponsored by Red Hat. It was launched in 2003 at the time the support for Red Hat Linux ended. As of 2018 it is available in the following versions:

- Workstation: for pc
- Server: for servers
- Atomic: for cloud computing

Fedora supports the architectures amd64, ArmHF, powerpc, mips, s390 and RISC-V. The distribution has a rather short lifecycle where a new release follows roughly every 6 months. The code name for a release does not follow a fixed naming scheme but mostly consists of city names.

CentOS

CentOS abbreviates from the name Community Enterprise Operating System. As with Fedora it is based on Red Hat Enterprise Linux, and compatible in terms of the binary packages. This allows the use of software on CentOS that is initially offered and developed with RHEL in mind. In contrast to Fedora it focuses on enterprise use for both desktop and server, with long-term support. The initial release of CentOS goes back to May 2004. The software packages come from three different repositories:

- Base: regular, stable packages
- Updates: security, bug fix or enhancement updates
- Addons: packages required for building the larger packages that make up the main CentOS distribution, but are not provided upstream

CentOS is available for the architectures i386 and amd64. Other architectures are not supported.

openSUSE

The Linux distribution openSUSE has its roots in the distributions SUSE Linux and the commercial SUSE Linux Professional that saw its first release in 1994. The name SUSE is an abbreviation for the original German owner named *Gesellschaft für Software- und Systementwicklung GmbH*.

OpenSUSE is based on the structures of Red Hat Linux and Slackware, and uses .rpm as a software archive format. It is available for the architectures i586, x86-64 and ARM. The openSUSE project aims to release a new version every eight months. As with Fedora, the code name for a release does not follow a fixed naming scheme.

Arch Linux

Arch Linux is a free Linux distribution that saw its first release in 2002. It follows the principle of a rolling release, which results in monthly releases of the distribution. Currently the core team consists of about 25 developers and is supported by a number of other developers, called trusted users. Arch Linux uses Pacman as a package management system. The single packages are held in four software repositories:

- Core: packages for the basic system
- Extra: additional packages like desktop environments and databases
- Community: packages that are maintained by trusted users
- Multilib: packages that can be used on several architectures

Arch Linux supports the architecture amd64. The early releases until 2007 had code names that do not follow a specific scheme.

Gentoo

As with Arch Linux, Gentoo follows the principle of a rolling release. New installation images are available weekly, with the first release available in 2002. Gentoo is special due to being a source code based distribution. Before installing the software, it has to be compiled first. Supported architectures are alpha, amd64, arm, hppa, IA-64, m68k, mips, powerpc, s390, sh and sparc.

Slackware

Slackware is the oldest active Linux distribution. The first release dates back to 1992. Regular releases are available without a fixed interval. It targets the professional user, and gives him/her as much freedom as possible. Slackware uses compressed tar.gz archives as a package format, and supports the four architectures i486, alpha, sparc and arm. The distribution was also ported to architecture s390.

4. Setting up a Linux System

As mentioned in the previous chapter, we will be using Debian for our demonstrations. To recap, Debian is a distribution that provides great stability and scales up exceptionally well once your skills and knowledge progress past the beginner stages. In this chapter we will install and configure Debian, showing you every single step along the way.

4.1 Types of Installations

Debian offers a variety of methods for a proper setup. This includes a graphical and a text-based installation; we will use the former. For installation media the Debian developers offer three variants:

- A CD or DVD for 32 bit and 64 bit
- A network image for 32 bit and 64 bit (a so-called Netinst-ISO)
- A tiny CD for 32 bit or 64 bit

We also have test media available. These include live images for 32 bit and 64 bit, and allow you to try Debian before installing it on your computer. During the time of writing this document, version 9.5 is the current stable release of Debian. The setup described here is based on this release and the amd64 architecture.

After downloading the network image from www.debian.org/distrib no further static images are required to be referenced in the system. Instead, it depends on the internet connection to retrieve the packages to be installed and keep your operating system up-to-date.

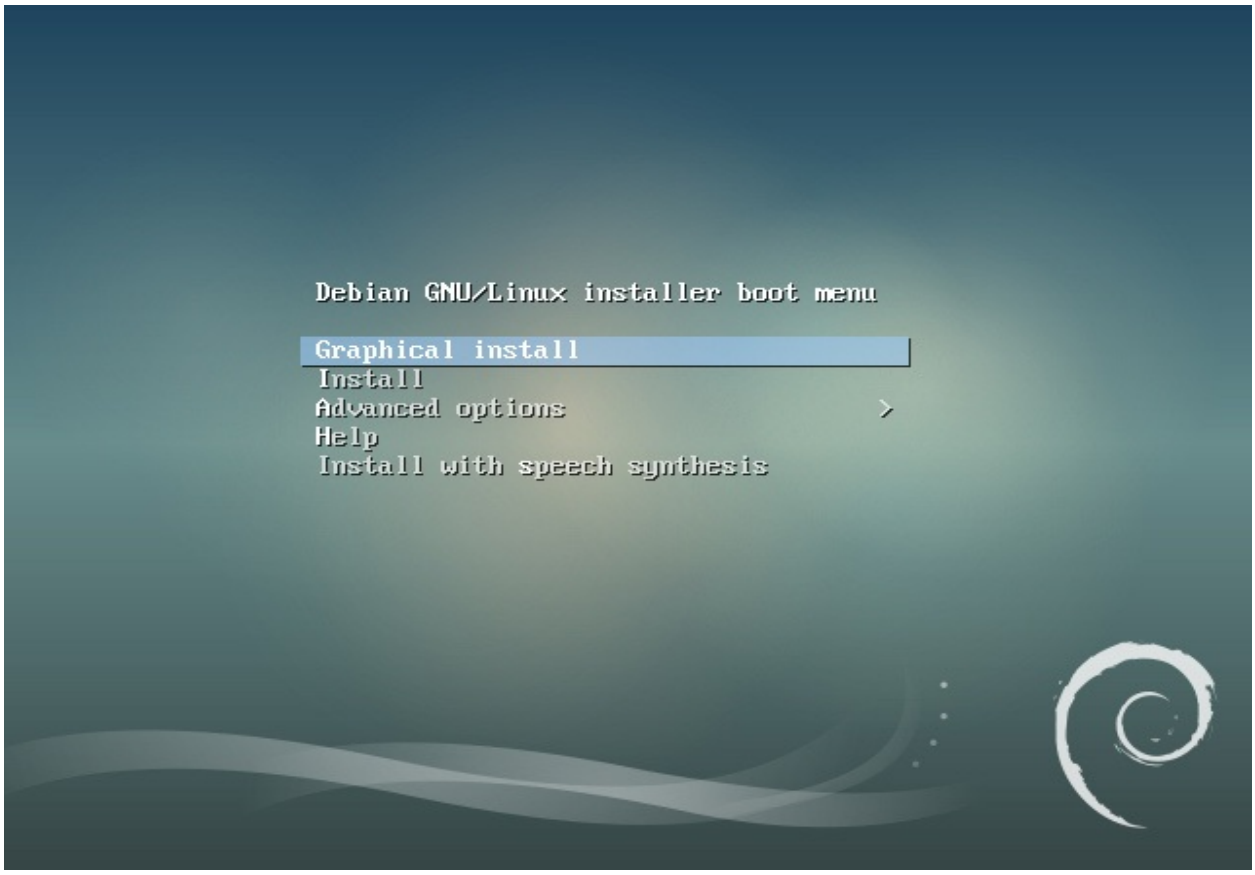
The entire process will take you about an hour and it allows you to have a lean software selection according to your specific needs. Software packages that you do not use will not be available on your system. They can be added whenever you feel the need for them.

The target system of our installation is an XFCE-based desktop system for a single user with a web browser and a music player. For the web browser we use Mozilla Firefox and for the music player, VLC. Both programs are a permanent component of the Linux distribution. The environment we use for demonstration purposes is a virtual machine based on VirtualBox with 4 GB of RAM and 15 GB of disc space.

4.2 Installing Linux Step-by-Step

Boot Menu

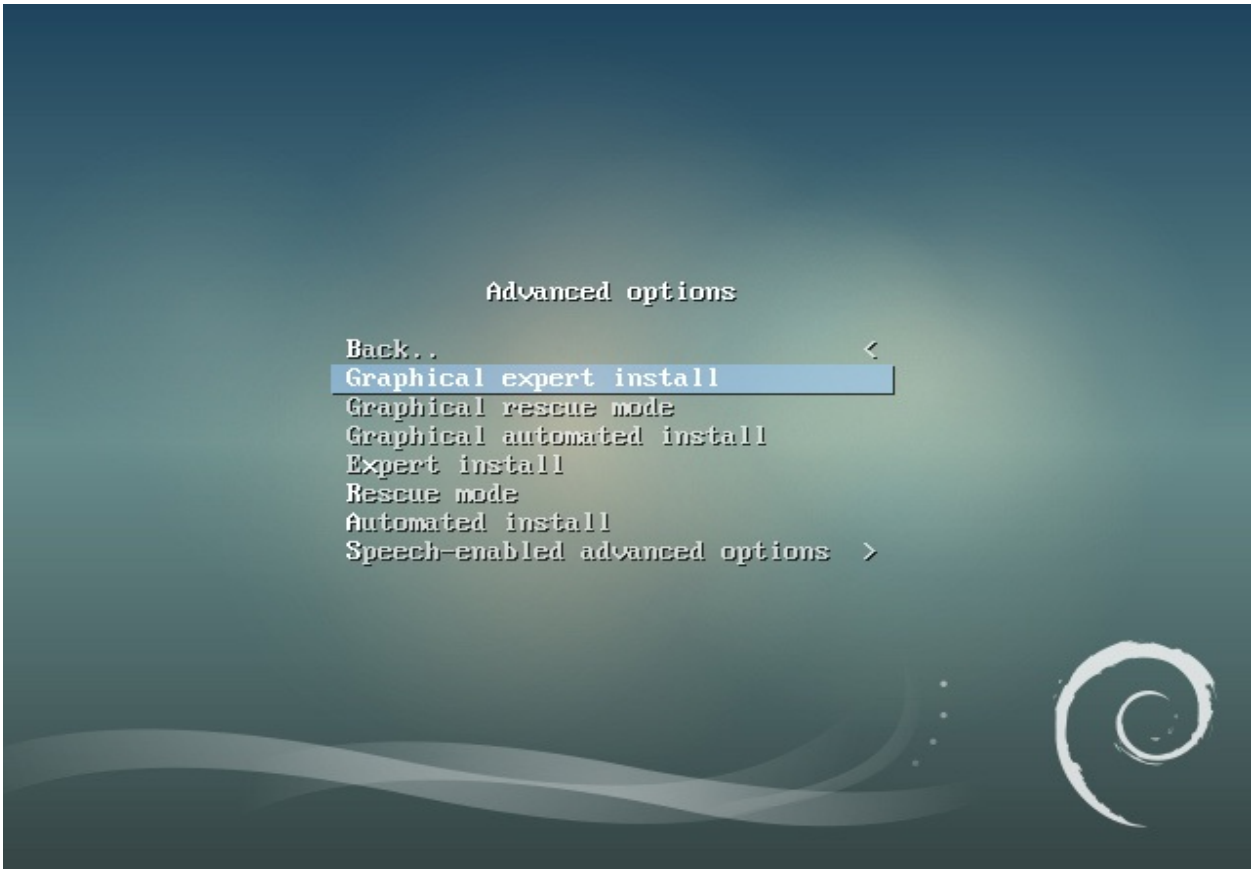
In order to begin with the installation of Debian, first boot the computer (in our case the VirtualBox image) from the ISO image you have downloaded. If you are also using a virtual machine, see your virtual machine vendor website for help on how to enable the ISO image. Next, wait for the boot menu to appear on the screen. The image below shows you the different options that are offered. Using the cursor keys you can navigate the boot menu, and the Enter key selects an entry.



The different options are:

- Graphical install: start the installation process using a graphical installer
- Install: starting the installation process using a text-based installer
- Advanced options: select further options like Expert mode, Automated install or Rescue mode (see image below for more details)

- Help: get further help
- Install with speech synthesis: starting the installation process with speech support



From the main boot menu choose the entry *Graphical install*, and press the Enter key to proceed.

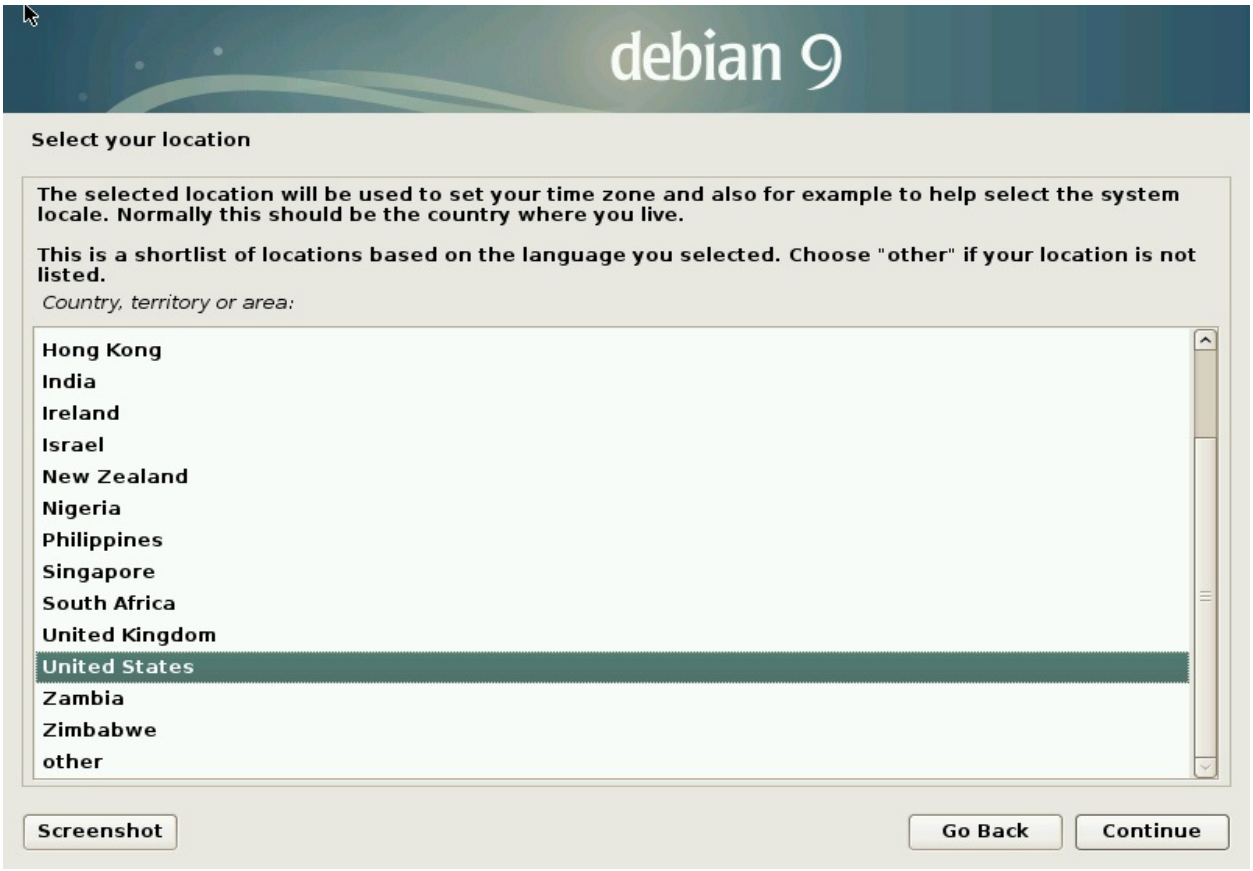
Language Selection

Next, choose the language you prefer to be used during the entire installation process. The dialogs and messages are translated accordingly. This selection does not determine irrevocably the language your Linux system will have, you can always choose a different language later.

The image below shows the dialog box. English is already pre-selected, and so you just have to click the Continue button on the lower-right corner of the dialog box to proceed.

Location Selection

Third, make a selection regarding your location (see image below). Based on your language setting made before, the countries are listed in which the chosen language is mainly spoken. This also influences the locale settings like the time zone your computer is in. In order to have a different setting choose the entry titled *other* from the end of the list and go on from there. When you are done, click the Continue button to proceed with step four.



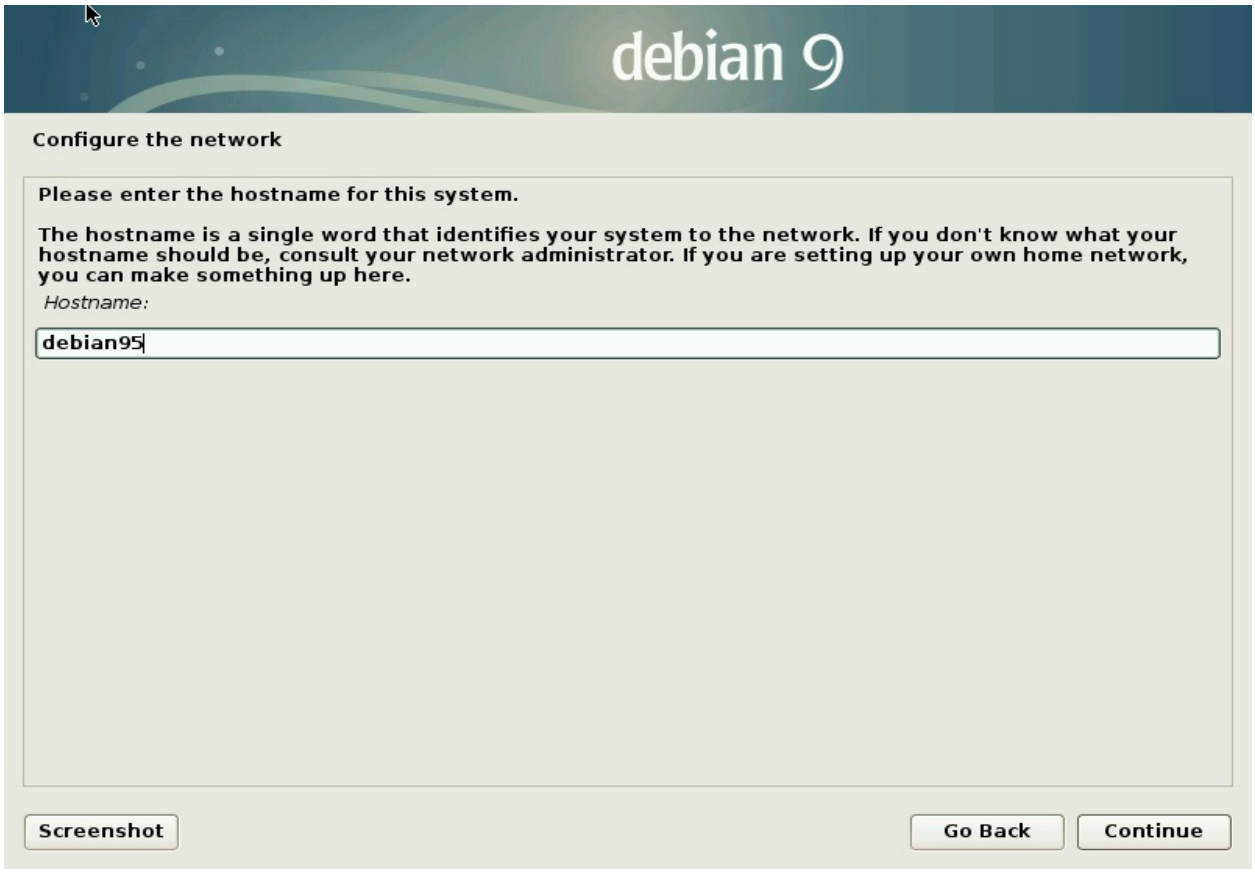
Keyboard Selection

Fourth, choose your keyboard layout from the list (see image below). For the United States the pre-selection is American English. If you use a different keyboard layout select the right one from the list. If done click the Continue button to proceed with step five.

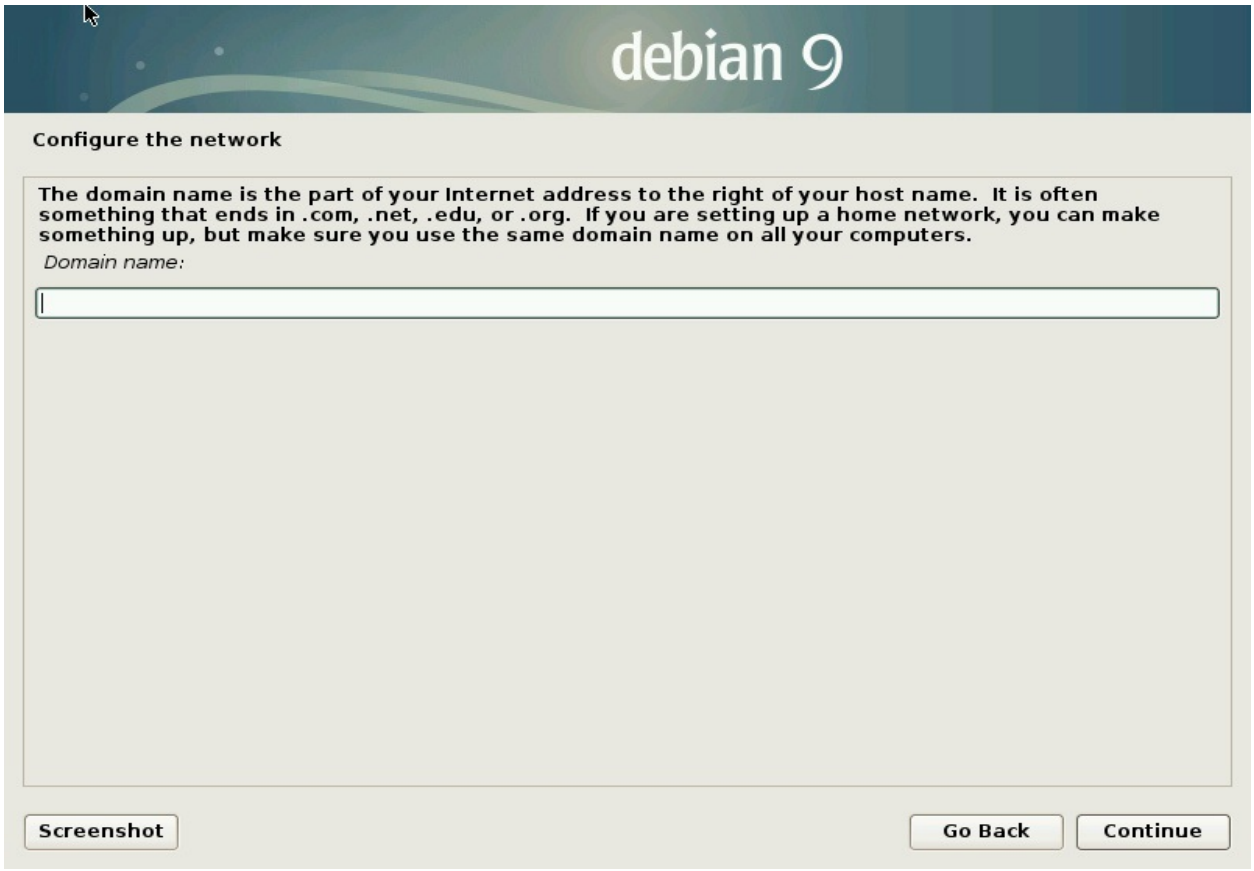
Network Setup

Step five includes loading the installer components from the ISO image, and the detection of the network hardware in order to load the correct network driver. Then, the installer tries to connect to the internet to retrieve an IP address via DHCP from your local network service.

When done, you can set up the hostname of your computer (see image below). Choose a unique name for your machine that consists of a single name and does not exist yet in your local network segment. It is common to use names of fruits, places, musical instruments, composers and characters from movies. In this case we choose the name debian95 that simply represents the Linux distribution and its version number.



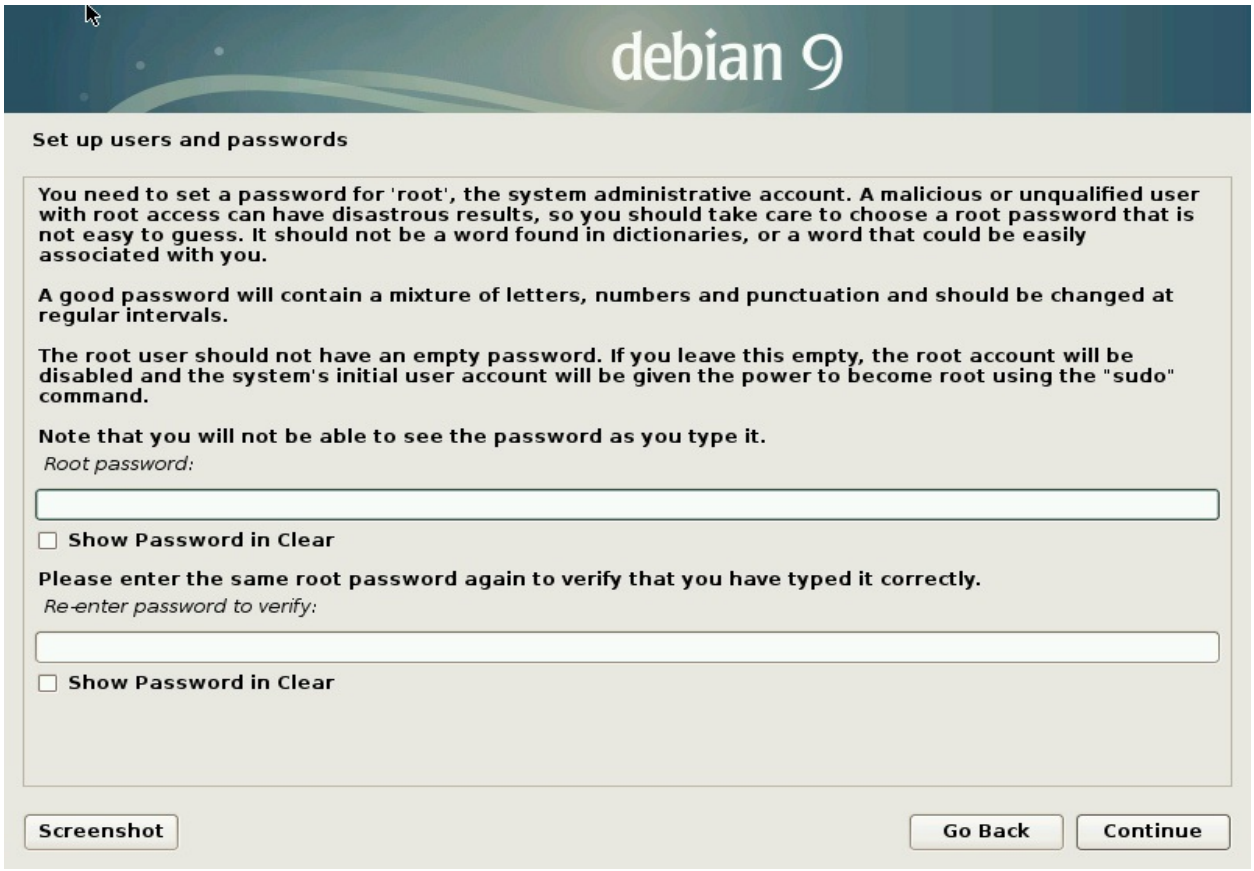
When you are done, click the Continue button to proceed with step six to add a domain name like yourcompany.com (see image below). In this case it is not needed. That’s why we leave the entry field empty. Click the Continue button on the lower-right corner to proceed with the installation.



Users and Roles

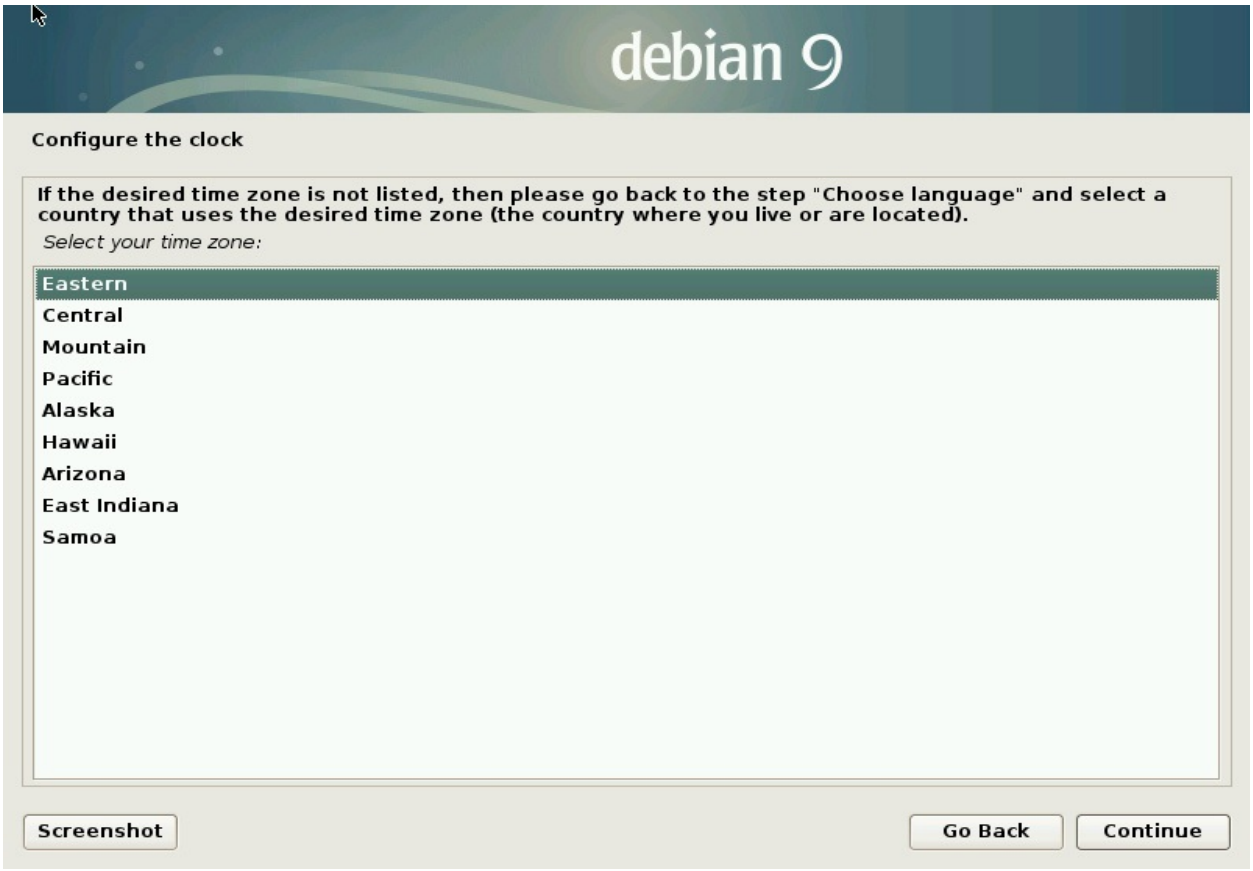
Our Linux system needs at least two users in order to be operated properly. One is an administrative user that has a fixed name root and the other is a regular user that we just give the name of *User* in this case.

In the next two steps you set the password for the user root (see image below) and both the full name and account name for the regular user. For simplicity we use *Debian User* as the full name and *User* as the account name. For both users, choose a password that is dissimilar and that you can remember. You will need these passwords later in order to log onto your computer.



Time Zone

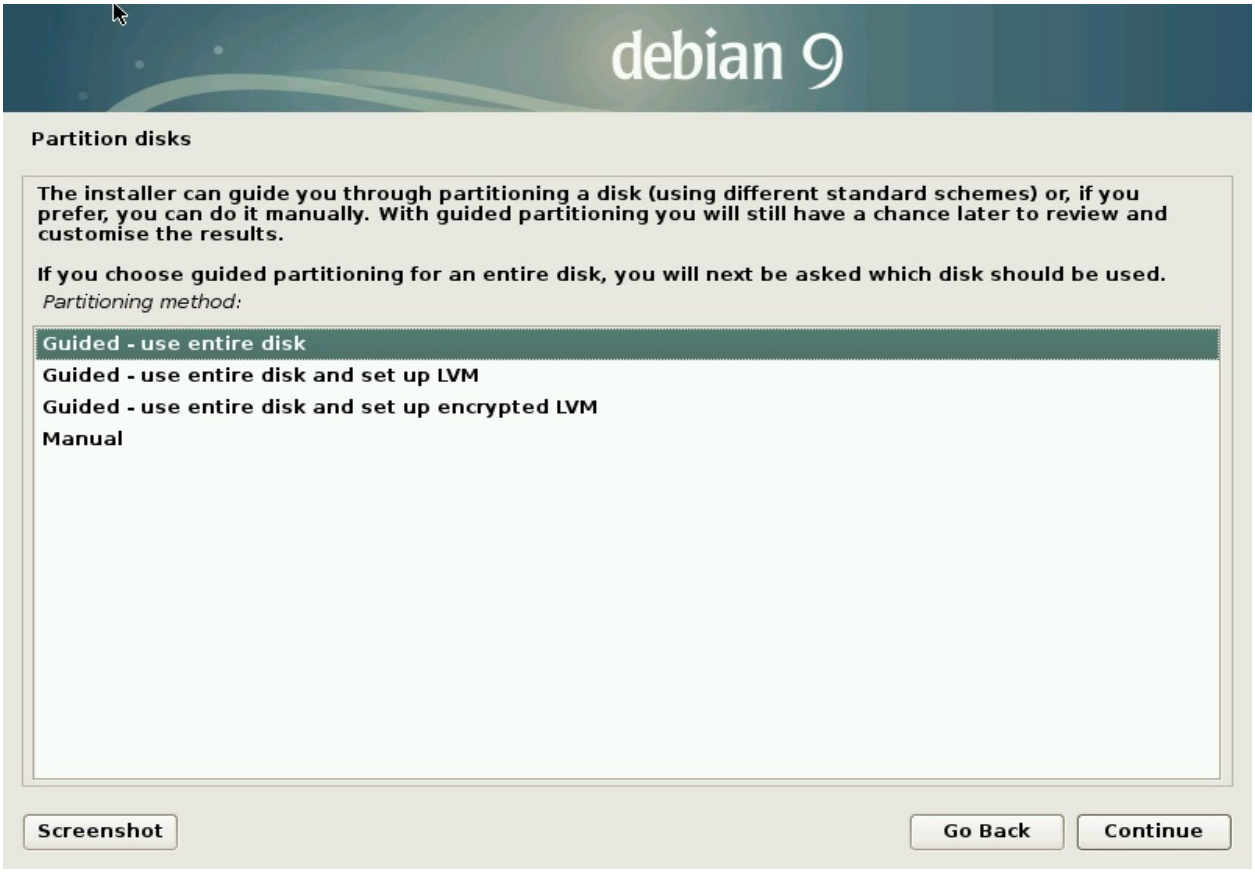
Setting the correct time zone is of significant importance for communication with other services, especially in a network. Choose the value from the list as seen in the image below. The entries in the list are based on the location you have selected before. When done, click the Continue button to define the storage media and the accompanying partitions.



Storage Media and Partitioning A Linux system can be distributed across a number of different storage media like hard disks and flash drives. Over and above, a storage media can be separated into multiple disk partitions. In order to do so, the setup program of Debian has the following methods available (see image below):

- Guided - use entire disk: follow the steps as provided and use the entire disk space for the Linux installation. This creates partitions with fixed sizes.
- Guided - use entire disk and set up LVM: follow the steps as provided and use the entire disk space for the Linux installation. This option makes use of Logical Volume Management (LVM) in order to create partitions with sizes that can be changed later on.
- Guided - use entire disk and set up encrypted LVM: follow the steps as provided and use the entire disk space for the Linux installation. This option makes use of Logical Volume Management (LVM) in order to create encrypted partitions with sizes that can be changed later on.

- Manual: create partitions individually. This is the expert mode and requires deeper knowledge about partitions and file system parameters.



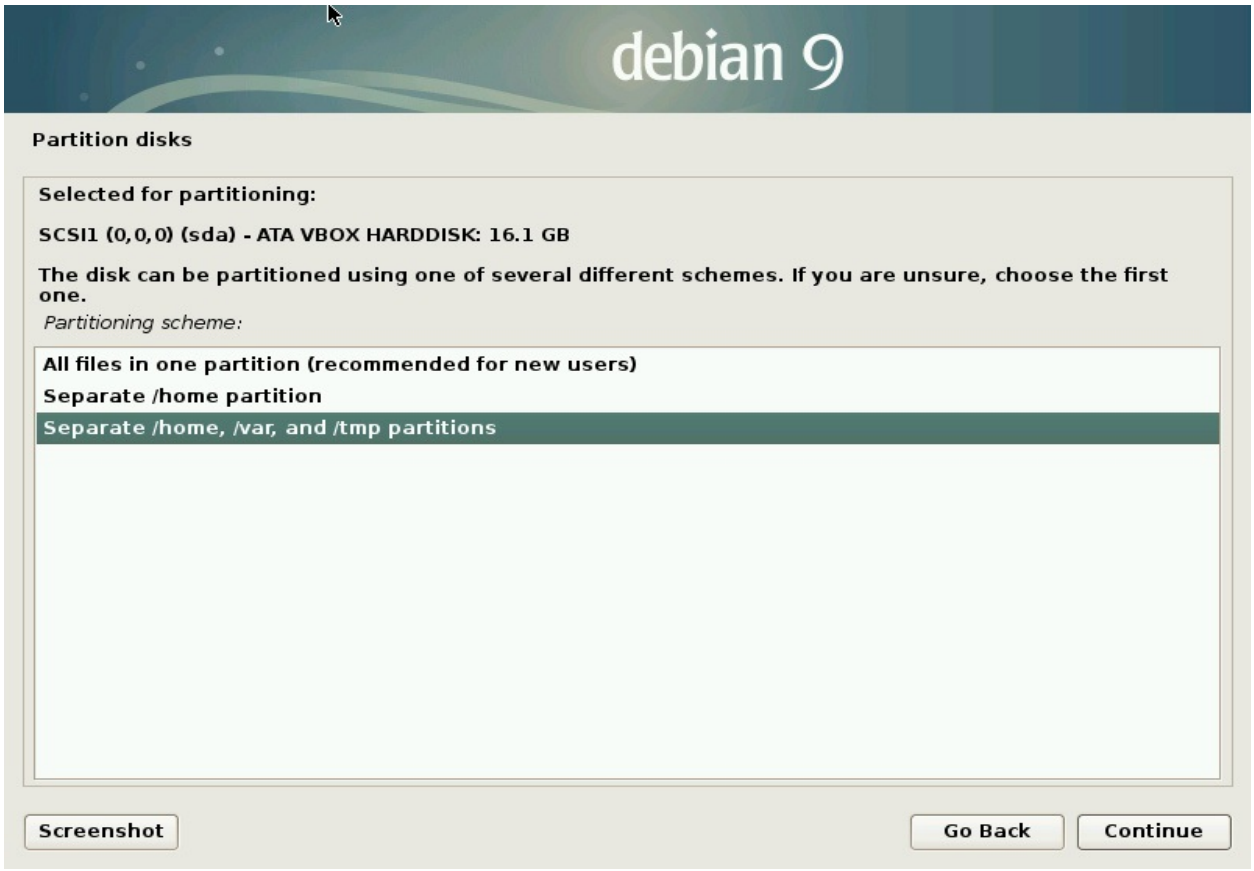
From the list choose the entry *Guided - use entire disk*. The values for partition sizes are chosen according to experience, implemented as an algorithm. A manual calculation is not required. Click the Continue button on the lower-right corner to proceed with the installation.

Next, select the disk to partition. In our case we have only one disk available (see image below). Later on in this guide the disk will be referred to as /dev/sda for the 1st SCSI disk.

A disk partition refers to a piece of the storage media that is organized separately and is intended to contain a branch of the Linux file system tree. There is no universal way to do this separation correctly. This guide shows a simple but safe solution that works for a basic system. The menu in the dialog box offers the following options:

- All files in one partition: use just a single partition to keep programs and user data
- Separate /home partition: store programs and user data in separate partitions
- Separate /home, /var, and /tmp partitions: keep user data, variable data and temporary data in separate partitions

As shown in the image below, choose the third entry *Separate /home, /var, and /tmp partitions*. Click the Continue button on the lower-right corner to proceed with the installation.

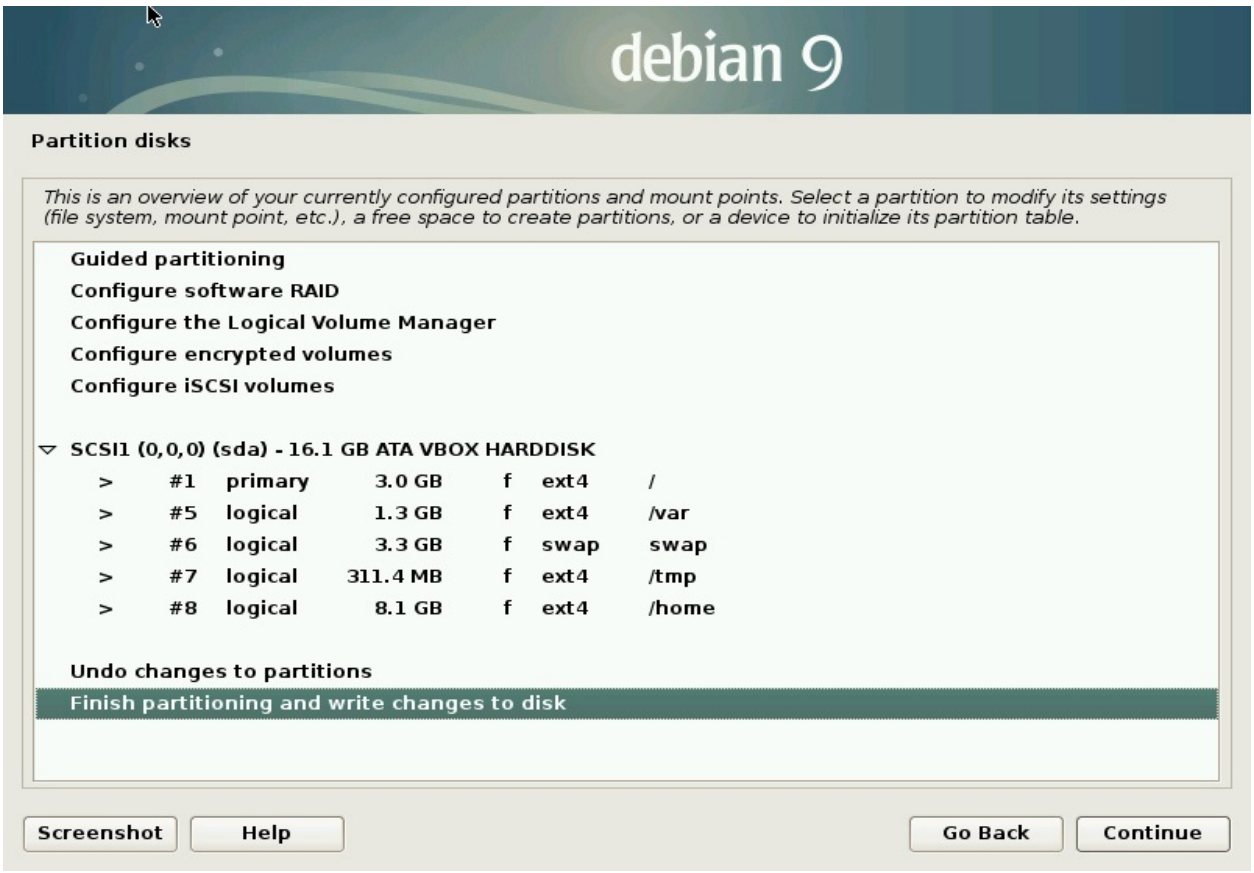


The next step is to confirm the partition scheme. This is calculated automatically based on experience and contains these partitions:

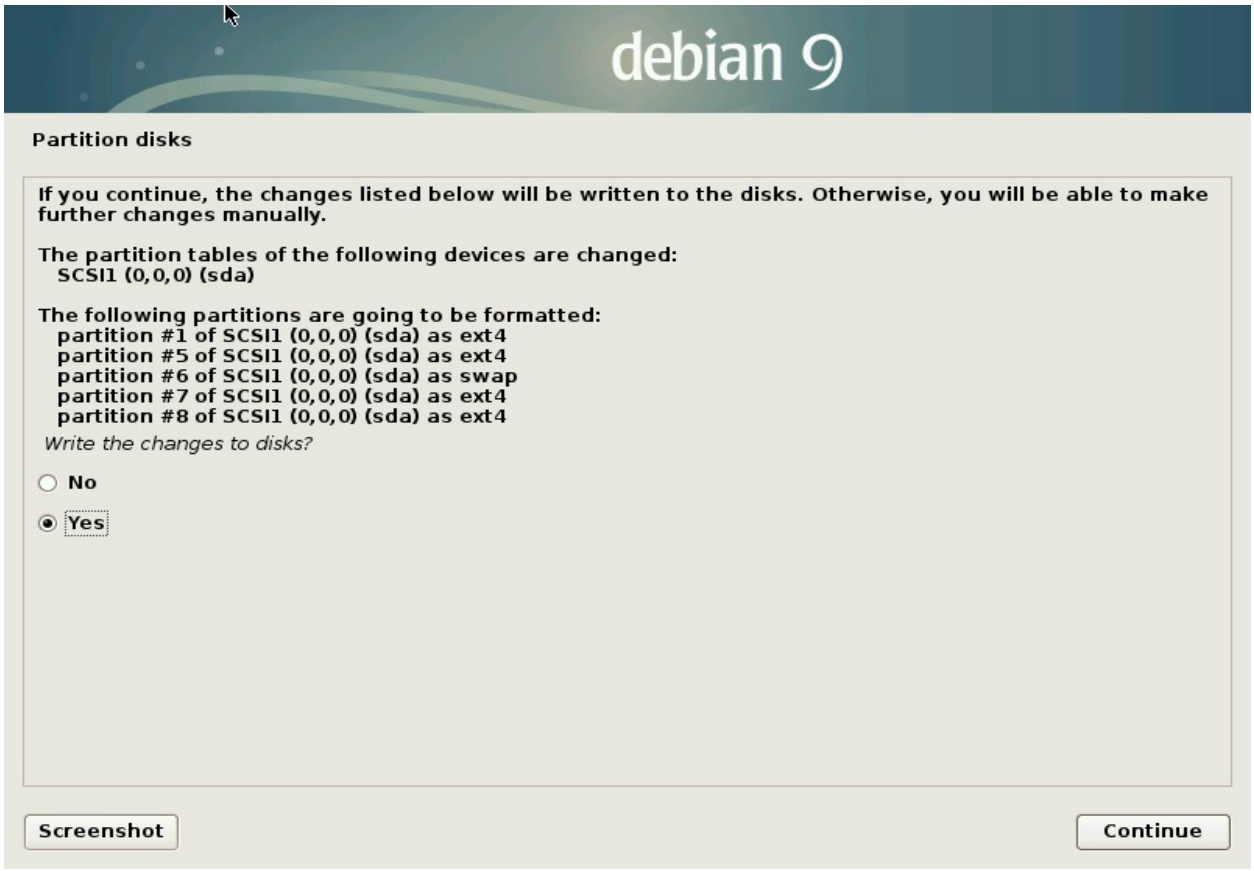
- sda1: the first partition of the first SCSI disk is a primary partition with a size of 3 GB, formatted with the ext4 file system, and referred to as the root part of the file system tree (indicated with /)
- sda5: the fifth partition of the first SCSI disk is a logical partition with a size of 1.3 GB, formatted with the ext4 file system, and reserved to store variable data of the file system tree (indicated with /var)
- sda6: the sixth partition of the first SCSI disk is a logical partition with a size of 3.3 GB, formatted as a swap file system
- sda7: the seventh partition of the first SCSI disk is a logical partition with a size of 311 MB, formatted with the ext4 file system, and reserved to store temporary data of the file system tree (indicated with /tmp)
- sda8: the eighth partition of the first SCSI disk is a logical partition with a size of 8.8 GB, formatted with the ext4 file system, and reserved to store the user data of the file system tree aka home directories (indicated with

/home)

Due to historical reasons a hard disk can contain four primary partitions only. The fourth one is called an *Extended Partition* if divided into so-called logical partitions or logical drives. In our case the logical partitions /dev/sda5, /dev/sda6, /dev/sda7 and /dev/sda8 are stored on the primary partition /dev/sda4. The partitions /dev/sda2 and /dev/sda3 are not in use.



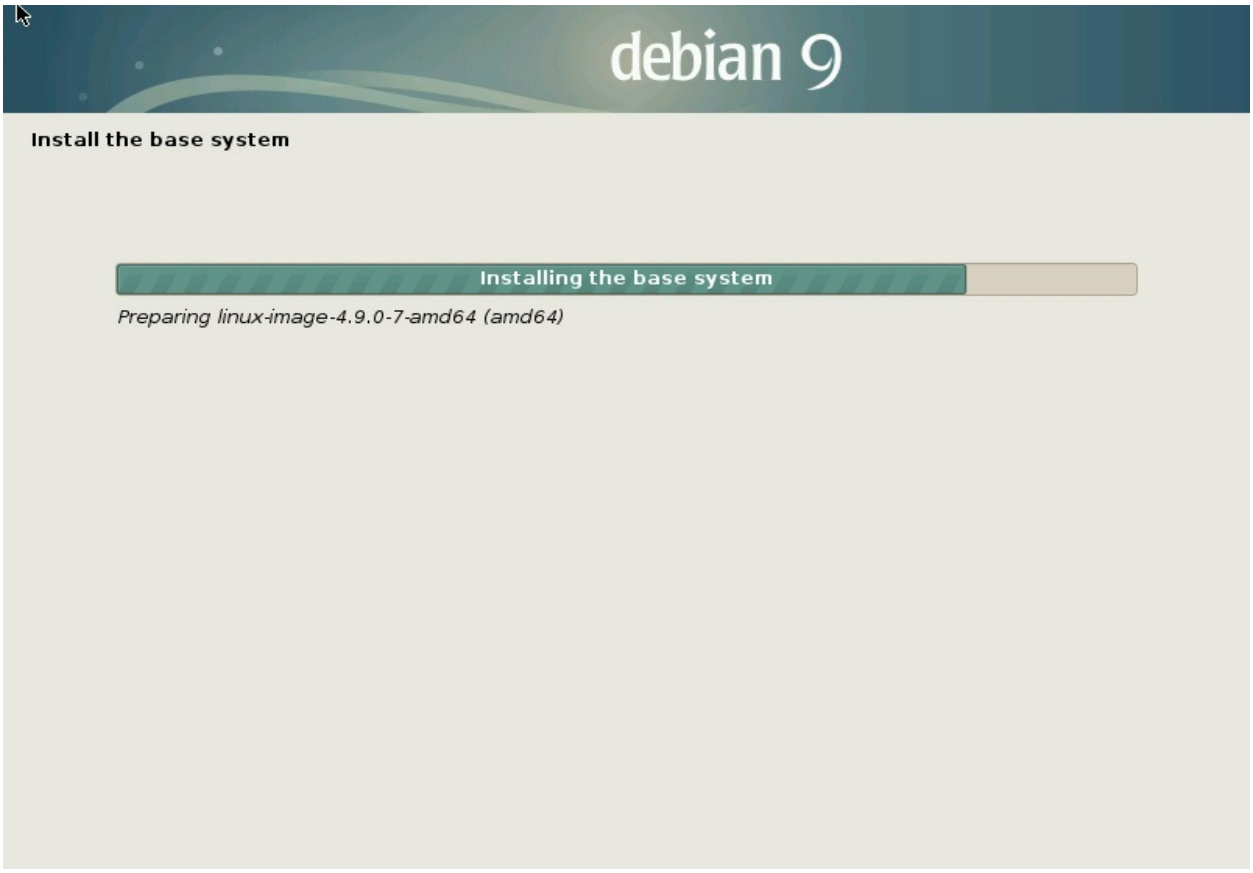
From the above list, choose the entry *Finish partitioning and write changes to disk*. Click the Continue button on the lower-right corner to proceed and to confirm the partition scheme (see below image). Choose *yes* from the list and click the Continue button to partition the disk. Note that all the data on the selected storage device will be lost and the disk will be empty.



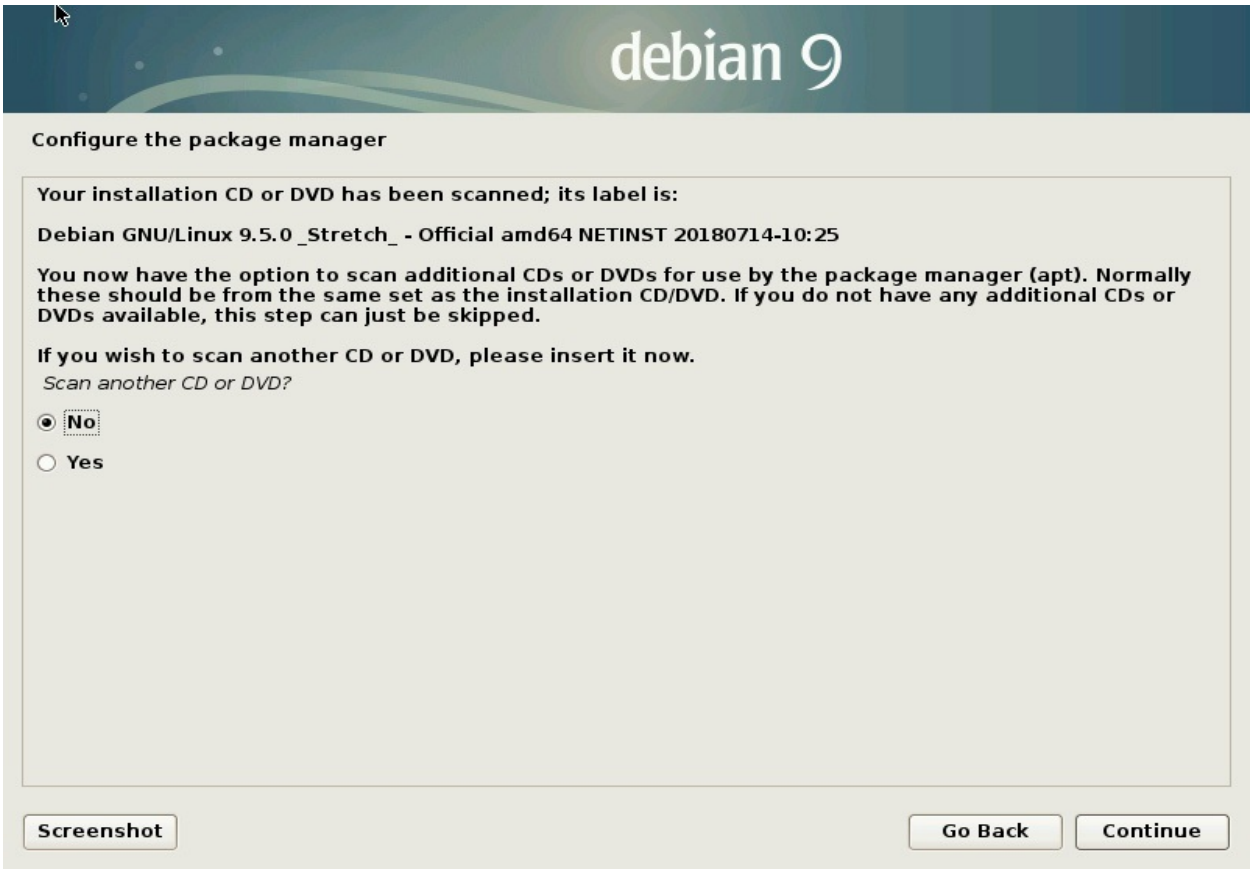
Having divided the storage media into single partitions, the partitions will be formatted with the file system as defined before. In our case the partitions /dev/sda1, /dev/sda5, /dev/sda7 and /dev/sda8 will get an ext4 file system, and the partition /dev/sda6 will get a swap filesystem. As soon as this step is completed, the base system of Debian will be installed next.

Package Management

The ISO image contains the installer and a number of packages to set up the Linux base system. For example this includes the Linux kernel being installed below.

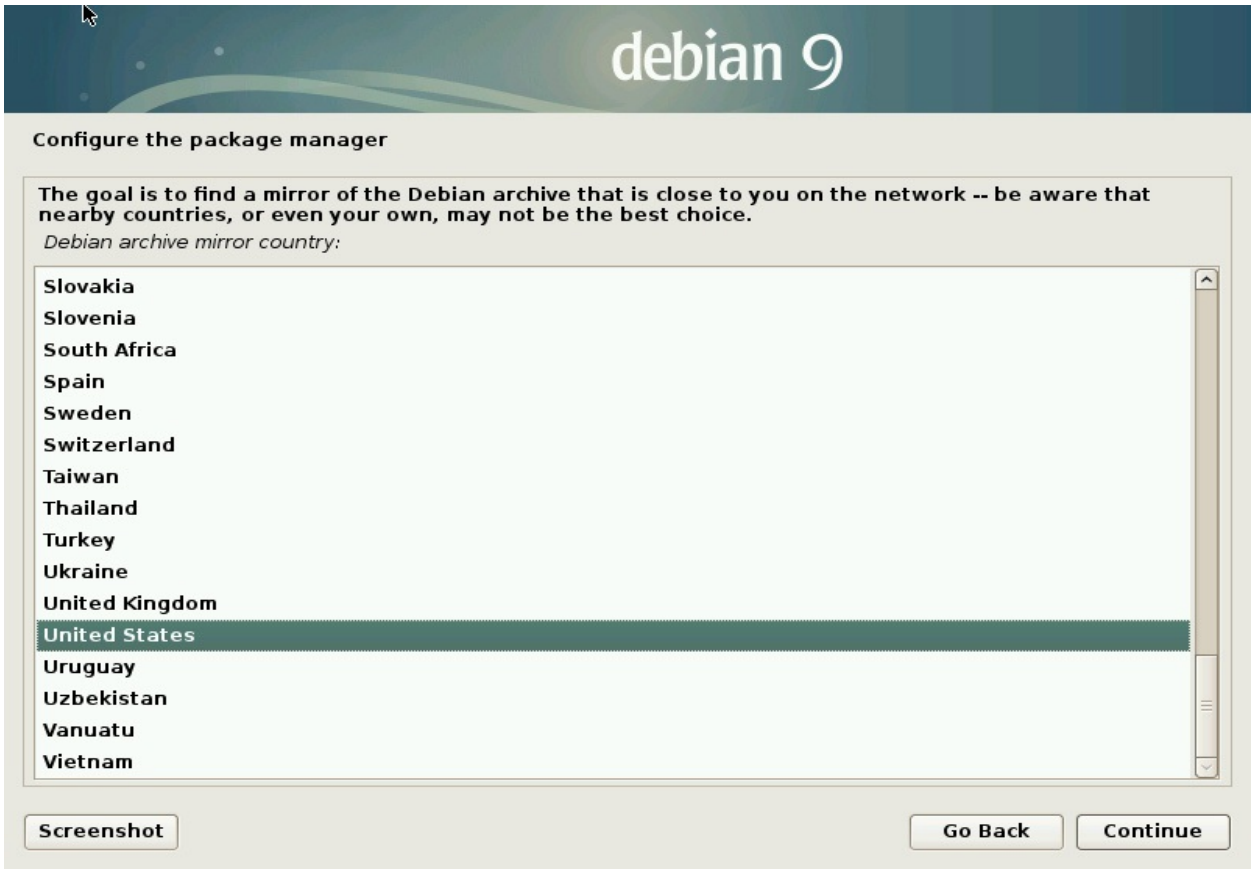


Next you need to decide whether to use additional installation media or not. In our case we have just a single installation disk and can consequentially select No from the menu (see image below). Then click the Continue button to proceed.



As pointed out earlier, the software for Debian is organized in packages. These packages are provided in multiple software repositories. The repositories are made available via package mirrors that are maintained by universities, private persons, companies and other organizations. These mirrors are located in different countries.

In the next step you will have to decide from which country you would like to retrieve your Debian packages. It is recommended to choose a mirror that is geographically located near you to minimize the time that is needed to transport the data from the mirror to your computer via network. As an initial step, choose your desired country.



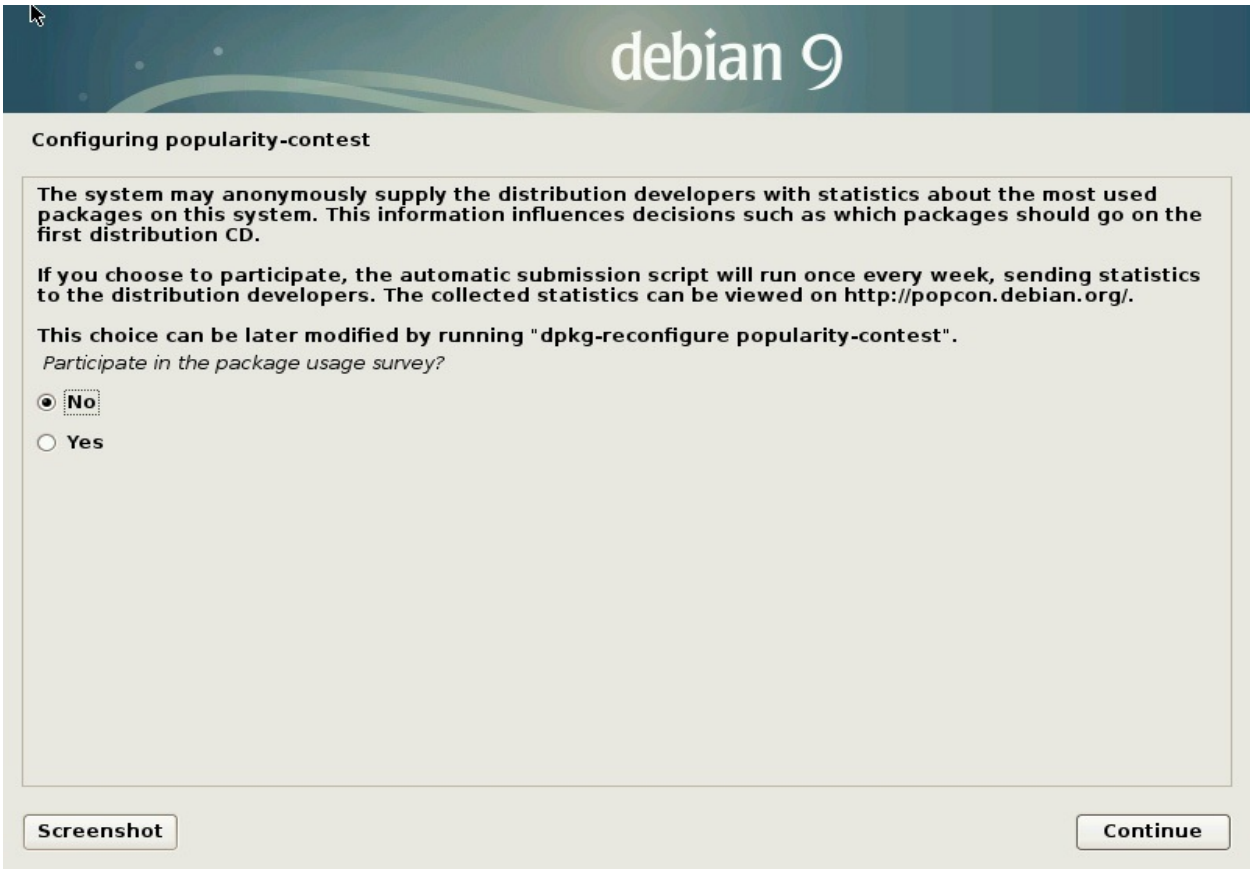
As a second step choose a preferred mirror from the list (see image below). The list contains universities, internet providers, government services and other organizations.

In case your computer network includes a proxy server to communicate with the outside world, enter the according information here. In this case we don’t have that and leave the entry field empty. Click the Confirm button to proceed.

As soon as the single parameters are set, the Debian installer connects to the previously selected package mirror and retrieves the package lists from there. A package list contains the packages that are available, including the name, size and description. Depending on the quality and bandwidth of your network connection this step can take a while.

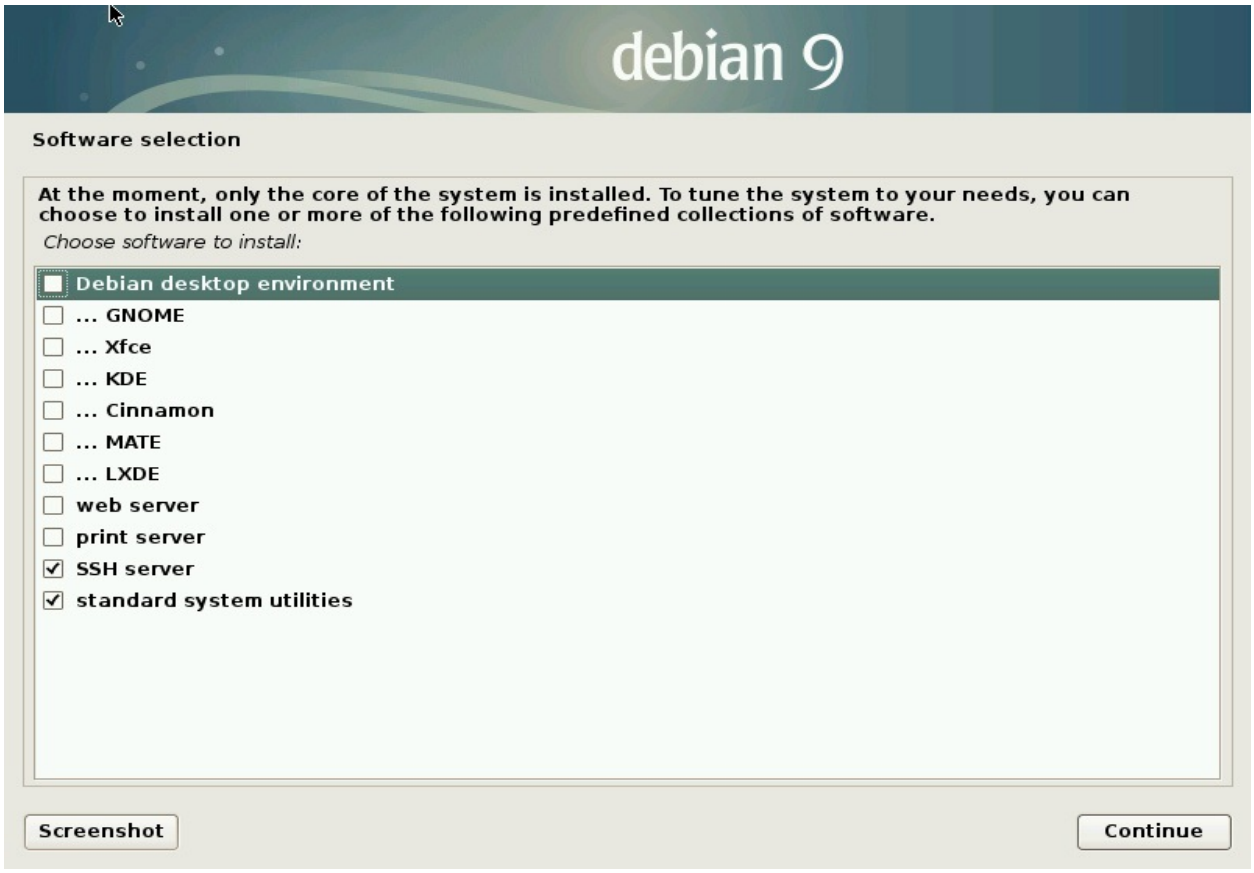
Next, you are asked to take part in Popcon, the Debian package popularity contest (see image below). This information is optional, and is used only by the team of developers that is responsible for the Debian packages. Based on Popcon, they figure out which Debian packages are the ones that are installed most often. The information has a direct influence on the preparation of installation images and which packages to keep or to dismiss for the different architectures.

In our case we do not participate in Popcon, and leave the selection to No. Click the Confirm button to proceed with the selection of software tasks (tasksel).

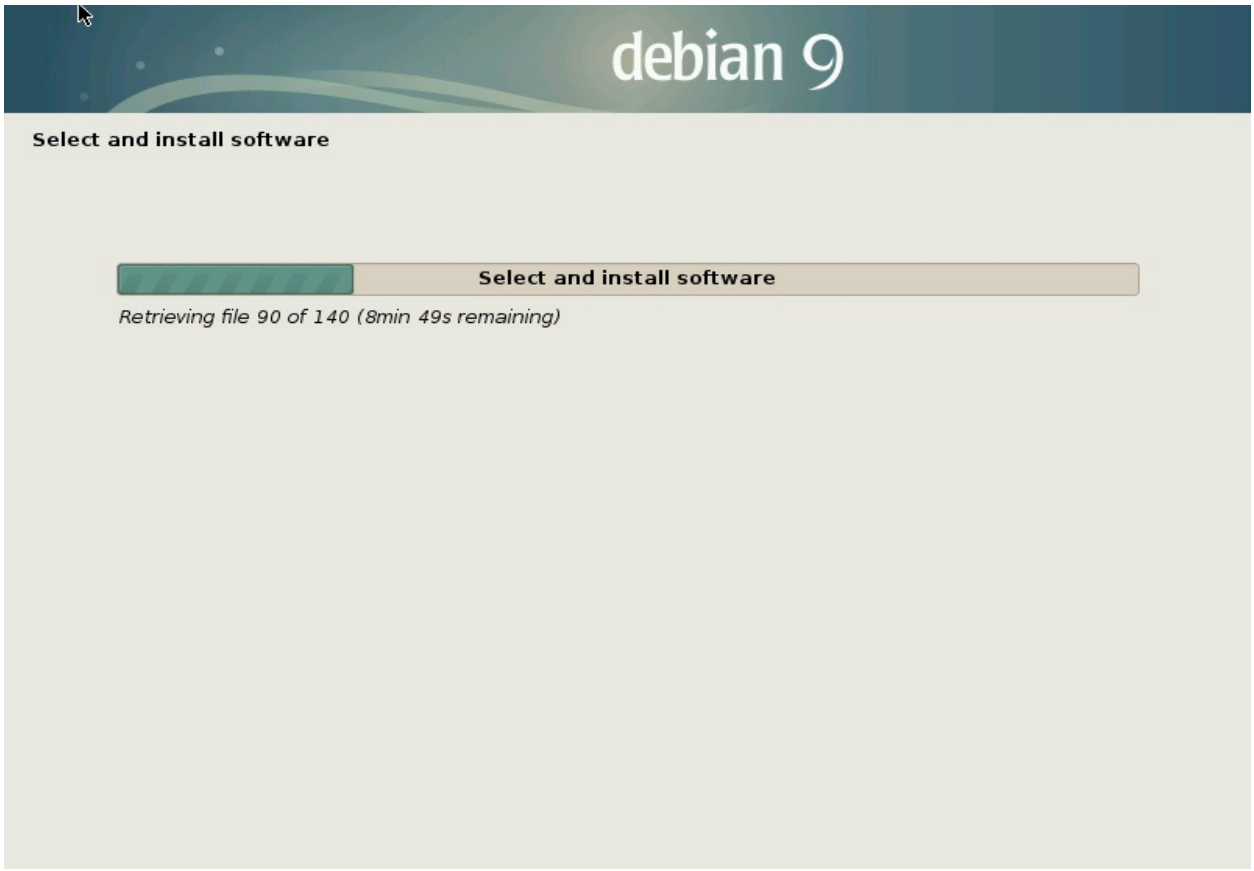


Software Selection

Debian offers carefully arranged selections of packages, so-called tasks. The idea behind them is to group packages for specific uses in order to simplify the installation. From the list in the dialog window (see image below) you can choose between different desktop environments, as well as a web server, a print server, a SSH server and the standard system utilities. To have a minimal installation just enable the last two entries from the list. We will install the XFCE desktop environment later.



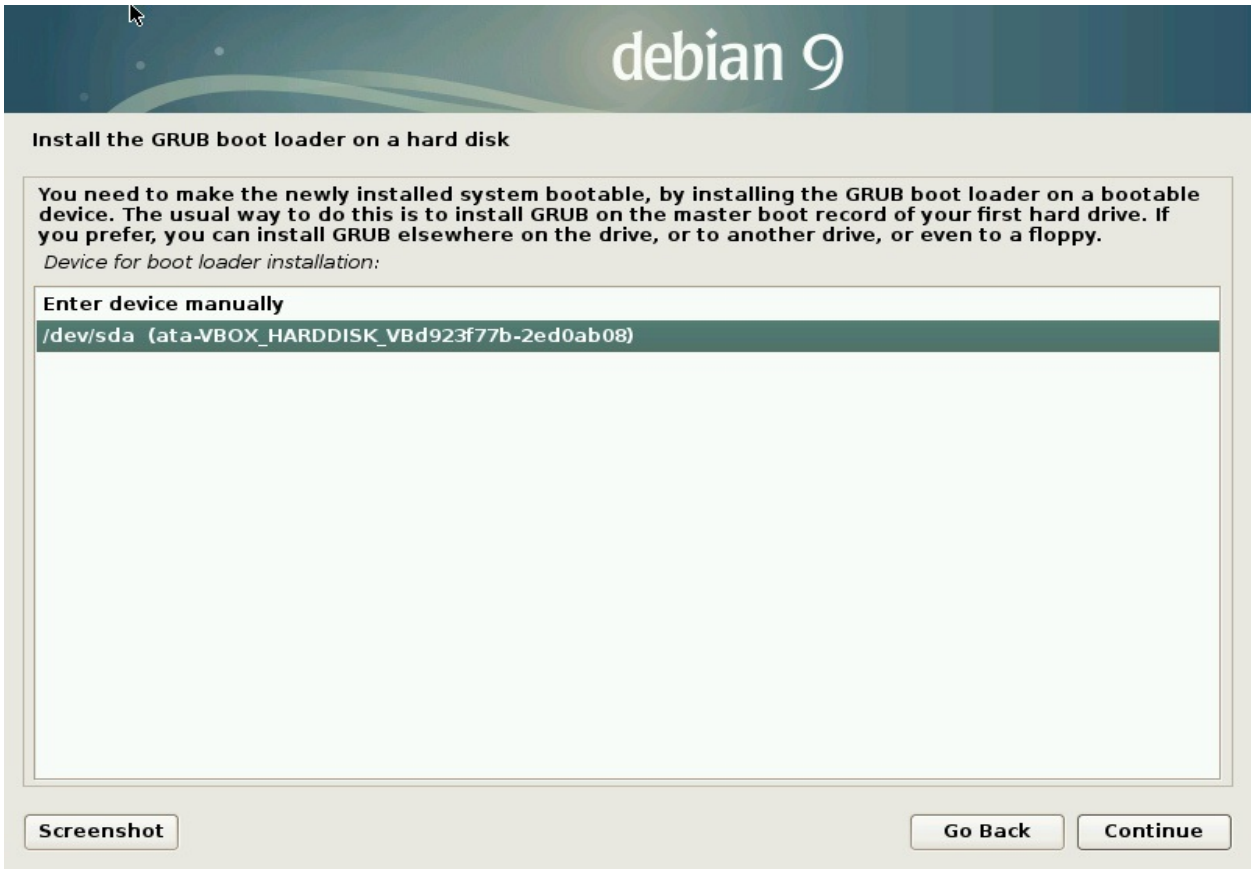
Having confirmed the software selection, the Debian installer retrieves the needed packages from the package mirror, unpacks and then installs them. The bar below shows the progress. In this case 140 packages have to be downloaded.



Setting up GRUB

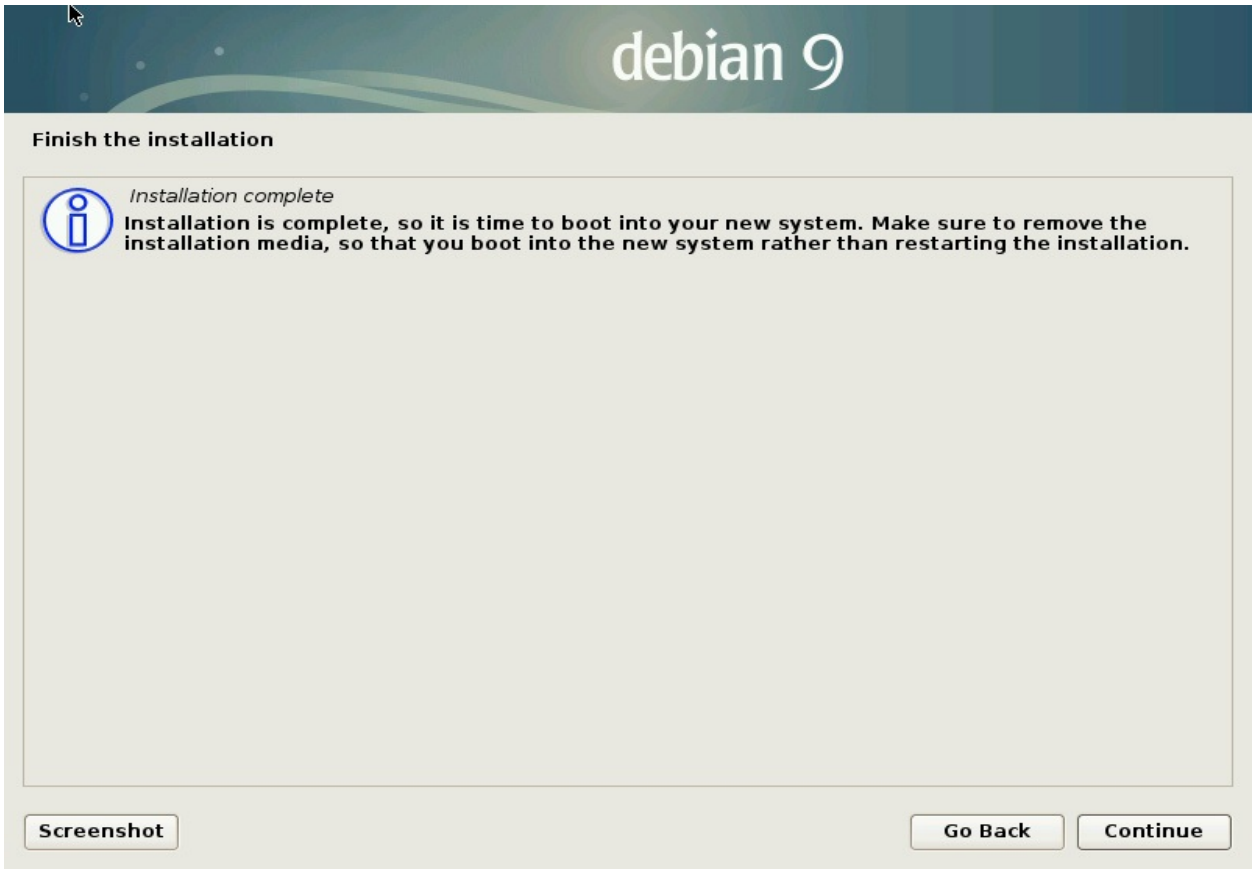
In order to start our newly installed Debian system, we have to set this information too. This process is called booting the system. The software component that handles this step is named Grand Unified Boot Loader (GRUB or GNU GRUB to be precise). The entry *Yes* is already pre-selected, and so we can click Continue to proceed.

The Debian installer needs to know where to install GRUB. The menu in the image below lists the storage media to be considered. In our case we choose the second entry from the list (/dev/sda). Then, click Continue to proceed.



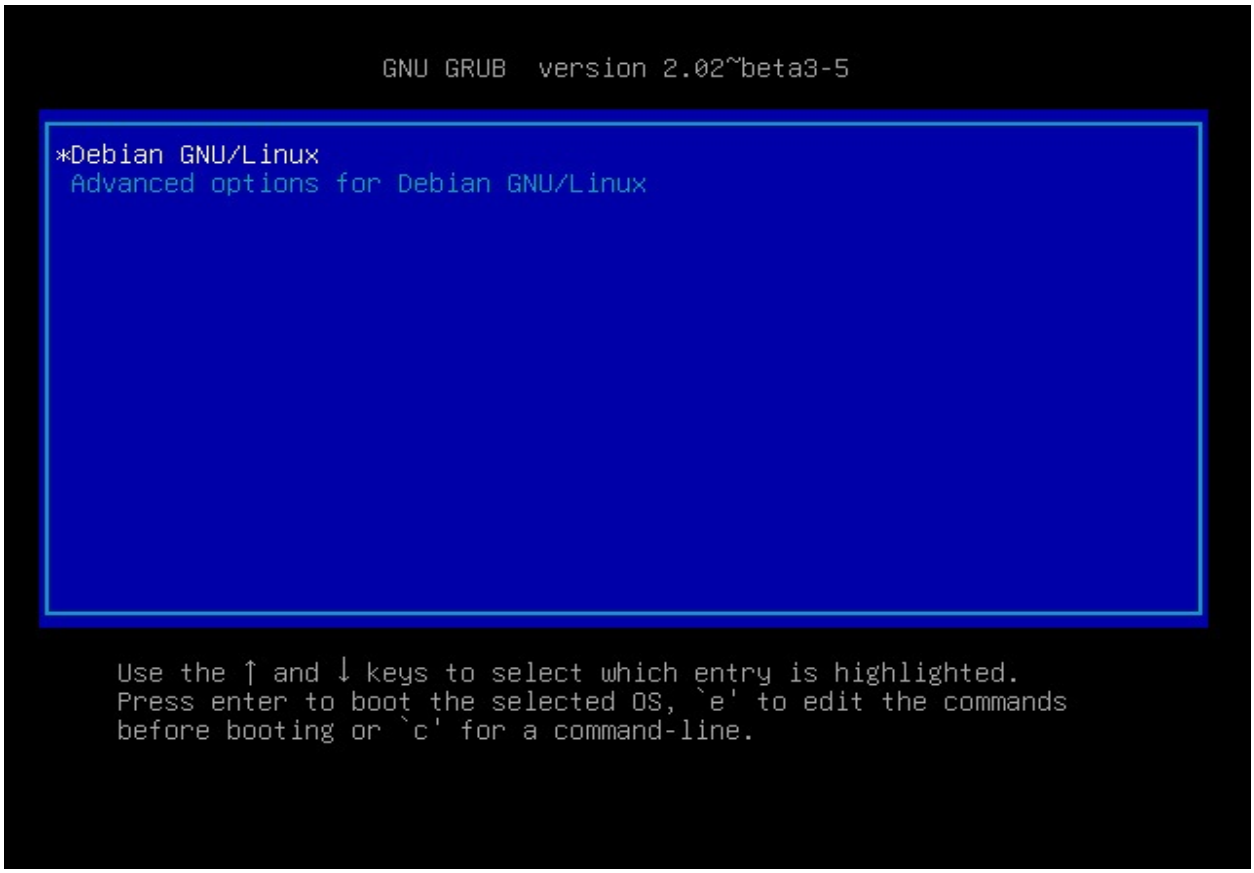
Now that we are nearly finished with the basic installation, the only thing left to do is just a single dialog box to read.

Finishing the Installation The following dialog box informs you that the installation is complete. Then click Continue to reboot the newly installed system.



After a few seconds, the text-based GRUB boot menu will appear on the screen (see image below). The boot menu includes two options: Debian GNU/Linux and Advanced options for Debian GNU/Linux. The first menu item is highlighted and pre-selected. The second menu entry allows you to set specific boot options. For a comprehensive list of these options have a look at the GNU GRUB manual at www.gnu.org/software/grub/manual/grub

Keep the first menu item selected and press Enter to proceed and boot the new system.



Your Debian system will start and initialize the necessary system services. This step will take a few seconds to be completed. Finally, a black-and-white screen will be visible on the first text terminal named tty1 (see image below) and ask you to log into the system.

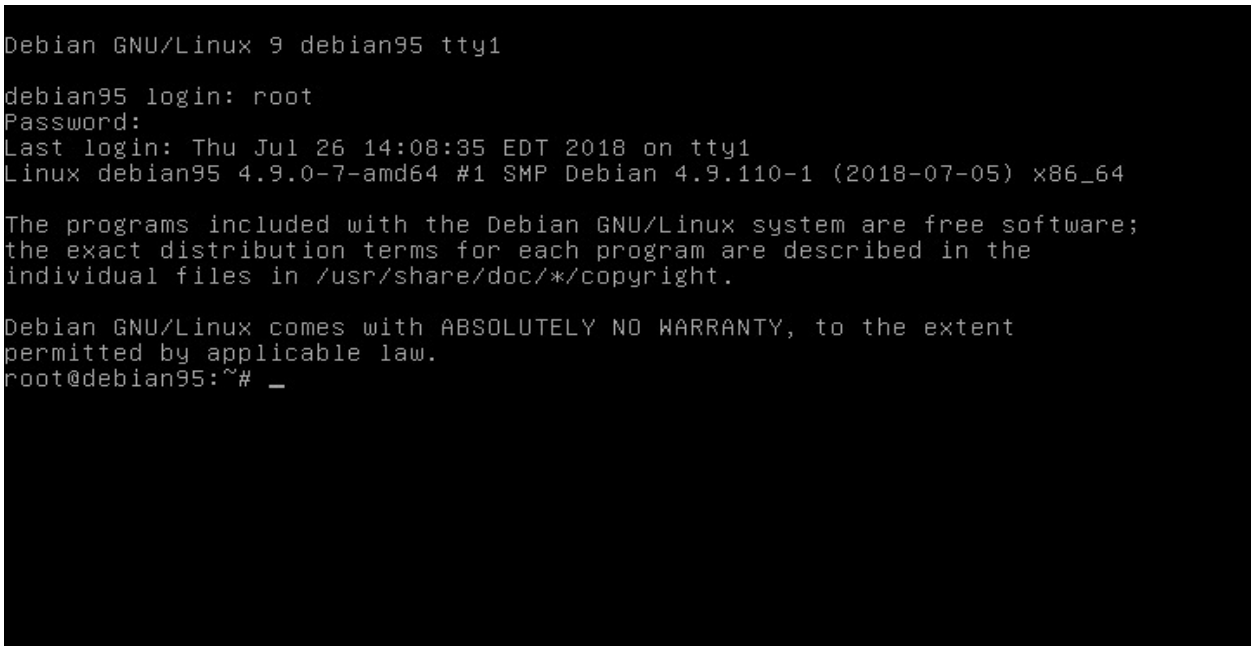
The login prompt consists of two components: the host name of your Linux system (debian95) followed by a space and the word login.

```
Debian GNU/Linux 9 debian95 tty1

debian95 login: _
```

Having logged into the system you will add further software to be able to use a graphical user interface, based on the XFCE desktop. At this step of the installation process, adding further software can only be done by logging in as the administrative root user.

In order to do so, type in *root* at the login prompt, press Enter, wait for the text *Password:* to appear and type in the password set for the root user you defined before. The Linux system will welcome you (see image below) with a login message. The first line of the login message will give you a display on the time of your last login followed by the version of the Linux kernel that is currently running (line two) and the usage advice (lines five to eight).



After the welcome message Debian opens a command-line prompt:

```
root@debian95 ~#
```

This command-line prompt consists of four components:

- root: the name of the user who logged in

- debian95: the hostname of the Linux system
- ~: the current directory. In this case it is the home directory of the user currently logged in. If not otherwise set, this is defined as /home/username for regular users and /root for the administrative user. ~ is just an abbreviation of it.
- #: the login symbol. The # symbol represents the user root whereas $ is used by regular users without administrative rights

Important: You are now logged in as the administrative root user. Take care which commands you type in, and check twice if in doubt. Any mistakes can lead to the necessity to install or repair your Linux system.

4.3 Adding a Graphical User Interface

At its current stage, the Debian system is fully active and can be used in production. For a desktop system suitable for a regular user, it still lacks a nice and easy-to-use graphical user interface. In this step we will change that and install the XFCE desktop manager.

In order to do so, we will install the following packages:

- The aptitude package manager
- The xdm display manager
- The xfce4 desktop environment

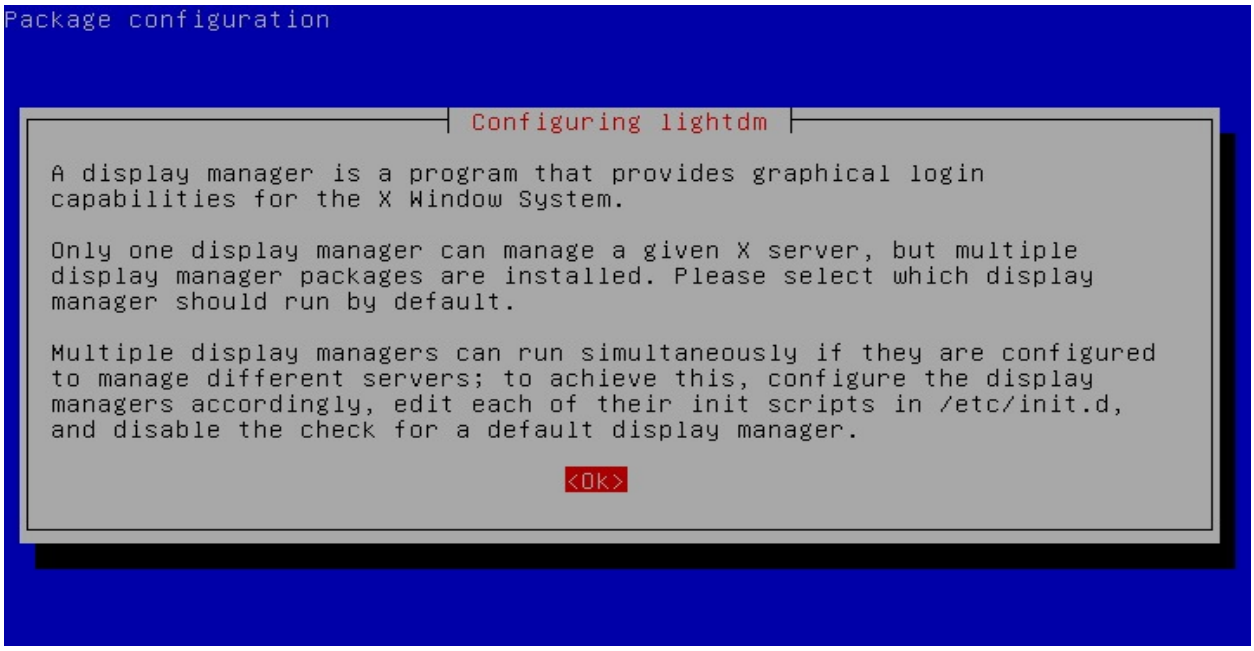
Debian uses the package manager *apt* to handle the installation, the update and the removal of software packages and the related package lists. Also, apt resolves all the package dependencies and ensures that the relevant software is available on the system. In our case a total of roughly 450 MB of data/software have to be retrieved from the package mirror and installed. As pointed out earlier, this requires a working internet connection to download the needed software packages.

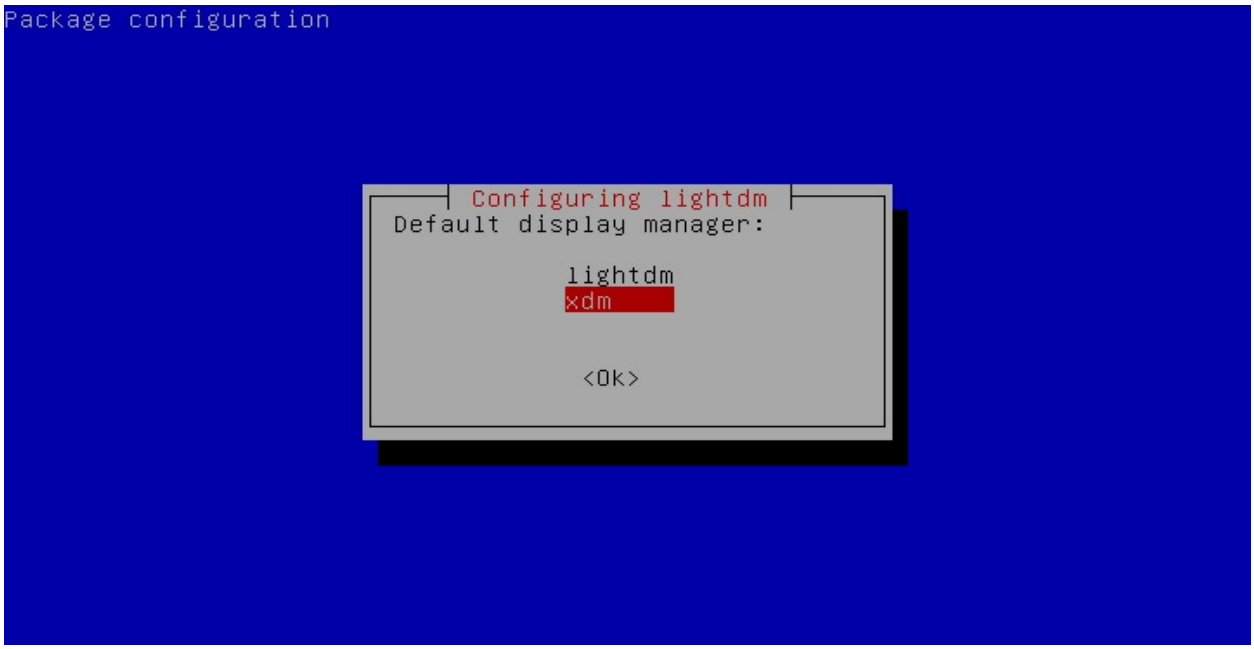
In order to install the three packages, type in the following commands at the command-line prompt. Just type the command after the # symbol:

```
root@debian95 ~# apt-get install aptitude xdm xfce4
```

apt will display further information regarding the packages to be installed. This includes the list of depending packages and recommended packages. At the end

you will see a command-line prompt. Type Y or press Enter to install aptitude, xdm and xfce4 as well as the depending packages. For the xdm display manager, two dialog boxes have to be confirmed (see images below). Press Enter two times to confirm the use of xdm.





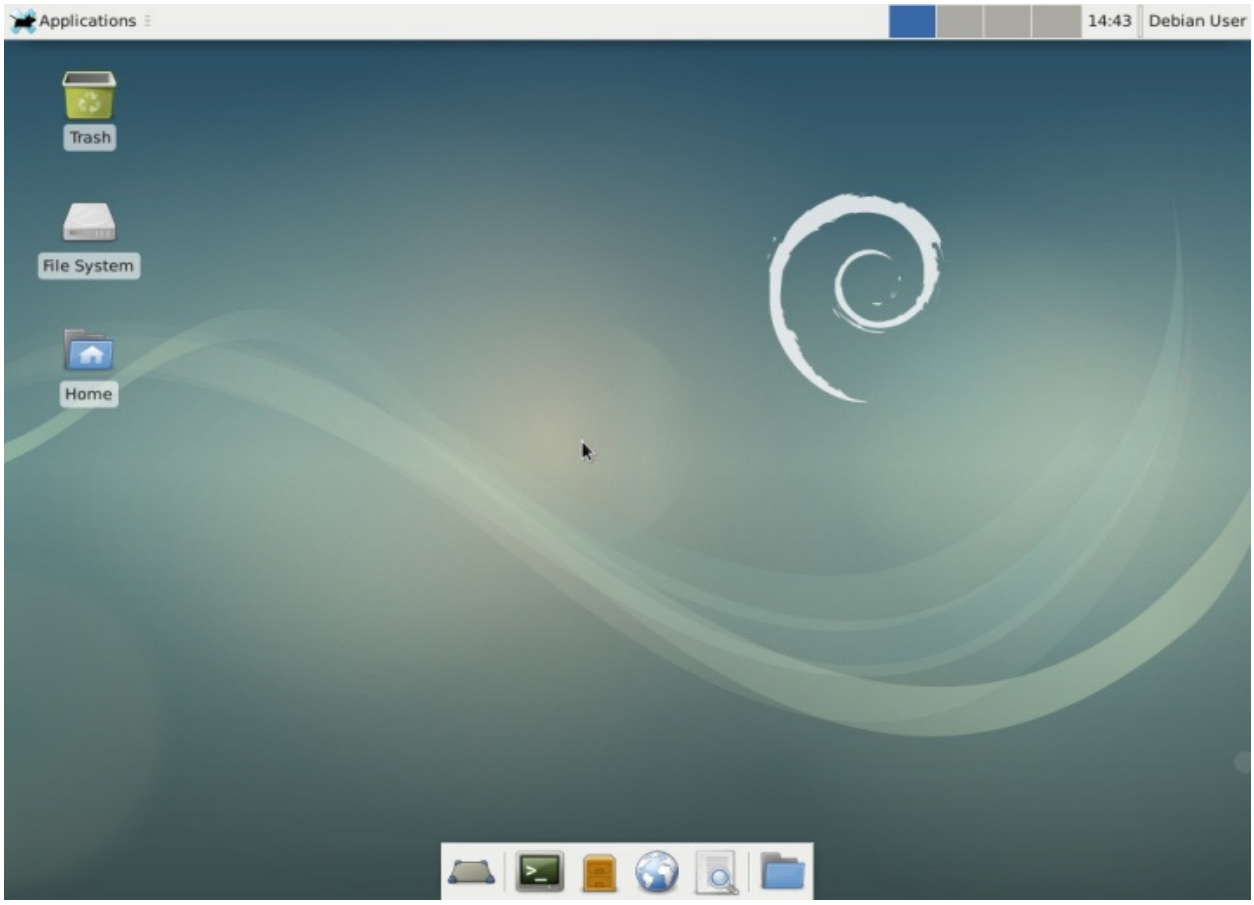
The retrieval of the software packages to be installed takes a while. When finished, the command-line prompt will appear again. In order to activate the changes regarding the graphical desktop environment, restart the system. Type the command *reboot* at the command-line prompt as follows:

```
root@debian95 ~# reboot
```

Having restarted the Linux system, Debian welcomes us with a graphical boot screen (see image below). Press Enter to start the Linux system with the default settings.

A few seconds later a graphical login screen will be visible (see image below). Log in to the system with the regular user named *user* as created earlier. Type in *user*, press Enter and type in the password for the user. Then, press Enter again to confirm and log in.

Next, the XFCE desktop will be visible (see image below). The desktop comes with a number of default elements: an upper navigation bar, a lower navigation bar and desktop icons. These elements are explained in more detail below.



- Upper navigation bar: this shows buttons to access the different applications, the four virtual desktop screens, the clock and a button for various user actions such as to lock the screen, change the user, change to stand-by mode and exit the current session
- Lower navigation bar: this bar contains several buttons to hide all the opened windows and show the empty desktop, to open a terminal, the file manager, a web browser, to find an application and to open the file manager directly with your home directory.
- Desktop icons: from top to bottom there is a trash bin, file manager icons that link to the computer, and the home directory

A right-click on the desktop opens a context menu that allows you to access the various applications (see image below).

4.4 Adding Additional Software

Up until now the software available to be used on your Linux system has been rather limited. The next thing to do is to add the following four Debian packages to make your life a bit easier:

- firefox-esr: the web browser Mozilla Firefox (Extended Support Release)
- gnome-terminal: a terminal emulation maintained by the GNOME project
- xscreensaver: a basic screensaver for the X11 system
- vlc: the video player Video Lan Client (VLC)

The installation of the three packages will be done using the command-line in a terminal emulator (we will look at terminals in detail later in the guide). Currently, on your system the X11 terminal emulator *xterm* is installed. In order to open xterm, click on the terminal button in the lower navigation bar or select the entry Application > System > Xterm from the context menu.

As step one, open xterm. Next, type in the command *su* next to the command-

line prompt as follows and press Enter:

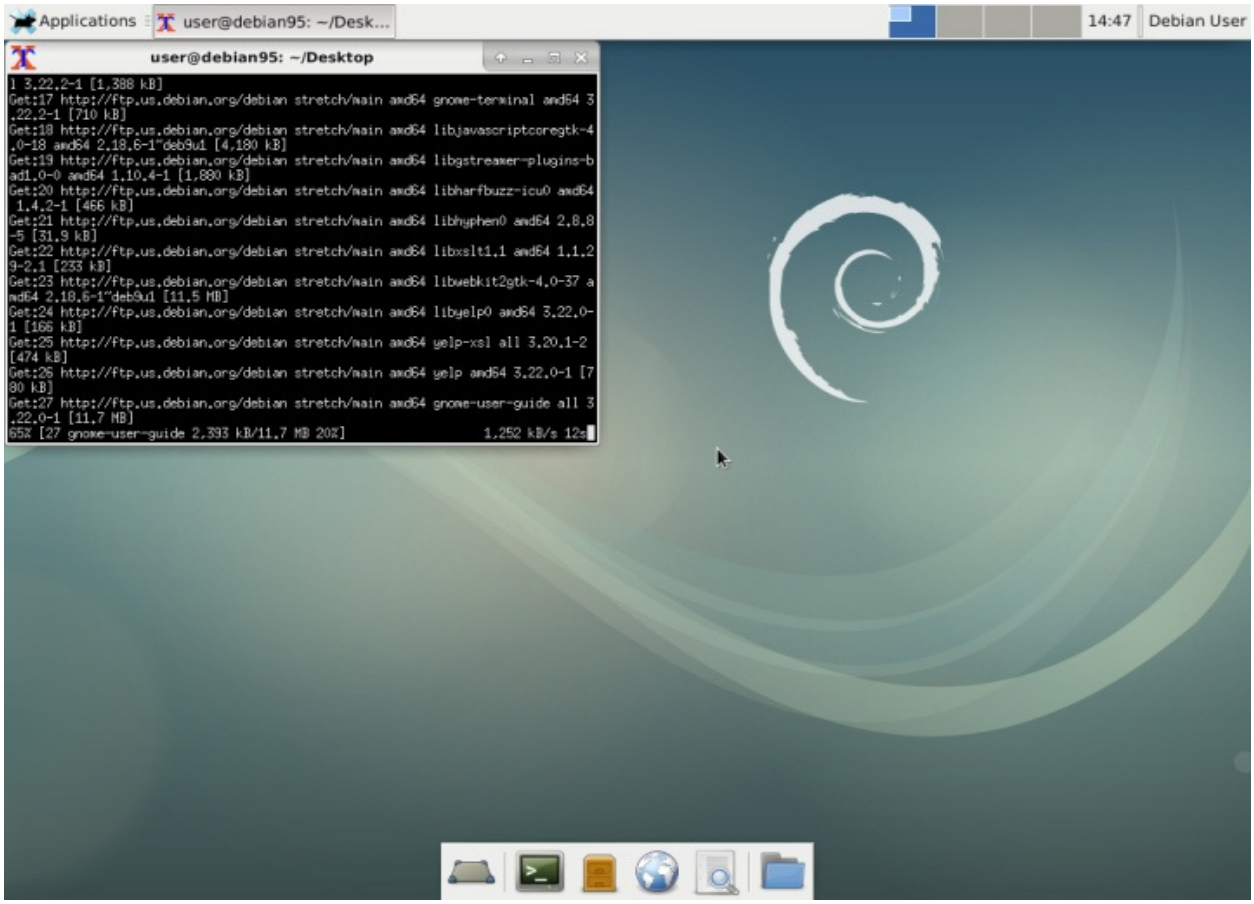
```
user@debian95: ~$ su
Password:
```

You may remember from the previous steps that only an administrative user can install, update or remove software on a Debian system. The *su* command abbreviates *switch user* and changes your current role. Used without an additional name, the role changes to the administrative root user. At the password prompt type in the password for the administrative user and press Enter.

As an administrative user, install the packages: firefox-esr, gnome-terminal, xscreenserver and vlc as follows:

```
root@debian95: ~# apt-get install firefox-esr gnome-terminal xscreensaver vlc
```

The output is seen below:



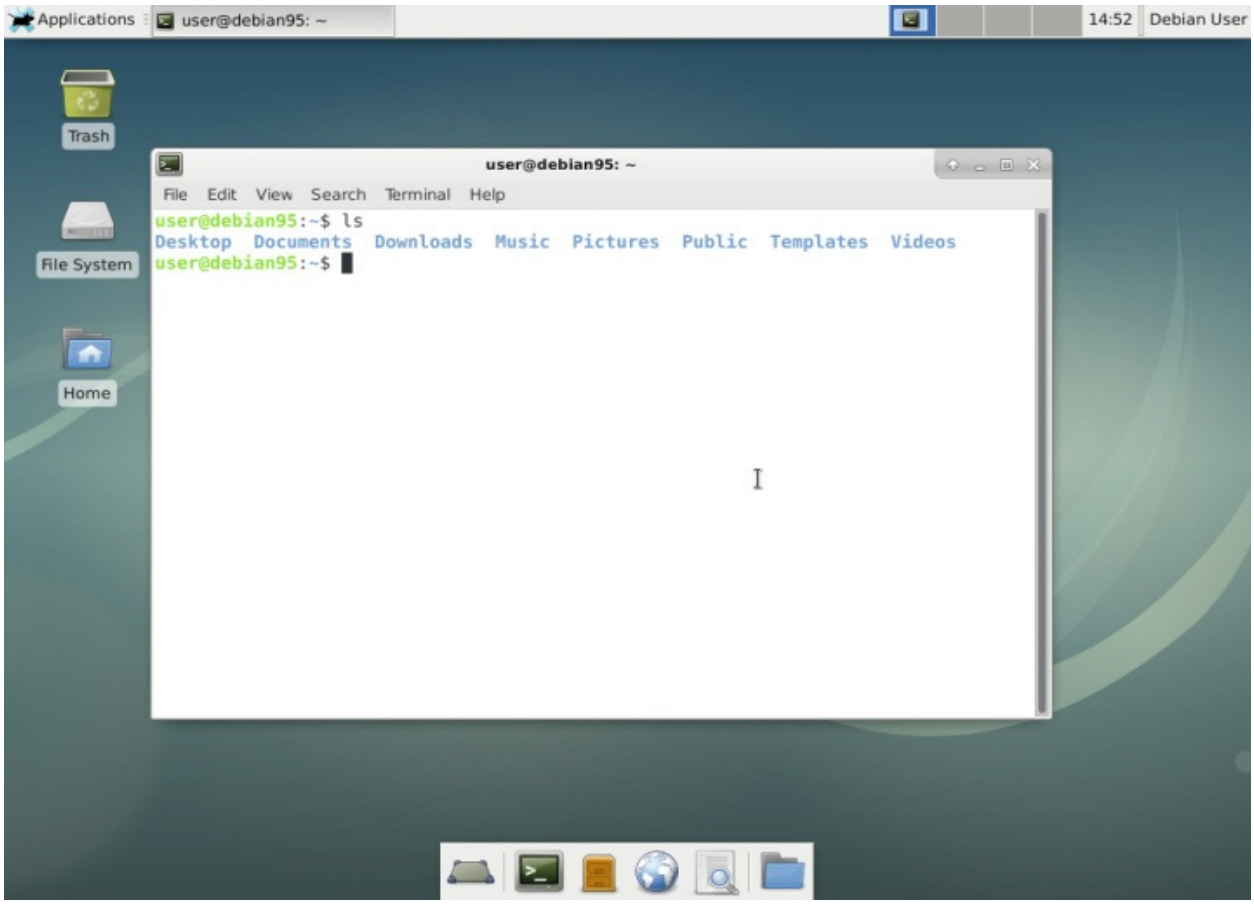
After the installation of the four packages, you can switch back to your role as a

regular user. Press Ctrl+D to quit the admin part, and press Ctrl+D again to close xterm.

The installation of Firefox has the following effects:

- The new software *Firefox* is available from the application menu
- The new command *firefox* is available from the command-line
- The earth icon from the lower navigation bar links to the Firefox web browser
- The installation of the GNOME terminal package has the following effects:
- The new software *gnome-terminal* is available from both the command-line and the application menu
- The terminal icon from the lower navigation bar links to the GNOME terminal
- The entry *Open Terminal Here* from the context menu refers to the GNOME terminal

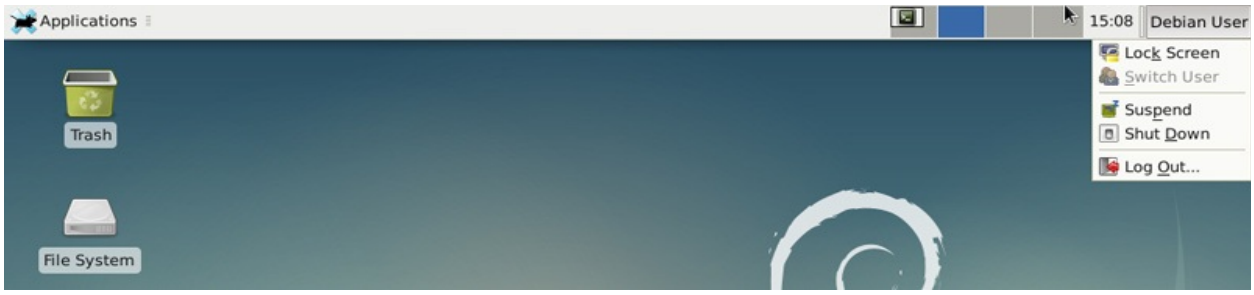
In order to see the changes, select the entry *Open Terminal Here* from the context menu. The image below shows the terminal window. It comes with a white background and a bigger font that is easier to read.



4.5 Exiting Linux

In order to use Linux properly, you also have to learn how to exit Linux and to reboot the system. Below you will learn how to quit the XFCE desktop environment, to shut down the Linux system and to reboot the system. We will look at two methods; the first is based on the graphical interface and the second can also be used on non-graphical systems like servers and wireless routers.

Quitting the XFCE Desktop Environment The easiest way to quit the XFCE desktop and to logout from your current session is to click on the button in the right corner of the upper navigation bar. The button is labeled with your user name, which in our case is Debian User. From the small menu select the last item labelled *Log Out* (see image below). Alternatively, you can choose either the item labeled *Log Out* from the Application button in upper-left corner or from the context menu.



A second window opens that contains the five buttons labeled Log Out, Restart, Shutdown, Suspend and Hibernate (see image below). Click on *Log Out* to quit your session. Subsequently you will return to the login screen.

Shutting Down the Linux System The way to shut down the entire Linux system is similar to exiting XFCE. Click on the button in the right corner of the upper navigation bar. The button is labeled with your user name which in our case is Debian User. From the small menu (see image above) select the item labelled *Shut Down*. From the next dialog window click on the button labelled with Shut Down to stop the Linux system.

From a terminal session you can run the commands *halt* or *shutdown -h now* as an administrative user. A regular user is not allowed to issue these commands.

```
# halt
```

Rebooting the Linux System In order to restart the Linux system, click on the button in the right corner of the upper navigation bar. The button is labeled with

your user name, which in our case is Debian User. From the small menu select the item labelled *Restart* (see image above). From the next dialog window click on the button labelled Restart to reboot the Linux system.

From a terminal session you can run the command *reboot* or *shutdown -r now* as an administrative user. A regular user is not allowed to issue these commands.

```
# reboot
```

Subsequently the system will reboot and you will return to the boot screen.

5. Navigating Linux

File systems on UNIX/Linux systems contain different types of entries. This includes regular files, directories (or folders), symbolic links, block and character devices, named pipes and sockets. This chapter deals with directories and the corresponding Linux tools.

5.1 The Filesystem Hierarchy Standard (FHS)

Modern Linux systems have a rather consistent directory structure, which is based on the Filesystem Hierarchy Standard (FHS). This definition specifies which main directories exist, and which content is stored in them. The FHS is maintained by the Linux Foundation. The latest version of the FHS is 3.0 and was released on 3 June 2015.

The Linux distributions mentioned in Chapter 3 follow the FHS structure. Nevertheless, there are distribution-specific extensions. Below are the common directories that are in use:

Directory	**Description**
/	primary hierarchy root and root directory of the entire file system hierarchy
/bin	essential command binaries for all the users
/boot	the boot loader
/etc	abbreviates etcetera and contains host-specific system-wide configuration files
/home	the home directories of the users
/lib	libraries essential for the binaries in /bin and /sbin
/media	mount points for removable media
/proc	abbreviates processes and is the virtual file system of the Linux kernel
/root	the home directory for the root user
/sbin	essential system binaries
/tmp	temporary data and files

/usr	Unix System Resources, contains user utilities and applications
/var	variable data, such as log files and database files

Some Linux distributions have additional directories like */opt* or use symbolic links. Linux also has some abbreviations in use. They can be quite helpful in your daily work:

Abbreviation	**Description**
$HOME	local variable that contains the user's home directory
~	points to the user's home directory like /home/tom
.	points to the current directory
..	points to the parent directory

You can use these abbreviations on the command line in a terminal and in shell scripts to shorten commands. Next we will look at some of these commands.

5.2 Commands for Directories

As with files, Linux offers a range of commands to list, create, remove, change and rename directories. You can use these commands in your terminal.

Listing Directories

In order to list the directories use the command *ls* with the switch *--directory* (short -d) as follows:

```
$ ls -d */
finances/
projects/
work/
...
$
```

This example lists the names of the directories in the current directory. The line "..." in the output indicates that the list is longer and the output is shortened here.

In order to list all the directories including the hidden ones starting with "." you may type the *find* command as follows:

```
$ find . -maxdepth 1 -type d
./projects
./.mozilla
./.ipython
./.dbus
./video
...
$
```

In order to learn more about the *ls* command and also list other entries, have a look at Chapter 6 which covers terminal commands.

The *tree* command searches the directory recursively and outputs an entire directory structure quite nicely. The switch *-d* limits the output to directories and looks as follows:

```
$ tree -d
.
├── archive
│   ├── ballet-zebola │   │   ├── documents │   │   └── invoices │   ├── charles-darwin-
university ├── projects
├── .dbus
├── .ipython
├── .mozilla
...
$
```

Creating Directories In order to create a directory, use the *mkdir* command (make directory). The next example creates a new entry named "training" in the current directory:

```
$ mkdir training
$
```

The switch *--verbose* (short -v) prints an additional message that the command succeeded.

```
$ mkdir -v training
mkdir: directory "training" created
$
```

The switch *--parents* (short -p) allows you to create entire subdirectory

structures. Parent directories that do not exist yet are created as well. The following example creates the directory “training” as well as its subdirectories “linux” (level 1) and “lpic1” (level 2).

```
$ mkdir -p training/linux/lpic1
$
```

In order to create two subdirectories on the same level, make use of the capabilities the underlying command line interpreter has. The Bash uses values in brackets {...} to achieve this. The next example shows you how to create the two directories “training/linux /lpic1” and “training/linux/lpic2” in one go:

```
$ mkdir -p training/linux/{lpic1,lpic2}
$
```

Removing Directories Removing a directory is done using the *rmdir* command (remove directory). The next example removes the entry named “training” in the current directory:

```
$ rmdir training
$
```

As with *mkdir,* the switch *--verbose* (short -v) prints an additional message that the command succeeded.

```
$ rmdir -v training
mkdir: directory is removed, "training"
$
```

This works well, as long as the directory is empty. In case your directory contains any entries like files or subdirectories, the *rmdir* command will complain when using this message, and keep the directory unchanged:

```
$ rmdir training
rmdir: failed to remove `training': Directory not empty
$
```

To solve this situation the *rm* command (remove) combined with the switch *--recursive* (short -r) becomes quite handy. Then, *rm* removes all the entries recursively. The following example combines the two switches *-r* and *-v* in order to delete one file and three directories.

```
$ tree training/
training/
├── lpic1
│   └── introduction.txt └── lpic2

2 directories, 1 file
$ rm -rv training/
„training/lpic1/introduction.txt“ was removed
Directory was removed: „training/lpic1“
Directory was removed: „training/lpic2“
Directory was removed: „training/“
$
```

Changing Directories The *cd* (change directory) command allows you to move through the system. To move one level upwards towards the root directory use this command:

```
$ cd ..
$
```

To move downwards into a subdirectory enter the *cd* command followed by the directory name. The command *cd /home/user/test* will take you straight into the /home/user/test directory.

As already explained earlier, there are some shortcuts available. The commands *cd ~ (tilde)* and *cd $HOME* will always take you to your home directory whereas *cd /* will take you to the / directory.

Tip: Because Linux is case-sensitive, *cd* is not the same as *CD*. You need to be careful when using upper and lower case.

Renaming Directories This can be achieved using the *mv* command (move). In order to change the name of a directory from “lpic1” to “lpic-1” type in the following:

```
$ mv lpic1 lpic-1
$
```

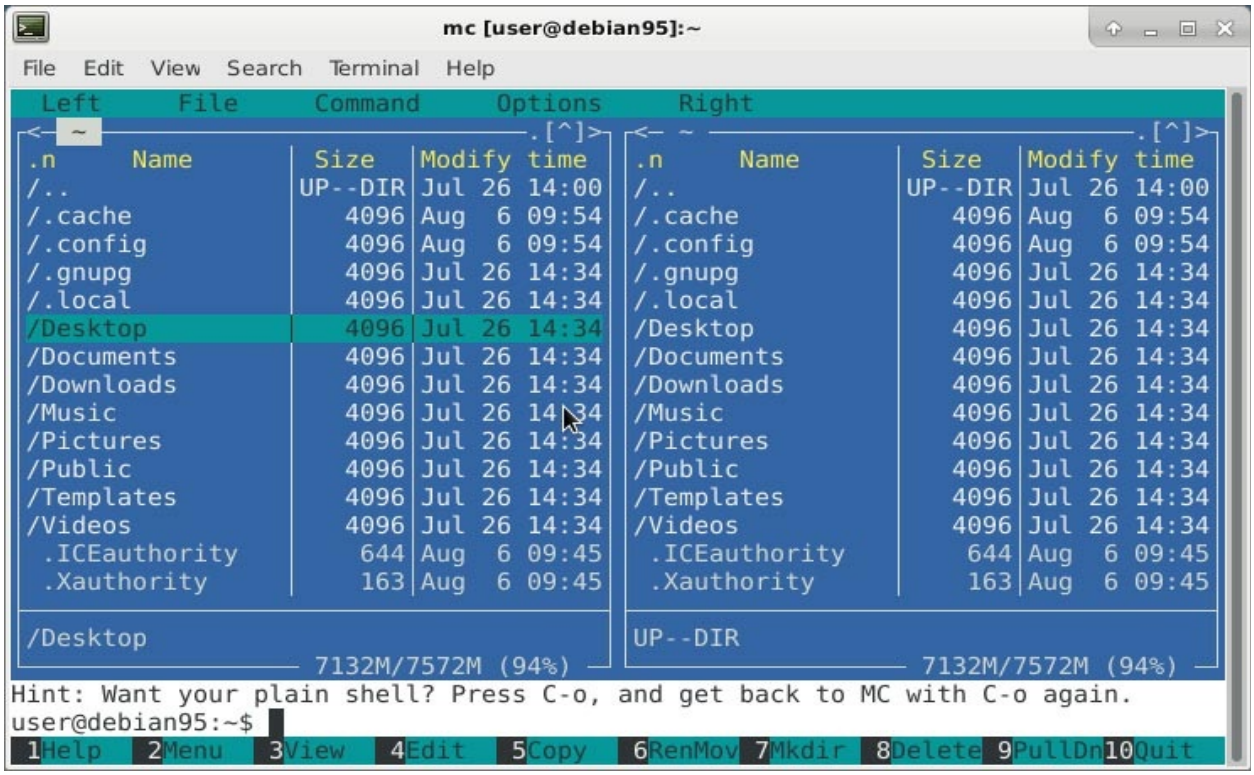
5.3 Terminal-based File Managers

Navigating Linux on the command line is not the most ideal situation. Terminal-based and graphical file managers are a great alternative. Terminal-based file managers run inside a terminal and do not open a separate window on your

desktop. There are tons of programs available, that's why we only present you with a selection of our favorites.

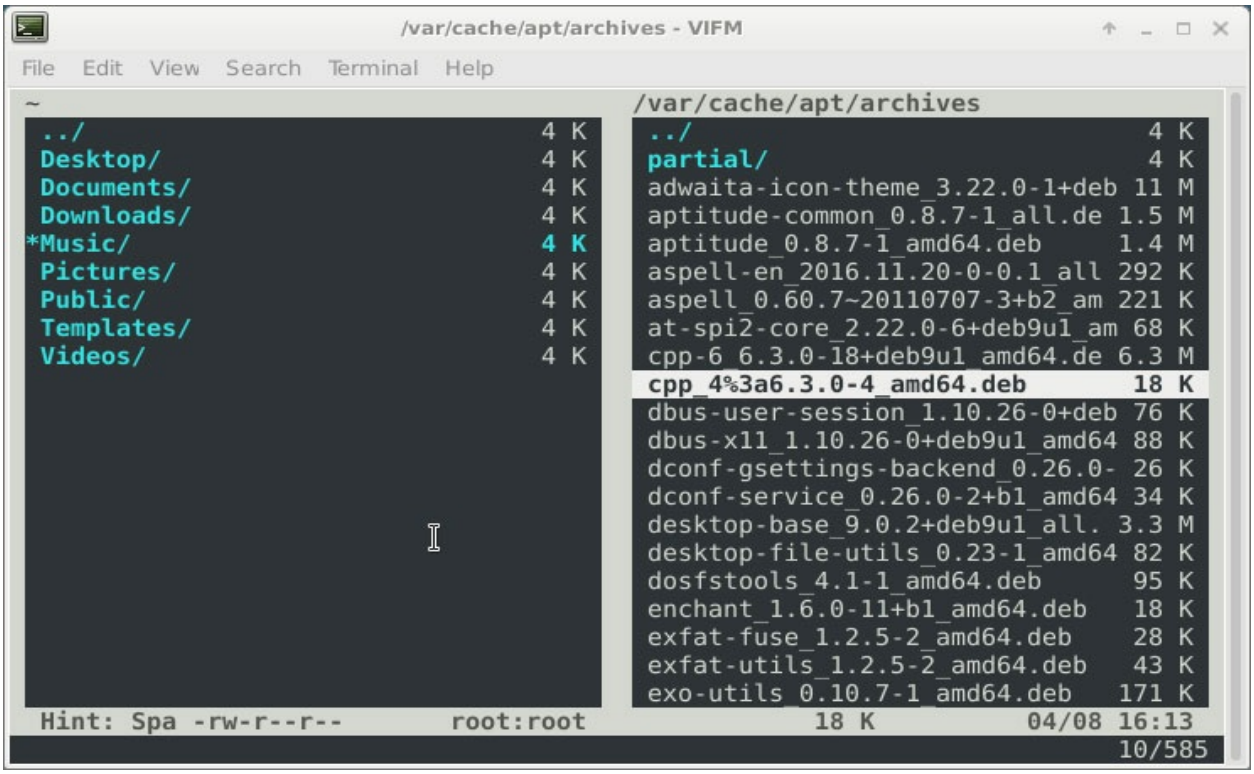
GNU Midnight Commander (mc) www.midnight-commander.org

This is a free software, full-screen, text mode, visual file manager that lets you search, copy, move and also delete both single and multiple files, and even a whole directory tree. The mc integrates a viewer and editor.



The Vi File Manager (vifm) https://vifm.info/

Similar to mc, the Vi File Manager (vifm) comes with two navigation windows based on a curses interface. It provides a Vi(m)-like environment for managing files and directories. If you are familiar with the Vi key bindings, this is the file manager for you.



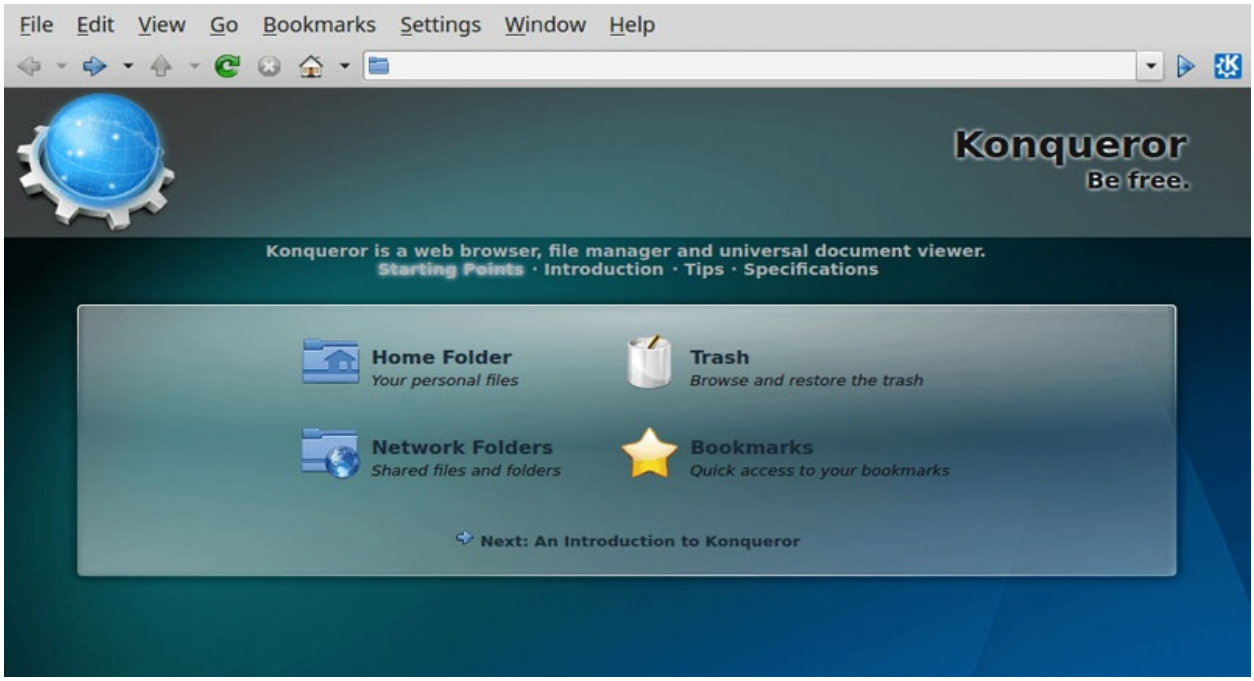
5.4 Graphical File Managers

Similar to terminal-based file managers, graphical file manager assist in navigating Linux. These file managers do open in separate windows, but provide a much nicer user experience. Again, there are tons of programs available, so we only present you with a selection of our favorites.

Konqueror

https://packages.debian.org/stretch/konqueror

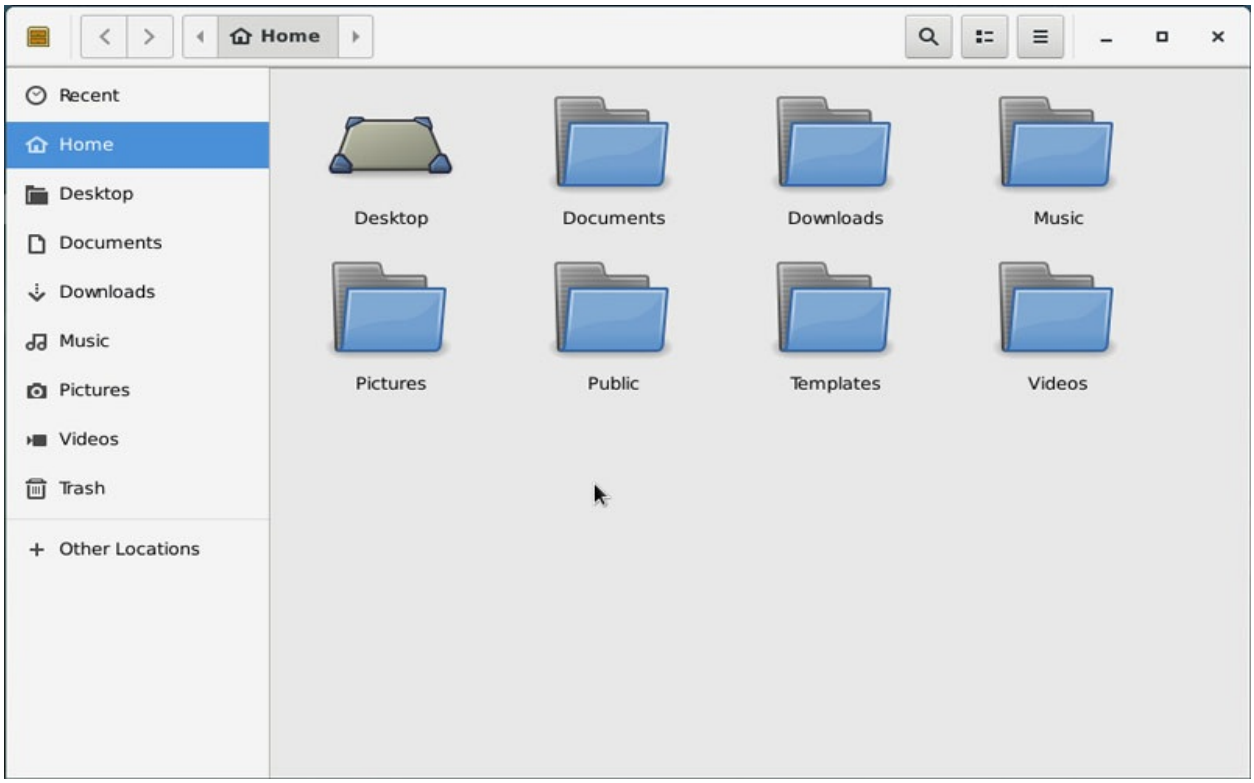
Konqueror is a powerful file manager for the KDE desktop environment. Konqueror offers simple file management functionalities such as copying, moving, searching and deleting files and directories plus some advanced features and functionalities, such as access to archives, browse and rip audio CDs as well as support for access to FTP and SFTP servers and SAMBA (Windows) shares.



Nautilus

https://wiki.gnome.org/Apps/Nautilus

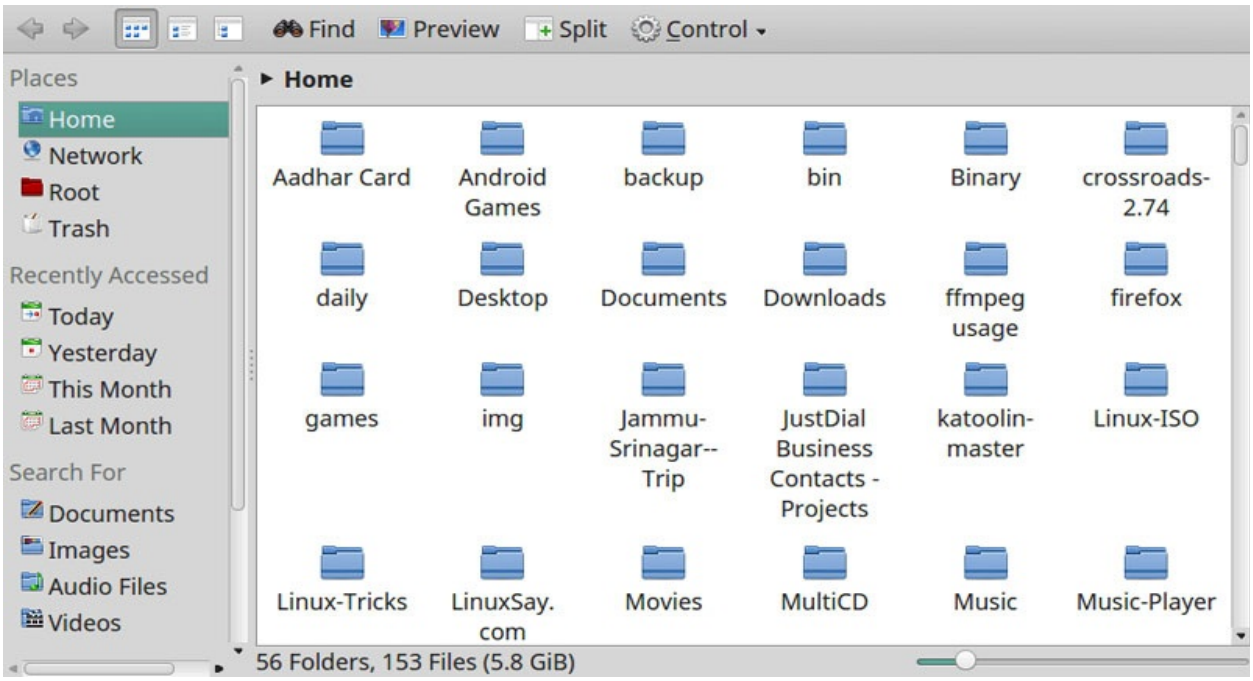
Nautilus is the file manager on the GNOME desktop. It offers easy navigation and management of files on a Linux system. This includes simple file management functionalities such as copying, moving, searching and deleting files and directories, as well as easy access to local and remote files.



Dolphin

www.kde.org/applications/system/dolphin/

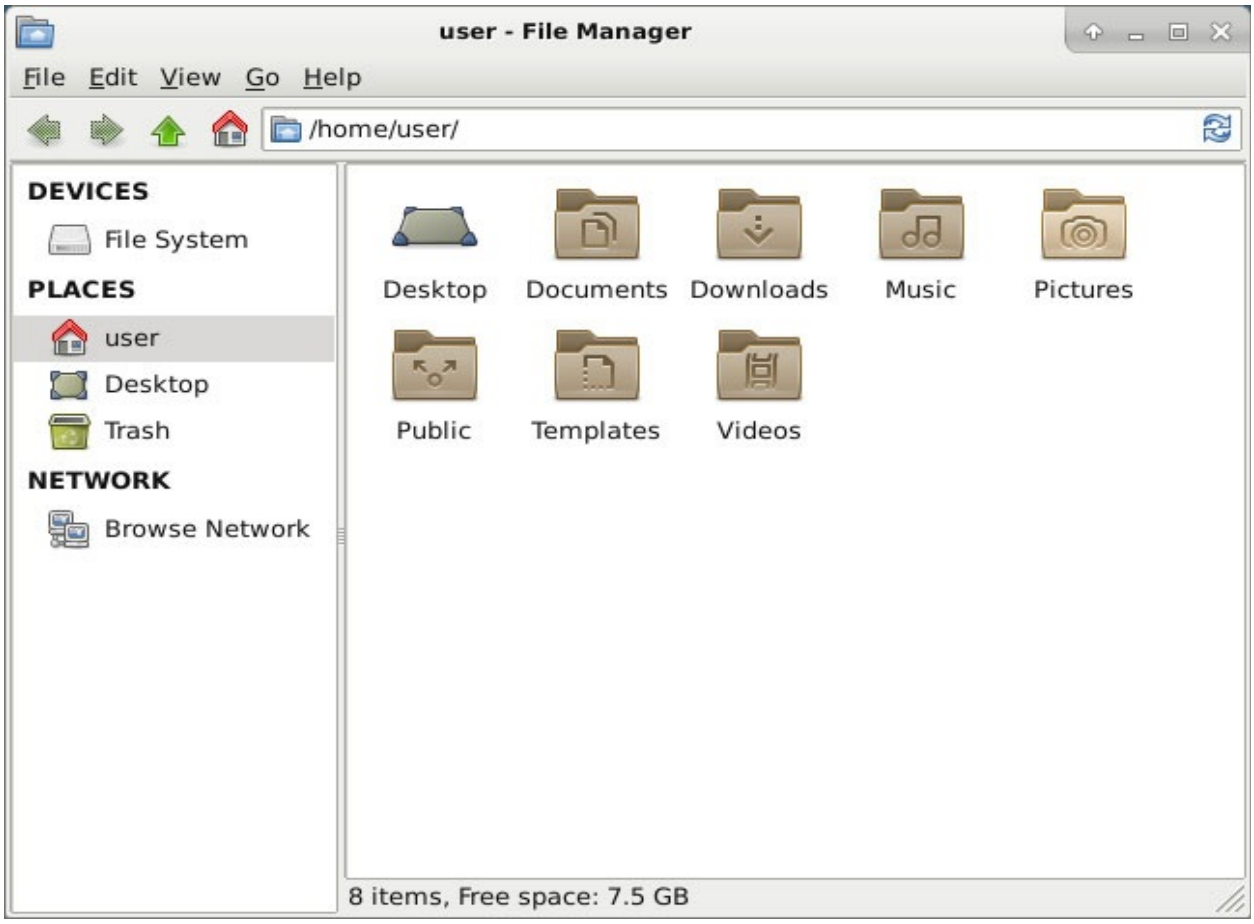
Dolphin is a lightweight file manager developed as part of the KDE applications package. Designed for simplicity, flexibility and full customization, it allows users to browse, locate, open, copy and move files around a Linux system with a lot of ease.



Thunar

https://docs.xfce.org/xfce/thunar/start

Thunar is a modern, lightweight file manager for the XFCE desktop. It is designed to be fast, responsive and easy to use. We use it here because it has a clean and intuitive interface with few and important user options available.


user - File Manager
File Edit View Go Help
/home/user/
DEVICES
File System
PLACES
user
Desktop
Trash
NETWORK
Browse Network
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
8 items, Free space: 7.5 GB

6. Introduction to Linux Terminals

Working with the Linux operating system requires you to have knowledge about the terminal and the command line. It is essential to know what these things are, to be at least slightly familiar with their usage and the standard commands available. A few of these commands have already been introduced in Chapter 5.

6.1 What is a Terminal?

A terminal is described as "a program that emulates a video terminal within some other display architecture. Though typically synonymous with a *shell* or text terminal, the term *terminal* covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window".

To be precise, a terminal is simply the outside. Inside a terminal runs a command line interpreter that is called a shell.

Debian Linux supports a long list of terminal software. This includes the Aterm, as well as the GNOME terminal, the Kterm, the Mate Terminal, the Rxvt, the Xterm and the Xvt. These different implementations of terminal software vary in terms of stability, support for character sets, design, colors and fonts, as well as the possibility to apply background images or work with transparency In this book we will use the GNOME terminal because of its stability, simplicity and adjustability. In order to increase and decrease the size of the content that is displayed inside the terminal window; use the two-key combinations CTRL+ and CTRL-.

6.2 What is a Shell?

Simply speaking, a shell is a sophisticated command-line interpreter. In a loop the shell reads characters, modifies them under certain conditions, and executes the result.

Under certain conditions, between reading from the command-line and the execution of the actual result, the shell has to interpret special characters that are part of your input. The table below displays the special characters that are available:

Character	Meaning
$name	substitution of a variable
name=	assignment of a variable
'command'	substitution of a command
$(command)	substitution of a command
< > >> 2> 2>>	redirection of input and output
|	pipelining of commands
" ' \	quoting
* [ ] ?	creation of file names (globbing)
;	separation of commands
&	run the command in the background
|| &&	run the commands under certain conditions
{ } ( )	cramp commands

For interest sake, the shell divides the command-line into single words in order to determine the commands, by means of the following process:

Step	Description	Special Characters
1	read until the command separator and tagging (substitution of commands)	\n & | || && ; $(...) \ "..." '..."
2	tagging (input-/output redirection, assignment of values)	< > >> 2> 2>> name=
3	division into single words	space, tabulator
4	substitution of variables	$name
5	substitution of commands	tagging, see step 1 and 2
6	input-/output redirection	tagging, see step 1 and 2
7	assignment of variables	tagging, see step 1 and 2
8	division into words	based on IFS
9	creation of file names	* [-]
10	command execution	

In steps 1 and 2 tagging happens for steps 5 to 7 only, and no further action takes place. Steps 5 to 7 are executed if tagging is completed. Next, the shell removes all the special characters. In step 10 the command-line contains only the arguments that are needed to execute the command. The shell interprets the first word as the name of the command to be executed, and the remaining words are its arguments:

```
command arg1 arg2 arg3 ...
```

6.3 Available Shells

Your Linux system allows the usage of various shells. Each shell is available as a separate software package through the Debian package manager, Aptitude, we installed earlier. The list of shells is quite long, so we list only a selection of the available shells that are the most popular:

- Almquist Shell (ash)
- Bourne Again Shell (bash)
- Debian Almquist Shell (dash)
- Z Shell (zsh)
- C Shell (csh)
- Korn Shell (ksh)
- Tenex C Shell (tcsh)

Unless otherwise stated the examples in this document are based on the Bourne Again Shell (bash). At the time of writing this document this is the default shell on Debian.

Which other shells are allowed in your Linux system, is set up in the configuration file */etc/shells*. Using the *cat* command you can see the list of allowed shells:

```
user@debian95:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
```

```
/usr/bin/screen
user@debian95:~$
```

The name of the shell that is currently in use in your terminal is kept in the shell variable *$0*. The following simple command outputs the shell's name:

```
user@debian95:~$ echo $0
bash
user@debian95:~$
```

In order to change the shell currently in use, have a look at the *chsh* command and the configuration file */etc/passwd* covered in the following section.

7. Essential Linux Commands

In this chapter we explain basic Linux commands that will help you to deal with files and directories, text processing commands, users and groups, process management, networks, and the help system.

7.1 Files and Directories

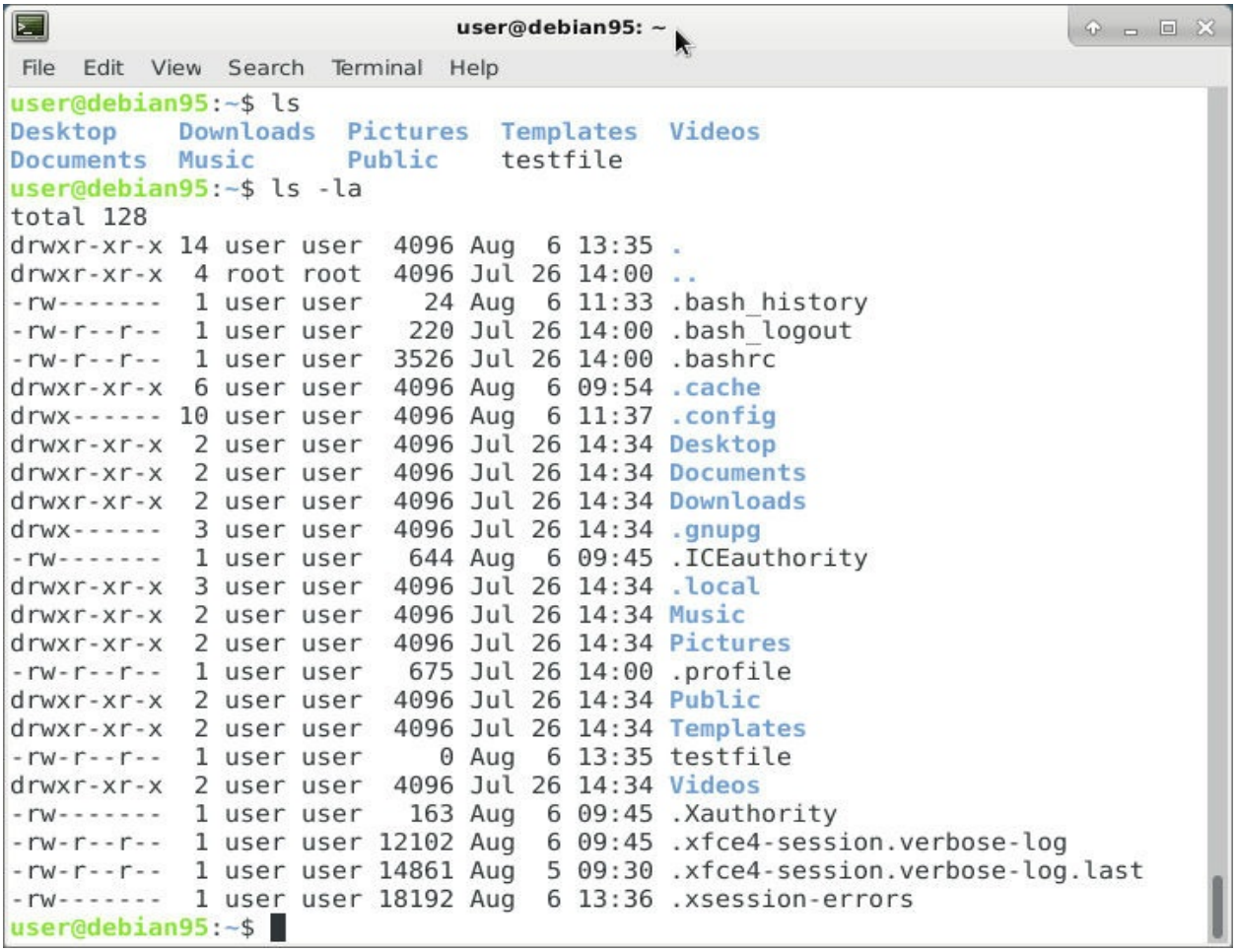
touch

The *touch* command is used to update the access date or modification date of a file and works in two ways: if the file already exists, the timestamp for access and modification of the file is set to the current timestamp. In case the file does not exist yet, an empty file will be created that has the current timestamp (see image below).

Use the touch command in order to set the timestamp of a file and to figure out if you have the appropriate permissions to write to a directory or an entire filesystem.

ls

This command lists the entries of a directory. The image below shows two common ways: without any options, and with the options *-la* (short for -l -a which means long all). The first output displays the entry names only for regular files, whereas the second output lists both regular and hidden entries. Over and above, it shows all the information like type of entry, permissions, name of owner, size of entry, access date and name of entry.



In order to list directories only, use the option *-d* (abbreviates --directory).

mkdir

This command is used in order to create a directory (make directory). The following example creates a new directory named “training” in the current directory:

```
$ mkdir training
$
```

rmdir

This command is used in order to remove an empty directory (remove directory). The following example deletes an empty directory named “training” in the current directory.

```
$ rmdir training
```

```
$
```

rm

This command abbreviates the word “remove” and deletes files and directories. In order to delete all the files ending with *.txt* that reside in the current directory issue the following command:

```
$ rm *.txt
$
```

In order to be on the safe side when deleting files and directories, use the option *-i* (or --interactive) in combination with *-v* (for --verbose). Before deleting a file *rm* will then request your explicit confirmation, and prints a status message:

```
$ rm -iv invoice156.txt
rm: remove regular file "invoice156.txt"? y
"invoice156.txt" was deleted
$
```

cp

The *cp* command copies files. In order to operate properly, it requires two names: the name of the original file and the name of the copy. The next example creates a copy of the calendar file that is named “calendar-2018”.

```
$ cp calendar calendar-2018
$
```

The original file is not touched and stays intact. The copy has the same content as the original, but with the current timestamp. Use the option *-i* (or --interactive) to prevent overwriting existing files.

mv

The *mv* command abbreviates the word “move” and moves and renames files and directories. It requires two names: the name of the original file and its new name. The following example renames the “calendar” file to “calendar-2018”.

```
$ mv calendar calendar-2018
$
```

The original file is removed and the new entry receives the current timestamp. Use the option *-i* (or --interactive) to prevent overwriting existing files.

cd

The *cd* (change directory) command allows you to move through the system. To move into the subdirectory “work”, use this command:

```
$ cd work
$
```

file

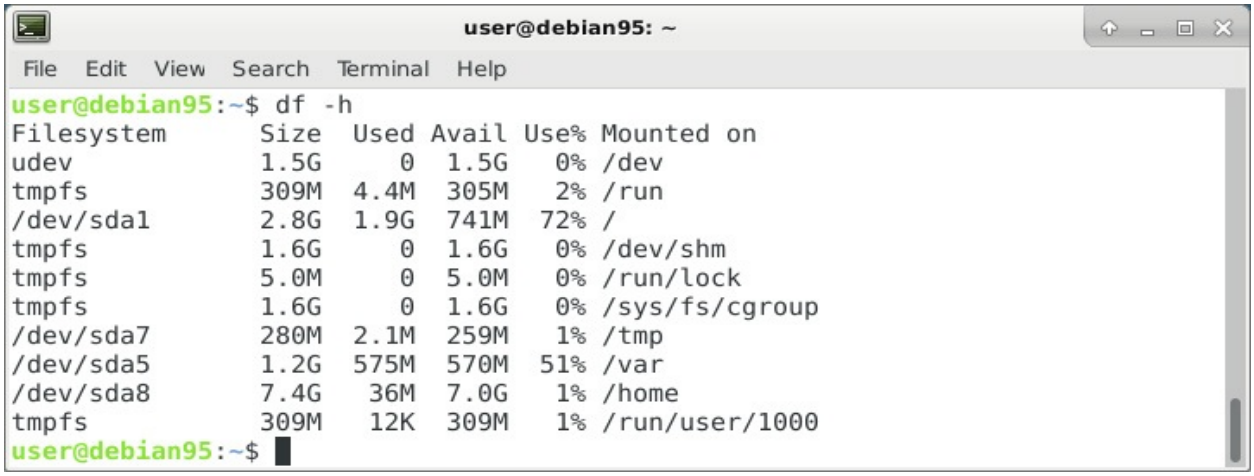
The *file* command determines the type of entry in the file system. There are three sets of tests that are performed in the following order: filesystem tests, magic tests, and language tests. The first test that succeeds causes the file type to be printed. As shown in the image below “Music” is classified as a directory, “testfile” as an empty file and “/etc/passwd” as a text file. This command also detects PDF files as well as various image formats.

du and df

These two very similar commands tell you more about the disk space that is in use, *du* abbreviates “disk usage” and *df* means “disk free”.

du calculates the amount of disk space that is used by a directory. The regular output states the value for every single entry and can be a bit confusing. In order to get a summary for a directory, extend the command line call by the three parameters *-s*, *-c* and *-h*. *-s* abbreviates “summary”, *-c* means “total” and *-h* outputs the value in human-readable format. The image below shows the disk usage of your home directory.

In contrast, the *df* command shows how much space is left on the devices. The image below shows the available disk space of your system. From left to right the columns cover the filesystem, the disk size, the amount of space that is used, the amount of space that is still available and the device that is mounted to that directory.



7.2 Output and Text Processing

echo

echo is a built-in shell command and is intended to output text:

```
$ echo help
help
$
```

As already shown earlier, the shell evaluates the command and prints the value of variables too. The next example prints the value of the shell variable *$HOME* which represents your home directory.

```
$ echo $HOME
/home/user
$
```

cat and tac

These two commands print a file, line by line. *cat* starts with the first line up to the last line, and *tac* starts with the last line up to the first line. The next example uses a simple plain text file named “places” that contains the names of one city per line. For *cat* Amsterdam comes first, and for *tac* it is Cape Town.

```
$ cat places
Amsterdam
Berlin
Bern
Cape Town
$ tac places
Cape Town
Bern
Berlin
Amsterdam
$
```

grep

This command abbreviates the description “global regular expression print”. The command acts as a filter that only prints the lines of text that match a given pattern. *grep* needs data to work on, in combination with a pattern to look for.

Based on the example used above for *cat* and *tac*, the following command line call only outputs the lines from the file that contain the character string “Ber”. Keep in mind that *grep* filters are case-sensitive, i.e. it looks for the character string that starts with an uppercase “B” followed by the two lowercase letters “e” and “r”. It does not matter whether the pattern is at the beginning, in the middle or at the end of text.

```
$ grep Ber places
Berlin
Bern
$
```

When it comes to patterns, *grep* supports character strings and regular expressions (RegEx). In order to find all the strings that end with the letter “n” use the option *-E* (or --extended-regexp) followed by the pattern “n$” as shown below:

```
$ grep -E "n$" places
Berlin
Bern
Cape Town
```

```
$
```

head and tail

head outputs the first lines from a file. Invoked without further parameters it outputs up to ten lines. In contrast *tail* does the same thing, but starts at the end of a file. In order to output the first two lines, add the option *-n 2* as follows:

```
$ head -n 2 places
Amsterdam
Berlin
$
```

In order to output the last two lines, do the same thing using *tail* as follows. In contrast to *tac* (see earlier in this chapter) *tail* does not change the order of output.

```
$ tail -n 2 places
Bern
Cape Town
$
```

nl

nl is similar to *cat* (see earlier in this chapter) but adds a line number at the beginning of each line of output.

```
$ nl places
    1 Amsterdam
    2 Berlin
    3 Bern
    4 Cape Town
$
```

wc

This command abbreviates the phrase “word count” and counts lines, words and single characters of the input data. Unless otherwise specified, all three values are printed:

```
$ wc places
4 5 32 places
$
```

Amongst others *wc* offers the following options to limit the output:

- -l: output the number of lines only, followed by the filename
- -w: output the number of words only, followed by the filename
- -c: output the number of characters only, followed by the filename

This example shows how to count only the lines in a file:

```
$ wc -l places
4 places
$
```

7.3 Users and Groups

These commands deal with a variety of actions in order to manage the users and groups of your Linux system. Unless explicitly stated, these commands can be run as a regular user.

whoami

This command returns your current user ID as follows:

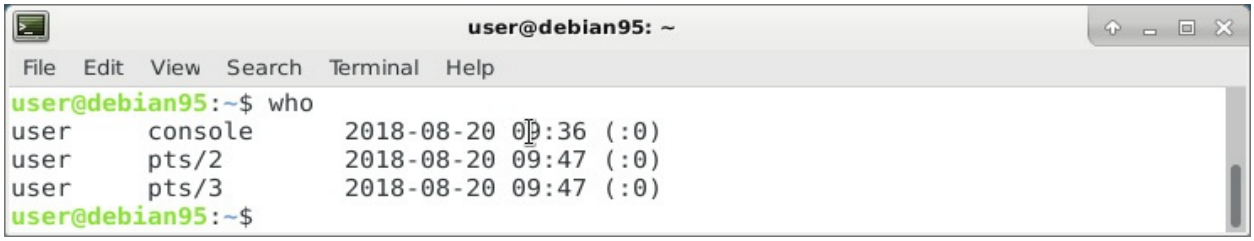
```
$ whoami
user
$
```

users, who and w

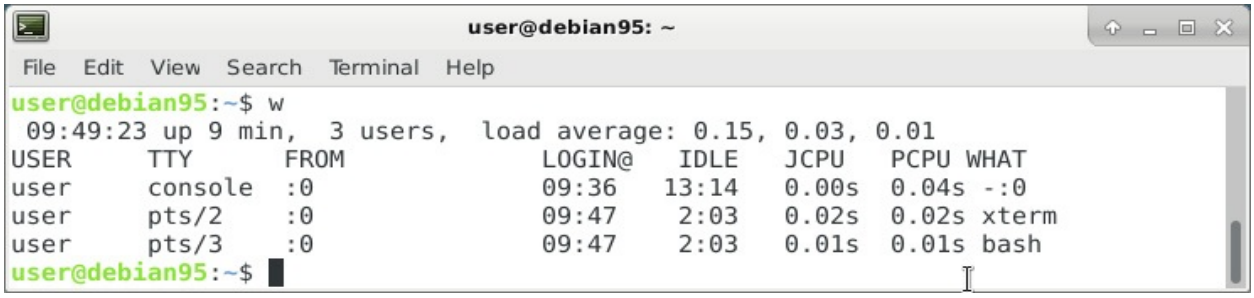
The *users*, *who* and the *w* commands show the users that are currently logged into your Linux system. *w* extends the output of *who* by the uptime information and another column that contains the last command that was executed. In contrast, *users* simply outputs the name of the users as a space-separated list in a single line (see image below).

The single columns for *who* start with the login name of the user. The output is followed by the name of the terminal, where “console” represents a login terminal, and “pts/1” abbreviates the first pseudo terminal session. The last two

columns contain the login time and the host the user comes from, in brackets (see image below).



The single columns for *w* contain the login name of the user (titled LOGIN), the name of the terminal (titled TTY), the name of the host the user comes from (titled FROM), the login time (titled LOGIN@), the activity (idle time and CPU usage titled IDLE, JCPU, and PCPU) as well as the last command the user executed (titled WHAT) (see image below).



id and groups

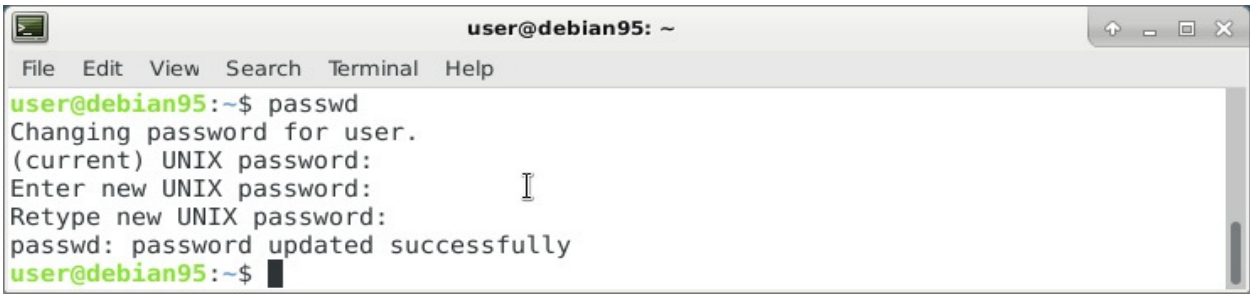
The *id* command outputs the user and group information of the current user (see image below). From left to right the columns show the user ID (uid=1000(user)), the group id (gid=1000(group)) and the name of the groups the user is a member of.

In order to list the names of all the groups the user belongs to, you can also invoke the *groups* command (see image below). The output is a space-separated list of the group names.

passwd

As described in Chapter 4, the Linux system has at least two users: an administrative root user and a regular user that we simply called *user*. Every account is also secured with a password.

In order to change your password, use the *passwd* command from the Debian "passwd" package (refer to Chapter 4 on how to install additional software). As shown below, type in the current password first, press Enter, type in the new password, press Enter to confirm, retype the new password and press Enter to confirm, again.



As a regular user, you are allowed to change your own password only. The administrative root user is able to set a new password for itself and other users as well. In such a scenario, we call *passwd* as follows:

```
# passwd felix
Changing password for felix.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
#
```

The password is stored as a hashed value in the configuration file "/etc/shadow". The content of this file is only visible to the administrative user. The example below shows how to extract the information for the user "user" with the help of the *grep* command (see earlier in this chapter).

chfn

This command is also from the Debian “passwd” package and changes the user information that is stored on your system in the file /etc/passwd. During the installation of your Debian system the basic setup was already done. In order to modify this information you can run *chfn* without further parameters in interactive mode, or with one or more of the following options to adapt only a specific value:

- -f or --full-name: change the full name of the user
- -h or --home-phone: change the home phone of the user
- -r or --room: change the room number of the user
- -w or --work-phone: change the work phone of the user

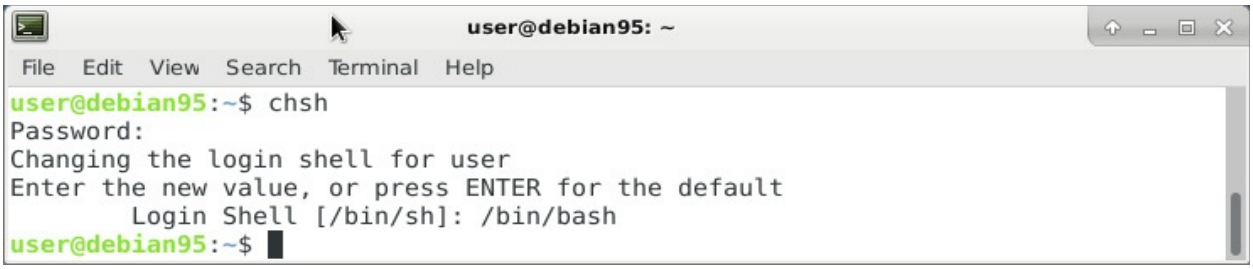
The following example changes the entry for the home phone number to 135:

```
$ chfn -h 135
Password:
$
```

chsh

This command (also from the Debian “passwd” package) changes the entry for the shell that you use to log into your Linux system. Again, this information is stored in the file /etc/passwd. Which shells are allowed to be used are limited by the entries in the configuration file /etc/shells.

chsh works similar to the *chfn* command. Invoked without further options an interactive method is used (see image below).



chsh accepts the option *-s* (short for --shell) in order to set the shell in non-interactive mode. The following example shows the according command line call:

```
$ chsh -s /bin/bash
$
```

In order to modify the shell for a different user other than yourself, invoke the *chsh* command with the user name as a parameter. Note that only the administrative user can do this for a different user. The next example shows how to do that for the user “felix”.

```
# chsh felix
Changing the login shell for felix
Enter the new value, or press ENTER for the new value
   Login shell [/bin/bash]:
#
```

su and sudo

In order to change your role from one user to another, you utilize the *su* command. *su* abbreviates “switch user”. Invoked without further options you change to the root user as follows:

```
$ su
Password:
#
```

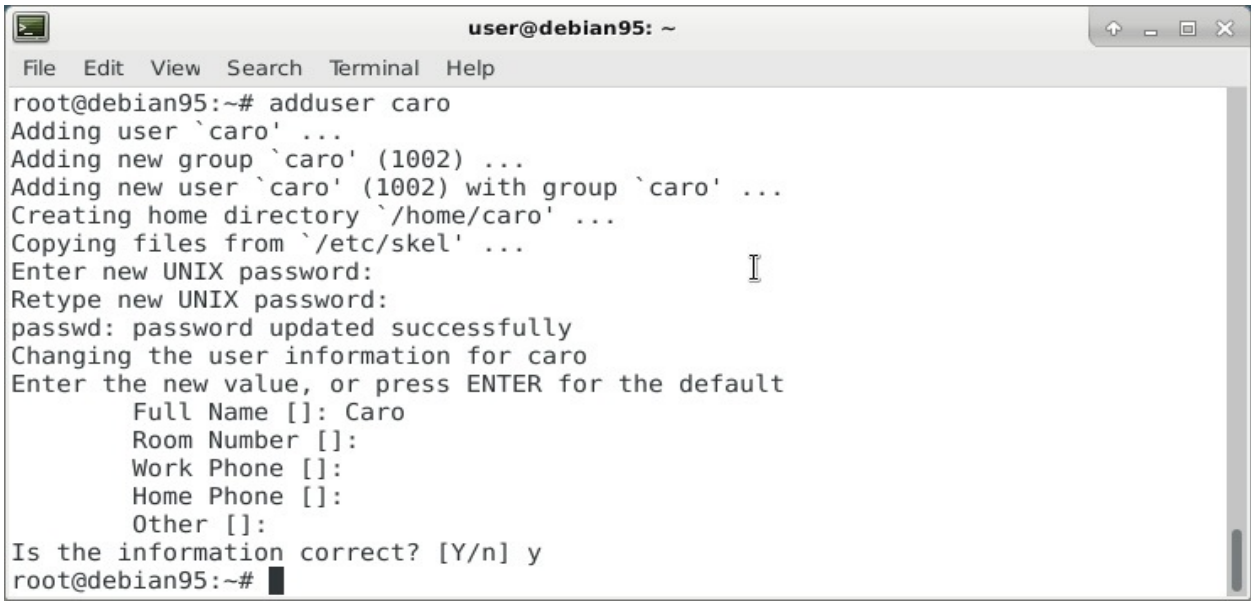
Working as the administrative root user comes with great responsibility and presumes that you know exactly what you are doing. To work as a different user than root, invoke the *su* command with the desired user name as follows:

```
$ su felix
Password:
$
```

The *su* command changes the current role permanently. In order to run only a single command as an administrative user, use the *sudo* command. This requires the Debian “sudo” package to be installed (refer to Chapter 4 on installing addition software) and the additional user to be added to the configuration file /etc/sudoers using the *visudo* command. This step will be explained in further detail later on.

adduser

The command *adduser* creates new user accounts. The image below shows the information that is required. This includes a new entry in the file /etc/passwd as well as the creation of a new group, plus home directory. Furthermore, prepared data from the directory /etc/skel is copied into the previously created home directory. Afterwards, the account information is modified using the *chsh* command (see earlier in this chapter).



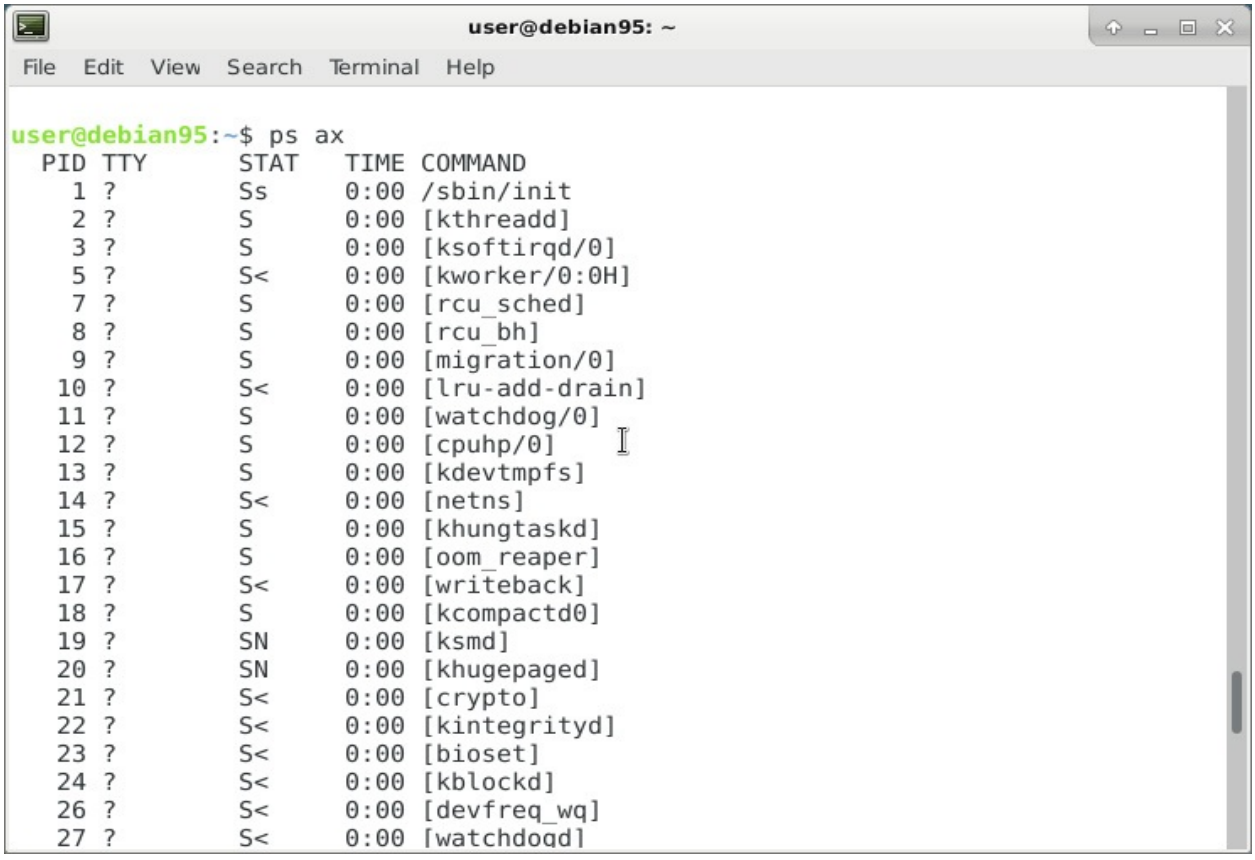
Having set up the new user, the entry in the file /etc/passwd looks as follows:

Deleting user accounts and modifying user accounts is done with the help of the two commands *deluser* and *usermod*. The usage of these commands will be explained in further detail later on.

7.4 Process Management

ps

This command lists the running processes of your Linux system. The image below shows the processes in your current terminal session, where the call *ps ax* (see the image thereafter) lists all the processes in your Linux system. The *ps* command has a long list of options that will be explained later on in more detail.



pgrep

pgrep is the *grep* command (see earlier in this chapter) for processes. This command looks through the currently running processes and outputs the process

ID’s that match the given pattern.

The image below shows two calls. The first call asks *pgrep* to search for the processes that have the string pattern “xterm” in its name, and to output the resulting process ID only. The second call uses the two options *-l* (short for --list-name) and *-a* (short for --list-full) in order to show the process ID and the resulting process name.

The partner of *pgrep* is *pkill*. Both commands are part of the Debian “procps” package (refer to Chapter 4 on how to install additional software packages).

kill

This command sends a specific signal to a process. In order to terminate this process use the *SIGTERM* signal. The next example shows how to end the process that has the process ID 12345.

The command *kill* requires a process ID to work properly. Use *ps* or *pgrep* to obtain the according process ID first (see earlier in this chapter).

```
$ kill 12345
$
```

killall

This command is part of the essential Debian package “psmisc” (refer to Chapter 4 on installing additional packages). *killall* sends a signal to all the processes that match the specified command. As an example, the command

```
killall firefox
```

sends the signal *SIGTERM* to all the processes that have the name “firefox” in order to terminate the process. Amongst other options, *killall* allows these parameters:

- -e (short for --exact): require an exact match
- -I (short for --ignore-case): treat the process name case-insensitive
- -i (short for --interactive): interactively ask for confirmation before terminating the process
- -v (short for --verbose): send a confirmation message if the process terminated

The following example shows how to combine the two commands *pgrep* and *killall*. First, obtain the process ID using *pgrep* as well as the exact name of the process. Second, invoke *killall* with the two options *-i* and *-v* to terminate all the processes that have that name. Prior to killing the processes *killall* asks you for confirmation, and afterwards outputs a confirmation message.

pkill

The two commands *pkill* and *pgrep* (see earlier in this chapter) belong to the same software package. Whilst *pgrep* scans the process list and outputs the process ID, the *pkill* command will send the termination signal to the processes that match the given pattern. In order to terminate all the “xterm” processes, invoke the following command:

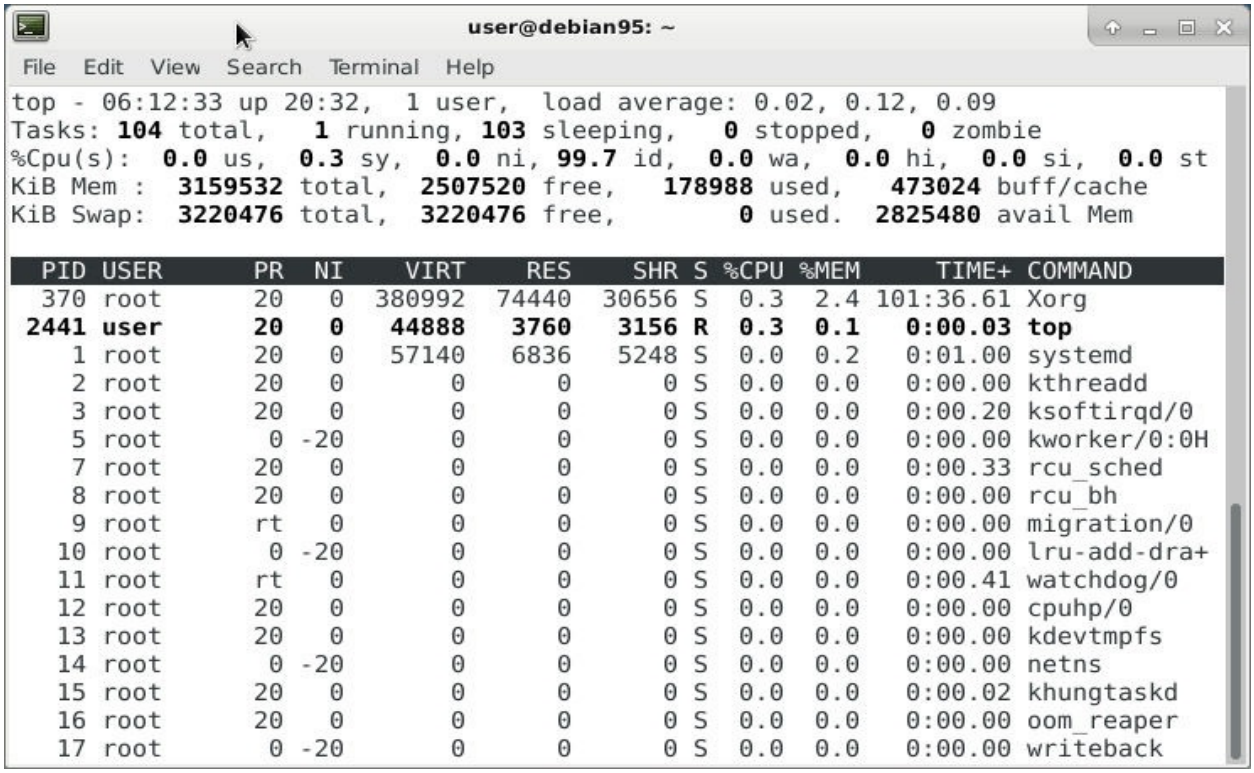
```
$ pkill xterm
$
```

top

The *top* command displays the processes according to their activity. The most active process is on top, followed by the less active ones. The list is updated every second. The single columns contain the process ID (titled PID), the user name of the owner of the process (titled USER), the process priority (titled PR), the nice level (titled NI), the virtual memory usage (titled VIRT), the reserved memory (titled RES), the shared memory (titled SHR), both the percentile CPU

and memory usage (titled %CPU and %MEM), the running time of the process (titled TIME+) as well as the command that was used to invoke the process (titled COMMAND).

In order to quit *top* press the *q* key.

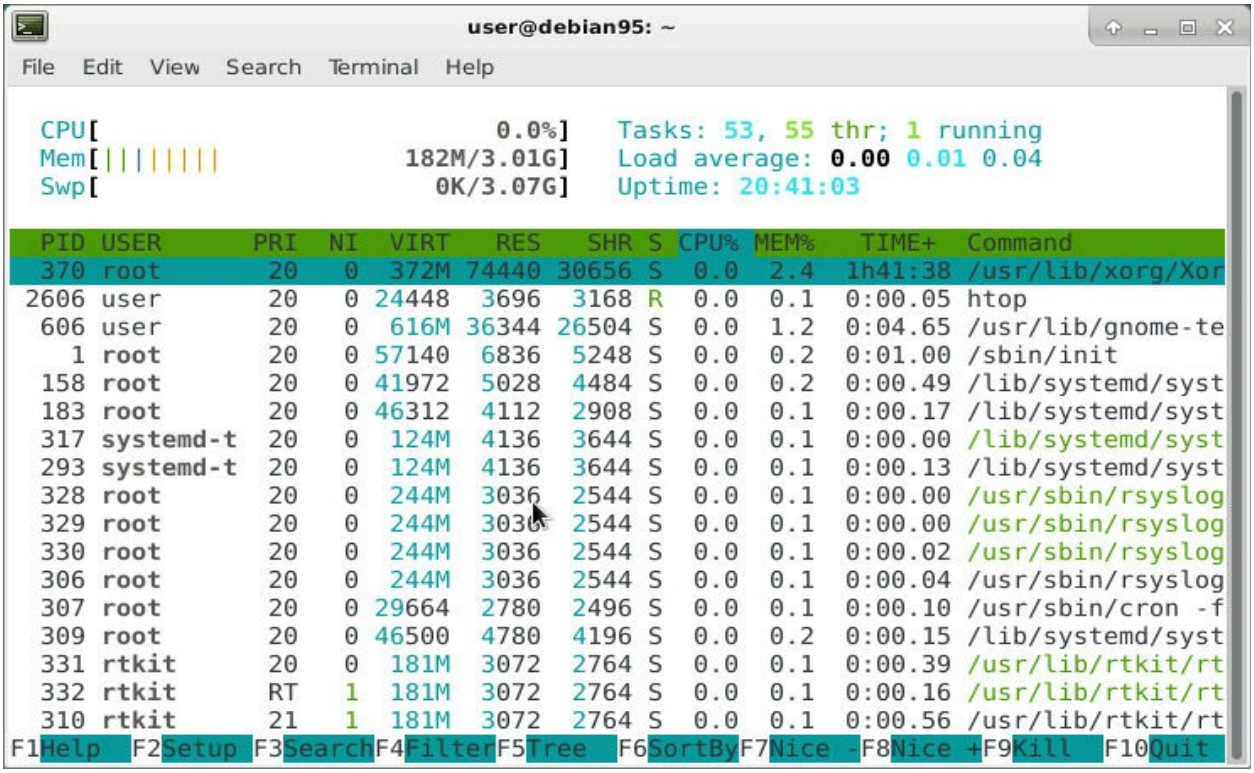


htop

“htop” is an additional software package (refer to Chapter 4 on installing additional packages) and contains a more interactive version of *top* (which we mentioned above). The arrangement of the columns is similar to *top*. Use the navigation and function keys in order to select the processes, sort them, or delete them. The function keys are:

- F1: show help
- F2: configure htop
- F3: search within the process list
- F4: filter the terminal output
- F5: display the processes as a process tree
- F6: change the sort order of the processes

- F7: decrease the nice level of the selected process
- F8: increase the nice level of the selected process
- F9: terminate the selected process
- F10: quit the program



7.5 Network and System Information

uname

uname abbreviates the term “UNIX name”. The command displays system information such as the exact name and version of the Linux kernel and the host name of your computer.

The image below shows the call of the *uname* command with the parameter *-a* (short for --all). The output contains the name of the operating system (Linux), the hostname (debian95), the kernel version and its build date (4.9.8-7-amd64 #1 SMP Debian 4.9.110-1 (2018-07-05)) as well as the architecture of the system (x86_64).

uptime

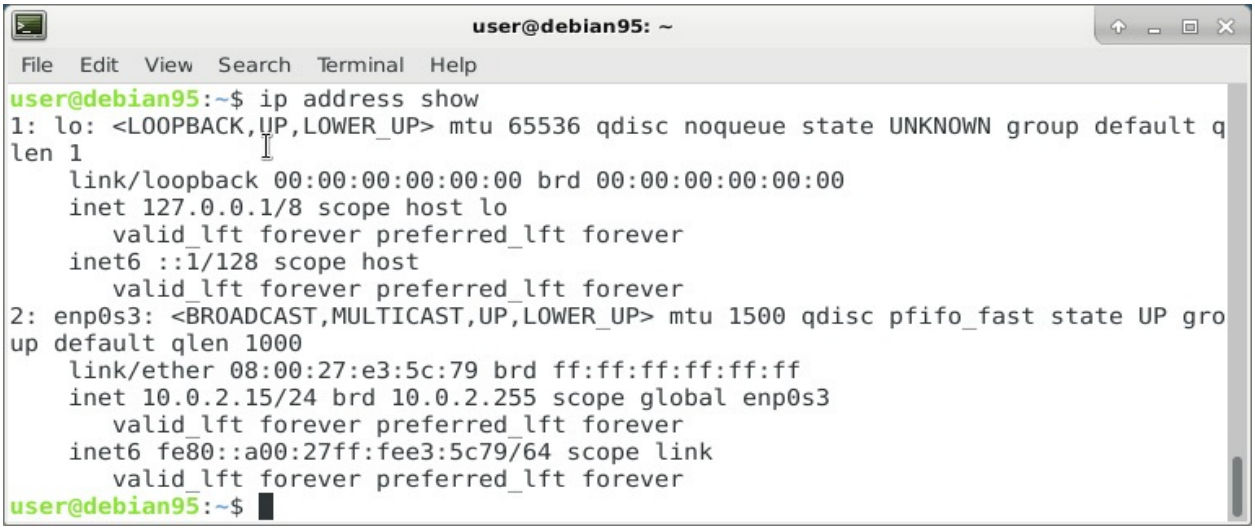
This command shows how long the system is running. It displays the current time (07:36:47) followed by the uptime in hours (21:56), the number of logged in users (1 user) and the average load (load average: 0.47, 0.43, 0.29) for the last 1, 5 and 15 minutes.

In order to see the *uptime* in a nicer way, use the option *-p* (short for --pretty). The image below displays a more human-readable version of the information. The system is up 22 hours and 3 minutes.

ip

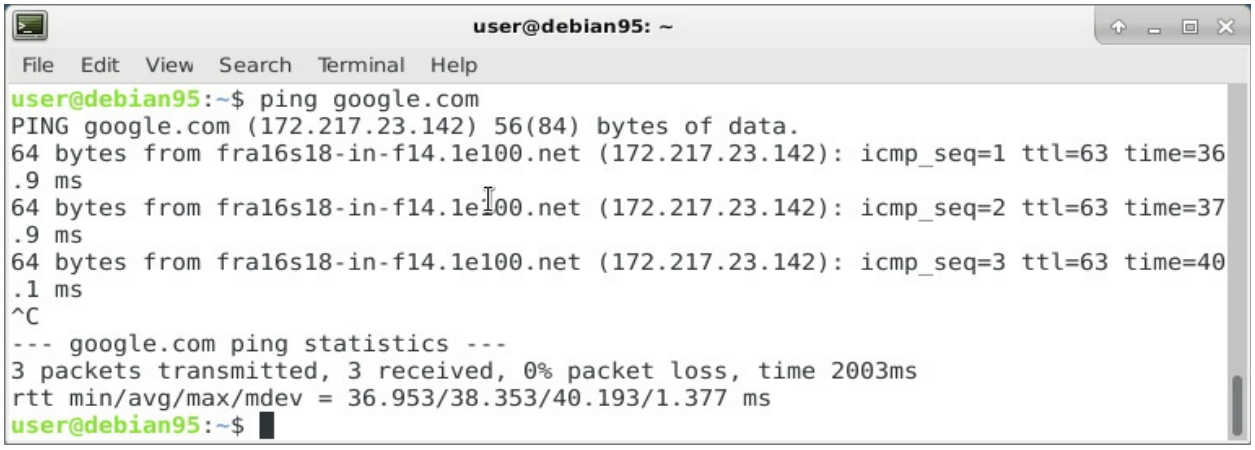
The *ip* command (along with the two keywords *address show*) displays the current network configuration. The image below shows the loopback interface (lo) and the ethernet interface (enp0s3). The ethernet interface is configured with the IP address 10.0.2.15. The network interfaces are abbreviated as follows:

- lo: the loopback interface. It is used to access local services such as a proxy or webserver http://127.0.0.1/
- eth0: the first Ethernet interface connected to a network switch or router
- wlan0: the first wireless interface
- ppp0: the first point-to-point interface, used to connect via VPN or dial up service



ping

ping sends ICMP network packets to the given IP address or hostname, and displays the turnaround time. The image below demonstrates this for the host http://www.google.com.



8. Getting Help

UNIX/Linux is a rather complex operating system, especially for beginners. When you get started, the list of programs can be quite confusing. Each program has a longer list of parameters or options than the last. And they are not easy to memorize, they have long and short names, and are not standardized. Asking for help when you get stuck is not a mistake or sign of low knowledge.

In order to get help on your Linux system, there are several ways to go about it. This includes the manual pages (called man pages), information pages (called info pages), as well as the integrated help of each command. In this chapter we will have a closer look at these help systems and get familiar with the commands *man*, *info*, *whereis* and *whatis*.

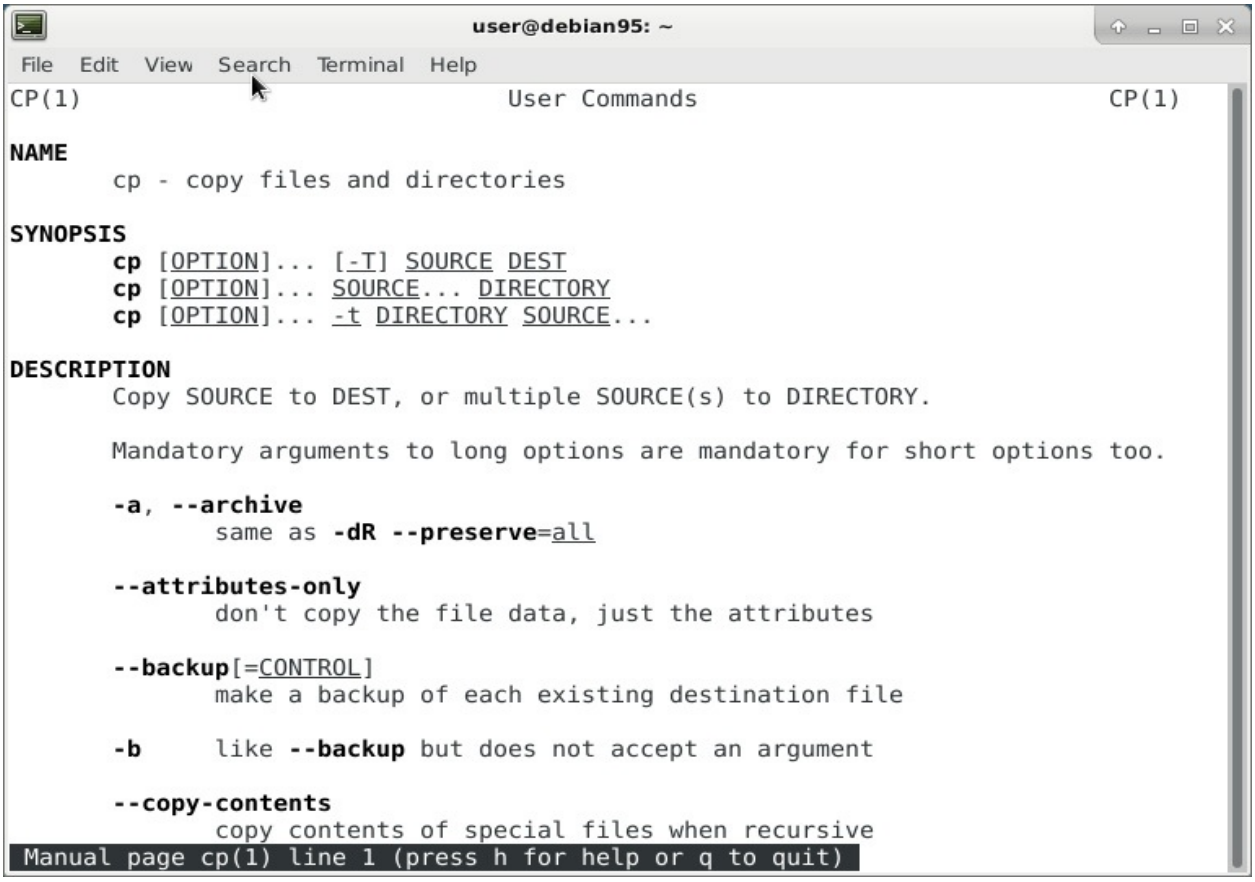
8.1 Man Pages

Man pages are the traditional way of distributing documentation about programs. The term “man page” is short for “manual page”, as they correspond to the pages of the printed manual. Man pages are part of the basic installation of your Linux system. The corresponding Debian package is named “manpages”.

A man page corresponds to a specific section in the full UNIX manual: 1 for commands, 2 for system calls, etc. To get information regarding a specific command, open the corresponding man page in a terminal window. As an example, for the *cp* command it is done as follows:

```
$ man cp
```

The resulting man page looks like this:



The key bindings are similar to the ones from the text editor *vi(m)*. Use the navigation keys to scroll up and down, and *q* to exit the man page and to return to the terminal.

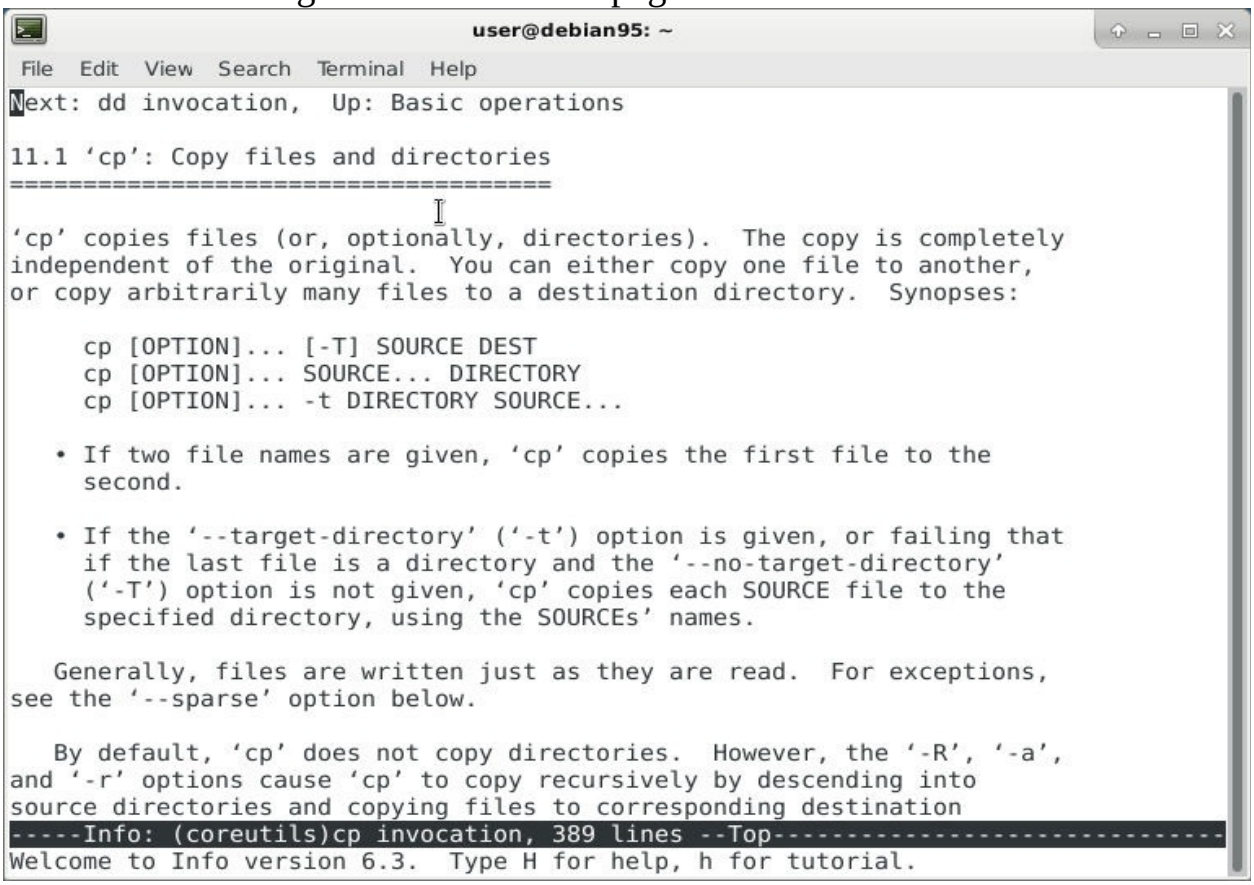
8.2 Info Pages

Info is the default format for documentation inside the GNU project. In the early 1990s, the GNU project decided that the *man* documentation system was outdated, and wrote the *info* command to replace it. *info* has basic hyperlinking features and a simpler markup language to use (compared to the *troff* system used for man pages). Nowadays both systems exist in parallel.

The basic installation of Debian does not contain the info pages. In order to use it, post-install the package named “info” (refer to Chapter 4 on how to install additional packages). To get information regarding a specific command, open the corresponding info page in a terminal window. As an example, for the *cp* command it is done as follows:

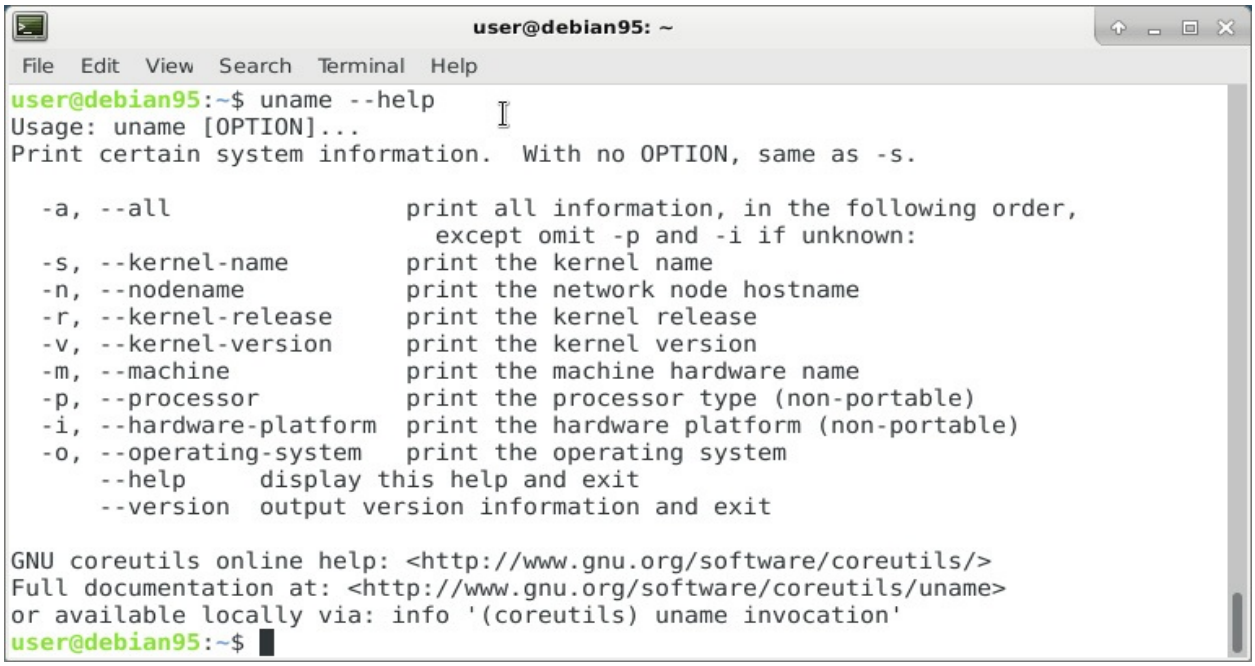
```
$ info cp
```

The resulting info page looks like this:



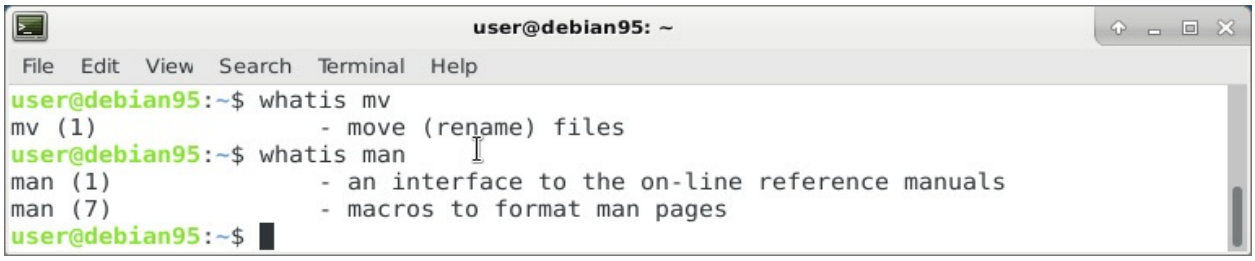
8.3 Integrated Help

Aside from *man* and *info* pages, many commands have an option *-h* (short for --help). Calling the command with this option opens a specific help section. The image below shows the built-in help of the *uname* command.



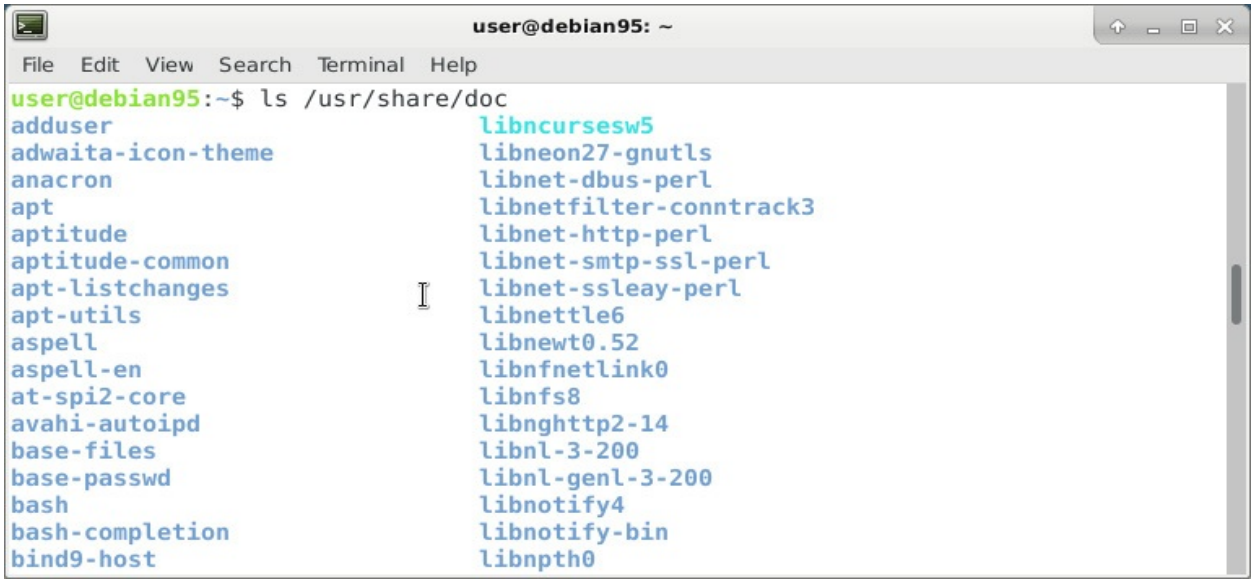
Alternatively, the *whatis* command is helpful to get brief information about Linux commands or functions. It displays a single line man page description for the command that matches the string passed as a command line argument to the *whatis* command.

The *whatis* command searches for the string in its index database, which is maintained by the *mandb* program, and picks a short description from the NAME section of the man page for that command. The image below shows this for the two commands *mv* and *man*.



Another option is the *whereis* command, which is helpful to locate the manual page of the command in the Linux system. It is very simple utility. The image below shows this for the two commands *mv* and *man*.

Every command also provides additional information and configuration examples. This is kept in the directory */usr/share/doc*, and every program has its own subdirectory. The content is standardized and differs from program to program. The content is also accessible for every user. A snippet of the directory is shown below.



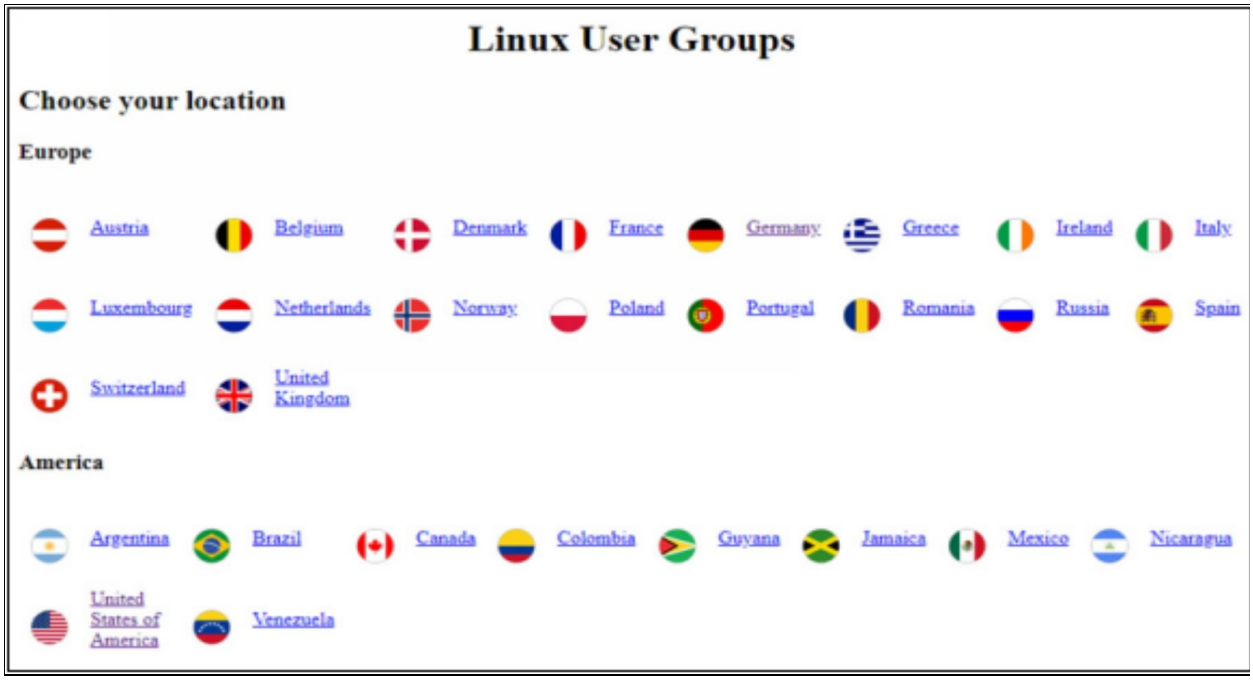
8.4 External Help

If the help already provided with the software tools is not enough, and further explanation is necessary, external resources come into play. There are a few online communities that can be consulted such as:

- LinuxHint - https://support.linuxhint.com
- LinuxHelp - https://www.linuxhelp.com

Another option is *Linux User Groups* (LUG). A LUG is a group of people from the same place sharing their common interest in the Linux operating system.

They are organized as a loosely associated number of people, or an association that meets regularly. A full list of LUGs worldwide is available from *LugsList* at http://lugslist.com



Many LUGs organize meet-ups and events in order to share ideas, exchange experiences, and to learn from each other. This includes local and international events.

Further Reading

If you found this book helpful, please consider leaving a review on Amazon. It shows my wife and I that the long nights we put into creating it were worth it ;) For further reading on Linux, keep a lookout for more books within this series coming soon.

About the Author

Nathan Clark is an expert programmer with nearly 20 years of experience in the software industry.

With a master’s degree from MIT, he has worked for some of the leading software companies in the United States and built up extensive knowledge of software design and development.

Nathan and his wife, Sarah, started their own development firm in 2009 to be able to take on more challenging and creative projects. Today they assist high-caliber clients from all over the world.

Nathan enjoys sharing his programming knowledge through his book series, developing innovative software solutions for their clients and watching classic sci-fi movies in his free time.