CHAPTER 9

Mageia

Mageia is one of the newer Linux distributions that I selected to analyze in this book, and it's the third most popular European Linux distro. Mageia is a fork of an older distribution, Mandriva (formerly Mandrake), which is discontinued but is the second most famous European Linux distro of all times. Mandrake was originally a fork of Red Hat. It was also inspired by SUSE, and today you can still find echoes of that in Mageia.

History

Mageia is the Greek term for "magic" and it is a gesture to the original Mandrake distro, which was named after the Italian-American magician Leon Mandrake. Mageia is a community distro supported by a non-profit organization, Mageia.org.

Mageia was born in the agony of Mandriva. Mageia is one of the heirs (along with PCLinuxOS and others) of a great lineage that started with Mandrake in 1998. It merged with another old Red Hat fork, Conectiva (1997), in 2005 and eventually changed its name to Mandriva (Mandrake + Conectiva). Mandriva/Mandrake was always a company-supported distro, but the company had problems staying afloat financially. Anticipating its coming demise, a bunch of developers of the company behind Mandriva and some other community members decided to fork the distro; as a result, the Mageia project was announced in September of 2010. The first version was released in June of 2011. Mandriva shipped is last release in the same year.

Mandriva/Mandrake was a French distro and it was commercialized, presented, and evolved similarly to the other great European distro, OpenSUSE/SUSE; this was no accident. You can see similarities and influences in both distros. In fact, I saw how these distros evolved, but even I can't remember who inspired who in each aspect.

Philosophy

Well, the truth is that the main purpose of Mageia was to perpetuate the legacy of Mandriva and continue the distribution where it left off. One of the major goals of Mageia/Mandriva was to create a pleasant and easy-to-use distro for all users. For a long time, this was the distro (along with SUSE) for new users who wanted an amicable distro. Mageia defines its mission these days as building great tools for people.

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_9

Distro Selection Criteria

Now that you know some history, let's see how Mageia rates on the selection criteria from Chapter 2.

Purpose and Environment

Mageia is a general purpose distribution that in the same manner as Mint, only it focuses on the desktop. Mageia offers a unique version for this environment.

Support

The old Mandriva had a great support; it even published a manual as a book (shipped with the Mandriva Powerpack), as SUSE did in the past. But now Mageia is a community-based distro and the support comes from members. It is not as complete as the support a company could provide. But there are many good ways to get support from the community:

- **Documentation**: `www.mageia.org/en/doc`
- **Wiki**: `https://wiki.mageia.org/en/`
- **Forum**: `https://forum.mageia.org/en/`
- **Mailing lists**: `https://ml.mageia.org/`
- **IRC**: `#mageia` at freenode

User Friendliness

Mageia (more exactly, the former Mandriva) had a reputation of being very user friendly. This prestige comes from the omnipresent Control Center (the equivalent of the OpenSUSE's YaST) from where you can configure almost all the most important parameters of your user distribution (from a user perspective). But it is important to know that this easiness is not an absolute value that you can measure objectively, because even when you can control almost anything from one place, you still have to know what you are doing. Thus, for a user used to Linux, Mageia or OpenSUSE can seem very easy to use and friendly, but to a newcomer it seems difficult. A newcomer will probably prefer other distributions like Ubuntu, Mint, or elementary OS. The reality is that that glory came from years ago, when installing Linux was not an easy task, and Mageia was a welcomed alternative for those afraid of installing Debian or harsher ones like Slackware. But when Ubuntu arrived, it changed all that; since then, Mageia/OpenSUSE and similar ones went down a step in what a new user considers friendly (and being frank, it would be stupid to ignore the influence that Mac OS X has had in all of this).

Stability

Mageia is a reasonable stable distribution. It follows a standard release model and tries to release a new version each nine months, followed by 18 months of support.

When this release cycle program was announced in 2011, a LTS version was considered but it never happened. The community lacks the muscle (maintainers and QA team) to achieve this big effort. This is a perfect example of a distro that was accustomed to having a company behind it but was later "diluted" by various community forks that want to keep the pace and the quality but are still working to return it to its

past glory. This does not mean that Mageia is not a stable distro with a reasonable quality; it just means that maintaining a LTS version, or several different versions, is an expensive task that not too many distros can afford (specially community ones).

There is a development version called Cauldron that is always up to date. Of course, it is inherently unstable and it is aimed at Mageia packagers.

Hardware Support

Mandriva was traditionally seen as a good distro regarding hardware support, and its successor seems to continue along this line. Mageia provides proprietary drivers and it does good work detecting hardware, so I can say it has good hardware support.

Aesthetics

There is not too much of a focus on aesthetics in this distro, apart from the backgrounds, colors, and logos. All of the rest is the default DE aesthetic.

Desktop Environment

Mageia supports various desktop environments, eight of them officially from the DVD ISO image of the distro: KDE, Gnome, XCFE, Mate, Cinnamon, LXDE, LxQT, and Enlightenment. The traditional DE of Mageia/Mandriva is KDE.

Init System

Mageia uses systemd as its init system. In fact, it was one the early adopters, in 2011.

Package Management System

As part of the legacy inherited long ago from Red Hat, the package management system used in Mageia is RPM. There is a graphic tool to manage the packages (`rpmdrake`) and several command line tools (`urpmi`, `urpme`, `urpmq`, `urmpf`) instead of a unique terminal tool to do all the operations.

Architecture

It only supports two major architectures, the AMD/Intel 32- and 64-bit architectures.

Security/Anonymity

The security in Mageia is very similar to other general purpose distros. A firewall is enabled by default and a “standard” level is selected by default in the security tool (called MSEC). Of course, as with almost every Linux, you can always harden your installation, but MSEC is very helpful to manage this topic in an easy way (or with grain too). MSEC is developed by Mandriva and it supports the usual tools like Apparmor or SELinux by default.

Principles and Ethics

The core repositories of Mageia offer only free software but you can always install proprietary software or mixed ones (e.g. that use patented and copyrighted code) from the "non-free" and "tainted" repos. Thus, the compromise about free software is addressed in a very pragmatic way, as it is in the majority of distros. On the other hand, Mageia was the first distro to make the switch to MariaDB from MySQL to avoid Oracle's copyright.

Live CD

Mageia has Live ISO images available for download as DVD and CD. The only inconvenience is that the CD images are only available in English, and you have to choose if you want the KDE or the Gnome desktop environment.

Professional Certification

There is no professional certification available specifically for Mageia.

Installation

Until now, all of the distro installations shown in this book were made on a system with a traditional BIOS, but for the past few years new computers (mostly laptops) have come with the modern "replacement" of that traditional, essential firmware, the UEFI. Some UEFI firmware has a legacy BIOS-compatible mode and you can choose which of those modes you want to boot your system in. The legacy BIOS mode is useful for those people who still feel uncomfortable with UEFI because even though it is better than BIOS, it is also more complex. But it is useful for installing distros that do not support UEFI. All of the distros shown previously in this book support installation in UEFI mode, but as the installation process is also more complex, in some distributions this can be more difficult or cumbersome (for all of the previous distros, it is not very hard to do), and some users prefer to boot in BIOS mode. In Mageia, following its tradition of making things easier, it's not much harder to install the distro in UEFI mode than in BIOS mode.

First things first, go to the Mageia site (`www.mageia.org/en/downloads`) to download the ISO image (see Figure 9-1).

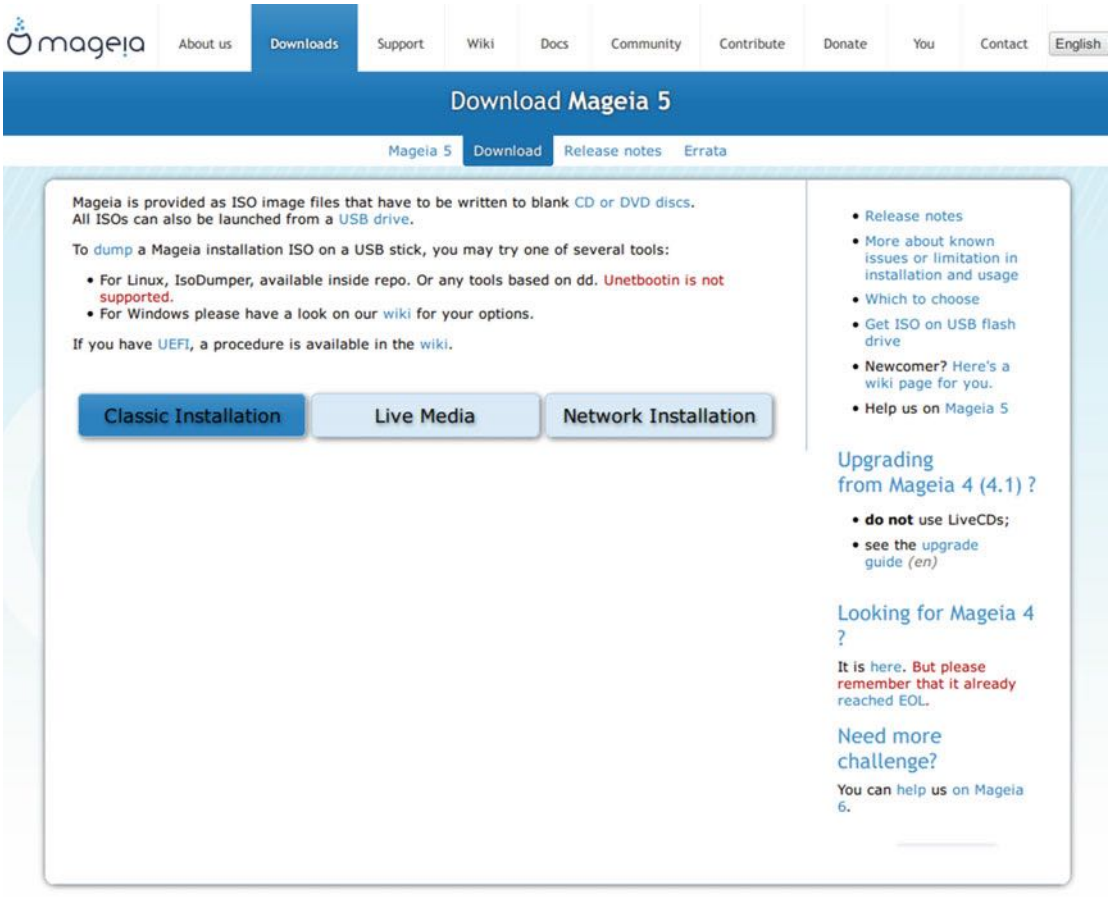


Figure 9-1. *The Mageia downloads page*

As with many other distros, you must choose the type of image you need. In this case, you can choose between the standard image, the live image, or the network one. Pick the standard one (Mageia call it "Classic") and you'll get three new options to select from, as seen in Figure 9-2.

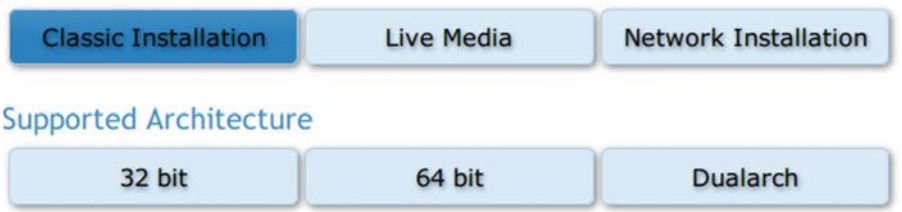


Figure 9-2. *The architecture options for the ISO image*

These options are the different architectures in which you can install the image. There is an ISO image for each one of them, and others that can be installed in both architectures. In this case, select the 64-bit architecture. This will lead to a new dual option in order to choose the way to download the image, via BitTorrent or the standard HTTP link (see Figure 9-3).

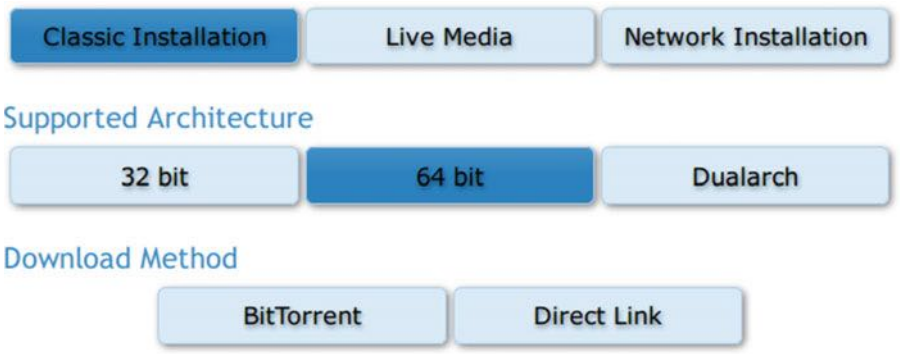


Figure 9-3. *The available protocols to download the ISO image*

Once you choose one of these ways and download the image, you can begin the installation. When you boot for the first time, you will see the first difference between installing the distro in a BIOS system or in an UEFI one. Figure 9-4 shows the boot screen for a BIOS system on the left and at the similar screen for an UEFI system on the right.

Figure 9-4. *The boot screen for both firmware types*

As you can see, the menus are different but they basically have the same options for installing Mageia. The other options are not essential for this task. One of the options available in the BIOS mode is the Hardware Detection Tool; this was not in the boot menu of any of the distros you saw previously, but it was very common in the past in many distros. It is a tool that can be very useful to troubleshoot hardware problems, and you can take a look at it in Figure 9-5.

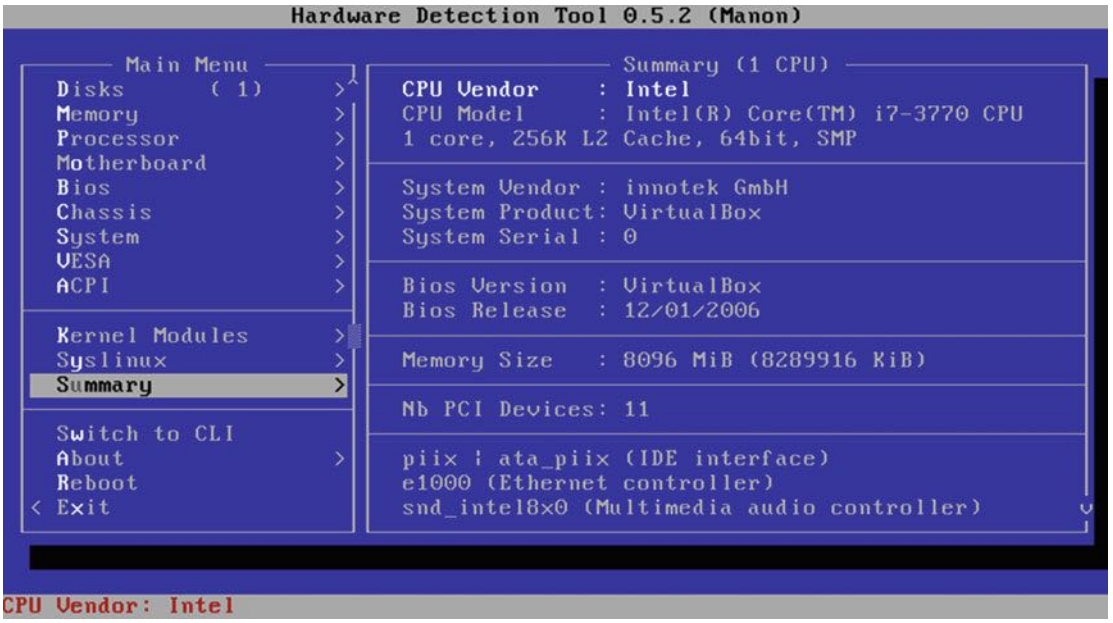


Figure 9-5. *The Hardware Detection Tool*

In the menu of the UEFI mode, choose the "Start Mageia install from ..." option (or wait a few seconds for it to start automatically) to go to the next step. From here, the installation is similar for both cases, so I'll show you only one screen. The first screen that appears in the Mageia installer is the one shown in Figure 9-6, the usual option to select the language to use in the installation process and/or distro. One peculiarity is that you can choose to use multiple languages from this screen (by clicking the "Multiple languages" button) and then you can then switch between them when you are using Mageia.

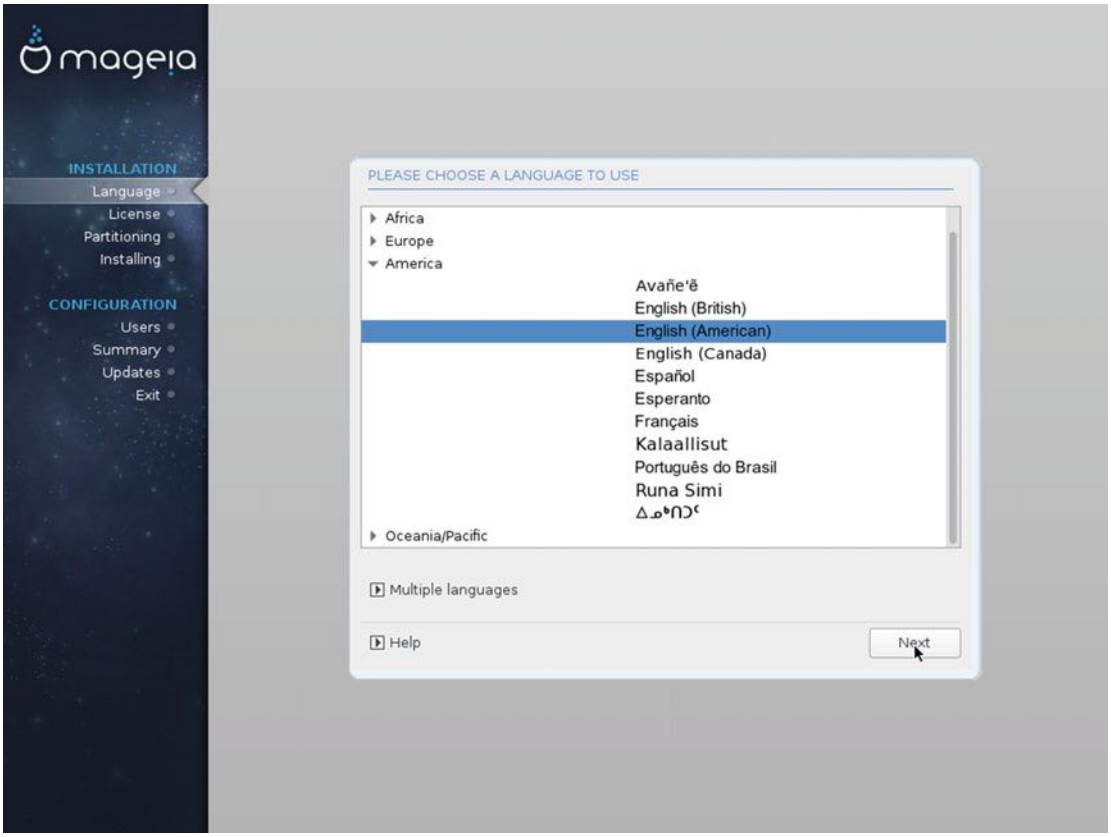


Figure 9-6. *The language selection screen in the Mageia installer*

The next step (Figure 9-7) is the one where you have to accept the License Agreement in order to continue the installation process (people don't usually read this, but I recommend you do so; it's always good to know something about how this license works and then you can compare it to the EULA of other OSes).

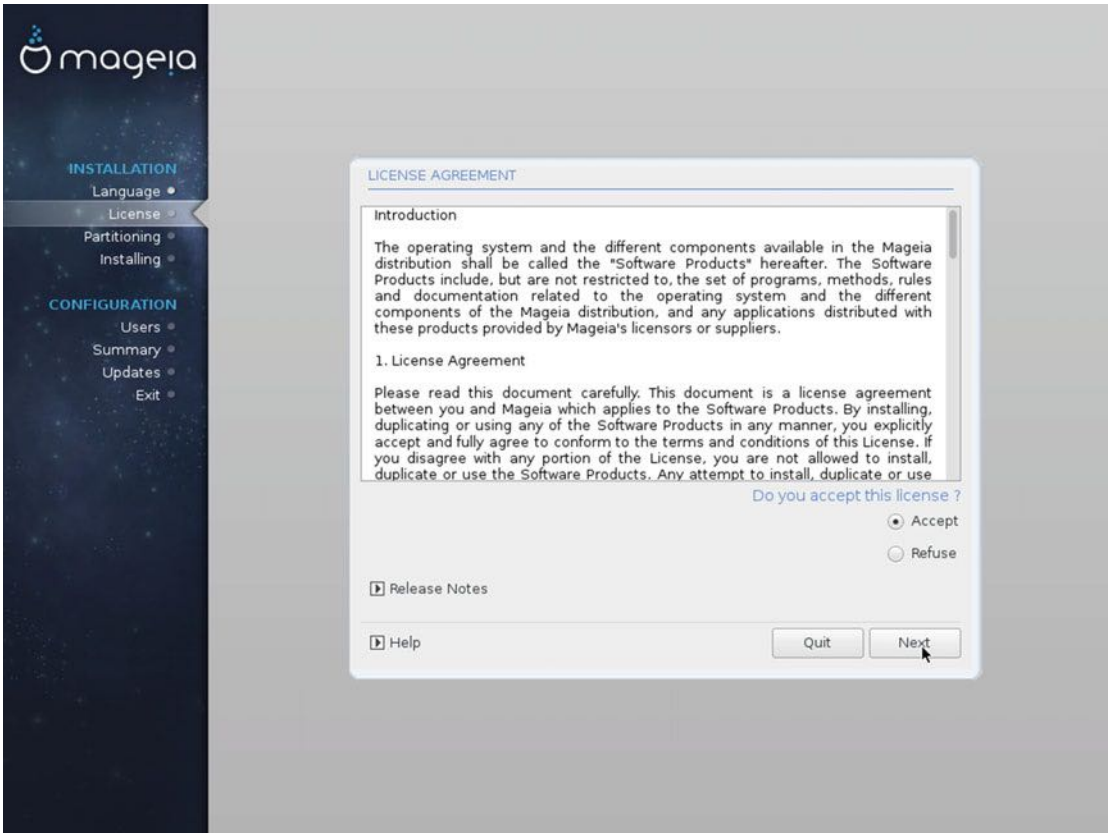


Figure 9-7. *The License Agreement screen*

If you accept the license, you can continue to the next screen (Figure 9-8) where you can choose the layout of your keyboard.

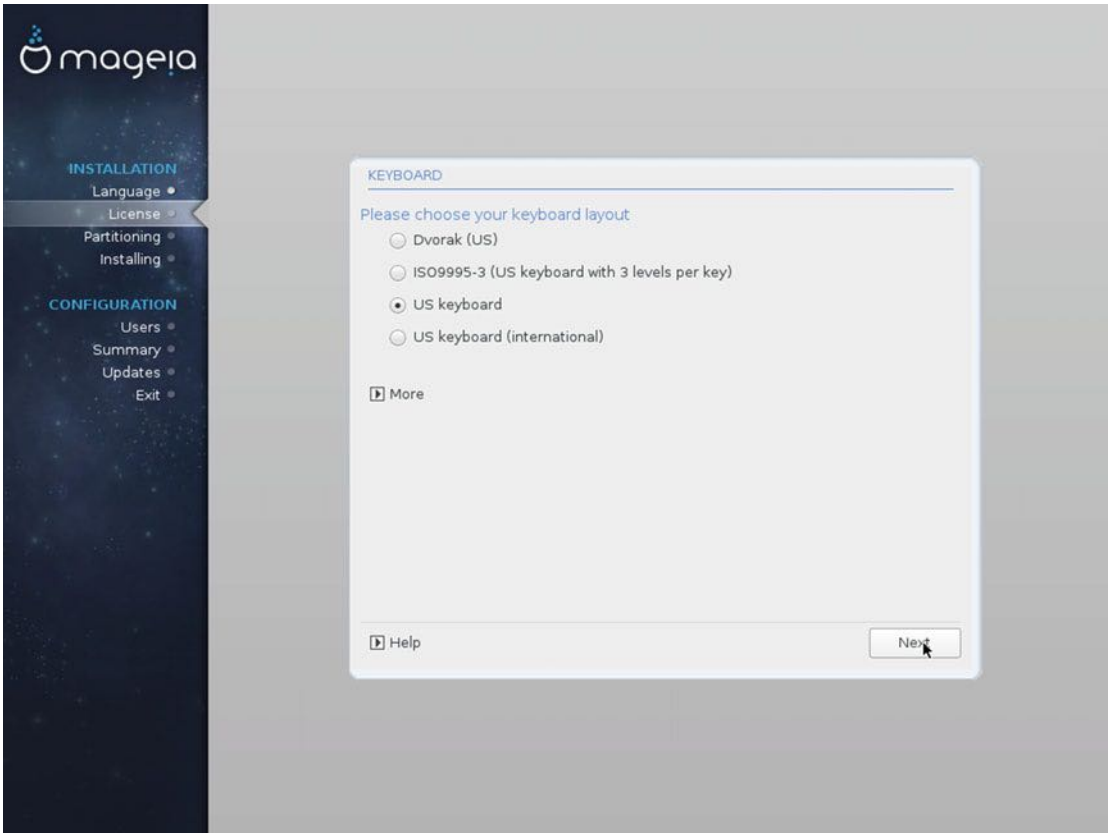


Figure 9-8. *Keyboard layout selection*

As in previous distros, the correct keyboard layout should be selected by default; if not, select the right one for you and press the Next button. This next step is always the most crucial one; it's where you choose how you are going to partition your disk drive. As you can see in Figure 9-9, this screen shows the current status of the disk drive selected and gives you two options to partition: an automated one using the free space on your disk or via a customized partitioning.

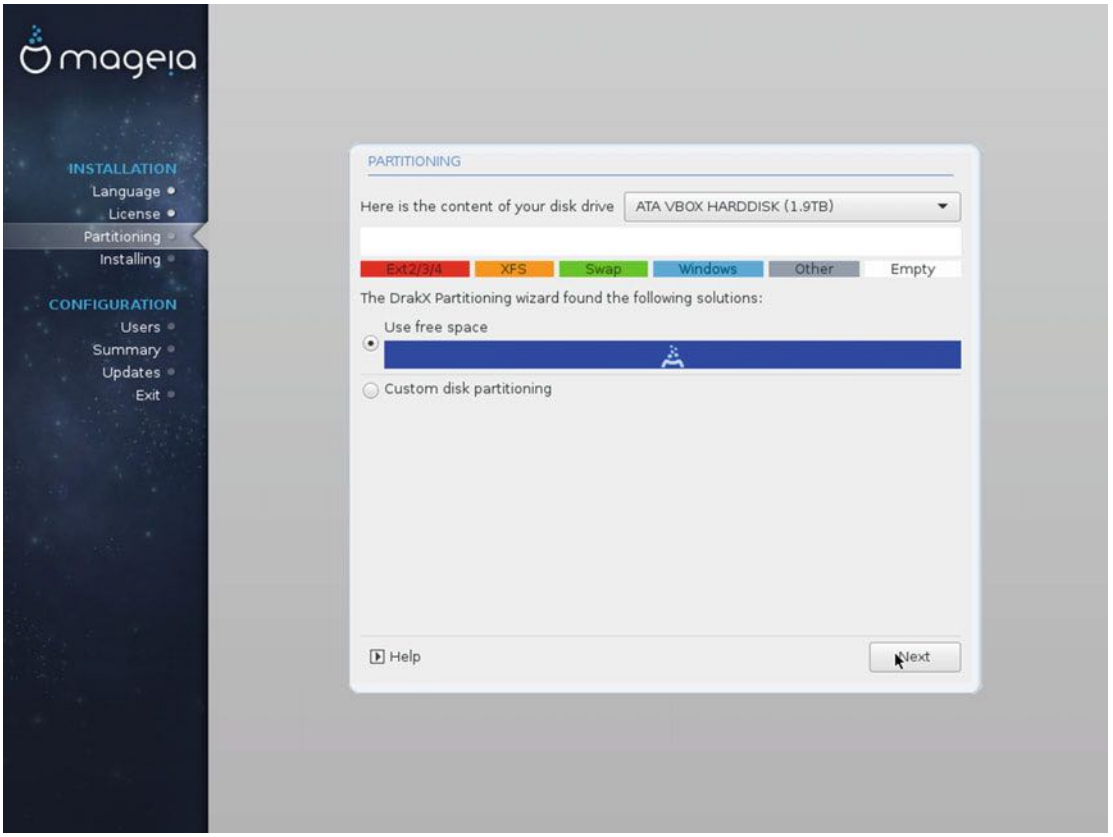


Figure 9-9. *The partitioning tool*

Let's do the custom partition option. This will also show the subtle but important differences between a BIOS and a UEFI installation. So select "Custom disk partitioning" and press the Next button.

Figure 9-10 shows how this custom option works. There are two options to choose from, an automated option (the same as before but now in this part of the tool) called "Auto allocate" or to manually partition the disk through the interface (by clicking on a partition or free space). Also, you can choose between the normal and expert mode, where more information is shown and more options are available to configure. In order to show the differences between the BIOS/UEFI installations, the best way is to select the automated partition option to view how the partition scheme is created in each case. Well, first, let's change to the expert mode to show more information. Next, create the predefined partition scheme by pressing the "Auto allocate button." As you can see in Figure 9-11, a new dialog emerges, asking you which type of partition you want (this dialog only appears in the expert mode).

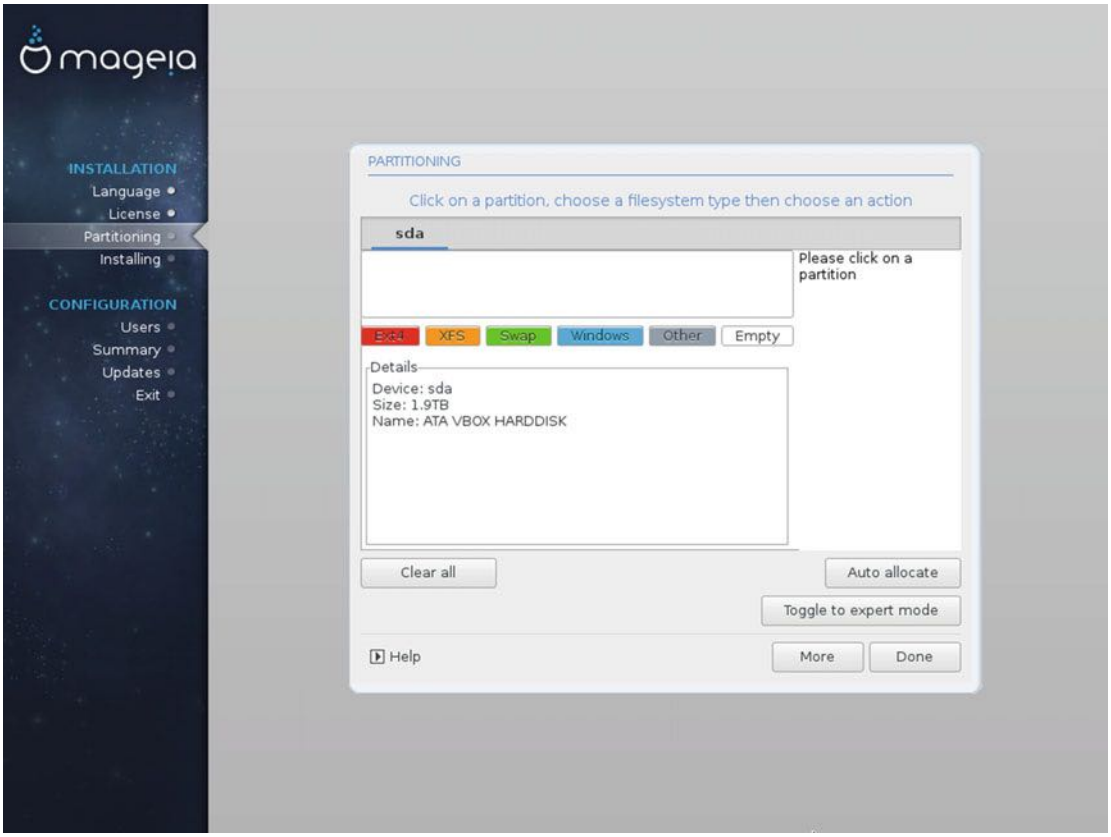


Figure 9-10. *The custom partitioning option*

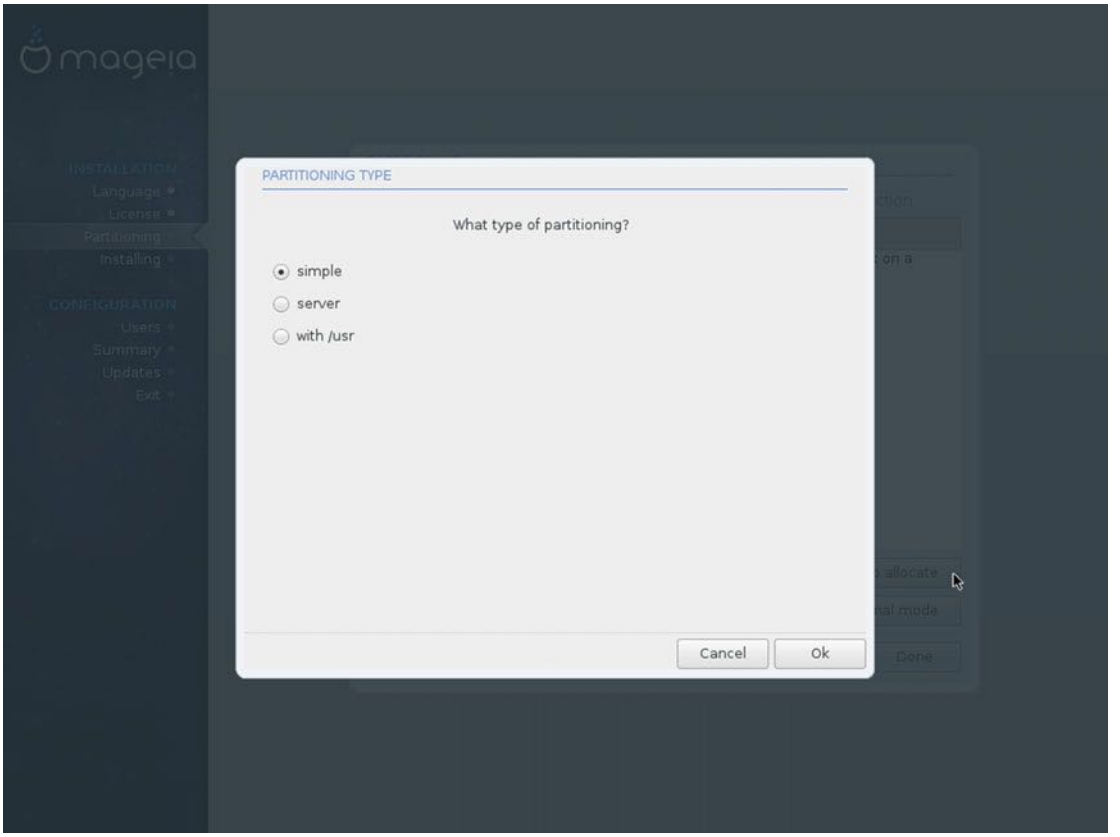


Figure 9-11. *Type of partition to choose*

Choose the "simple" option and click the Ok button. A new partition scheme is created where appears for first time in the book a new partition, allocated at the beginning of the disk that is "colored" as a Windows partition (in blue) and is mounted under `/boot/EFI` . This partition stores the necessary system files to boot the system and it is a DOS partition type, which is why it appears as a Windows partition. The rest of the partitions are the usual ones (`/`, `/home` and `swap`). You can see the result in Figure 9-12 and compare it to Figure 9-13, which shows doing the same in a BIOS system.

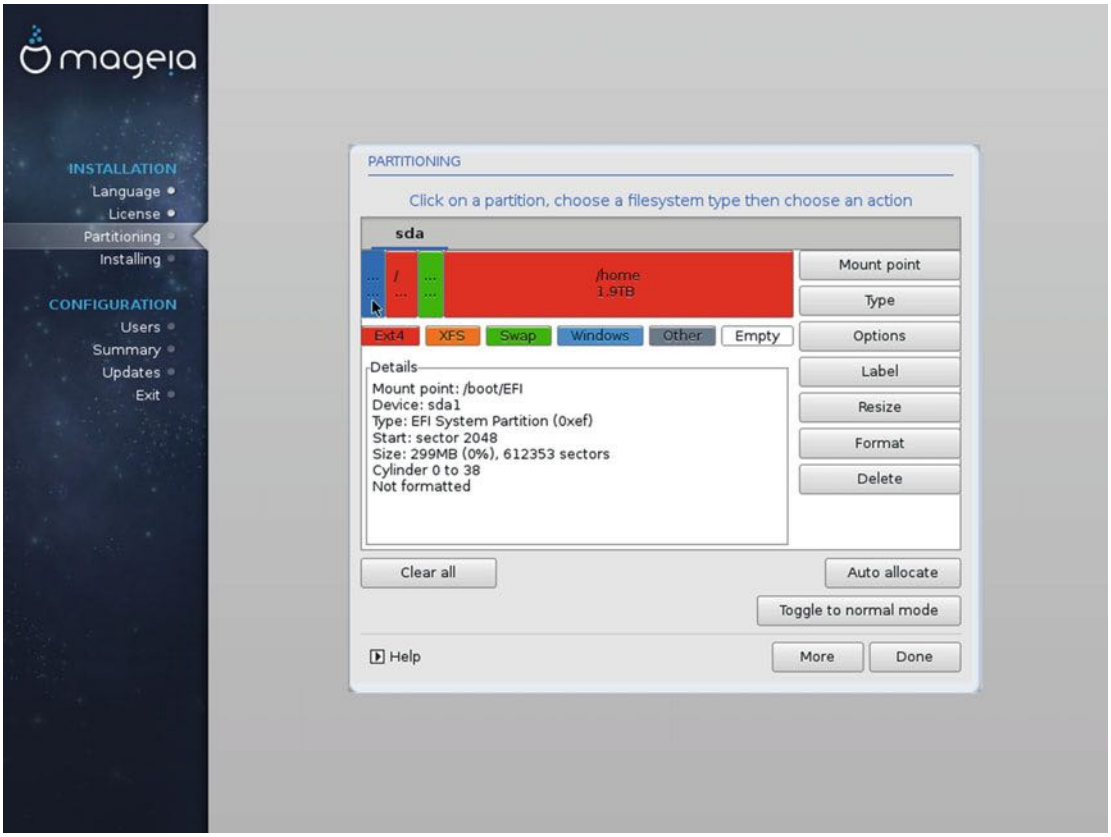


Figure 9-12. *The default partitions in a UEFI system*

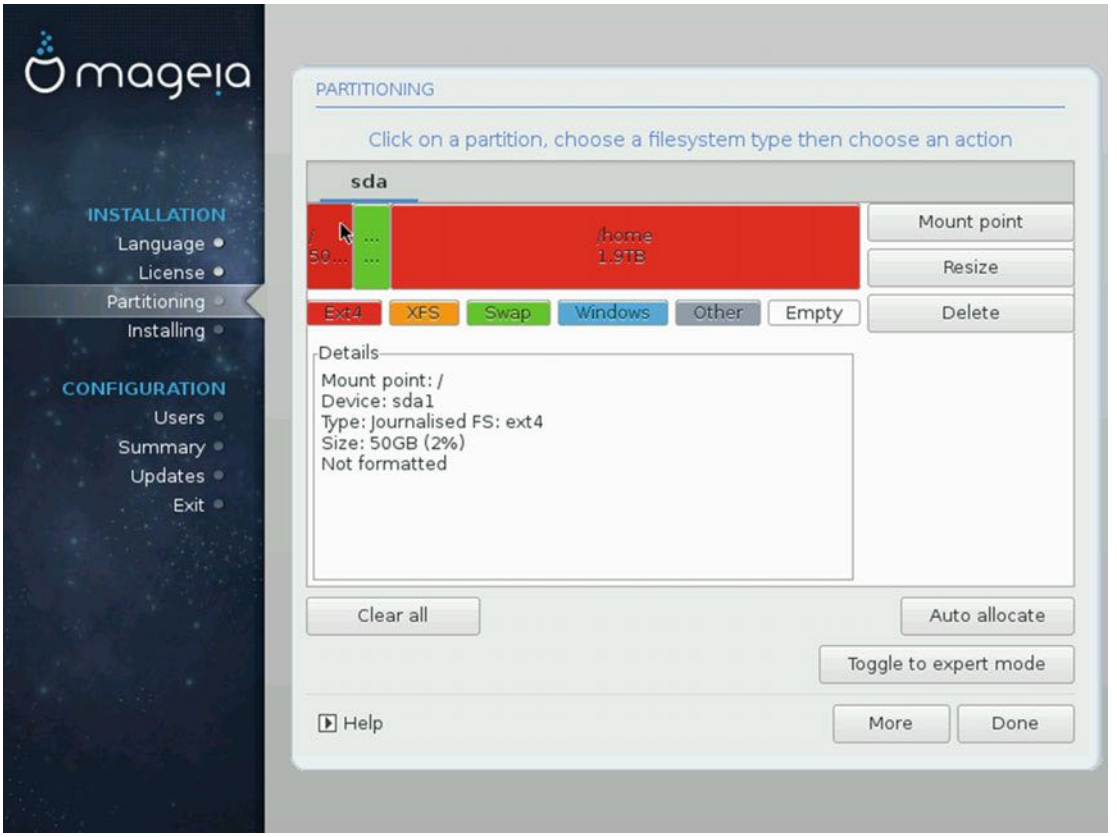


Figure 9-13. *The default partitions in a BIOS system*

Now that you have created the partitions, you can press the Done button. After the usual warning about making the changes to the disk, the partition process completes and you advance to next step. This step is something you saw in other distros like OpenSUSE, where you are asked if you have another media available to use in the installation (shown in Figure 9-14).

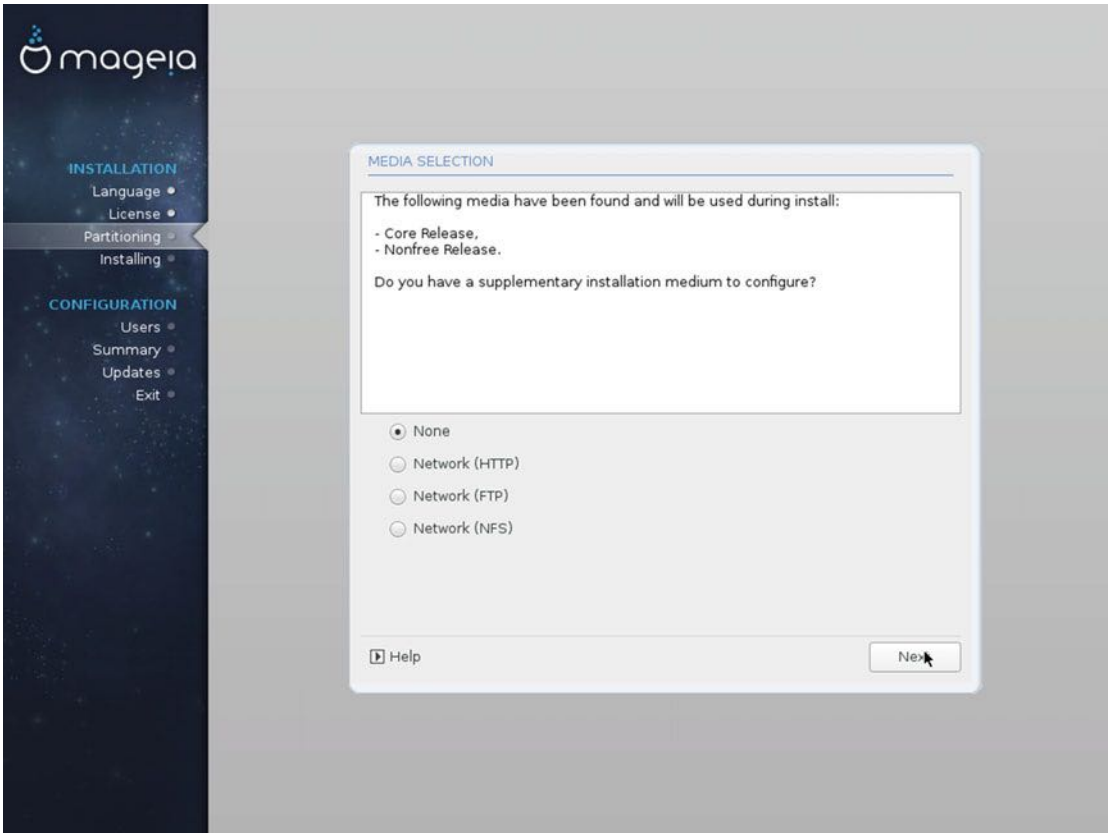


Figure 9-14. *Is there another installation medium available?*

In the following screen you can pick packages you want to install (Figure 9-15). You can stick with the core packages or you can add non-free packages (e.g. private firmware) to the installation. This screen seems a little confusing to me, because some people could confuse the local repositories with the online ones and get the wrong idea.

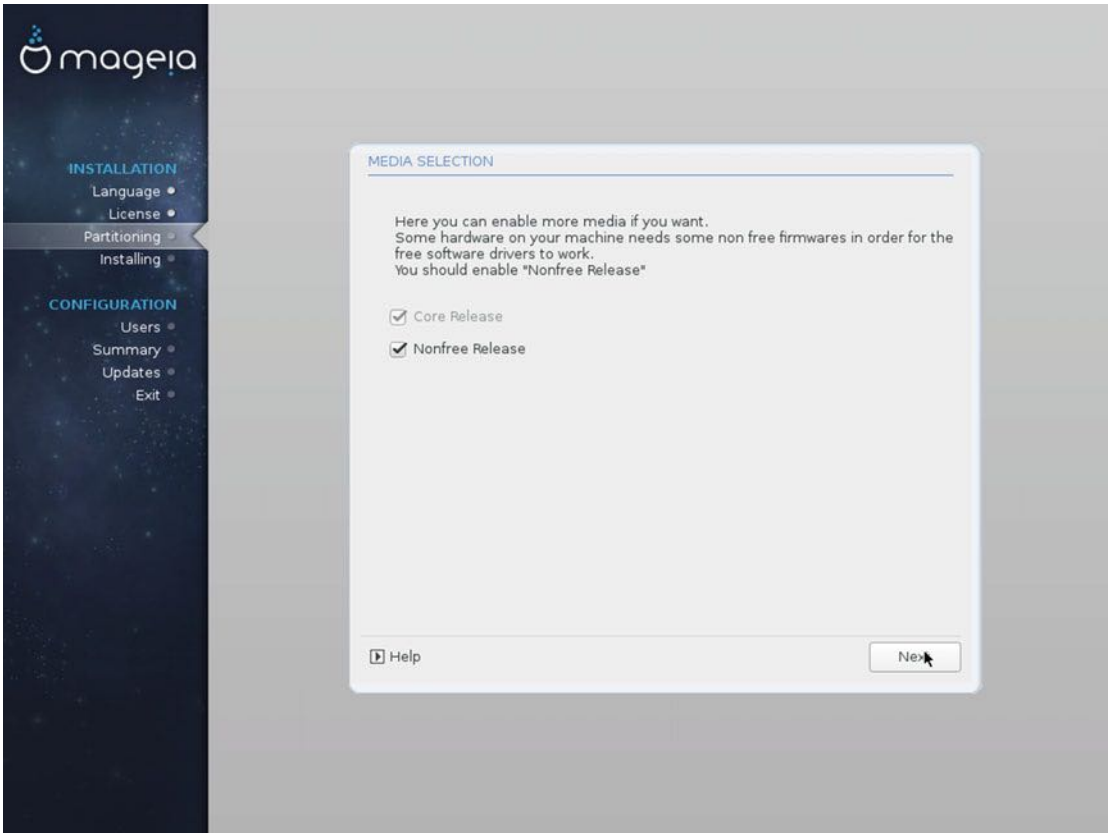


Figure 9-15. *Select local repositories*

Next, you get to choose which desktop environment (Figure 9-16) you want to use in your Linux installation. You can opt for one of the major ones, KDE or Gnome or another alternative (Cinnamon, LXDE, LxQT, Enlightenment, MATE, and XFCE). The default DE is KDE because it was the traditional DE of Mandriva. So let's choose it and continue the installation process. It copies the system files to the disk, as shown in Figure 9-17.

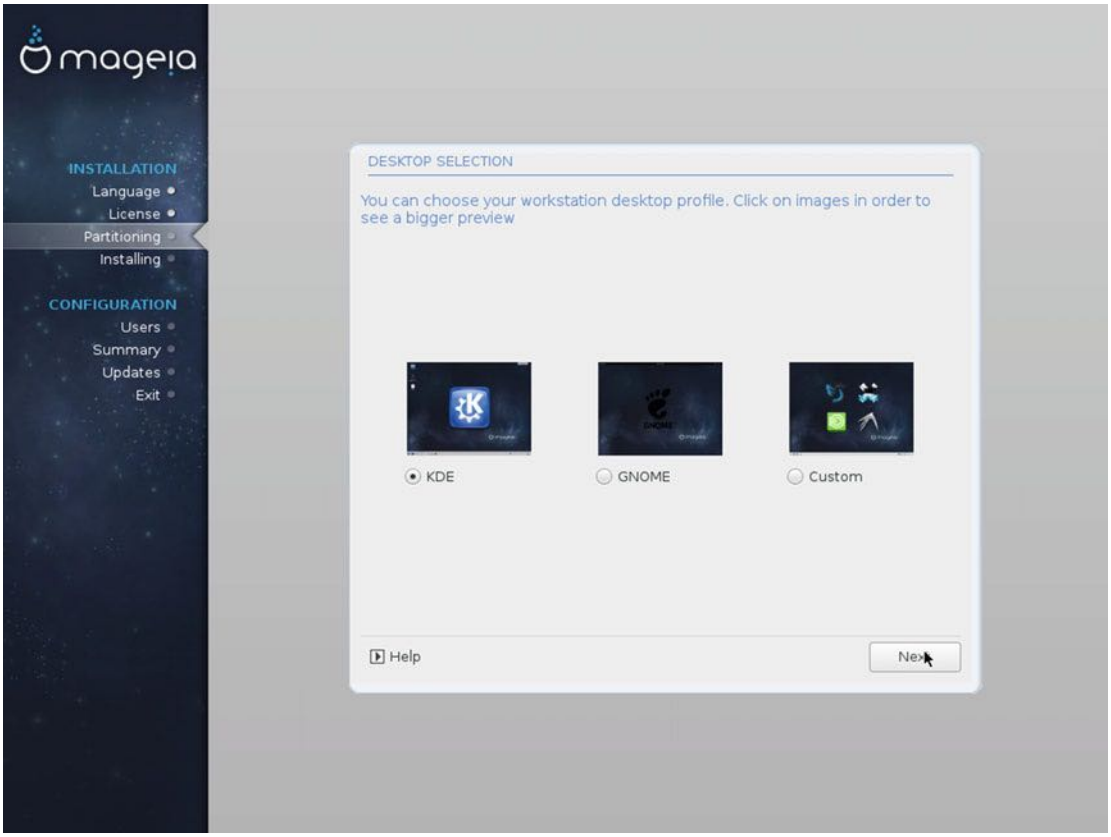


Figure 9-16. *Select the desktop environment*

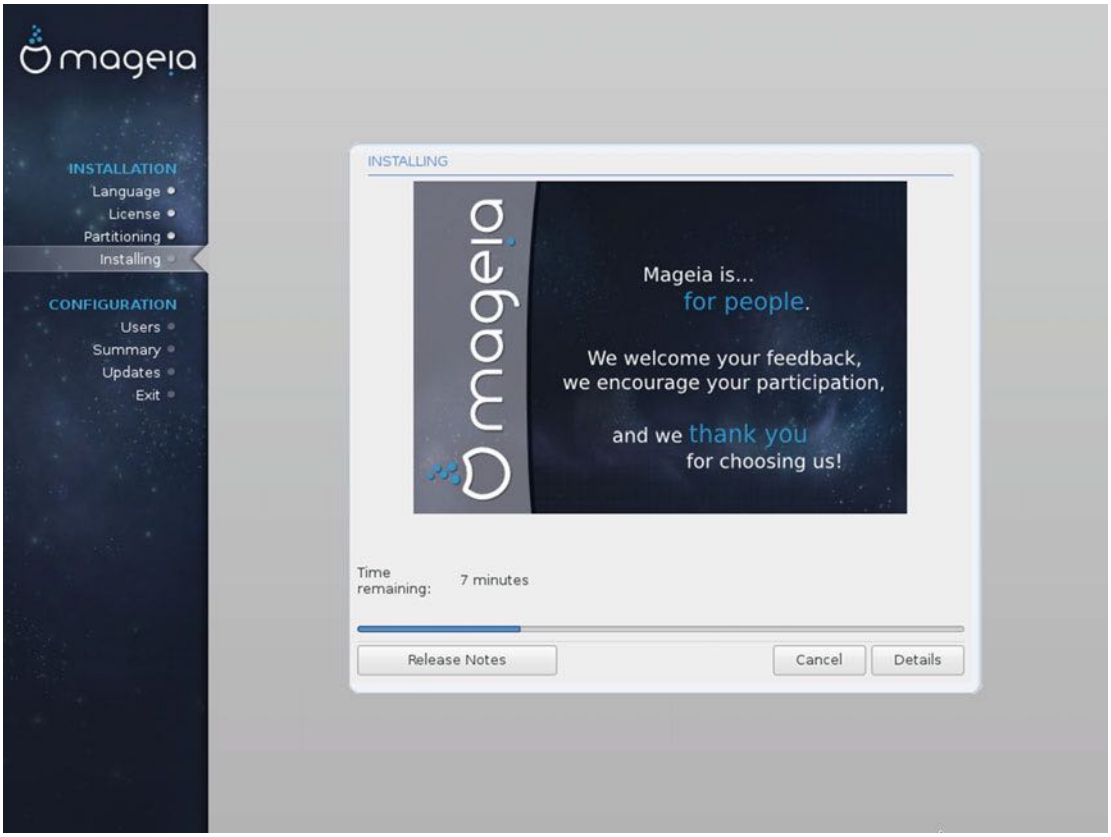


Figure 9-17. *Copy the files to the disk*

When the Mageia installer finishes the task, you must introduce the root and user information. There's nothing here that you haven't seen before. As it should be, Mageia will show you warning that the password couldn't resist a basic attack, which implies that it is not a strong password (Figure 9-18).

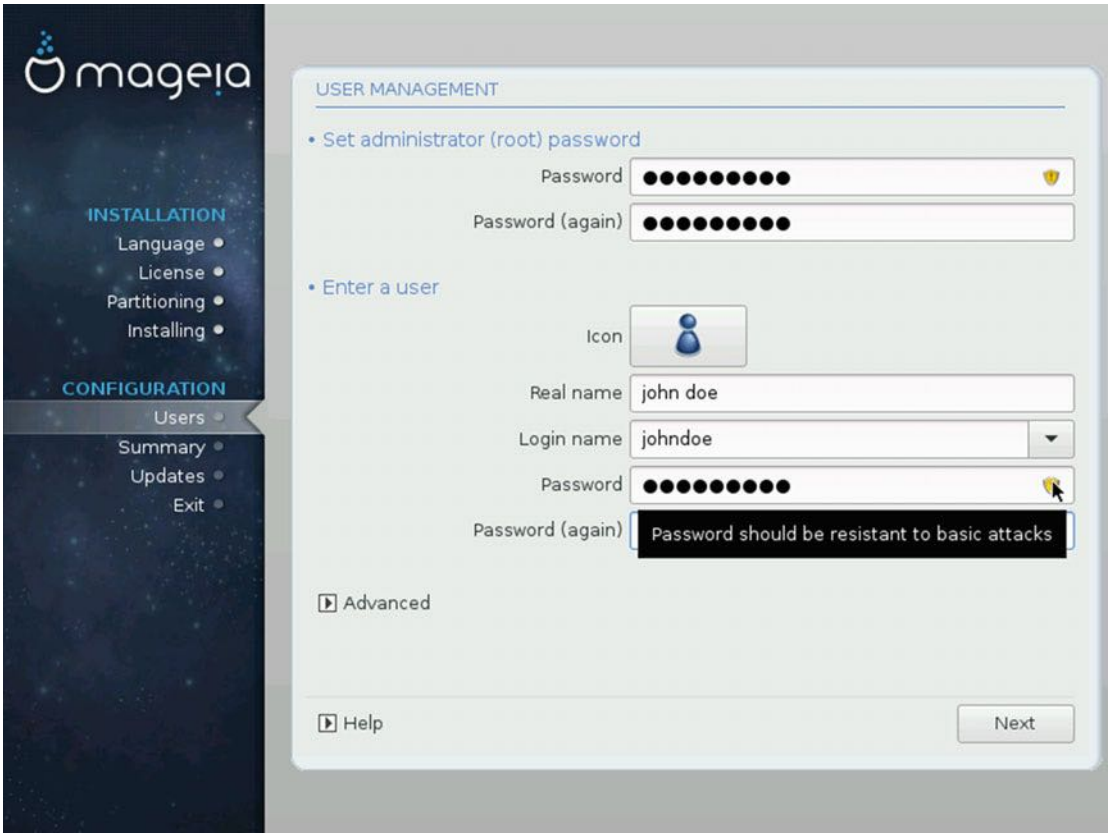


Figure 9-18. *The user information*

The next is an odd request: the installer tool asks you to choose a monitor from a set of options (Figure 9-19). Most distros detect this setting automatically, so it's rare to see this request these days. I suspect that it is related to the installation on a virtual machine; it is possible that installing this distro on a real system would skip this step. Either way, if you see this screen, select a plug'n play one (or a generic one) if you don't know exactly which monitor you have.

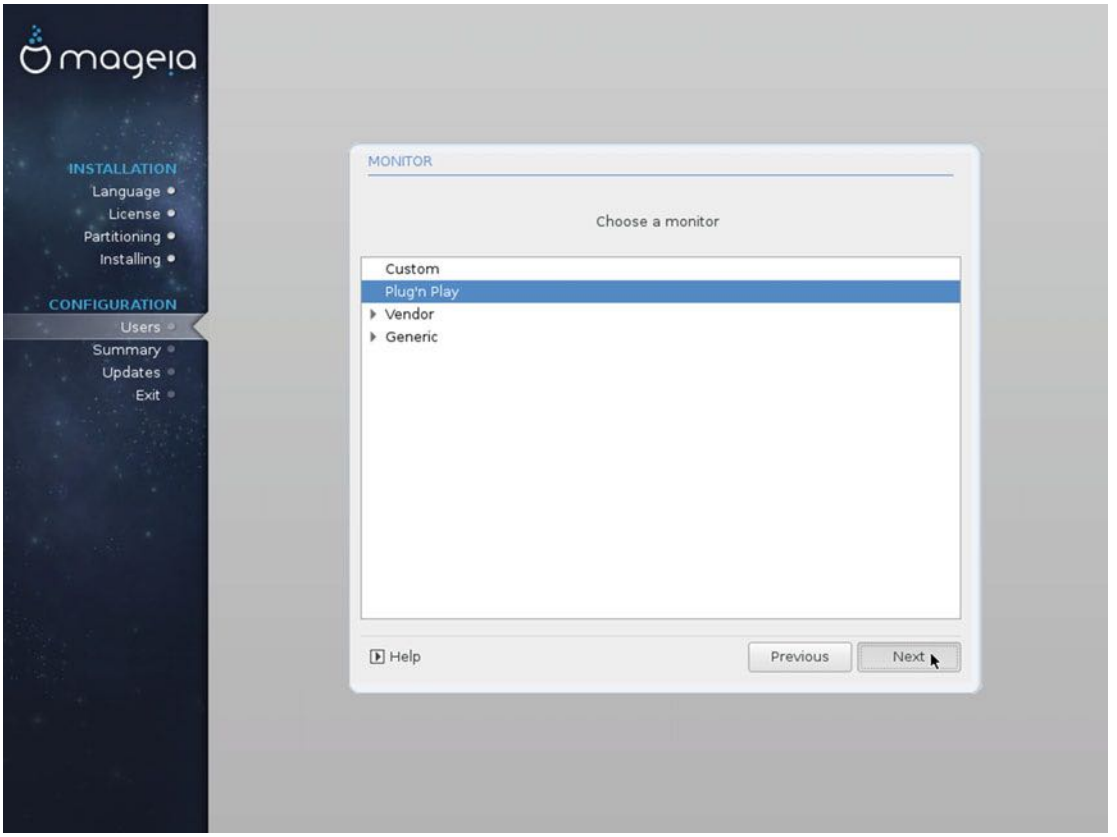


Figure 9-19. *Monitor selection*

After that, the installer will show you a summary of the installation, with details about the system and the settings; you can make corrections here (Figure 9-20). The summary is divided in four sections: System, Hardware, Network & Internet, and Security. There are some advanced options that only can be changed from here (of course you can always change the system configuration after the installation). If you don't know or understand each option, leave it as is.

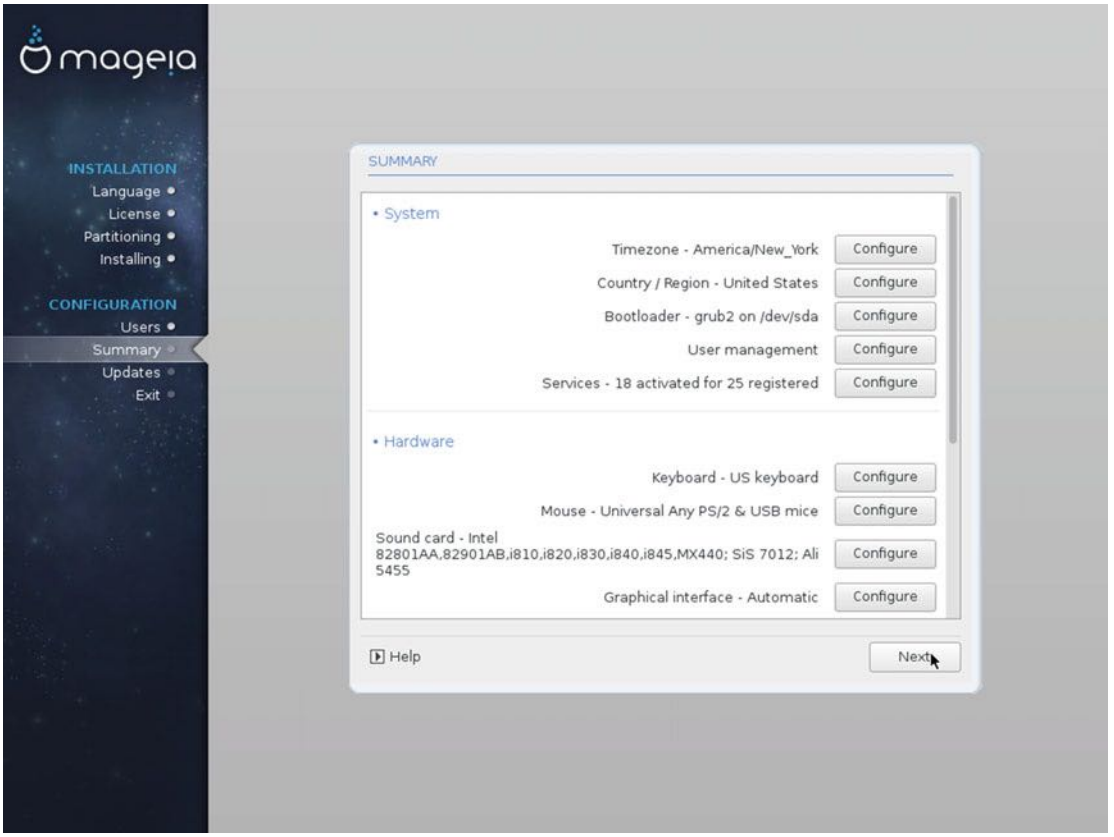


Figure 9-20. *The installation summary*

The following step is to decide if you want to use the online repositories to get any possible package updates that could exist there (Figure 9-21).

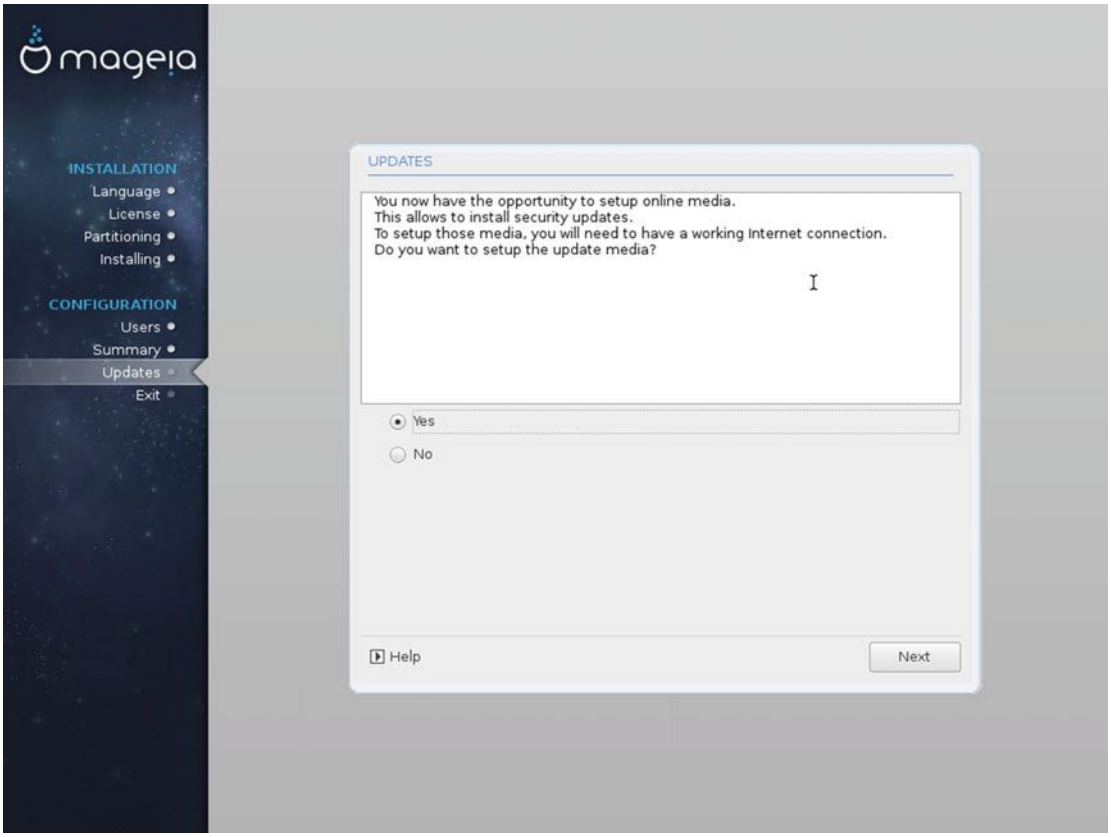


Figure 9-21. *Use the online repositories to install the last security updates*

If you elected to use the online repositories, it will start downloading all of the security updates. After downloading all of the security updates, a screen asks you to confirm that you wish to download all of the updated packages (Figure 9-22).

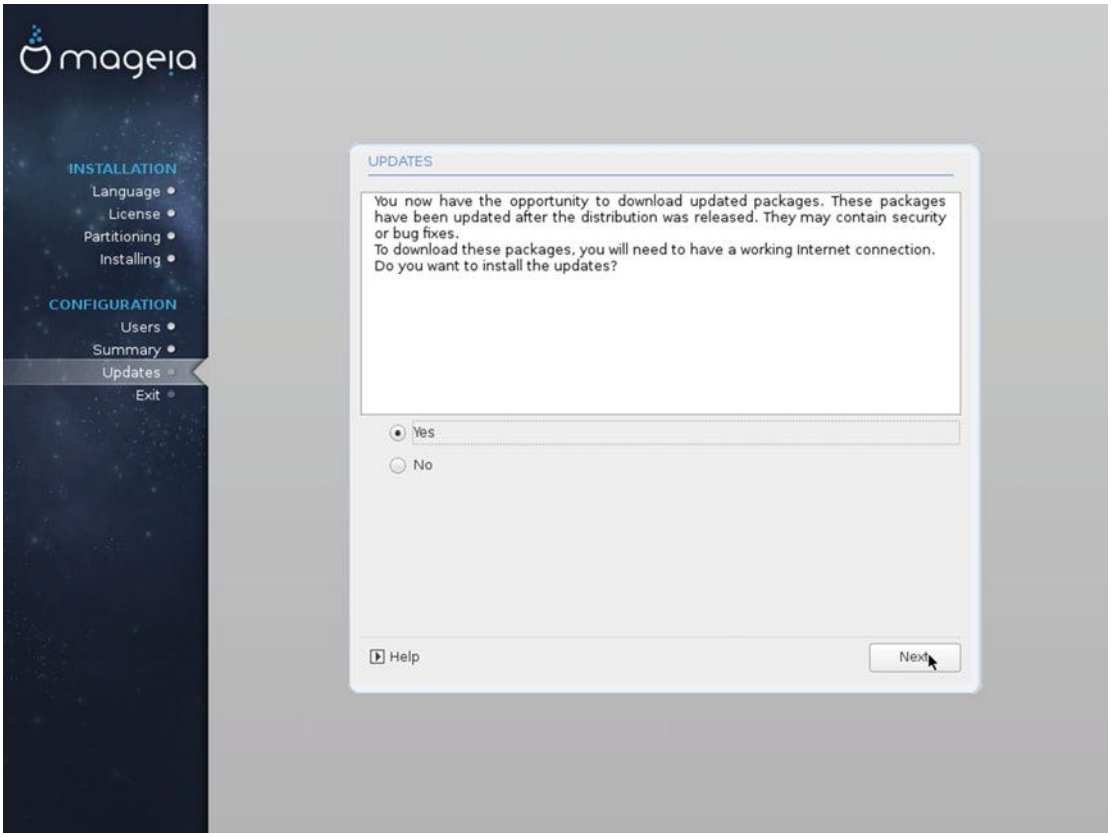


Figure 9-22. *Confirmation screen to install all of the packages that have an update*

Some pop-up dialogs will appear, asking for your permission to install those packages. If you give your permission, the installation ends when all of the packages are installed on the disk drive. You know that the installation is finished when you see the screen shown in Figure 9-23.

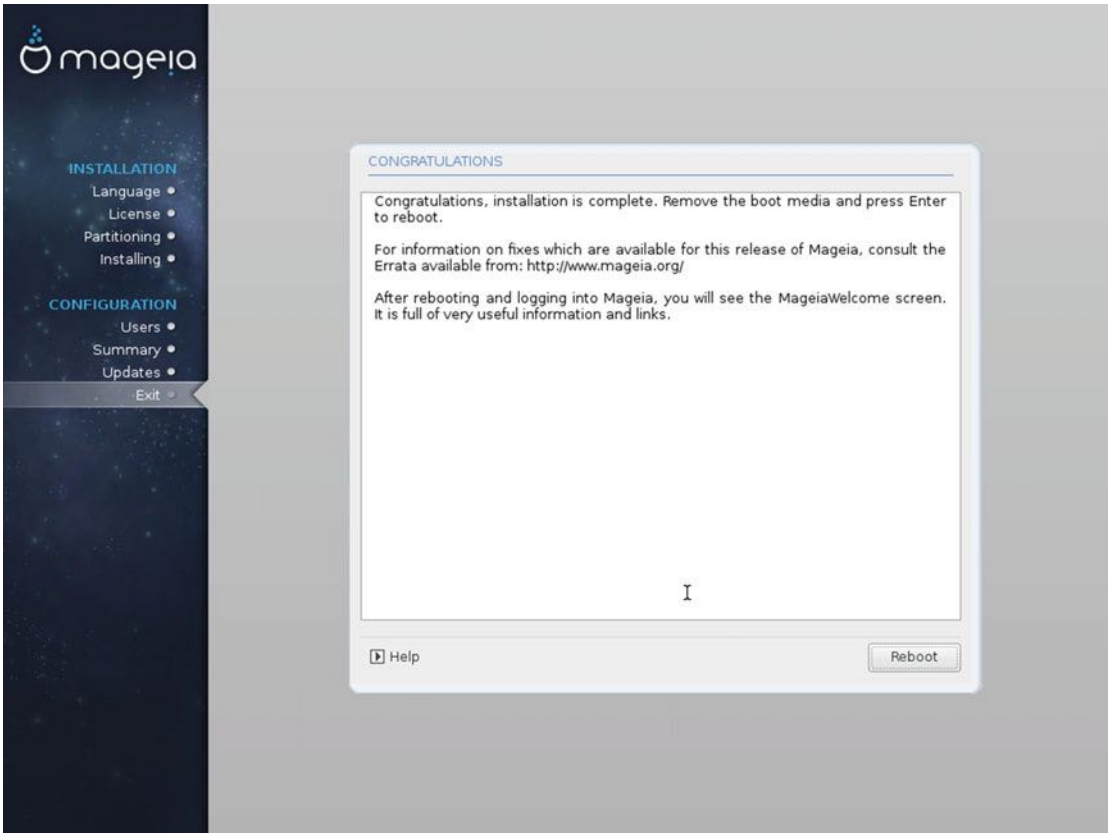


Figure 9-23. *It's done. You have installed Mageia*

Now is the time to restart your system and enjoy your Mageia Linux for the first time. The first screen that you see when you boot your system is the last difference that you will see between the traditional BIOS and the modern UEFI. They are very similar (both have the boot loader Grub) but there are subtle differences in how they are presented. Figure 9-24 shows the BIOS Grub and Figure 9-25 shows the UEFI Grub.

Figure 9-24. *The Grub screen in a BIOS system*

Figure 9-25. *The Grub screen in an UEFI system*

The login screen is your welcome to your new Mageia distro. Enter your password and go to the desktop. This login screen has an older design (Figure 9-26) because it still uses KDE 4.

Figure 9-26. *The login screen for KDE*

The desktop is typical of KDE 4 with a Mageia custom background (Figure 9-27). After a few seconds, a window appears in the center of the screen welcoming you to the distribution (Figure 9-28); you saw something similar in other distros.

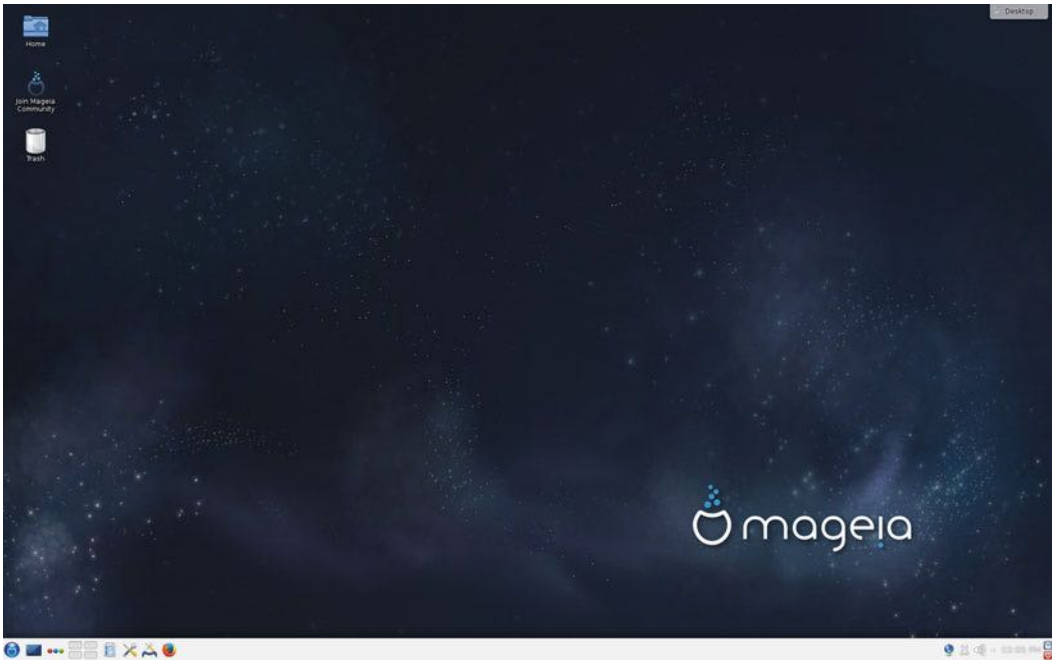


Figure 9-27. *The KDE Mageia desktop*

Figure 9-28. *The welcome window*

Like OpenSUSE, a distro very similar to Mageia, it detects that is being installed inside a VirtualBox machine and shows you a warning to update the Guest Additions to the last version (Figure 9-29). This is a nice touch.

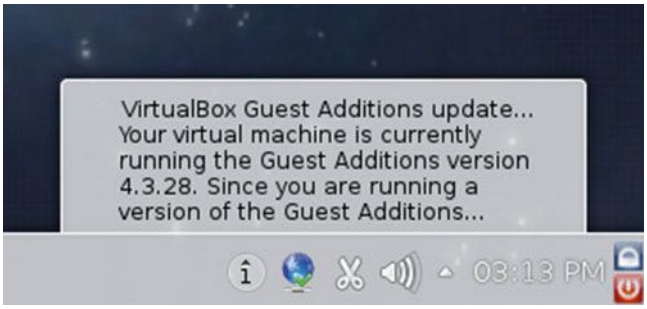


Figure 9-29. *Mageia detects VirtualBox*

As you can see, there are not many differences in installing the distro in a BIOS system versus a UEFI system, and this is why I chose the traditional BIOS system to install all the previous distros. In some distros, these differences are bigger, but these are usually the advanced ones.

Maintenance

The maintenance tasks in Mageia are done in a similar way to OpenSUSE, and you can do almost all of them from GUI interfaces without needing to use the command line. The exception is the distro upgrade.

Updating and Managing Apps

The obvious path to follow in Mageia is to go to the Control Center for essential administration tasks. This is also the ideal way to deal with upgrading and managing apps (Figure 9-30).

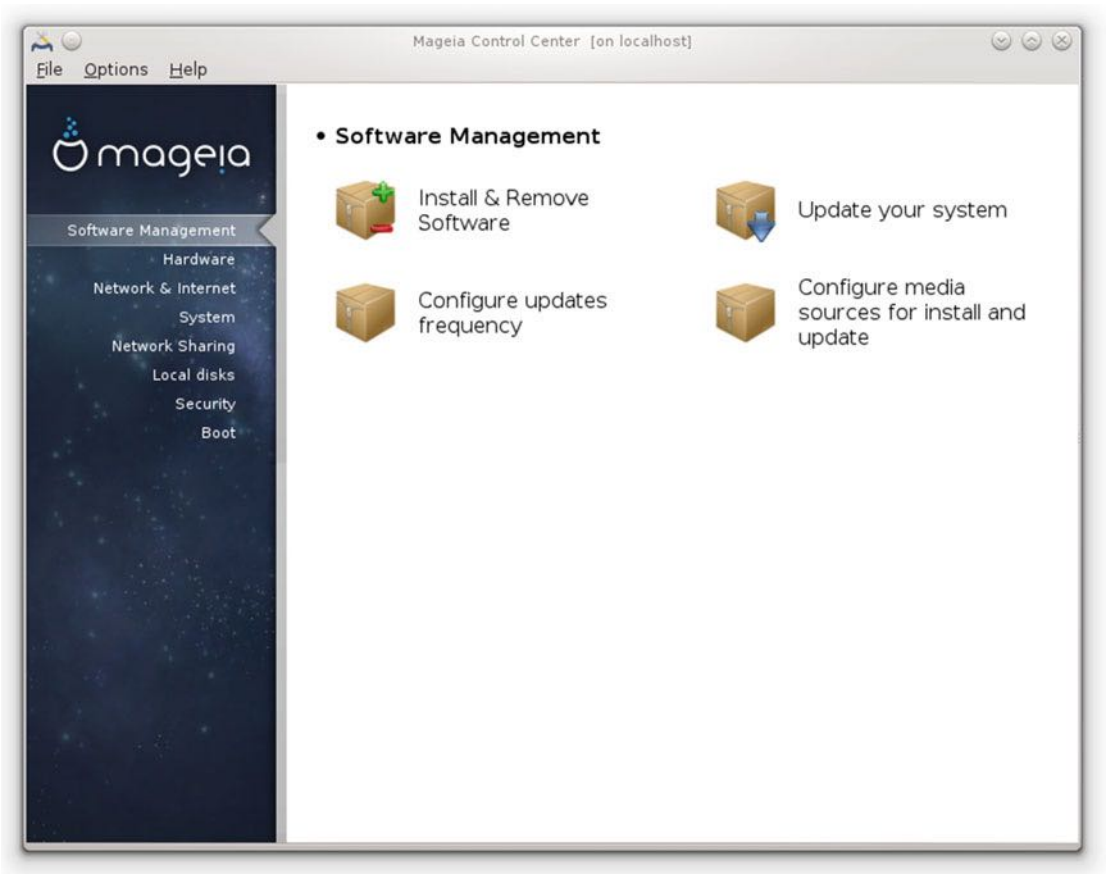


Figure 9-30. *The Software Management section of the Mageia Control Center*

From there you can manage your apps or update them. Both options will open a GUI program (which you can call directly if you wish), rpmdrake and drakrpm-update, respectively (Figure 9-31). These tools are only a front end to the urpmi tool, and you have the option to use it from the console instead, which many users prefer.

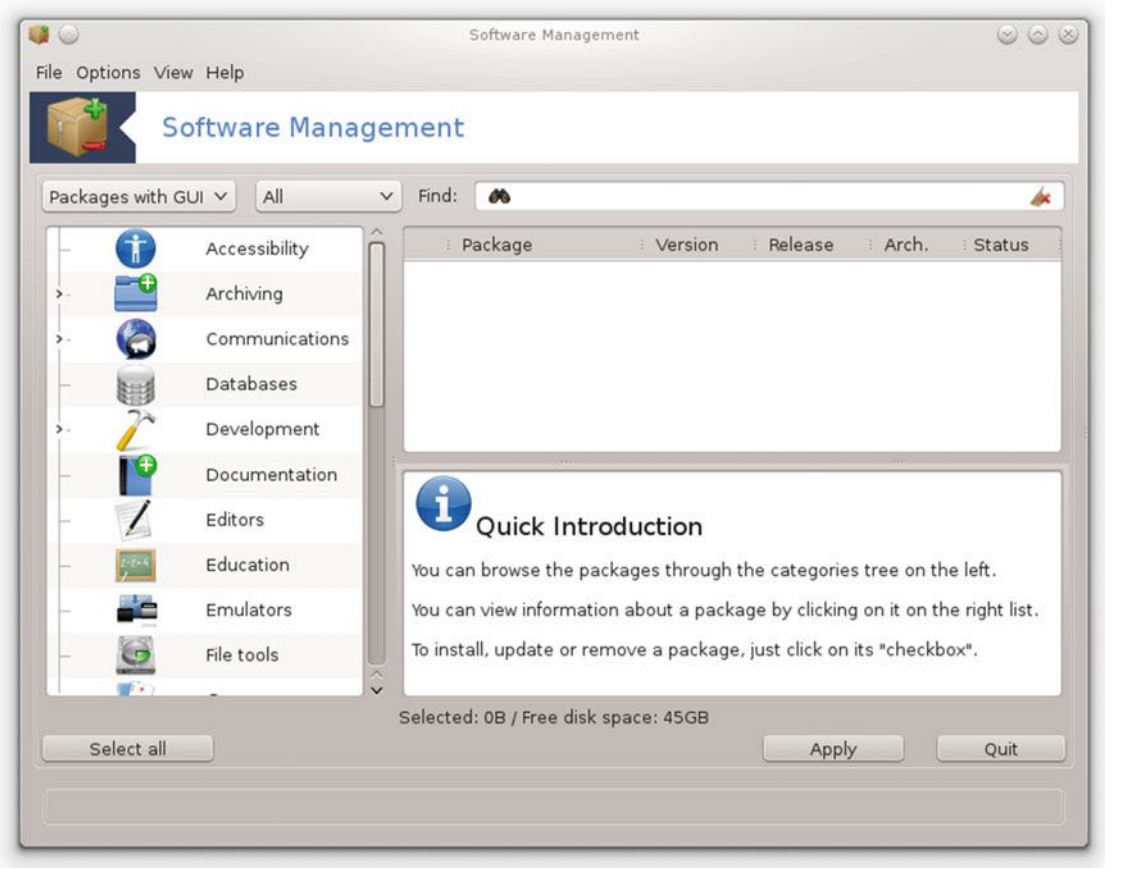


Figure 9-31. *The Software Management tool (rmpdrake)*

As with other distros, you will be warned when new updates are available and you can use the same dialog to perform the task from here.

Upgrading

Upgrading is always one of the most sensitive tasks to perform in a distribution; some distros don't have a proper way to do it, and others have an intricate way to do it. Mageia has two options:

- Follow the instructions that will appear when a new release is available. The packages will be fetched from the Internet repositories.
- Use the ISO image of the new release and select the upgrade option from the installation program. You can use the online repositories to update all the packages not included by default in the ISO image (and this is recommended). You can take a look at the upgrade option in the installation menu in Figure 9-32.

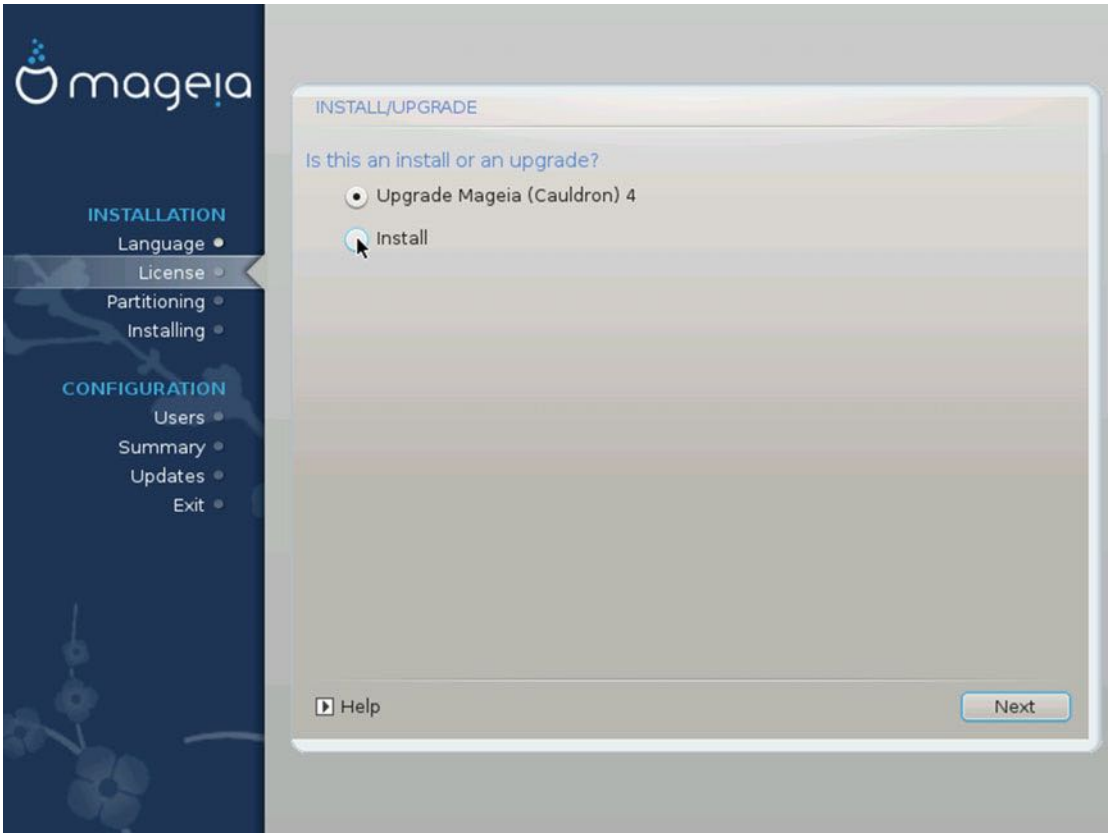


Figure 9-32. *An upgrade option is available on the installation program*

The non-obvious way to do this is to use the command line and the urpmi tool to upgrade the distro. This way is best suited for advanced users, but it is also the most reliable way to upgrade the distro.

Pros and Cons

The following is a list of some things that I personally see as pros and cons of the Mageia distribution. There's always room for discussion in this matter, but I've tried to be as objective as possible.

Pros

- The Mageia Control Center makes it easier to tweak and admin the distro.
- Mageia uses the rpm package format.
- It has a regular release scheme.
- It's a community-driven distro.
- It's a stable distro, but it's a bit outdated.

- It has a great legacy and it seems to continue the tradition.
- It's an original distro, not a derivative (although Mandrake was a fork of Red Hat).
- It offers reasonable security and it's easy to harden it a bit more.

Cons

- It only has an official desktop version.
- Fewer packages are available from the official repositories than with other distros.
- It is not a free software-only distro for purists.
- It doesn't take enough care of aesthetics.
- It has no commercial support.
- Only two architectures are supported.
- Although it's a very popular distro (mostly in Europe), its community is not very big compared to other great distros.

Summary

In this chapter, you saw the perfect example of how free software is a fundamental aspect of the survival of Linux distros. Mageia was an abandoned project that was rescued by a community. It started life as Mandrake, then Mandriva, and finally resurfaced as Mageia. And the community keeps improving the distro, with an eye on the past to maintain its great legacy and the other eye on the future to improve itself and the future of Linux.

In the next chapter, I analyze a distro focused in ease of use and aesthetics, elementary OS.

CHAPTER 10

elementary OS

elementary OS is the newest distro that I analyze in this book. It is only five years old and has only delivered three major releases to date. It also has an unusual origin, a very small community behind, and it is an Ubuntu derivative. Despite all these odds against it, elementary OS has become a very popular distro and the best known representative of a new generation of Linux distros (together with Solus, Cub, deepin, and others) that want to establish a new level of friendliness and design in Linux, as Ubuntu did years ago.

These distros do not pretend to be one-for-all distros. On the contrary, elementary OS and the others focus on the user who has simple needs, like browsing the net and managing a few documents, videos, and photos. Minimalism, great design, and ease of use are the pillars of this new generation of Linux distros. (And yes, in case you are wondering, "elementary" always start with a lower case "e," even at the beginning of a sentence.)

History

In the beginning, elementary OS was not a Linux distribution. In fact, it was not even a software project, but a design one. elementary OS started as a Gnome 2 icon set and then a Gnome theme that was intended to be used with Ubuntu. Daniel Foré created it in 2009, and the theme quickly become very popular. Foré even ended up working for Canonical (the corporation behind Ubuntu) as a designer for almost two years, thanks to this notoriety. Before leaving Ubuntu, he began a new project that could fulfil his vision of what a Linux distro (actually an OS) should be: the elementary project.

The elementary project started first as a mock-up/concept of an ideal desktop environment. Then some developers developed a few things, like a modified Nautilus (the Gnome 2 file manager), and Dexter and Postler (a contact manager and e-mail client, respectively). Then they built its own desktop environment, Pantheon, and finally its own distribution, elementary OS.

The first release of elementary OS, based on Ubuntu 10.10, was Jupiter in March of 2011. Luna was released in August of 2013 and Freya was released in April of 2015. Jupiter was released with a customized Gnome desktop; Luna was the first release to have Pantheon as its desktop environment.

Philosophy

The elementary OS philosophy is contained in its own name. Simple, easy, minimalistic, beautiful, basic, and uncomplicated are the words that define the goals and vision of this project. The developers say that they want to be the beautiful and intuitive alternative to MS Windows and OS X. A more realistic description is that they are an alternative to Chrome OS or the different projects that are going to bring Android to the PC (like Remix OS). In a few words, if you ever thought that a tablet could cover all of you needs and you want to ditch your laptop/desktop in favor of one, this is a good clue about what you can expect from elementary OS (I'm not saying that elementary OS works on tablet, because it doesn't; I'm talking about the kind of

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_10

tasks that you would expect to perform on a tablet). Being a Linux distro you can always go beyond that basic functionality, of course, but at the cost of breaking the simplicity/design consistency across all of the applications.

In order to do maintain design consistent, the developers use Ubuntu as a basis and then add their own desktop environment plus a few minimalistic apps to cover the simple needs of a big group of users. The out-of-the-box apps include a mail client, a calendar, multimedia management (photos, music, and videos), a text editor, a way to browse the Internet, and some system management apps (files, terminal, and settings). Basically, they replace the usual DE and apps with a very simple and friendly alternative. Again, a similar alternative is Chrome OS, which uses the cloud and the simpler Google apps to do tasks; however, elementary OS uses a traditional local storage and local apps approach.

I think that elementary OS may offer very interesting things in the future, even if it just works as an inspiration for other projects. But, at the same time, it has some obstacles to address. As I see it, it only can grow in one direction, that of adding more built-in apps, because by its own definition, if it adds more features to the current ones, it will break its own principles.

Distro Selection Criteria

Let's see how this particular distro fares on those decision-making points from Chapter 2.

Purpose and Environment

elementary OS focuses only and exclusively on the desktop; it only has a desktop version.

Support

The community behind elementary OS is very small. There are currently about 30 regular developers. So you cannot expect the same level of support as with a big corporate or community distro. However, elementary OS has a reasonably good level of support from various sources:

- **Documentation**: `https://elementary.io/docs/`
- **Q&A:** `https://elementaryos.stackexchange.com/`
- **Forum**: `https://elementaryforums.com/index.php`
- **IRC**: `#elementary` on `irc.freenode.net`

User Friendliness

The intentions of the elementary OS project are very clear on this topic.

> *Our primary motivation for developing elementary OS has always been to build the absolutely best computing experience we possibly can.*
>
> —Daniel Fore, in an interview for LME Linux, January 2014

Have they succeeded? Well, many people feel that elementary OS is the best Linux distro available; others feel that it is on its way to achieving greatness; still others feel it's merely a bad copy of OS X; and still others think there are more important things than design and simplicity. I classify computer users into two

big groups: content creators and content consumers. For the first group, clearly elementary OS has little to offer, even if some of them like the design and vision a lot. The majority of elementary OS fans are in the second group, where many of them will have all they need with only the built-in applications and others will only have to install a few more.

It is precisely in that last point where the major weakness of elementary OS appears. If you only need to use the built-in apps, then you have an install-ready OS that is easy to use and comes with what you need. It's very basic and amicable. But once you install more apps, you may maintain the Pantheon theme, but other than that, the consistency is gone. You may even experience some basic problems due to poor integration with the desktop environment. In fact, I saw that with an app like Blender and another KDE app.

But is elementary OS a very friendly Linux distro? Sure, if you use it as intended and you do not need something more from it. The reality is that it is very hard in a small project like this to get consistency because of the nature of open and free software and the lack of universal guidelines for design, GUI development, and user experience. It's difficult even for the large Linux distros In fact, even Apple, with its closed ecosystem, has problems with third party apps that do not always respect their guidelines. Likewise, Android has the same problem with the small adoption of Material Design by app designers.

Having said that, it is quite remarkable how a small group of people achieved such a level of ease-of-use and excellence with so few resources. Imagine what they could do with the resources of a corporation like Canonical.

Stability

After the second release, Luna, elementary OS switched to being based on the LTS releases of Ubuntu. Thus, it follows a regular release scheme, but it is not a rigid schedule; the new releases appear when they are ready. The release number scheme began with a 0.1, and each major release will add another 0.1 to that number, like so:

- **0.1 Jupiter**, based on Ubuntu 10.10
- **0.2 Luna**, based on Ubuntu 12.04 LTS
- **0.3 Freya**, based on Ubuntu 14.04 LTS

0.4 Loki (based on Ubuntu 16.04 LTS) is expected to be the next elementary OS release. Two minor releases of Freya, 3.1 and 3.2, fixed some errors and introduced new features and updates.

So, because it's based on the LTS versions of Ubuntu, elementary OS should be a little outdated but rock solid, right? Actually, there are always minor errors and glitches, due to the small team behind elementary OS, and this makes the distribution a little unstable sometimes, and even a bit annoying occasionally.

Hardware Support

The hardware support is essentially the same as the underlying Ubuntu version that it is using at the moment, so it is good support.

Aesthetics

Since the goal of this project is minimalistic and beautiful design, the aesthetic of every little thing is very well polished. The use of their own built-in apps helps to ensure consistency of design across the environment (but not if you install third party applications). As soon as you boot up the distro, you notice the clean design.

Some detractors say that elementary OS is merely a copy of the OS X design, and the inspiration is undeniable. The color palette, the look of the icons, and the dock all reference OS X (Figure 10-1). But this is not anything new; there are and were a lot of themes, icons, and distros that wanted to replicate the look

and feel of other OSes like OS X or Windows. In fact, you can tweak the desktop environment of almost any distro until you get close to looking like one of those OSes. Regardless, you cannot deny the effort put into the distro to get a professional-level design and the further steps of developing new apps to get a cohesive design.

Figure 10-1. *The elementary OS dock (Plank)*

Is the elementary OS design the best and the most beautiful? Well, I don't want to introduce an opinion here. I want to show you how the design of an essential tool, a file manager, can vary between distributions. You can judge for yourself how important design is to you and which distros offer the most eye-candy. To do this, look at three distros that handle aesthetics (Figures 10-2, 10-3, and 10-4) and another distro that doesn't (Figure 10-5). You can see how some of them cast shadows and others don't; that's a desktop environment feature, but it's also a design decision. It's obvious that the first three offer better designs, taste aside, than the last one. But between the first three, which one is the best? It's quite subjective.

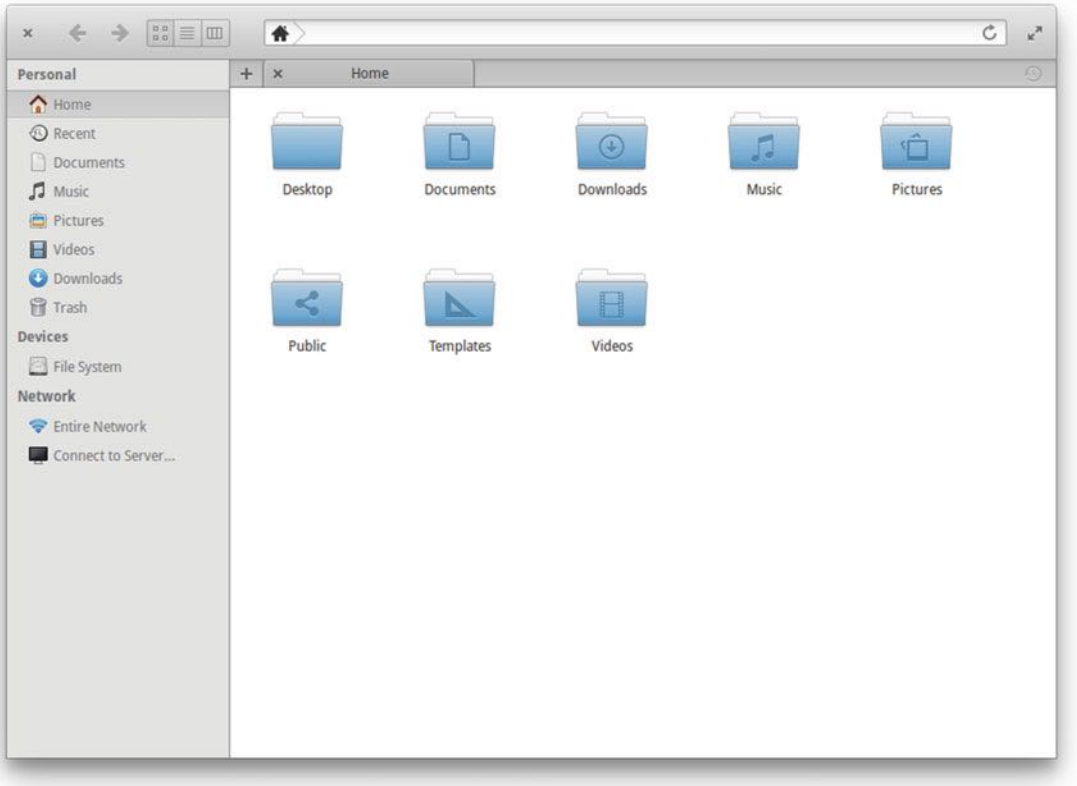


Figure 10-2. *The elementary OS file manager on a Pantheon DE*

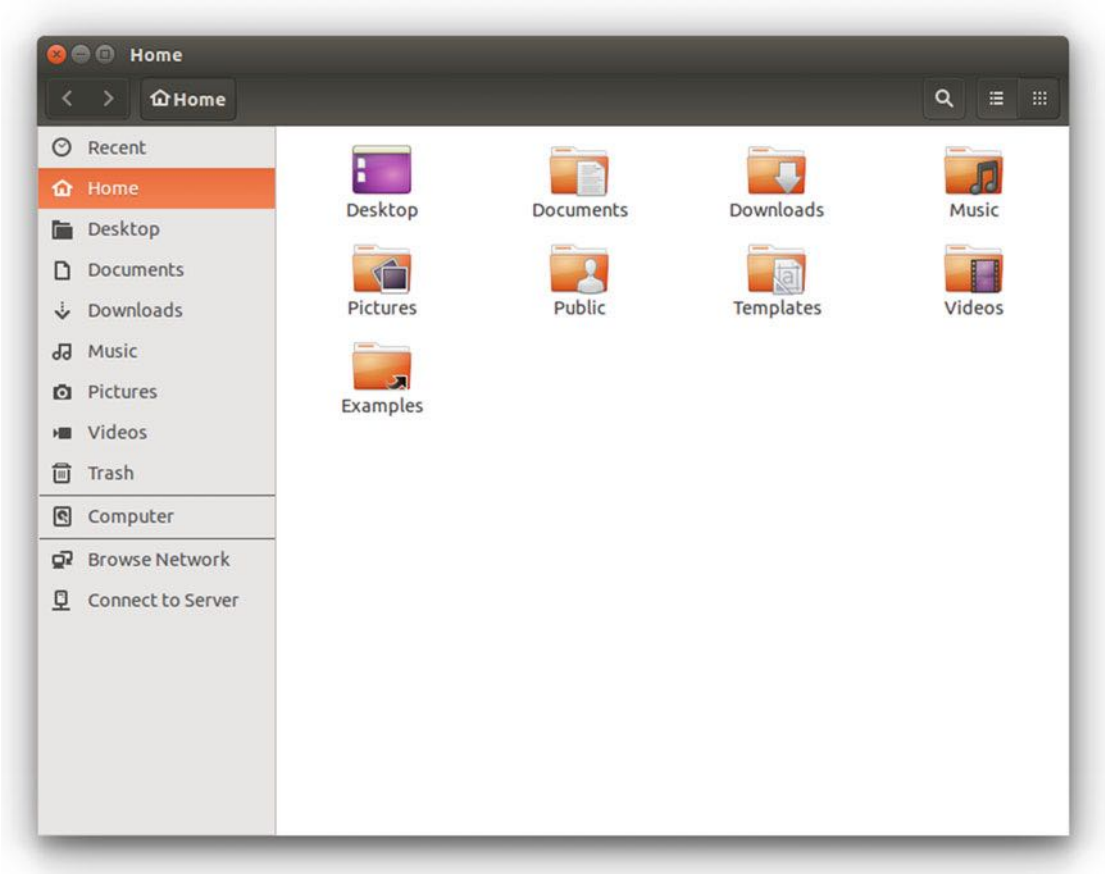


Figure 10-3. *The Ubuntu file manager on a Unity DE*

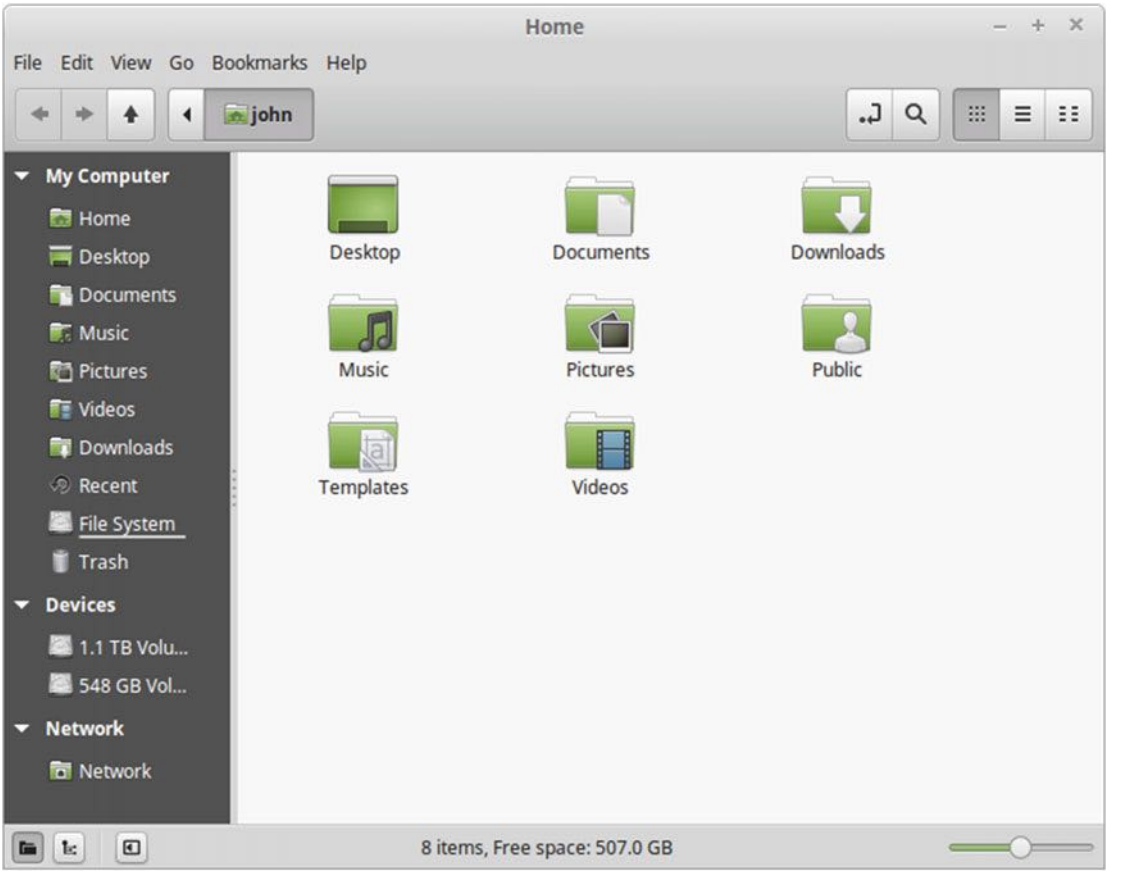


Figure 10-4. *The Linux Mint file manager on a Cinnamon DE*

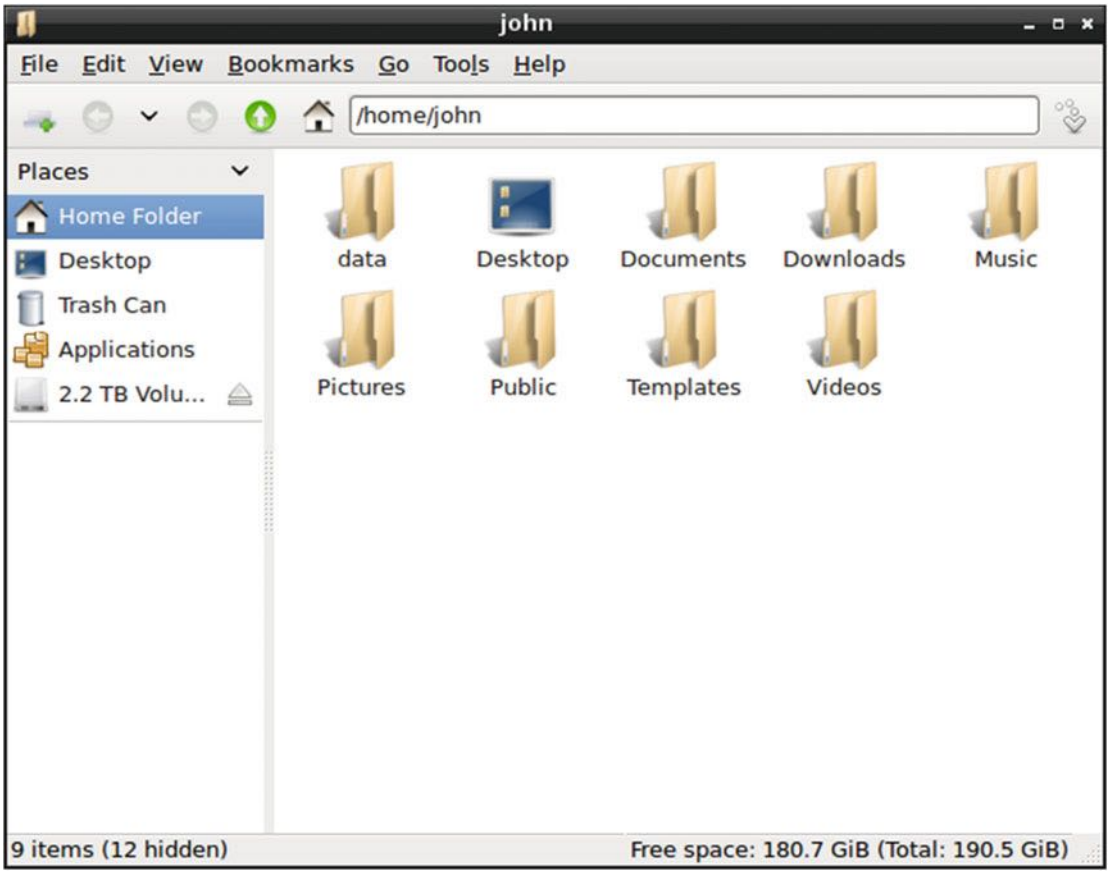


Figure 10-5. *The Debian file manager on a LXDE DE*

Desktop Environment

elementary OS uses its own desktop environment, Pantheon. It uses the GTK 3 toolkit (originally from Gnome) and is programmed in Vala and C. Although this DE was developed by the elementary OS team, you can install it unofficially on other distros, like Arch Linux or Ubuntu.

Init System

Since elementary OS is based on Ubuntu, it uses the same init system, but because the current Freya release is based on an old Ubuntu version, 14.04, it is still using the sysv init system. This will change after the next release, Loki, which will be based on Ubuntu 16.04 LTS and thus will use systemd.

Package Management System

Because it's based on Ubuntu, elementary OS uses not only the same package management system, but also the Ubuntu repositories and software management apps. In fact, elementary OS's software is obtained and updated through PPAs (Personal Package Archives) hosted in Launchpad (from Canonical).

Architecture

elementary OS only supports the Intel and AMD 32/64-bit architectures.

Security/Anonymity

In terms of security, elementary OS is based on Ubuntu, so it should be equal to Ubuntu. As for anonymity, the elementary OS project claims that it doesn't make advertising deals or collect any sensitive personal data.

Principles and Ethics

Also, the principals and ethics are similar to Ubuntu, since elementary OS uses Ubuntu's repositories.

Live CD

The ISO image of elementary OS also works as a Live DVD.

Professional Certification

Obviously, in a very small project like this, there is no professional certification available.

Installation

Installing elementary OS is very easy, and since it is based on Ubuntu, like Mint, it uses the same installer. Since I already covered in detail the installation of Ubuntu and Mint, please refer to those chapters. Here I will just mention the main differences you will encounter.

As always, you should go to the elementary OS web site, `https://elementary.io`, to download the ISO image. In Figure 10-6, you can see that it seems like you have to pay to download it; in fact the button says "Purchase elementary OS." This is merely a way to suggest that you make a donation to sustain the project, but you can enter a $0 quantity to download it. This was a controversial step at the time, but the reality is that Ubuntu does something similar without too much noise. So select a predefined quantity or introduce a custom one and click the "Purchase elementary OS" button. The next window asks you to choose the version you want to download, 32- or 64-bit (Figure 10-7). I recommend choosing the 64-bit version.

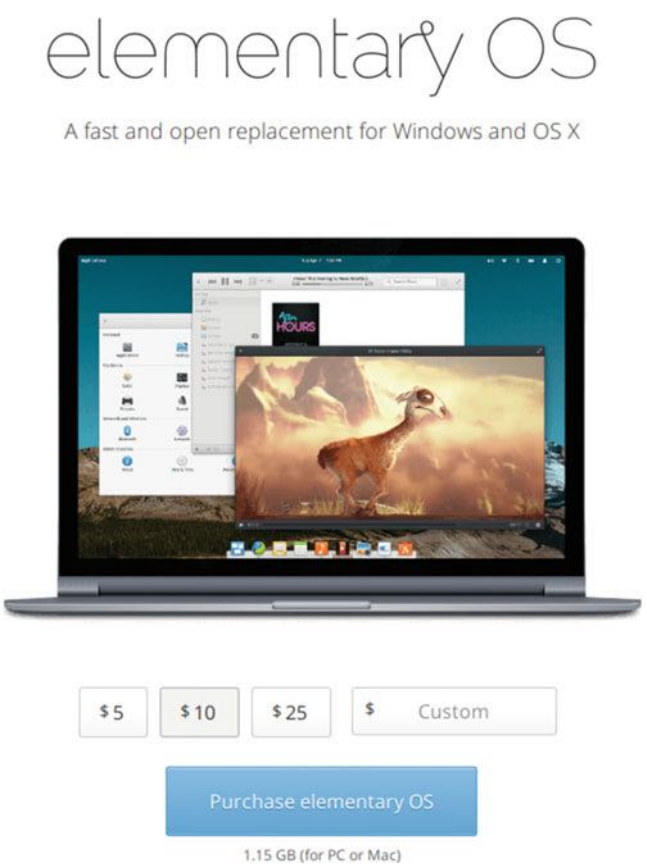


Figure 10-6. *The elementary OS site where you can download the OS*

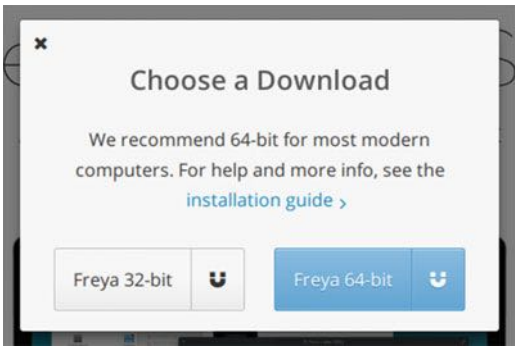


Figure 10-7. *Choosing the elementary OS version to download*

Then you have to install the distro, which is exactly like Ubuntu but in a grey color palette. When you first boot the system, and after you log in, you will see the desktop shown in Figure 10-8. And those are the only differences that you will experience when installing elementary OS.

Figure 10-8. *The elementary OS desktop*

You can learn more about the elementary OS installation at `https://elementary.io/docs/installation`.

Maintenance

Maintaining elementary OS is similar to maintaining Ubuntu, for obvious reasons.

Updating and Managing Apps

Currently elementary OS uses the same application as Ubuntu, the Software Center, to perform updating tasks. A new application for this same purpose, AppCenter, is being developed, and it will be the default probably in the next release, 0.4 Loki.

Upgrading

There is no predefined or easy way to upgrade elementary OS. Even the developers suggest you make a clean install.

Pros and Cons

The following are some of the things that I personally see as pros and cons of the elementary OS distribution. There is always room for discussion in this matter, but I'll do my best to be as objective as possible.

Pros

- It is one of the easiest to use.
- It has an exquisite, modern, and consistent design.
- Because Ubuntu is its base system, you can benefit from its advantages.
- The built-in apps are simple, intuitive, lightweight, and fast.
- It is suitable for users with few requirements or computer experience.
- It is a good start pointing for users coming from other OSes.

Cons

- Technically, it is Ubuntu with a different desktop environment and other default apps.
- Due to the small team behind it, it is not rare to find several errors every time a new release is launched, which are usually resolved via small updates.
- The built-in apps are too simple to perform any semiprofessional tasks.
- You cannot expect the same pure design and simplicity from third party apps.

Summary

In this chapter, you saw a new paradigm in Linux distros, a distro that could be attractive to those who choose with their eyes and those who want uncomplicated things. elementary OS is basically all about design and simplicity. You won't choose this distro for technical prowess. It covers a niche, and based on its popularity, it seems to do it pretty well.

In the next chapter, I talk about another distro that covers another niche, advanced users and tweakers who wants a very customizable system: Arch Linux.

CHAPTER 11

Arch Linux

Arch Linux is the first distro covered here that is targeted to advanced users and the first one that is a pure rolling release distro. It's also not user-friendly, but that hasn't stopped it from becoming very popular. Perhaps its success lies in its simplicity, because Arch only gives you a very basic foundation upon which you can build the Linux you need or want. The other key to its fame is its appeal to the type of user who always wants to have the most recent version of any software. But none of this would be possible without the support of the impressive documentation and the great community behind it.

History

In 2001, a Canadian programmer named Judd Vinet started developing a new Linux distro inspired by the simplicity of others like CRUX and BSD Unix. He liked the inherent elegance that came with that simplicity, but he strongly disliked the lack of package management in those options, so he also developed a new package manager, pacman, based on the same principles. He released the first version of this new distro, Arch Linux 0.1 (Homer), on March 11, 2002.[1]

Judd Vinet passed the role of main developer and leader of the distro to the American programmer Aaron Griffin in late 2007, who remains at the helm to this day.

Arch Linux's reception by the community would be gradually, growing steadily over the years. Even today it still has a modest install size compared to distributions like Ubuntu or Debian.

Philosophy

Minimalism, simplicity, and code elegance (the KISS Principle) were the core guidelines of the development of Arch Linux. The Arch Philosophy, commonly known as the Arch Way, is defined as follows:

- **Simplicity**: No unnecessary additions or modifications to the software. The packages are almost identical to their original developer's release of them; essential (and minimal) changes are made only when necessary to run the distro.
- **Modernity**: Offer the latest stable releases of software and the newest features available in Linux. Be a bleeding-edge distro.
- **Pragmatism**: Arch is a facts-and-needs–based distro, not an ideological one. This approach spans the development decisions to the license of the software; you are free to choose between free and proprietary software.

[1] www.archlinux.org/news/arch-linux-01-homer-released/

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_11

- **User centrality**: Arch Linux does not try to be everything to everyone; it tries to satisfy the needs of its contributors and community. It promotes a DIY (do it yourself) attitude.
- **Versatility**: Arch provides a very basic command-line system. You can install and set up whatever you want for whatever tasks and needs you have in mind.

■ **Note** The KISS (Keep it simple stupid) Principle states that any system works at best if simplicity is a key goal in its design. Unnecessary complexity is to be avoided. It is a widely used principle in engineering and computer science, and it is behind great achievements like the design of the UNIX OS, and therefore Linux too.

Distro Selection Criteria

Now that you know some history, let's see how Arch Linux rates on the selection criteria outlined in Chapter 2.

Purpose and Environment

Arch Linux is a general purpose distribution. In fact, it's hard to be more general than this distro. Since nothing is predefined, you can build the system you want, even a task-oriented one (which proves that Arch Linux is a general as it can be). It offers one version as one ISO image.

Support

As a community distro, the support is obviously less professional and extensive than in a company-backed one. Furthermore, its philosophy is one of expecting its users to solve their own problems. So you might think that the community support would be small and of poor quality. But here is the paradox, Arch Linux has one of the most open, friendly, and helpful communities of all distros, and it has perhaps the best documentation around (which is maintained by the community itself). It's a known fact that many users of other distros use the Arch Wiki to answer questions about their distros. Perhaps the reason for the quality of the documentation is because it's better to write down a topic on the wiki once than answer the same question several times in other channels. Everyone expects you to consult the docs first before asking any questions.

The channels you can use to get support from the Arch community are the following:

- **ArchWiki** (Documentation): `https://wiki.archlinux.org`
- **Forums**: `https://bbs.archlinux.org/`
- **Mailing lists**: `https://lists.archlinux.org//listinfo/`
- **IRC**: `#archlinux` at `irc.freenode.net`
  More at `https://wiki.archlinux.org/index.php/IRC_channels`

User Friendliness

As mentioned, the Arch Linux philosophy does not emphasize user friendliness. The typical Arch Linux user is one who has advanced knowledge or is willing to do it all by themselves by reading the documentation first (tinkerers).

I do not recommend this distro for beginners. However, many people are attracted by the rolling release scheme and are tired of upgrading their user-friendly distros. Thanks to the extraordinary quality of the documentation, they were tempted to test this distro, and a good part of them are still Arch Linux users. There is also a kind of user that wants to know more about Linux itself, and using a low-level distro like this one can teach you a lot. Of course, if you hate the command line, avoid this distro.

Stability

Arch Linux is as unstable as it can get. It follows a rolling release scheme, so you always have the latest version of the software, from the kernel to the rest of the packages. This way you never need to upgrade your distro; you install Arch Linux once and you only have to do regularly updates to keep current. Arch Linux updates its ISO image monthly, but you only need to make a fresh install if something goes terribly wrong.

You can adjust your level of stress and instability to a more reasonable level or just stay on the bleeding edge. You can do this by selecting which repositories you want to enable. If you enable the testing repositories, you will always have the latest and most unstable version; if you don't, you will have a reasonable stable system, given the context. For example, if you don't use the test repositories, you may experience a delay of a few months before you get the latest version of the kernel. Also, one of the advantages of the rolling release scheme is that if one of the releases has a bug, and that particular bug is solved in the next version, you have the solution available in your system almost at the very moment as it is released by its author. In a standard release scheme, it could take years to get the fix. If you want more stability, you can also install the `linux-lts` kernel, which is a more stable kernel that does not upgrade frequently.

So how reliable can an Arch Linux system be? Well, it depends. First, you should not use Arch Linux in a production server unless you really know what you are doing. Otherwise, there a good number of programmers, system administrators, and other IT professionals who use Arch Linux on their main computer without problems for years. It all depends on how you manage it and your level of knowledge. Let's use an example: me. I've used Arch Linux on several machines for years. The oldest is a desktop of four years, the same age as its Arch Linux installation. I haven't had to make a fresh install on any of them. But I use a minimal setup. I don't use a desktop environment, just a window manager (Awesome WM). I don't have a dedicated graphics card; I use the one integrated in the processor. I also disabled the testing repositories. I always update the packages on a daily basis. As a result, I've only had one or two annoying problems in those machines, and they were easily solved in a couple of hours or less. Part of the secret is that the desktop managers and hardware drivers (especially graphics ones) are the most troublesome packages in almost every Linux distribution. If you remove these parts of the equation, you will have statistically less problems.

Hardware Support

The hardware support in this distribution relies on the kernel and the drivers available in the repositories. Also, you can use the ABS system and AUR repository to find or build additional drivers. As with other aspects of this distro, this topic depends heavily on the user and his capabilities or willingness to learn.

Aesthetics

Obviously, in a very minimal distro like this one where you are responsible for what you build, there is no focus on aesthetics.

Desktop Environment

There is nothing predefined in this distribution, and this includes the desktop environment. Do you want a command line-only distro? This is it. Do you want to install a traditional desktop environment? You can do so, choosing from eight officially supported ones (Gnome, KDE Plasma, Cinnamon, MATE, LXDE, LXQt, Enlightenment, and Xfce) or from about fifteen unofficial ones (including Unity, Pantheon, Deepin, etc.). Do you want to install a window manager instead? You can do so, choosing from about sixty different ones. You can even choose the windows server, either Xorg or either Wayland. Basically there are many possibilities available so you only have to choose one (or more, if you want) and use it.

Init System

Arch Linux adopted systemd as the default init system in October of 2012.

Package Management System

Arch Linux uses its own package management system, pacman, a name that is shared by the command line tool used to manage the packages. It is not as popular as dpkg or rpm (it is used mostly by its derivatives), but it has a great reputation of being a simple and solid package manager. The pacman packages have the `.pkg.tar.xz` extension.

It also has the Arch Build System (ABS), which allows any user to customize any official package or even create her own packages from third party sources. These packages are configured in the form of a package description known as PKGBUILD (a shell script) that can be compiled from the source and built into a package that can then be installed via pacman. This characteristic is exploited to create the Arch User Repository (AUR), a repository of PKGBUILDs created and maintained by the users. These packages are by default not supported officially and are insecure, but this system is broadly used and has been proven over time to be a great way to make a very big number of packages available to users. Thanks to the AUR, a lot of packages are available a few hours after the author releases them, which gives Arch Linux a great advantage over other distros, which have a wait period before you can install the same package. And there's the option of making your own PKGBUILD and submitting it to the AUR; this one of the reasons why Arch Linux is so popular with some developers.

Architecture

Arch Linux only supports the Intel and AMD 32-/64-bit architectures. There is also an unofficial and independent port to the ARM architecture, the Arch Linux ARM.

Security/Anonymity

As with every other aspect of this distribution, making Arch Linux secure is mostly the responsibility of the user. You can harden Arch to paranoid levels if you want (by using the grsecurity kernel or sandboxing your browser), but it is up to you. You can refer to the wiki for guidelines to help you; go to https://wiki.archlinux.org/index.php/security.

Principles and Ethics

As stated by the Arch Way, this distro is pragmatic to a fault, and the ideological or ethical motivations are the responsibility of the final users. You can build a distro using only free software, or not; it's up to you.

Live CD

A Live CD makes no sense in a distro like this, one that is tailor-made by each user.

Professional Certification

There is no specific professional certification for this distribution.

Installation

Installing Arch Linux is a completely different process from the other distros previously covered in this book. The difference is not that all of the installation is done on the command line; you can do that in Debian and in the majority of the other distros if you wish. The main difference is there is no program to assist you in the workflow. The installation process is completely manual and unassisted. In the past, there was a text-based app to help you, but now only a few derivative distros have a program to do this (some of them graphical). Keep in mind that this is a distro inspired by the KISS principle and the DIY culture; however, you are not alone. There is excellent documentation to help you. So, if you are new to this way of installing a Linux distribution, I strongly recommend you go to the Beginner's Guide at https://wiki.archlinux.org/index.php/Beginners'_guide.

Well, first things first. Go to the Arch Linux Downloads page at www.archlinux.org/download/. As you can see in Figure 11-1, it lists the release info about the current month's release and several ways to download it. In this case, get the worldwide HTTP direct download by clicking the `rackspace.com` link.

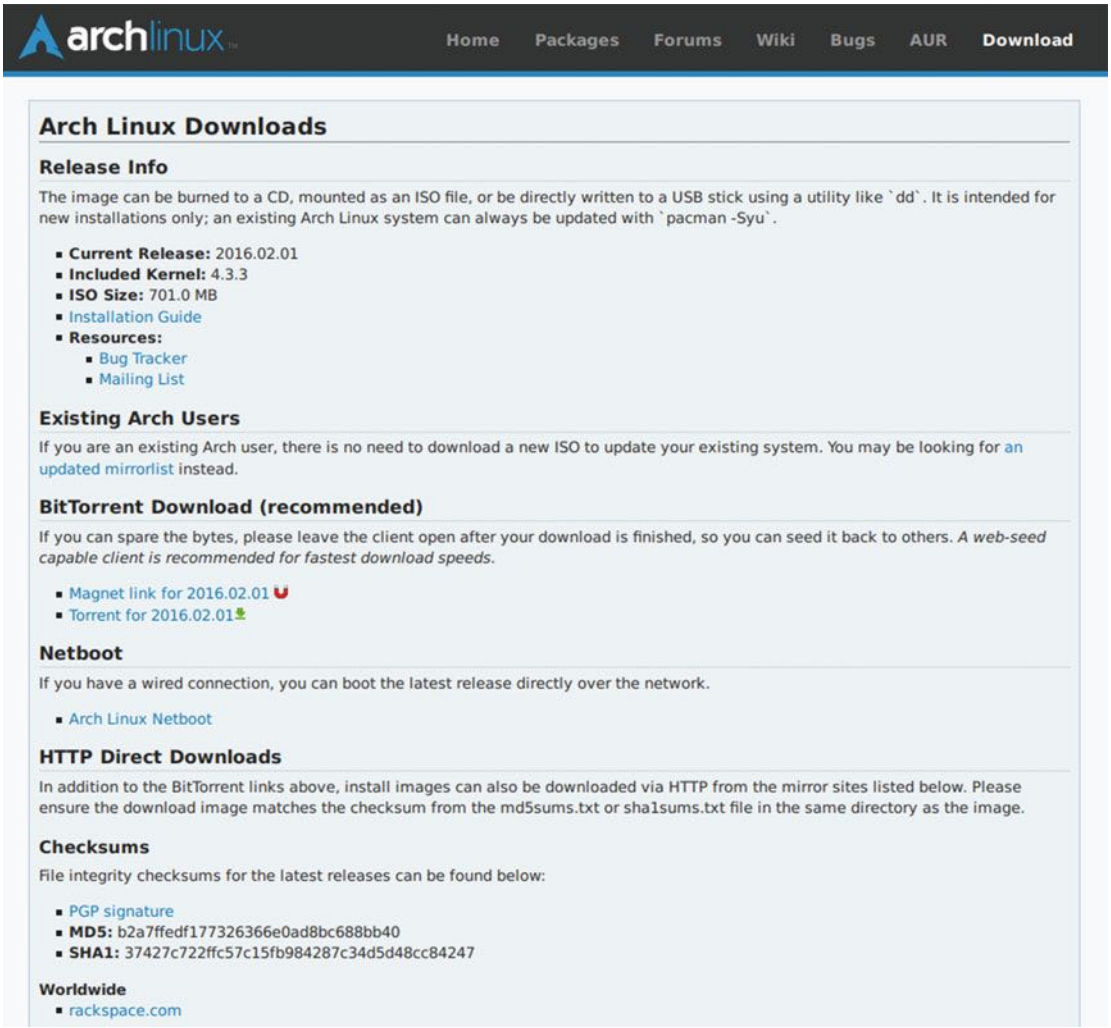


Figure 11-1. *The Arch Linux Downloads page*

If you use an HTTP server to download the image, you will see the different files available (Figure 11-2). Click the ISO image file link and proceed to the download.

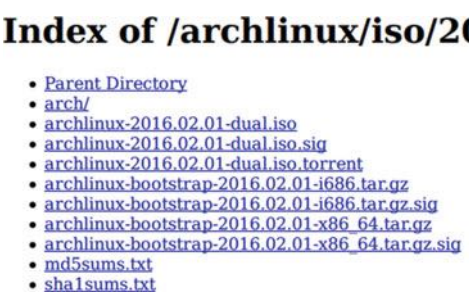


Figure 11-2. *The Arch Linux files available to download*

Once you have downloaded the ISO image and boot your system from it, the first screen you will see if you boot in a classic BIOS system (shown in Figure 11-3) is very similar to the ones you saw in the other distros. The main difference here is it is a dual architecture ISO image, so it lets you choose to boot in the corresponding one. It does not even try to detect it by itself.

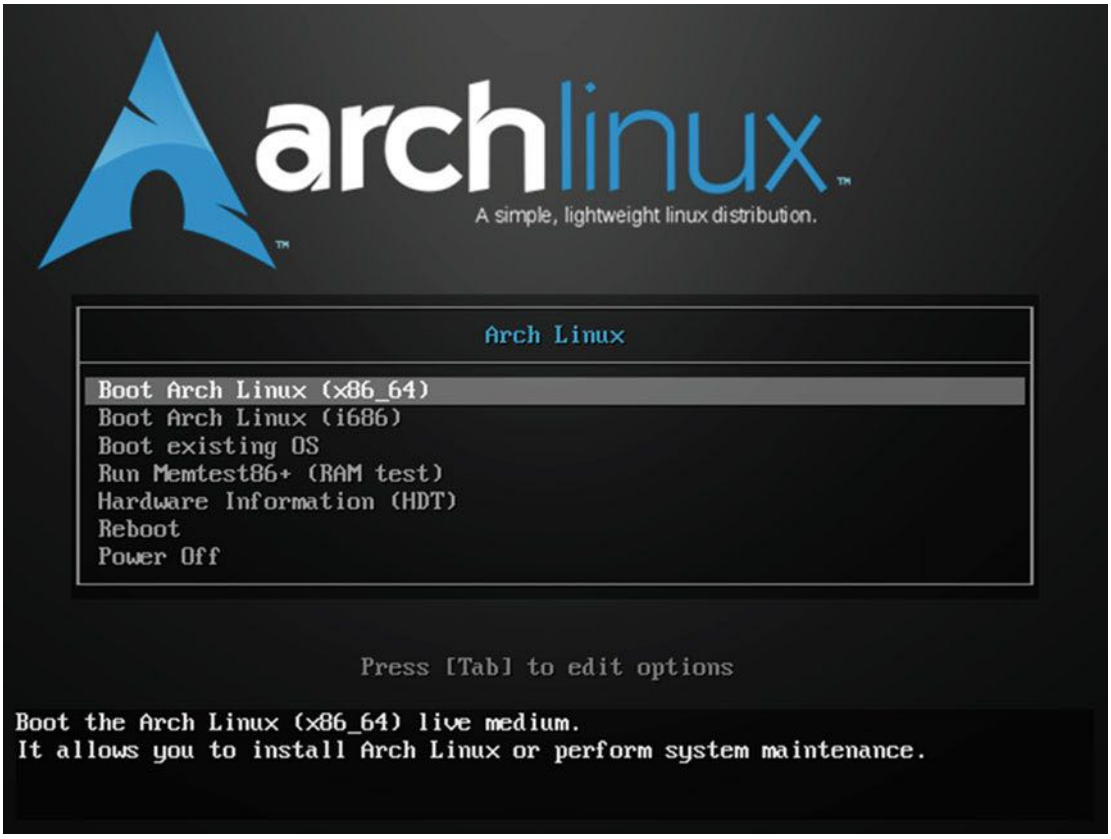


Figure 11-3. *The first Arch Linux boot screen*

Think of a modern UEFI system. In this case, the system boots automatically and you end up in a simple zsh shell prompt, logged in as the root user (you would get the same end result if you had booted in a BIOS system). You can see this in Figure 11-4. For beginners, this is hell on earth because they are totally lost, but there was a time when almost all of the distros were installed this way.

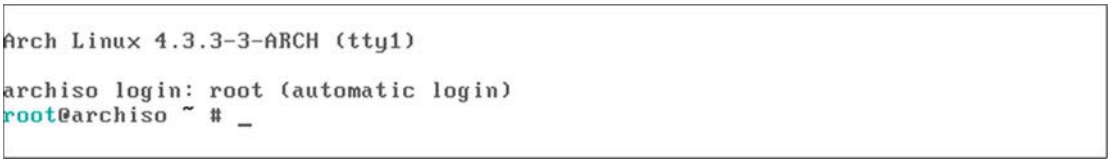


Figure 11-4. *The root zsh shell where you boot for the first time*

I will guide you through a simple installation, and let you decide if it as horrible as the rumors say, or not.

Note that the kernel version is a very recent one; it's usually the latest available in the core repository at the moment of building the ISO image. The first step is to configure your keyboard layout. If you are using an ANSI layout keyboard (US), do nothing. Otherwise, you should configure it with the following command:

```
# loadkeys es
```

In this example, if you were using a Spanish ISO layout, you would now have the correct layout for your keyboard loaded. Usually the layout is a country code of two letters, but there are some variants. I'm from Spain but I use an ANSI keyboard regularly because it is a better suited for admin servers and programming, so I did nothing here. But do not take this step lightly; when you have to configure passwords, this could make the difference between being able to log in or not. (When you enter or create the password, you cannot see the characters that you are typing. If the layout is not the one that you are using on your keyboard, well, you can imagine what could go wrong.)

The next step is to configure your network. If you are using a wired network, as in my case, and you have a router with a DHCP server, you will probably already be connected to the Internet because the systemd dhcpd daemon has already started. The best way to check this is to ping to a known address like google.com or one of its DNS servers like 8.8.8.8. If your router or firewall does not block the ICMP response, you should see the server answering you.

```
# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=46.6.ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=14.5.ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev/ = 14.505/30.585/46.666/16.081 ms
```

Another way is to use the already installed elinks text web browser to open a known URL (you can exit later by pressing "q" and then "yes"). This can be a very helpful resource to assist you in installing the distro: you can switch to another terminal (via CTRL+ALT+F2), log in with root (without a password), and then open the ArchWiki and navigate to the installation section to check it when you have questions (Figure 11-5 shows what it actually looks like).

```
# elinks wiki.archlinux.org
```

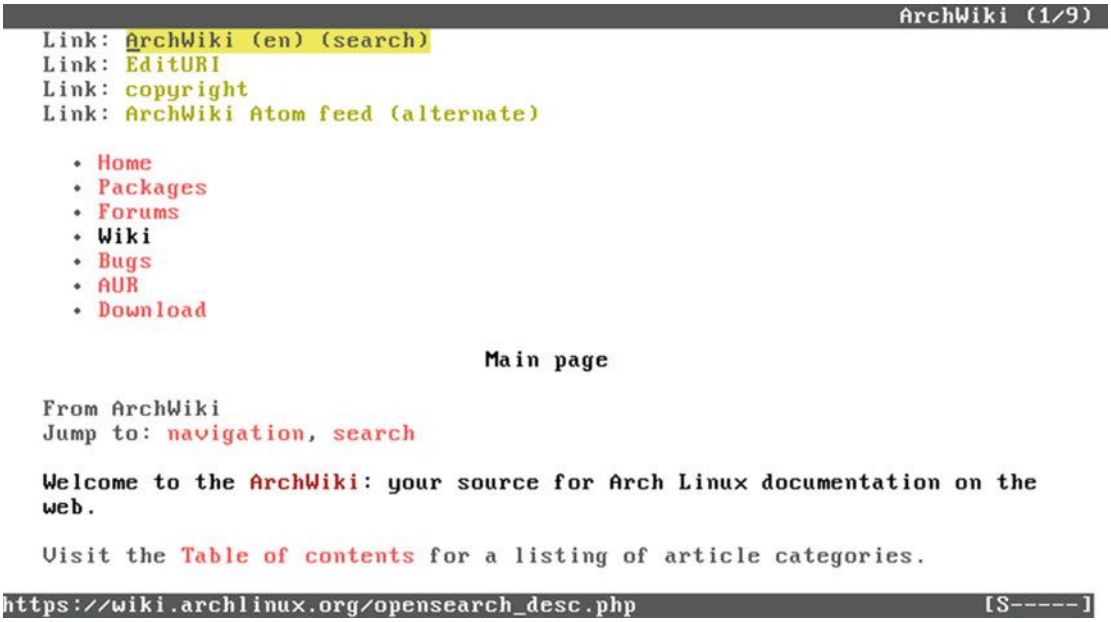


Figure 11-5. *The ArchWiki in a elinks browser in a terminal*

I'm going to suppose here that everything is ok and you are actually connected to the Internet. If not, consult the ArchWiki to help you set up the network (there are several situations you could have here, and I can't cover all of them in this book).

Now you must sync your system clock to ensure that the time is accurate. This could seem like a superfluous step, but if your system time is not correct, you could have problems installing packages later because it might be hard to import the needed pgp keys. To do so, use the following command:

```
# timedatectl set-ntp true
```

Next, check the status with

```
# timedatectl status
```

The critical step in all distro installations is the disk partitioning, and that's your next step. First of all, you need to identify your current disks and partitions to operate on them. In this example, I am using an empty 2TB hard disk. So execute the following tool, `lsblk`, to list the block devices in your system like hard drives, CD-ROM readers, DVD readers, and so on:

```
# lsblk --paths
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
/dev/sda          8:0    0     2T  0 disk
/dev/sr0         11:0    1   701M  0 rom  /run/archiso/bootmnt
/dev/loop0        7:0    0 307.8M  1 loop /run/archiso/sfs/airports
```

What we have here is a hard disk of 2TB called `sda` in the path `/dev/sda`. The Arch Linux ISO image mounted as `sr0` and there's another auxiliary virtual device used only in the installation. It's not showing any partition on the `sda` device.

Now you must do the partition. In this case, I am installing Arch Linux in a UEFI system, so I am going to use a GPT partition table on this disk. If you were using a BIOS one, you would use a MBR partition table. To do the partition, use the `parted` tool, which supports both partition tables. The process is very similar in both cases (you could use other tools).

First, start the program, specifying the correct device to use:

```
# parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Next, create your partition table:

```
(parted) mklabel gpt
```

In this case, let's create a very simple partition scheme with only three partitions, one for the EFI, another for the / mount point, and another for the `/home` mount point. I will show all the steps, but they are simple enough to understand. I chose a 50GB partition for the / directory; it could be smaller, but with a 2TB disk it's better to not be miserly here. After all, with Arch Linux you don't have to perform a fresh installation again for a long time and you don't know what you're going to install in the future. (Of course you could have a more complex scheme with lvm or btrfs and not worry about the size now, but let's continue with a simple scheme.)

```
(parted) mkpart ESP fat32 1MiB 513MiB
(parted) set 1 boot on

(parted) mkpart primary ext4 513MiB 50.5GiB
(parted) mkpart primary ext4 50.5GiB 100%
```

The final scheme is like the one you can see in Figure 11-6, after executing the following commands inside parted:

```
(parted) unit GiB
(parted) print
```

```
(parted) unit GiB
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 2048GiB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start    End      Size     File system  Name  Flags
 1      0.00GiB  0.50GiB  0.50GiB  fat32              boot, esp
 2      0.50GiB  50.5GiB  50.0GiB  ext4
 3      50.5GiB  2048GiB  1997GiB  ext4

(parted)
```

Figure 11-6. *The partition scheme*

Finally, exit the parted tool with the `quit` command, and get the status of your block devices:

```
(parted) quit
# lsblk --paths
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
/dev/sda          8:0    0     2T  0 disk
 ├─/dev/sda1      8:1    0   512M  0 part
 ├─/dev/sda2      8:2    0    50G  0 part
 └─/dev/sda3      8:3    0     2T  0 part
/dev/sr0         11:0    1   701M  0 rom  /run/archiso/bootmnt
/dev/loop0        7:0    0 307.8M  1 loop /run/archiso/sfs/airports
```

Now it's time to format the partitions and mount them. The EFI (`sda1`) partition needs to be a FAT32 one, and for the rest, choose a safe bet, like ext4. Let's format them:

```
# mkfs.fat -F32 /dev/sda1
# mkfs.ext4 /dev/sda2
# mkfs.ext4 /dev/sda3
```

You must mount them in the live system to perform the installation; those are not the definitive mount points:

```
# mount /dev/sda2 /mnt
# mkdir -p /mnt/boot
# mount /dev/sda1 /mnt/boot
# mkdir -p /mnt/home
# mount /dev/sda3 /mnt/home
```

You can see the current status of the partitions in Figure 11-7.

```
root@archiso ~ # parted /dev/sda unit GiB print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 2048GiB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start    End      Size     File system  Name  Flags
 1      0.00GiB  0.50GiB  0.50GiB  fat32              boot, esp
 2      0.50GiB  50.5GiB  50.0GiB  ext4
 3      50.5GiB  2048GiB  1997GiB  ext4

root@archiso ~ # lsblk --paths
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
/dev/sda      8:0    0     2T  0 disk
├─/dev/sda1   8:1    0   512M  0 part /mnt/boot
├─/dev/sda2   8:2    0    50G  0 part /mnt
└─/dev/sda3   8:3    0     2T  0 part /mnt/home
/dev/sr0     11:0    1   701M  0 rom  /run/archiso/bootmnt
/dev/loop0    7:0    0 307.8M  1 loop /run/archiso/sfs/airootfs
root@archiso ~ #
```

Figure 11-7. *The disk is ready to perform the installation and the partitions have been made*

After preparing the disk, continue the installation by installing the base system first. You do this with the `pacstrap` script (the `base-devel` is to build packages with ABS or from the AUR). Install this package in the `/mnt` directory that is going to be your future / root directory:

```
# pacstrap -i /mnt base base-devel
```

The script will show you the packages that will be installed and it will ask for your confirmation, as you can see in Figure 11-8. There are around 150 packages in this release and although they will take up 750 MiB on the disk, pacman uses a very good compression algorithm (lzma2) so you will download only about 230 MiB of data.

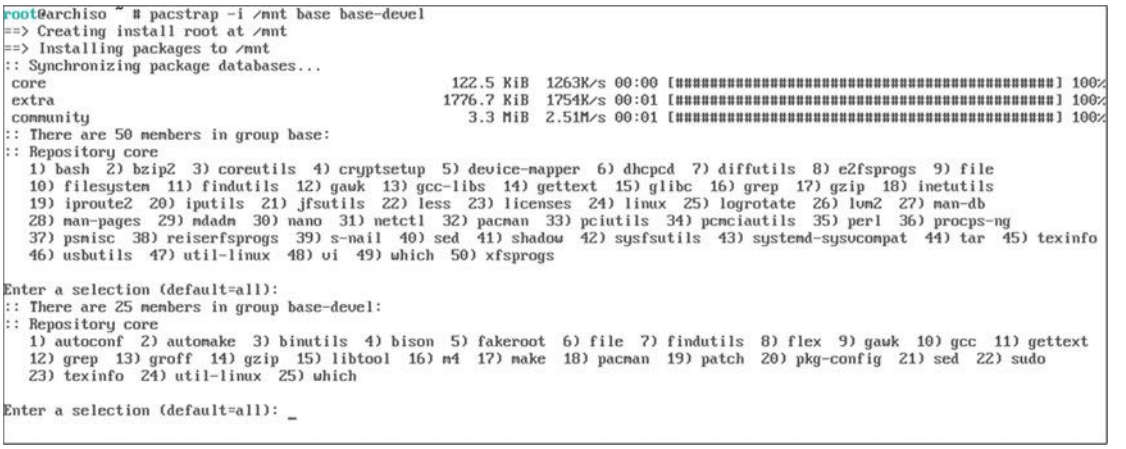


Figure 11-8. *The base system installation script*

After all of the base packages are installed, you will create the `fstab` file that manages how the partitions are going to be mounted on your system with the right mount points (which are identical to the current ones without the preceding `/mnt` directory) and parameters.

```
# genfstab -U /mnt >> /mnt/etc/fstab
```

Next, create a chroot jail, which changes the root directory from the current one (in the virtual system booted from the ISO image) to the real one that is on your disk. Thus, the rest of the commands will consider your disk as the current booted system and apply the changes. Note that the command prompt changes slightly to reflect the change:

```
# arch-chroot /mnt /bin/bash
```

In the next steps, you need to use a text editor. If you know how to use `vi`, I strongly recommend using it; if not, use `nano` instead, which is a more traditional text editor. Of course you can use `pacman` to install another terminal text editor that you like, such as `emacs` or `joe`.

To continue, you need to choose the language (and regional settings) that you are going to use in the system. To do so, you need to edit the `/etc/locale.gen` file and uncomment (remove the # character at the beginning of the line) the language code lines that you want to install (e.g. `enUS.UTF-8 UTF-8`). Then execute the command `locale-gen`:

```
# vi /etc/locale.gen
# locale-gen
```

Now you need to create the `/etc/locale.conf` file where you specify the language that you are going to use as default, such as `LANG=enUS.UTF-8`. If you also set a different keyboard layout than ANSI US, you need to edit the `/etc/vconsole.conf` file too; check the ArchWiki if you need help.

```
# vi /etc/locale.conf
# cat /etc/locale.conf
LANG=en_US.UTF-8
```

The time zone is the next step. First, get the code for the time zone with the tool `tzselect`, which saves you from the tedious work of having to look up your time zone in the `/usr/share/zoneinfo` directory. When you invoke this tool, it shows several options to choose from and narrows the search until you have your local time zone. At the end, it shows you the path of the appropriate file. So run `tzselect`, and after following the instructions, use that time zone code (for example, America/New_York) to make the changes permanent via a symbolic link to its path in the global `/etc/localtime` configuration file. Finally, adjust the time and set the time standard to UTC (this is strongly recommended).

```
# tzselect
# ln -s /usr/share/zoneinfo/America/New_York /etc/localtime
# hwclock --systohc --utc
```

Next, you need to install a boot loader. There are several options (I use rEFInd) but let's use the one that comes with systemd, `systemd-boot`. Next, configure a default boot entry on the `/boot/loader/loader.conf` file. Finally, add that entry to the boot loader, using a helper command to get the PARTUUID of the root partition.

```
# bootctl install
# vi /boot/loader/loader.conf
# cat /boot/loader/loader.conf
default  arch
timeout  3
editor   0
# blkid -s PARTUUID -o value /dev/sda2
58507240-c577-41c0-b228-e5fee0dfaee3
# vi /boot/loader/entries/arch.conf
# cat /boot/loader/entries/arch.conf
title          Arch Linux
linux          /vmlinuz-linux
initrd         /initramfs-linux.img
options        root=PARTUUID=58507240-c577-41c0-b228-e5fee0dfaee3 rw
```

Next, configure the network. In my case, this was very simple and only required two steps: setting the name of the system and enabling the `systemd-dhcpcd` service.

```
# echo myarch /etc/hostname
# cat /etc/hostname
Myarch
# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:2a:3b:64:15 brd ff:ff:ff:ff:ff:ff
# systemctl enable dhcpcd@enp0s3.service
```

The last step is to get a basic Arch Linux system installed. First, set the root password, then unmount the partitions and reboot. Remember to set a very strong password for your root user.

```
# passwd
# exit
# umount -R /mnt
# reboot
```

And that's it! You have now a very basic Arch Linux distro installed on your system. When you boot your new Linux distro for the first time, if you press Enter over the Arch Linux entry or wait 4 seconds, you will end up again in a Linux terminal session (see Figure 11-9). And that's because all you have installed is the distribution, nothing more; it's the basic core. As mentioned, now you build into the installation what you really want; nothing is predefined. Thus, contrary to other Linux distros, when you boot up the system for the first time, your real job begins; friendly distros are ready to use immediately. At this time, you only have a root user and no graphical environment of any kind. But this is a DIY distribution, so this is what you get.

Figure 11-9. *An Arch Linux base system before any customization*

From here, it's hard to say what a "standard" path is. I will show you a few very basics steps to help you get an idea of where to go. Don't follow them as a guide, because I do not cover security, multimedia, repositories, and so on. To completely customize an Arch Linux installation can take several hours, depending on what you want to do and your knowledge. But remember, you install this once and then you don't have to reinstall it for years, and you always have an up-to-date system.

The first thing to do is add a user to use the root user only for admin tasks. Also, let's install `sudo` to escalate the privileges of your user to do admin tasks like updating the distro.

```
# useradd -m -G users,wheel johndoe
# passwd johndoe
# pacman -S sudo
```

Now let's give your user the authorization to use `sudo`, by giving those privileges to the wheel users group to which you previously added for user. The first command is necessary only if you don't want to use the `vi` editor and use `nano` instead. With the command `visudo` you have to edit the `/etc/sudoers` file and uncomment the line `# %wheel ALL=(ALL) ALL` The last command installs the package `bash-completion` to make it easier to write commands in the terminal with auto-completion.

```
# EDITOR=nano visudo
# visudo
# sudo -lU johndoe
User johndoe may run the following commands on myarch:
    (ALL) ALL
# pacman -S bash-completion
```

The last step that I will show you is to install a graphic environment and a very simple window manager, (instead of installing a classic desktop environment, which you can do if you wish). Start by installing the Xorg server and the video drivers. It will present some choices; if you don't have an Nvidia graphics card, the defaults are fine. The second command is to identify your video card, and the third lists all of the video drivers available. I'll pick an Intel driver.

```
# pacman -S xorg-server xorg-server-utils xorg-apps
# lspci | grep -e VGA -e 3D
00:02.0 VGA compatible controller: Intel Corporation Xeon E3-1200 v2/3rd Gen Core processor
Graphics Controller
# pacman -Ss xf86-video
# pacman -S xf86-video-intel
```

Now it's time to install the window manager. First, you need to install a display manager to be able to log in and select the window manager/desktop environment that you want to use. You could start the window manager directly or boot in the terminal and start the window manager manually, but I think it is friendlier (and equivalent to what you saw in other distros) to do it in this way. I used to do this with Slim, but it is currently discontinued. Now I prefer to use LightDM, which works with multiple DMs and WMs and is very light. There are many options and you can use the one you prefer.

```
# pacman -S lightdm lightdm-gtk-greeter
# systemctl enable lightdm.service
```

There are also several choices of window manager. Let's choose a tiling window manager. (I personally use Awesome WM, and I also like i3 or dwm.) Select i3 because it is one of the easiest to learn. Also, let's install some auxiliary programs; when you are asked about what to install after this command, choose all by default. Finally, let's install a graphical web browser and a terminal tool to show you how they look in i3. Then you need to restart your system to see how it boots directly to the display manager.

```
# pacman -S i3 dmenu termite
# pacman -S chromium htop ttf-dejavu
# systemctl reboot
```

After the system restarts, you can see that you end up in LightDM and not the terminal (Figure 11-10). Introduce your password and then go to the i3 window manager. If it asks you about the configuration, simple press ESC to go with the default one. What you will see is a black screen with a minimal text status bar below; it's as minimal as you can get. If you open a pair of programs like a terminal with `htop` and the Chromium browser, you will see something similar to Figure 11-11. One of the advantages of a minimal system like this is that, as you can see in the `htop` output, only a few programs started and the resource use is minimal. To put this into context, I tested a similar scenario in Ubuntu 15.10 and I got 715MB of memory and 120 tasks versus the 212MB and 34 tasks (I only show you the ones initiated with the current user) that you can see here. And in a system configured like mine, the difference with this example is not too much and even very far from the Ubuntu example (that could be Fedora, openSUSE …).

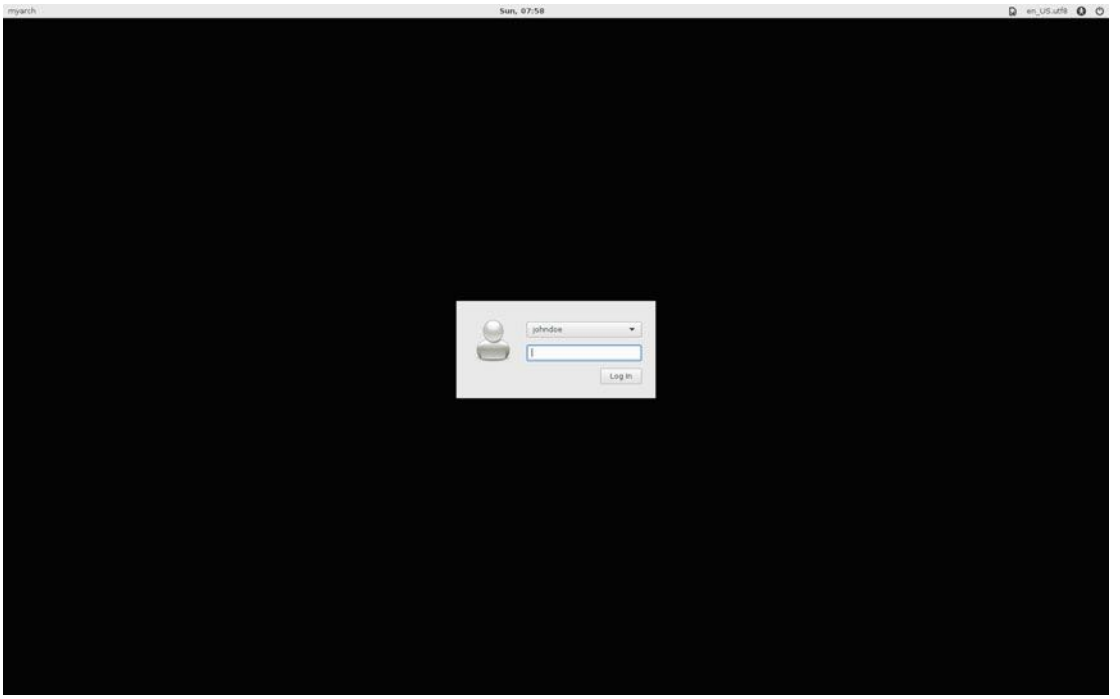


Figure 11-10. *The LightDM screen to log into your i3 session*

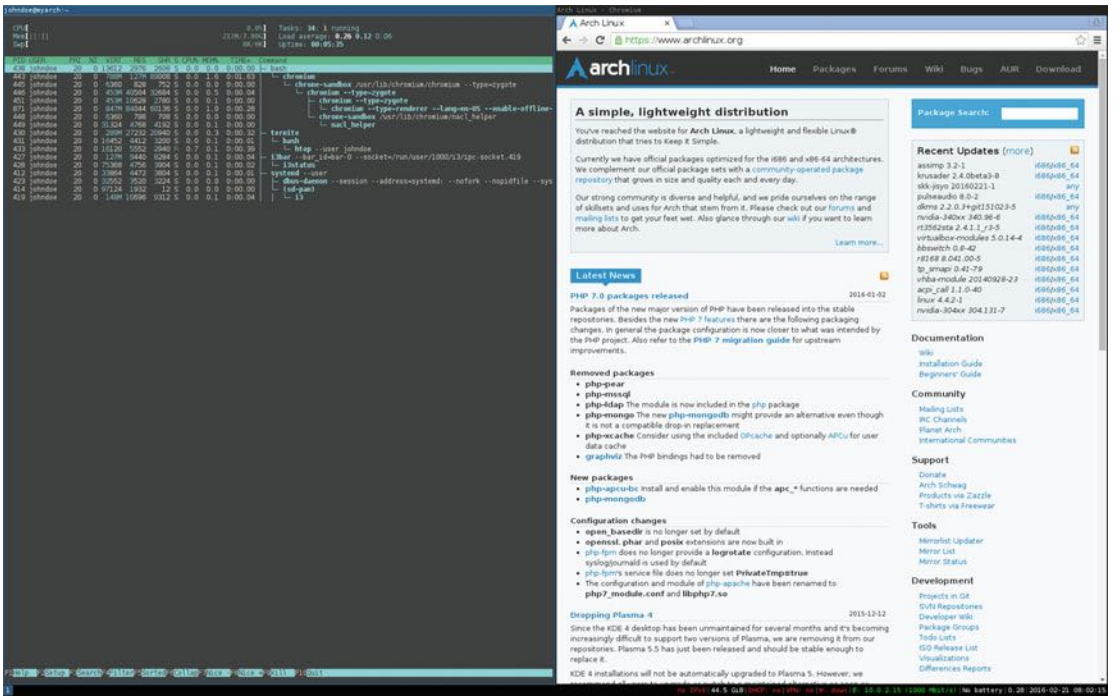


Figure 11-11. *The window manager i3 with a pair of windows opened*

To be fair, with a distro like this, where the job of installing is much more complex than with a traditional distro, you must consider the level of customization and lightness that you can achieve. For example, in a case like mine, where I prefer to work with a window manager instead a desktop environment, if I installed an Ubuntu distribution I would have two options after installing and setting up the window manager:

- Use the window manager and keep the default DE (Unity, in this case) with the accompanying waste of space and resources. Also, I'd need to update the large number of packages that usually come with a DE.
- Use the window manager and uninstall the default DE. This is not an easy task. In a distribution like Ubuntu, because of the dependences, it could be impossible to perform a clean uninstall; it might even break the system if you are not careful.

So, in a case like mine, it's usually a better choice to build up a system from a minimal base than to try to make a traditional distro lighter or under-use it. Of course, you have to accept several compromises and give up some conveniences.

Maintenance

In Arch Linux, you only have to managing apps and perform updates; it's a rolling release distro so there's no need to upgrade it.

Managing Apps and Updating

The tasks of managing apps and upgrading them are performed in Arch Linux from the command line. There are some tools that work as graphical front ends, but no one uses them and they not are officially supported. You use the same app for both tasks, `pacman`. To install an app, use this command:

```
# pacman -S firefox
```

To remove it, use this command:

```
# pacman -R firefox
```

And for making regular updates, the command should be like this one:

```
# pacman -Syu
```

With those simple commands a majority of people can survive for a long time without major problems. Of course, there are some necessary long-term tasks, like refreshing the keyring, but this distro expects that you feel comfortable in a DIY culture and can take care of yourself. In a distro like this, it is strongly recommended that you perform updates frequently, even daily, because if there are changes that require manual intervention or ones that introduce changes in the setup of the distro or the way it works, it could stop working properly if these updates are postponed for too long. The current package releases expect these tasks to be performed immediately. In brief, updating regularly saves you from some unnecessary headaches.

As for packages installed from the AUR, there are several tools that can help you do that automatically, like `pacaur` or `yaourt`, or semi-automatically, like `cower`. All of them are unofficial apps. Some people prefer to do this by hand in order to have more control in the process.

■ **Note** I'm using the term "updating" in all of the chapters in this book for consistency, but Arch Linux people prefer to use "upgrade" because they say that what you are really doing is upgrading the full system regularly and not simple updating it. And that's true since it doesn't support partial upgrades; you must update all of the packages available. (You still can block package updates, and do partial upgrades, but it is not recommended.) So technically you can say that you are fully upgrading the system each time. However, to compare distros, I think it is easier to understand it as "updating."

Pros and Cons

The following points are things that I personally see as pros and cons of the Arch Linux distribution. I tried to be as objective as possible.

Pros

- It is a rolling release distro, so it is always on the bleeding edge.
- It offers great documentation.
- It has a very welcoming community.
- It comes with a great package manager, and ABS and AUR.
- Arch Linux is what you make of it; it's fully customizable.
- You can use free and proprietary software, and if none is available officially you can probably make your own package.
- It is an original distro, not a derivative.

Cons

- It is oriented to users willing to do it all by themselves, so it's not a friendly distro.
- It has the instability inherent to being a rolling release distro.
- You have to be comfortable with the command line to even install it.
- No aesthetics at all; the look is up to you.
- It has no commercial support.

Summary

Arch Linux was the first Linux distro I covered that is suited for advanced users; it's also the first genuine rolling release distro I covered. You can see the huge differences between this kind of distro and the traditional ones. This kind of distro not only requires you to do it all by yourself, but it also requires that you know what you want to build because there are no predefined setups for you.

In the next chapter, you will see another distro of this kind, but Gentoo goes a step further.

CHAPTER 12

Gentoo Linux

Gentoo Linux is the quintessential custom Linux distro. In fact, it's so adaptable that the Gentoo people prefer to call it a metadistribution. As with Arch Linux, Gentoo can be whatever you need it to be. It's also a rolling release distro, but it goes a step further; you can build every package from source code to optimize them for your system and needs. Its popularity has taken the opposite trajectory of Arch Linux's: it was once a very popular distribution but this has decreased over the past few years. However, it still has a strong and loyal community behind it.

History

In April of 1999, Daniel Robbins, an American programmer, started writing the Enoch Linux distro. The idea was to create a Linux distro that was as fast as possible by compiling each package tuned to specific hardware instead of using generic precompiled binaries. You only had to install the packages you really needed/wanted, so it was also a light and versatile distribution.

In May of 1999, after buying a new machine (a dual Celeron), Robbins found out that the Linux kernel would not boot on it, so while other programmers continued to maintain the Enoch distro, he switched to FreeBSD until the bugs were fixed. After several months of using FreeBSD, he returned to developing Enoch, and he started with two important moves:

- He changed the name of the distribution to Gentoo Linux (this name was a suggestion from another contributor).
- He incorporated some ideas from FreeBSD to create Portage, a modern ports system.

The first release of Gentoo Linux (there was a previous one as Enoch) was released on March 31, 2002.

Two years after that release, Daniel Robbins left the Gentoo project as Chief Architect and transferred all Gentoo intellectual property to the non-profit Gentoo Foundation, which he created. The Gentoo Linux project still is under the umbrella of the Gentoo Foundation today, so it is a community-based distribution. Daniel Robbins now works at Zenoss and simultaneously collaborates on the Funtoo project, a Gentoo derivative he created in 2007.

One proof of the power of the customization and versatility of Gentoo, and also of the type of user/use that it has, is that Google used Gentoo in July of 2009 as the base for its own web-based OS, Chrome OS, to work with its cloud.

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_12

■ **Note** The Gentoo name was an idea from Bob Mutch, and it is related to the Linux mascot. As mentioned in Chapter 1, the Linux mascot is a penguin. Gentoo is a species of penguins; they are considered the fastest underwater swimming penguins, reaching speeds of 22 mph (36 km/h). The idea was to transmit the association between the fastest penguins and the fastest Linux distro.

Philosophy

The philosophy of Gentoo shares some points with that of Arch Linux, but Gentoo takes a more radical approach. Everything can be customized and optimized to suit the user. You can compile the kernel and practically all of the packages from the source code, customizing at very basic levels like what compiler options use or which dependencies (if any) you want to install for each and every package. Thus, if in Arch you get to build your system as you want it, in Gentoo you have almost full control about every tiny aspect of your distribution. And on top of that, Gentoo also follows a rolling release model, so the most recently updated packages are available.

The goal is to have a full customized Linux installation that is much more finely tuned, optimized to your hardware and needs. But this comes at a huge cost in terms of time invested in the installation (and updating it). If installing an Arch Linux system takes several hours for a hugely customized installation, Gentoo can take more than a day just to compile the packages you want to install. As a result, Gentoo users are very particular ones who prefer full control and are prepared to take the time to make it happen; this includes system administrators, developers, enthusiasts, and in general very advanced users. If I had to define the Gentoo philosophy in only one word, it would be "choice."

The strong constraints that both distros impose generate a similar outcome in two critical aspects of any distro: a very welcoming and helping community and documentation of extraordinary quality.

Distro Selection Criteria

Now that you know a little history, let's see how Gentoo Linux fares on the selection criteria from Chapter 2.

Purpose and Environment

Obviously Gentoo is practically the definition of a general purpose distro. Because you can built whatever Linux you want with Gentoo for whatever purpose, it can be used to create a task-oriented installation too. Therefore, there is only one version of this distro. Actually there is another version available for the two main architectures (amd64/x86) called Hardened Gentoo, but it's essentially a same version that has been predefined with security measures to obtain a very hardened Linux system. So I hardly consider it as a different version.

Support

Gentoo gets support from its community, which is of a modest size, but they are technically well-versed and helpful. The community maintains a detailed and quality documentation. Perhaps it doesn't covers as many themes as the Arch Linux one, but it covers some topics in at more technical and deeper level. The channels for support are the following:

- **Documentation**: `www.gentoo.org/support/documentation/`
- **Wiki**: `https://wiki.gentoo.org/wiki/Main_Page`
- **Forums**: `https://forums.gentoo.org/`

- **Mailing lists**: www.gentoo.org/get-involved/mailing-lists/
- **IRC**: #gentoo at irc.freenode.org
  More at www.gentoo.org/get-involved/irc-channels/

User Friendliness

This is not a user-friendly distro, not only because you need to do it all on your own, without a tool to assist you, but also because you need to compile the packages, which requires more knowledge. Happily, there is excellent documentation to guide you. Another unfriendly aspect is the time it takes to compile just one big package (e.g. OpenOffice, Mozilla Firefox, or a desktop environment); this is probably the main cause of the decrease in popularity of the distro.

A few inexperienced users are still attracted to this distro because is one of the best for learning about Linux at a low level; also the documentation has very good coverage on Linux fundamentals. Aside from that, I would not recommend this distribution to beginners, especially newcomers, because the compilation process is one of the main points used to criticize Linux (which is unfair because it's something you rarely do in more conventional distros nowadays). But if you want to really learn how an OS works at a low level, Gentoo is one of my recommendations for sure.

Stability

As with Arch Linux, Gentoo is inherently unstable because it is a rolling release distribution. As with Arch, you can choose your level of recklessness; you can choose the stability of your packages from three levels:

- **Stable**: Stable packages with no issues, security or otherwise.
- **Keyword masked**: Packages are still insufficiently tested.
- **Hard masked**: It has broken or very insecure packages.

The stable packages can be a bit outdated (but still fresher than classical distros) but the system is reasonable stable. If you follow a strategy similar as the one outlined in Chapter 11, you can end up with a more-than-reasonably stable system, for a rolling release distro. Also, contrary to Arch Linux, you can be very selective about the packages that you want to update without risking the stability of the distribution (in Arch, doing this is not recommended). Moreover, you can select the stability for individual packages, such as setting stable packages for the core system and testing ones for developer tools. By adhering to this practice very rigorously and meticulously you can achieve a very robust system, one that can be as stable as Debian stable. After all, Gentoo is whatever you want it to be.

Hardware Support

Gentoo's hardware support relies on the kernel, firmware, and the drivers available on the Portage tree and overlays. As with other aspects of this distro, this topic depends heavily on the user and her capabilities or willingness to learn.

Aesthetics

Gentoo follows the philosophy of building a unique installation suited for you. It doesn't impose any particular aesthetic at all. Only a few branding patches are applied here and there, but they're mostly incidental (and you have to compile the packages with that option).

Desktop Environment

There is almost nothing predefined in this distro, and this extends to the desktop environment. You get to decide what you want to install. There is a wide variety of desktop environments you can install (Gnome, KDE, Unity, Mate, Cinnamon, Pantheon, LXDE, LXQt, Budgie, and Xfce). In a similar way, if you prefer a window manager instead, you can also choose from a variety (Awesome, Xmonad, Dwm, i3, Openbox, Sawfish, Fluxbox, Enlightenment, Ratpoison, FVWM, and Wm2). As with Arch Linux, you can even decide if you want to use the traditional X.org window server or the new Wayland, or none at all if you want just a command line.

Init System

Gentoo is one of the few distros that still does not implement systemd as its init system, and there is no intention to do so in the future. Gentoo uses OpenRC, a BSD-based init system. But you can optionally install and use systemd, if you want to use Gnome 3, for example, but again, it's your call.

Package Management System

Gentoo uses its own package management system, Portage. Portage is based on the experience that Robbins had with FreeBSD; it's based on its ports system and written in Python. Portage is the soul of Gentoo and it defines the distribution. Portage manages the packages in a different manner than the traditional Linux package managers. It doesn't get binaries from repositories; it compiles the packages from the source code. Even when you can also compile packages in the other distributions, or some others like Arch Linux have its own ports based system like ABS, in Portage the norm is to compile all the packages.

Portage has a collection (usually known as Portage tree) of ebuilds (shell scripts), which are recipes that define how each package is going to be build. To manage this, there's another tool called emerge that works as an interface to Portage. There's also a local ebuilds collection, which you can think of as a sort of local repository that can be synced with the main repository, the Portage tree, and emerge is the tool to manage this local repository. Take this with a grain of salt; I won't get into detail here. I just want you to get a rough idea of how it works.

There are some GUI interfaces for Portage that can be used instead of emerge, like Porthole and Himerge, but usually the people that choose Gentoo as their distribution are more inclined to use command line tools.

Portage/emerge is a very powerful tool, and it allows you total control over the packages and your system. You can configure global settings to compile all the packages via the profile and "CFLAGS and USE flags." You can also set individual settings by package or temporary ones.

Compiling the packages instead of using precompiled binaries has some inconveniences, such as the following:

- **It's a very slow process**: A simple package like Chromium took me about three hours to compile when using the setting `MAKEOPTS="-j2"` (two simultaneous jobs, one core), and about one hour with `MAKEOPTS="-j5"` (five jobs, four cores) with an Intel i7 processor. So, imagine building a complete system like the default installation of Fedora or Ubuntu. Of course, you can use some precompiled packages (you can find them using `# eix *-bin`; currently there are 94 packages), and this will help a little, but it will still take a long time. However, this factor is getting less and less annoying as computers get faster over time and package sizes grow at a much slower pace.
- **A package build can fail**: And sometimes there is no easy to fix, even after changing compilation options. On rare occasions it's even impossible, and only upstream changes can fix it.

But it has some advantages too, like the following (among others):

- **You can choose the dependencies/components**: You can choose, for example, to remove the support for the desktop environment KDE/Qt and only have support for Gnome/GTK in packages that support both. Or you can remove the graphical interfaces for the tools that support it and the command line in servers.
- **You can compile packages for your hardware**: For example, this can allow you to build an updated Linux system in obsolete architectures that are merely supported for other distros or not at all (e.g. PowerPC). It's also great for taking advantage of computing/multimedia packages.
- **You can choose versions and maintain various versions**: You can manually choose the version of a package from the available ones and even have various ones installed in the same system with the slots functionality. This used to be a great advantage for developers before the arrival of virtual machines and containers.
- **You can update an unmaintained system**: You can easily resuscitate a Gentoo system that has not been updated for a long time. As always, it is not immediate; it takes time and effort, but you can't do this in the majority of distros (even in Debian unstable or Arch Linux this can be very painful, something I know firsthand), and it would probably be impossible without a fresh install.

Aside from the Portage main repository, Gentoo also supports overlays, which are a way to create additional repositories of ebuilds, something roughly equivalent to Arch's AUR (or Ubuntu's PPAs). There are official and unofficial overlays; the latter ones are supported and maintained by the community. There are some tools to manage overlays like layman, eix, and emaint. Currently Gentoo only keeps about 19,000 packages in the Portage tree, and the rest of the packages are usually obtained from these overlays.

Years ago, the gain in performance was a strong point in favor of compiling all the packages suited to your hardware, but nowadays the difference in performance against binary package distros is not that big and it doesn't make up for the time spent on the compilation. Only when you need an extremely tuned system for specific tasks can you justify to choosing Gentoo over other distros in terms of performance. In a few words, performance is not the reason why users choose Gentoo as their distro anymore, even if they say it is.

Architecture

Gentoo supports a wide spectrum of hardware architectures. It covers the usual Intel/AMD 32- and 64-bit options plus Alpha, ARM, Sparc, PowerPC, Itanium, and PA-RISC. Three more are supported only as experimental ones: MIPS, S390, and Super H.

Security/Anonymity

In a distro like this one, it all depends how you manage security and anonymity. However, Gentoo is well known for taking this topic seriously. From the very beginning, packages compiled with Portage/emerge are optionally built and installed in a sandbox (isolated) environment before they are merged into the actual system. Also, the stable packages are free of known security flaws, and Gentoo includes tools for checking for insecure packages in your system.

Also, the fact that binaries or the kernel can be heavily customized for your hardware and taste makes it less likely that a specific package vulnerability can affect you in the same way it would in other distros.

There is an entire project to help you to harden your Gentoo system: the Hardened Gentoo project. It has a great reputation in the security world. You can use it to harden your Gentoo system and get a very secure one, up to paranoia levels. You can see it at `https://wiki.gentoo.org/wiki/Hardened_Gentoo` and `https://wiki.gentoo.org/wiki/Project:Hardened`.

Principles and Ethics

Gentoo is a very pragmatic distro in terms of principals and ethics. It does not impose any particular vision; you choose your way to address these issues. But you can easily set up the system to only install free software by change an option in Portage to `ACCEPT_LICENSE="@FREE"`.

Live CD

Gentoo offers a LiveDVD that it calls Hybrid ISO (more about this in the Installation section). Obviously, in this case (unlike in other distros), it is only a proof of concept of what you could build, not what you would have at the end of the installation process (no two Gentoo installs are alike). Also, this DVD is released sporadically; the last one available is from August 2014.

■ **Note** Gentoo calls also their minimal ISO image a Live media. It is true; it's a live Linux system, but not in the sense that I'm using in this book.

Professional Certification

There is no professional certification for this distribution.

Installation

The Gentoo philosophy, summarized by choice, is something that you can perceive from the very first moment you try to install Gentoo. Unlike the norm in other distros, you can start the installation of Gentoo from its minimal installation ISO or from its LiveDVD, as well as from another LiveCD/DVD, a netboot image, another distro already installed, etc. Basically you can install Gentoo from almost any basic minimal Linux environment. You can do something similar with other distros like Slackware, but Gentoo does not impose anything from the beginning.

It's impossible for me to cover all of the possible scenarios here, especially with a distro like this one, so I strongly recommend taking a look at the Gentoo Handbook at `https://wiki.gentoo.org/wiki/Handbook:Main_Page`. Also, I'm not going to spend too much time on topics I covered in Chapter 11 about Arch Linux because I want to focus on topics specific to the Gentoo installation.

I'm going to use the minimal installation CD that Gentoo provides on its web site because I think this is the choice of the majority of users, although a lot of them prefer to use a LiveCD that gets them a live desktop environment like the Ubuntu one to prepare the installation with graphical tools like Gparted to do the partitions.

As always, first go to the Downloads page of the Gentoo web site at `www.gentoo.org/downloads/`. You can see how it currently looks in Figure 12-1. Note how the images are organized by architecture; in each architecture you have "Boot Media" (the ISO images) and "Stage Archives" (more about this later). At the bottom of the page (not shown in the picture) you can also select other architectures or advanced choices for the architectures already shown at the top. Some of these choices include a "Hardened Stage 3" (only for amd64 and x86), which you can use to obtain a previously secured environment to get a very secure Gentoo system. In this case, download the Minimal Installation CD for the amd64 architecture. These minimal installation ISOs are released every week, so they are always updated.

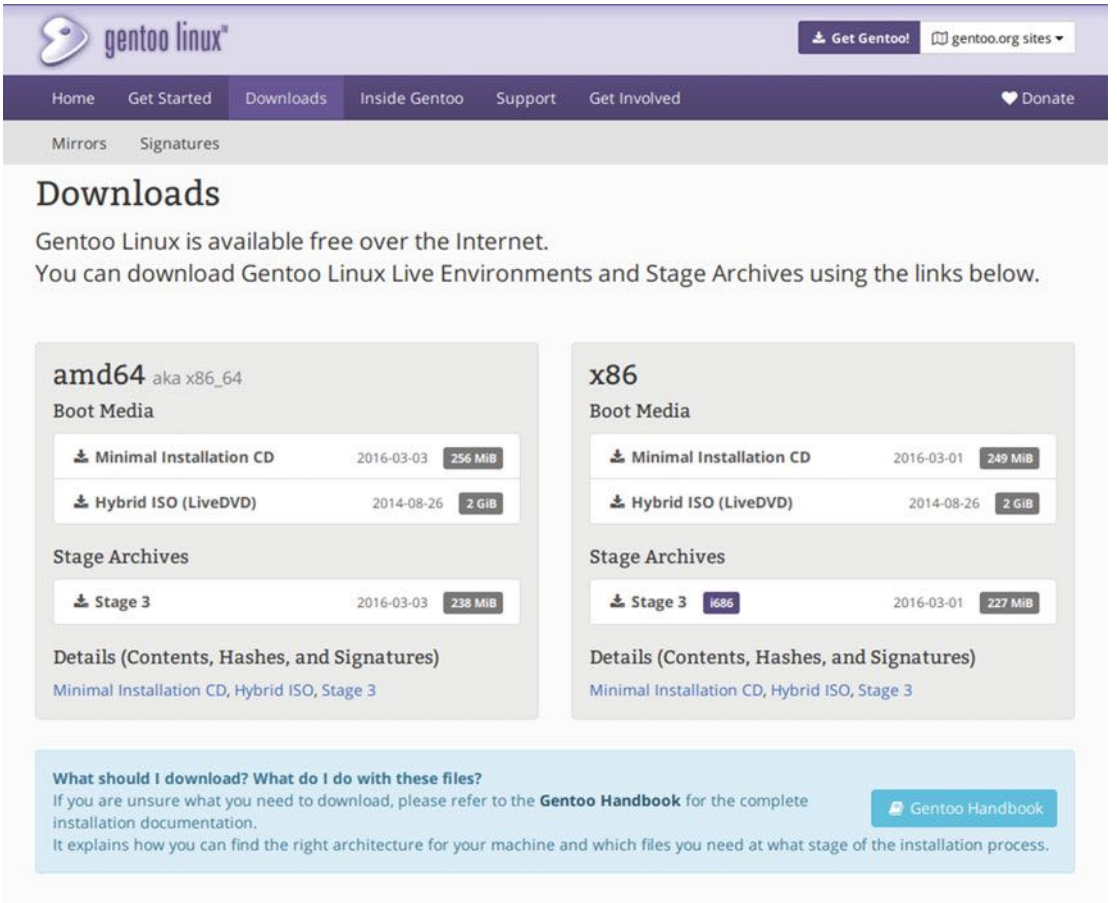


Figure 12-1. *The Gentoo Linux downloads page*

Once the download is finished, you must boot your system from the ISO. The first screen that you get (like the one in Figure 12-2) is harsher that what you've seen in the other distros earlier in this book. And if you don't press any key in 15 seconds, it will try to boot from your hard disk. So if you want to continue, press Enter or, as in other distros, you can opt to change the defaults with the F1 (kernels) and F2 (options) keys. So press Enter to continue.

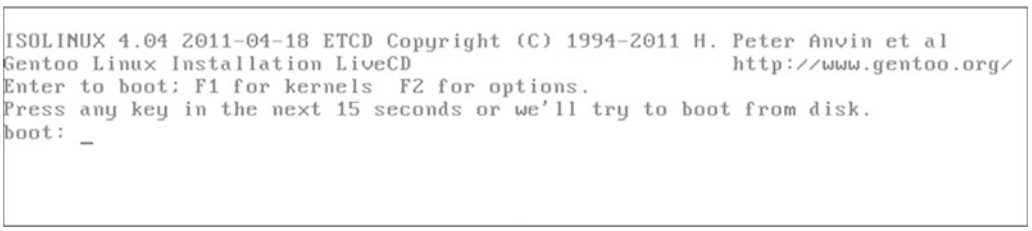


Figure 12-2. *The ISOLINUX boot loader of Gentoo's Minimal Installation CD*

The system will start to boot, but after a moment it will stop to ask you about your keyboard layout, as you can see in Figure 12-3. You only have a few seconds to introduce your layout or press Enter (if you are going to use the default) before it automatically selects the default one.

```
>> Loading keymaps
Please select a keymap from the following list by typing in the appropriate
name or number. Hit Enter for the default "us/41" US English keymap.

 1 azerty   8 croat   15 fi  22 jp  29 pt    36 slovene  43 sf
 2 be       9 cz      16 fr  23 la  30 ro    37 trf
 3 bg      10 de      17 gr  24 lt  31 ru    38 --
 4 br-a    11 dk      18 hu  25 mk  32 se    39 ua
 5 br-l    12 dvorak  19 il  26 nl  33 sg    40 uk
 6 by      13 es      20 is  27 no  34 sk-y  41 us
 7 cf      14 et      21 it  28 pl  35 sk-z  42 wangbe
<< Load keymap (Enter for default):
```

Figure 12-3. *Keyboard layout configuration*

Then it will continue. You end up at a command line prompt (as you do in the Arch Linux installation); see Figure 12-4.

```
livecd login: root (automatic login)
Welcome to the Gentoo Linux Minimal Installation CD!

The root password on this system has been auto-scrambled for security.

If any ethernet adapters were detected at boot, they should be auto-configured
if DHCP is available on your network.  Type "net-setup eth0" to specify eth0 IP
address settings by hand.

Check /etc/kernels/kernel-config-* for kernel configuration(s).
The latest version of the Handbook is always available from the Gentoo web
site by typing "links http://www.gentoo.org/doc/en/handbook/handbook.xml".

To start an ssh server on this system, type "/etc/init.d/sshd start".  If you
need to log in remotely as root, type "passwd root" to reset root's password
to a known value.

Please report any bugs you find to http://bugs.gentoo.org. Be sure to include
detailed information about how to reproduce the bug you are reporting.
Thank you for using Gentoo Linux!

livecd ~ # _
```

Figure 12-4. *The prompt of Gentoo's Minimal Installation CD*

This is where all the work starts. By default, it should have set your Ethernet connection already if your Internet card was detected and you have a DHCP server. If not, follow the instructions or take a look at the Gentoo Handbook. Open the Handbook via the text browser links (better yet, do it in another console). You can check the connection with a ping to `gentoo.org` or open the site with links (remember to exit with the "q" key).

```
# ping gentoo.org
# links gentoo.org
```

The first step is to prepare the disk. In my case, I am going to install the distro in a BIOS system with a 2TB disk, so I'm going to use a MBR partition table (I could use a GPT, but this option could cause some problems in some scenarios). As in the Arch Linux example, it first shows how many devices there are and how they are partitioned.

```
# lsblk --paths
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
/dev/sda          8:0    0     2T  0 disk
/dev/sr0         11:0    1   256M  0 rom  /mnt/cdrom
/dev/loop0        7:0    0 224.5M  1 loop /mnt/livecd
# parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) u GiB
(parted) p
Error: /dev/sda: unrecognised disk label
Model: ATA QEMU HARDISK (scsi)
Disk /dev/sda: 2048GiB
Sector size (logical/physical): 512B/512B
Partition Table: Unknow
Disk Flags:
```

What it shows is a disk without any partition or partition table yet. Let's create the partitions in the disk using parted. Use a very simple partition scheme with only the root partition and the /home partition, both ext4.

```
# parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) u GiB
(parted) mklabel msdos
(parted) mkpart primary ext4 1MiB 50.5GiB
(parted) set 1 boot on
(parted) mkpart primary ext4 50.5GiB 100%
(parted) q
# lsblk --paths
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
/dev/sda          8:0    0     2T  0 disk
├─/dev/sda1       8:2    0    50G  0 part
└─/dev/sda2       8:3    0     2T  0 part
/dev/sr0         11:0    1   256M  0 rom  /mnt/cdrom
/dev/loop0        7:0    0 224.5M  1 loop /mnt/livecd
```

Now format the partitions and mount them.

```
# mkfs.ext4 /dev/sda1
# mkfs.ext4 /dev/sda2
# mount /dev/sda1 /mnt/gentoo
# mkdir -p /mnt/gentoo/home
# mount /dev/sda2 /mnt/gentoo/home
```

■ **Note** Take into account that with the Minimal Installation CD you only have the tools to make ext2, ext3, and ext4 filesystems. You should use another medium to install it if you want to use another filesystem, such as btrfs.

Finally you have a hard disk partitioning scheme, as shown in Figure 12-5.

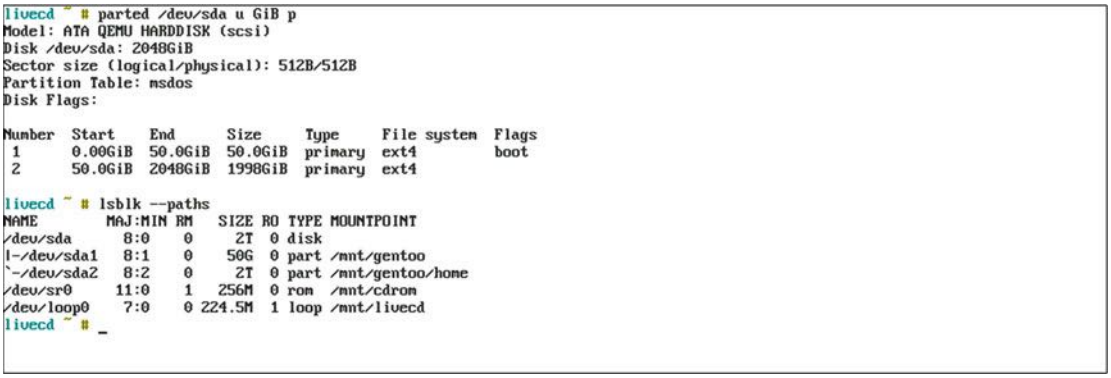


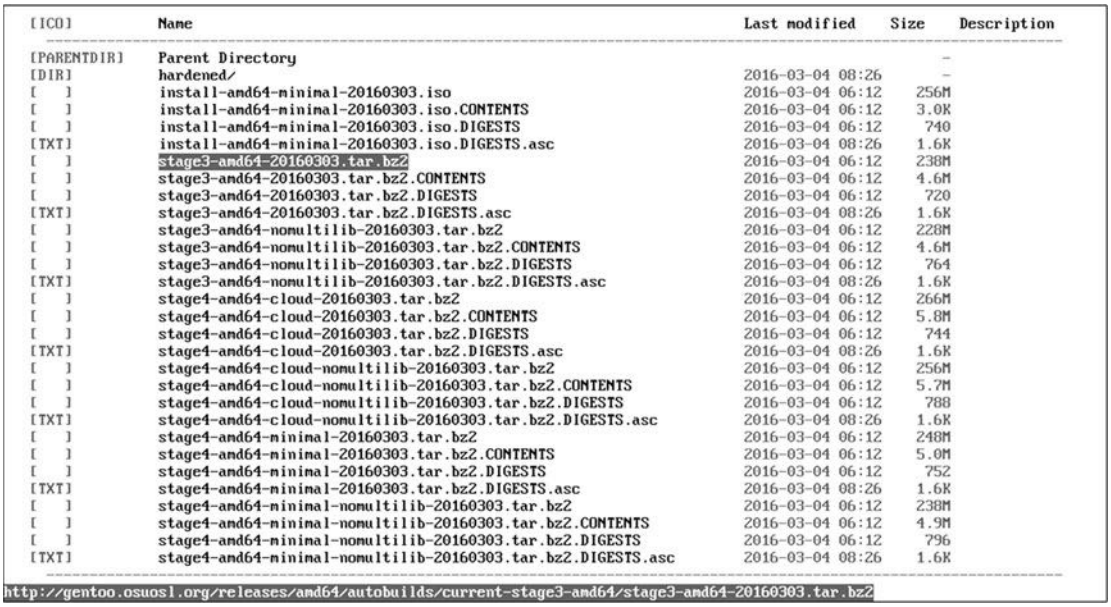
Figure 12-5. *The disk partitioning scheme*

Next, check the date, and correct it if necessary. Note that the syntax is very particular: *MMDDhhmmYYYY*. This stands for month, day, hour, minute, and year.

```
# date
```

The next step is the first one that is properly exclusive of Gentoo. You will download and install the Stage 3 tarball that you see on the Gentoo downloads page. A stage3 is an archive containing a minimal Gentoo environment to continue with the installation. The result of this process is something almost similar to the `pacstrap` script execution in the Arch Linux installation. The best way to download the stage 3 file is to use the links browser to go to the download mirror, choose the one nearest to you, and download the file. You could use another tool like `wget` or `curl`, but you should know the complete URL (and write it). Getting accustomed to this way will make easier to adapt to future changes in the Gentoo web site. Once you choose one mirror and open it, you should look for the `releases/amd64/autobuilds/` directory. Once there, look for the current stage 3 file for amd64 (or your architecture) and download it by pressing "d". It should be a compressed tarball file (with extension `.tar.bz2`) like the one shown in Figure 12-6. There are other stage 3 files there, but you don't care about them. Later, if you grow fond of Gentoo, you'll learn about them for sure.

```
# cd /mnt/gentoo
# links https://www.gentoo.org/downloads/mirrors/
```



[ICO]	Name	Last modified	Size	Description
[PARENTDIR]	Parent Directory		-	
[DIR]	hardened/	2016-03-04 08:26	-	
[ ]	install-amd64-minimal-20160303.iso	2016-03-04 06:12	256M	
[ ]	install-amd64-minimal-20160303.iso.CONTENTS	2016-03-04 06:12	3.0K	
[ ]	install-amd64-minimal-20160303.iso.DIGESTS	2016-03-04 06:12	740	
[TXT]	install-amd64-minimal-20160303.iso.DIGESTS.asc	2016-03-04 08:26	1.6K	
[ ]	stage3-amd64-20160303.tar.bz2	2016-03-04 06:12	238M	
[ ]	stage3-amd64-20160303.tar.bz2.CONTENTS	2016-03-04 06:12	4.6M	
[ ]	stage3-amd64-20160303.tar.bz2.DIGESTS	2016-03-04 06:12	720	
[TXT]	stage3-amd64-20160303.tar.bz2.DIGESTS.asc	2016-03-04 08:26	1.6K	
[ ]	stage3-amd64-nomultilib-20160303.tar.bz2	2016-03-04 06:12	228M	
[ ]	stage3-amd64-nomultilib-20160303.tar.bz2.CONTENTS	2016-03-04 06:12	4.6M	
[ ]	stage3-amd64-nomultilib-20160303.tar.bz2.DIGESTS	2016-03-04 06:12	764	
[TXT]	stage3-amd64-nomultilib-20160303.tar.bz2.DIGESTS.asc	2016-03-04 08:26	1.6K	
[ ]	stage4-amd64-cloud-20160303.tar.bz2	2016-03-04 06:12	266M	
[ ]	stage4-amd64-cloud-20160303.tar.bz2.CONTENTS	2016-03-04 06:12	5.8M	
[ ]	stage4-amd64-cloud-20160303.tar.bz2.DIGESTS	2016-03-04 06:12	744	
[TXT]	stage4-amd64-cloud-20160303.tar.bz2.DIGESTS.asc	2016-03-04 08:26	1.6K	
[ ]	stage4-amd64-cloud-nomultilib-20160303.tar.bz2	2016-03-04 06:12	256M	
[ ]	stage4-amd64-cloud-nomultilib-20160303.tar.bz2.CONTENTS	2016-03-04 06:12	5.7M	
[ ]	stage4-amd64-cloud-nomultilib-20160303.tar.bz2.DIGESTS	2016-03-04 06:12	788	
[TXT]	stage4-amd64-cloud-nomultilib-20160303.tar.bz2.DIGESTS.asc	2016-03-04 08:26	1.6K	
[ ]	stage4-amd64-minimal-20160303.tar.bz2	2016-03-04 06:12	248M	
[ ]	stage4-amd64-minimal-20160303.tar.bz2.CONTENTS	2016-03-04 06:12	5.0M	
[ ]	stage4-amd64-minimal-20160303.tar.bz2.DIGESTS	2016-03-04 06:12	752	
[TXT]	stage4-amd64-minimal-20160303.tar.bz2.DIGESTS.asc	2016-03-04 08:26	1.6K	
[ ]	stage4-amd64-minimal-nomultilib-20160303.tar.bz2	2016-03-04 06:12	238M	
[ ]	stage4-amd64-minimal-nomultilib-20160303.tar.bz2.CONTENTS	2016-03-04 06:12	4.9M	
[ ]	stage4-amd64-minimal-nomultilib-20160303.tar.bz2.DIGESTS	2016-03-04 06:12	796	
[TXT]	stage4-amd64-minimal-nomultilib-20160303.tar.bz2.DIGESTS.asc	2016-03-04 08:26	1.6K	

Figure 12-6. *The directory for the stage 3 tarball file in a Gentoo mirror*

After you download the file, you need to extract and decompress it in the hard disk (remember that now you are in the `/mnt/gentoo` directory that is actually `/dev/sda1`, your future root partition).

```
# tar xvjpf stage3-*.tar.bz2 --xattrs
```

This is going to take a little while, about a minute in my machine. Next, you need to change a few things in the compilation options configuration file that is at `/etc/portage/make.conf`. You can change several things here to optimize the compilation, but let's only change two things: the `CFLAGS` line and a new one for the variable `MAKEOPTS` (see the following code). I have chosen a value of 7 for this last variable because I created a virtual machine with six cores and I want to use them all to speed up the compilation. If I left the default of 2, I would spend a lot of time waiting. To do this, you can use one of these text editors: `nano`, `vi`, and `Emacs`.

```
# vi /mnt/gentoo/etc/portage/make.conf
# cat /mnt/gentoo/etc/portage/make.conf
# These settings were set by the catalyst build script that automatically
# built this stage.
# Please consult /usr/share/portage/config/make.conf.example for a more
# detailed example.
CFLAGS="-march=native -O2 -pipe"
CXXFLAGS="${CFLAGS}"
# WARNING: Changing your CHOST is not something that should be done lightly.
# Please consult http://www.gentoo.org/doc/en/change-chost.xml before changing.
CHOST="x86_64-pc-linux-gnu"
# These are the USE flags that were used in addition to what is provided by the
# profile used for building.
USE="bindist mmx sse sse2"
PORTDIR="/usr/portage"
```

```
DISTDIR="${PORTDIR}/distfiles"
PKGDIR="${PORTDIR}/packages"
MAKEOPTS="-j7"
```

Now it's time to begin the installation of the base system. The first thing is to indicate the mirrors you want to use to download the source code. It's better if the mirror is fast, and usually that means the one closest to you. This is set with a variable in the `make.conf` file, but instead of selecting them by hand, there is a tool that allows you to select them (I suggest two or more) from a list and add them to the file. Next, copy the main Gentoo repository configuration file to its destination and copy the DNS info to keep the Internet connection alive when you restart the system.

```
# mirrorselect -i -o >> /mnt/gentoo/etc/portage/make.conf
# mkdir /mnt/gentoo/etc/portage/repos.conf
# cp /mnt/gentoo/usr/share/portage/config/repos.conf
  /mnt/gentoo/etc/portage/repos.conf/gentoo.conf
# cp -L /etc/resolv.conf /mnt/gentoo/etc/
```

You must also mount certain necessary filesystems to continue working normally in the new environment (which you are going to jail root in a moment).

```
# mount -t proc proc /mnt/gentoo/proc
# mount --rbind /sys /mnt/gentoo/sys
# mount --make-rslave /mnt/gentoo/sys
# mount --rbind /dev /mnt/gentoo/dev
# mount --make-rslave /mnt/gentoo/dev
```

Let's get into the new environment with a jail root (chroot). Then all the commands will apply to that environment that is set with your hard disk as the base, and not in the live one. The last line works as a good hint to remind you that you are in a chroot environment.

```
# chroot /mnt/gentoo /bin/bash
# source /etc/profile
# export PS1="(chroot) $PS1"
```

Next, configure Portage. First, get a snapshot of the current Portage tree and then sync the latest updated version with your new local tree. Ignore the errors about the `/usr/portage` location and the warnings about the news (this is a very nice feature and I recommend you look at it in the documentation).

```
# emerge-webrsync
# emerge --sync --quiet
```

You should have to select a profile now that, without diving into it now, should define the global settings for compilation of the future packages and more. Since I want to show you something similar to the Arch Linux installation, the right choice in this case is to select the "desktop" profile.

```
# eselect profile list
Available profile symlink targets:
  [1]   default/linux/amd64/13.0 *
  [2]   default/linux/amd64/13.0/selinux
  [3]   default/linux/amd64/13.0/desktop
  [4]   default/linux/amd64/13.0/desktop/gnome
  [5]   default/linux/amd64/13.0/desktop/gnome/systemd
```

```
  [6]   default/linux/amd64/13.0/desktop/kde
  [7]   default/linux/amd64/13.0/desktop/kde/systemd
  [8]   default/linux/amd64/13.0/desktop/plasma
  [9]   default/linux/amd64/13.0/desktop/plasma/systemd
  [10]  default/linux/amd64/13.0/developer
  [11]  default/linux/amd64/13.0/no-multilib
  [12]  default/linux/amd64/13.0/systemd
  [13]  default/linux/amd64/13.0/X32
...
# eselect profile set 3
```

Next, you could/should configure the `USE` variable to set the global options to compile the packages. It's an important step, but in this case you are going with the defaults. I suggest you get comfortable with Gentoo and read the docs carefully before dealing with this.

Let's take another step, a traditional one in all distros: setting the time zone and the language and local values. The first command will show you the directory tree where the files for the time zones are stored; then you must set one and update the date automatically. Next, edit the `locale.gen` file (in this environment you only have nano to edit the files) and uncomment the language(s) that you want to use (in this case, `enUS.UTF-8 UTF-8`). Finally, use `eselect` again to select the system-wide language and reload the environment.

```
# ls /usr/share/zoneinfo
# echo "America/New_York" > /etc/timezone
# emerge --config sys-libs/timezone-data
# nano -w /etc/locale.gen
# locale-gen
# eselect locale list
Available targets for the LANG variable:
  [1] C
  [2] POSIX
  [3] en_US.utf8
  [ ] (free form)
# eselect locale set 3
# env-update
# source /etc/profile
# export PS1="(chroot) $PS1"
```

It's time for the most daunting task among Linux newcomers, the stuff of many rumors and myths: installing and configuring a Linux kernel. Sorry, but you must do it, so let's do it in the least painful way possible: automatically. I assure you, it is not that scary or terrible, but I understand that you may need some experience to gain confidence. First, you need to install it.

```
# emerge --ask sys-kernel/gentoo-sources
```

This will take a while, about three minutes on my system. To configure the kernel, you are going to use a tool named `genkernel` to do it automatically. First, you need to install the tool.

```
# emerge --ask sys-kernel/genkernel
```

Now you need to edit the /etc/fstab file and comment the /boot and swap lines since you didn't create those partitions; you also need to create a new line for the /home partition.

```
# nano -w /etc/fstab
# cat /etc/fstab
# /etc/fstab: static file system information.
#
# noatime turns off atimes for increased performance (atimes normally aren't
# needed); notail increases performance of ReiserFS (at the expense of storage
# efficiency). It's safe to drop the noatime options if you want and to
# switch between notail / tail freely.
#
# The root filesystem should have a pass number of either 0 or 1.
# All other filesystems should have a pass number of 0 or greater than 1.
#
# See the manpage fstab(5) for more information.
#

# <fs>          <mountpoint>    <type>      <opts>      <dump/pass>

# NOTE: If your BOOT partition is ReiserFS, add the notail option to opts.
#/dev/BOOT      /boot       ext2        noauto,noatime  1 2
/dev/sda1       /           ext4        noatime     0 1
/dev/sda2       /home       ext4        noatime     0 2
#/dev/SWAP      none        swap        sw      0 0
/dev/cdrom      /mnt/cdrom  auto        noauto,ro   0 0
#/dev/fd0       /mnt/floppy auto        noauto      0 0
```

Next, configure the kernel automatically with genkernel.

```
# genkernel all
```

This will take a while, about seventeen minutes on my system. This first not-fast compilation shows you what it will take to compile almost everything. Next, configure the modules that you need to start automatically via /etc/conf.d/modules. In this case, let's assume that you don't need any additional modules. But I suggest that you install the firmware because you never know, and some drivers require it (especially in laptops).

```
# emerge --ask sys-kernel/linux-firmware
```

Continue configuring the system. Let's set up the network now. The first step is to change the default hostname of "hostname" to the one you want. After that, configure the Internet to use DHCP, and make it the network that starts at boot.

```
# nano -w /etc/conf.d/hostname
# cat /etc/conf.d/hostname
# Set the hostname variable to the selected host name
hostname="does_pc"
# emerge --ask --noreplace net-misc/netifrc
# nano -w /etc/conf.d/net
# cat /etc/conf.d/net
config_eth0="dhcp"
```

```
# cd /etc/init.d
# ln -s net.lo net.eth0
# rc-update add net.eth0 default
```

You must set a password for the root user. Remember to use a strong password, especially for this user.

```
# passwd
```

If you have a keyboard layout different from the default ANSI US, and you selected "other" when you booted from the minimal installation image, you should now configure the keyboard layout to avoid problems (like not being able to recreate the password later depending on the characters you used) when you reboot the system. To do so, you must edit the `/etc/conf.d/keymaps` file.

```
# nano -w /etc/conf.d/keymaps
```

Let's install some tools necessary for the system. Let's start with the logging tool to log what happens in your system, then a cron tool, a file indexing tool, some file system tools, and a DHCP client. Here you're using the classic or easiest tools for each task, not necessarily the best ones.

```
# emerge --ask app-admin/sysklogd
# emerge --ask app-admin/logrotate
# rc-update add sysklogd default
# emerge --ask sys-process/cronie
# rc-update add cronie default
# emerge --ask sys-apps/mlocate
# emerge --ask sys-fs/e2fsprogs
# emerge --ask net-misc/dhcpcd
```

You are getting close to the moment of booting up your new Gentoo system for the first time. First, you need to install and configure a bootloader, Grub2 in your case. Prepare to wait again; this will take a while to compile (about 10 minutes on my system).

```
# emerge --ask sys-boot/grub:2
# grub2-install /dev/sda
# grub2-mkconfig -o /boot/grub/grub.cfg
```

Finally, exit and reboot the system. Cross your fingers!

```
# exit
# cd
# umount -l /mnt/gentoo/dev{/shm,/pts,}
# umount /mnt/gentoo{/boot,/sys,/proc,}
# reboot
```

And voila! You have started your brand new Gentoo Linux system for the first time! After it completes the boot up, you should find a screen like the one in Figure 12-7.

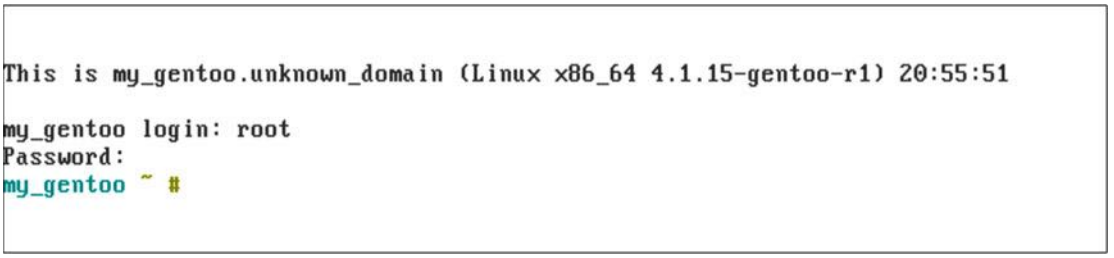


Figure 12-7. *The first boot up of your brand new Gentoo system*

All that work and you end up at a simple command prompt again! Well, from here it is up to you; you can build what you want. The profile you selected previously and the dependencies of each package are going to "spend your time." I'll give you an example: in my new system, I didn't have the tools installed that I had in the minimal installation environment, like emacs, vi, or links. I want to have links to be able to access the documentation from the terminal in case something goes wrong. Well, this package has as a dependency on certain components of the X windows server. It took 20 minutes to install a text mode browser. Do the math: how long would it take to install Gnome or KDE from source? On the other hand, you only have to install those dependencies once.

But first you must create a new user. To do that, you need to log in with the root account and then add it. Add the user to the group wheel (to use `su` and `sudo`) and `audio`.

```
# useradd -m -G users, wheel,audio -s /bin/bash johndoe
# passwd johndoe
```

You can now safely remove the stage3 tarball file from your system.

```
# rm /stage3-*.tar.bz2*
```

From here, things get a little more complicated and they depend heavily on your specific hardware. If you want a similar scenario as the one we set up in Arch Linux, the first step is to install a video driver for X.org. But here is where the things get messy, because depending on your graphics card, emerge is going to tell you that you need to add some options to the USE variable, and these options are not always the same. So, you should add those variables in the `/etc/portage/make.conf` file and also add a new pair of variables to define the input devices and the video card. In this case, let's suppose that you have an Intel card: `INPUT_DEVICES="evdev synaptics"` and `VIDEO_CARDS="intel"`. And it gets better; after this you probably need to recompile your kernel and (wait for it) update your whole system because of the changes in the USE variable (and it could take a while!). And finally, install the X.org server. As a rough resume,

```
# emerge --ask x11-drivers/xf86-video-intel
# genkernel all
# emerge --ask x11-base/xorg-server
```

Next, let's install a display manager. In this case, even though it's obsolete, I've chosen Slim because it's light and takes less time to compile. Next, let's install a tiling window manager, Awesome WM. But don't forget that you have to also configure them; look at the documentation to do so. You should then init your display manager to get a login screen (Figure 12-8) and finally log in to get the window manager (Figure 12-9).

```
# emerge --ask x11-misc/slim
# vi /etc/conf.d/xdm
# cat /etc/conf.d/xdm
DISPLAYMANAGER="slim"
# rc-update add xdm default
# emerge --ask x11-wm/awesome
$ mkdir -p ~/.config/awesome/
$ cp /etc/xdg/awesome/rc.lua ~/.config/awesome/rc.lua
# emerge --ask xterm
# emerge --ask feh
# /etc/init.d/xdm start
```

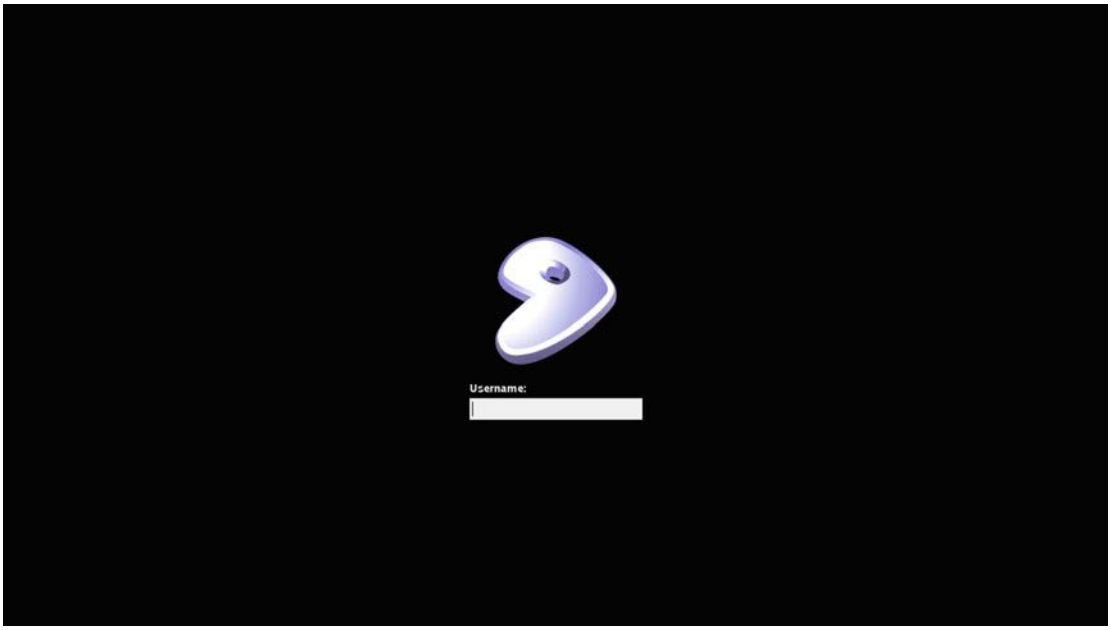


Figure 12-8. *The Gentoo Slim login manager*

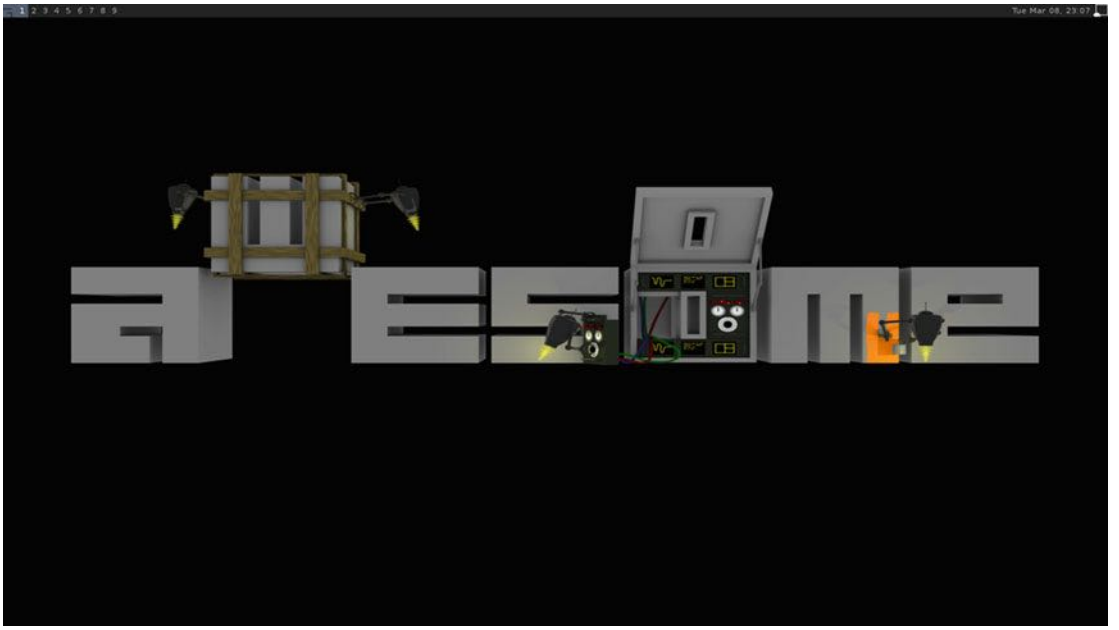


Figure 12-9. *The Awesome WM window manager in Gentoo*

There are still a lot of things to do to get a reasonable system, so be prepared to work with it on a daily basis. At this pace, I would need at least another chapter to cover all of that here.

Don't worry if you fail the first time and don't get a working Gentoo system; it's normal. There are a lot of things to take into account; something as simple as a typo could keep the system from booting. This is essentially the way Linux was installed in almost every distro years ago, and the origin of so many rumors and myths about the harshness of Linux. If you read the previous chapters, you already know that there is a huge difference between installing Linux in this way and installing a user friendly distro like Ubuntu. But I'm sure you've already realized too that the level of control and customization that you can get with an advanced distro like Gentoo or Arch is miles away from what you can get with a distro like elementary OS.

Maintenance

In Gentoo, all of the essential maintenance is done with the Portage/emerge tool. Being a rolling release, you can keep it permanently updated. Basically you don't need to upgrade it in the traditional way.

Updating and Managing Apps

Although I'm going to show you an easy way to perform these tasks, note that Portage is a very powerful tool. It lets you completely control how you compile and install every package. There are other ways of doing this, with slightly different commands. If you want to use this distro, you need to get familiar with this tool through Gentoo's docs. As you know already, to manage Portage you use the emerge tool and a lot of patience.

Installing an app using emerge can be as simple as this:

```
# emerge firefox
```

But usually it is better to first search for the app and then install it in this way:

```
# emerge --search firefox
# emerge --ask www-client/firefox-bin
```

If you want to remove a package:

```
# emerge --unmerge firefox
```

To update your system, it's also very simple. First, sync the Portage tree. Then, launch the automate update.

```
# emerge --sync
# emerge --uD world
```

I mentioned that you can do all of this with graphical tools like Porthole, but usually the people that are willing to go through the effort of installing this distro from the command line are the least inclined to later use a graphical tool to maintain it. The users that choose Gentoo are usually pursuing total control and fine-grain settings. Porthole is a very good tool, but it can't compete with emerge.

Upgrading

As with Arch Linux, upgrading Gentoo has totally different meaning from that for traditional distros. Even though the packages are always updated, sooner or later there will be changes in the core of the distro. Arch has a special way to do this: when a change is made, it is announced and you are given instructions about the changes that you must make (usually by hand). This modality has great inconveniences, because if you leave a system unattended for a long time, reconstructing these changes could be very painful or even impossible without doing a fresh install. In other words, Arch Linux waits for no one. If you want it updated, you have to keep the pace.

But Gentoo is very good at this. It solved this problem in a smart way that allows you to make "upgrades" smoothly and at your pace. The instrument to do this is the "profile," which if you remember from the "Installation" section is also used to determine the type of systems that you want to build in a certain manner. Also, remember that the new Gentoo releases are done weekly, and some of these releases bring a new profile. When a new release comes with a new profile, you can choose to migrate to the new profile or perform only the package updates. But don't get me wrong; these migrations can be easy or a titanic job. You can safely ignore one profile migration and after it becomes deprecated, do the task. It is not a perfect solution, but at least it is a solution. You can leave a system unattended a long time, and then with patience, time, dedication, and care, you can end up with an updated system without having to make a fresh install. I suggest you look at the documentation to get a complete understanding of this (this is a little lie; you never really understand these things until you do them yourself) at the Gentoo Wiki at https://wiki.gentoo.org/wiki/Upgrading_Gentoo.

Pros and Cons

The following list is some things that I personally see as the pros and cons of the Gentoo Linux distribution. There's always room for discussion in this matter, but I've been as objective as possible.

Pros

- It is a rolling release distro, so it's always on the bleeding edge.
- It has great documentation.
- It has a good and helpful community of advanced users.
- You can compile the packages optimized to your hardware and needs.
- It's fully customizable, offering amazing flexibility and total control.
- It has an astounding package manager, Portage.
- It has a big focus on security. Although it's up to you, you have the tools and docs.
- It is an original distribution, not a derivative.
- It supports multiple architectures, even obsolete ones.
- If you stick with this distro, you'll learn a lot about Linux.

Cons

- To compile almost all the packages could be a waste of time.
- Thanks to modern hardware, the performance gain is negligible (with exceptions).
- It's not user friendly. It's one of the most demanding of all distros.
- Maintaining it costs time and dedication.
- It has no commercial support.
- It does not focus on aesthetics.
- It's unstable by nature (unless your learn enough and have a lot of free time).

Summary

Here you saw a distro that allows the maximum level of customization possible. To go further you could only take one more step: build your own distribution. Gentoo is a distribution for advanced, patient, and very meticulous users. It's an example of the freedom and power that Linux can provide to the user at the cost of the necessary knowledge.

In the next chapter, you are going to see an historical and ancient distribution that has not changed much through time: Slackware. It's known for being harsh and unpractical, but it shares some points with Gentoo.

CHAPTER 13

Slackware

Slackware is the most ancient distribution alive. It was born a few months before Debian, which seems hard to believe, seeing the evolution of both. A lot of people think Debian is much too conservative; these same people would call Slackware a dinosaur. Yes, Slackware has not changed too much over the years, but far from being ashamed, the Slackware community is proud of this. Slackware claims to be the most "Unix-like" Linux distribution.

History

Slackware and Debian don't just share a long history as Linux distros; they also share a common origin: the SLS (Softlanding Linux System) distro. While Debian was created as a better alternative to SLS, Slackware was created as an evolution of it.

Patrick Volkerding was a student at the Minnesota State University Moorhead (MSUM) in 1993 when his AI professor asked him to install SLS in the computer lab to get the benefit of its better LISP interpreter. Volkerding did so, and then he improved the distro itself by fixing bugs and making modifications for future installations at the lab. Later, seeing that the SLS distro didn't have any more releases, that the bugs remained unfixed, and that there was great demand on the Internet for a good product, he asked if anyone was interested in his work. After many positive responses, and encouraged by his friends at the university, he uploaded to one of the university's anonymous FTP servers the first Slackware 1.00 release on July 17, 1993.

Today, Volkerding is still the main developer of Slackware; he is known as "The Man" or Slackware's BDFL (Benevolent Dictator for Life). And Slackware continues to be a small but popular distribution.

Note The origin of the Slackware name comes from an internal joke related to the slack term as homage to J.R. "Bob" Dobbs, the figurehead of the Church of SubGenius (a parody religion which was very popular in Internet subculture in those days). It was never intended to be the definitive name of the distro; he kept the name when he first released the distro with the intention that nobody would take it too seriously.

Philosophy

The philosophy of Slackware is difficult to understand without much knowledge of the history of UNIX and Linux itself. While the distro claims to be conservative in terms of simplicity, stability, and power (and yet user friendly), a new user coming from a friendly OS/distro would probably have a very different opinion about that. First of all, most people would not considerate Slackware a friendly distro; in fact, quite the opposite. And they would find it hard to consider it simple and powerful since it lacks tracking package

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_13

dependencies. But if you see it from the perspective of Slackware's author and the community, it seems easier to understand. They didn't want to stray too far from the original UNIX design, like the current BSDs do, for a simple and powerful reason. An experienced UNIX user would feel almost at home with Slackware, and it would be very easy for her jump in and use it. Regardless of whether you share this opinion, you must recognize that time supports their decision; other friendly distros disappeared a long time ago.

Thus, Slackware tries to follow the KISS principle (explained in Chapter 11), being conservative in its core design and not introducing changes in its packages. Broadly speaking, Slackware is a BSD with a Linux kernel. In that search for simplicity and purity, each package is configured independently; there is no global configuration. The majority of the configuration is made through simple text files. Nothing is predefined by default; you are supposed to know what you are doing and what you want.

Don't get me wrong; being conservative about its core design does not mean it lacks innovations or necessary changes (like support for 64-bit architecture). Slackware tries to be in sync with the times and supports "recent" innovations like UEFI or the Btrfs file system, and the packages are very recent when a new release appears. But generally Slackware follows the motto of "If it's not broken, don't fix it."

Distro Selection Criteria

Now that you know a little of Slackware's history, let's see how this particular distribution fares on the selection criteria from Chapter 2.

Purpose and Environment

Slackware is a general purpose distro with a unique version for all purposes and environments. Still, there are two different branches of flavors that you can install: the stable one and the "current" one, which works as the development tree of the distro.

Support

Although Slackware (and Volkerding himself) are supported by the sales from the store, and it once had a commercial distribution (for a brief period of time), it is not a purely commercial distro (you still can get it free), and it does not have commercial support at all. Support only comes from its developers and its small, but loyal, community. The ways to get support from the community are the following:

- **The Slackware Book Project**: `www.slackbook.org`
- **Wiki**: `http://docs.slackware.com/`
- **FAQ**: `http://docs.slackware.com/slackware:faq`
- **Slackware forum at Linux Questions**: `www.linuxquestions.org/questions/forumdisplay.php?forumid=14`
- **Mailing lists**: `www.slackware.com/lists/`
- **IRC**: `#slackware` at `irc.freenode.net`

User Friendliness

Slackware is not for beginners. Although it's easier than Arch or Gentoo, it is not a user-friendly distro. You still need to know a lot of things about the command line and Linux itself, and you should be comfortable with how Slackware manages its packages. Traditionally the Slackware user was stereotyped as a typical "neck beard" system administrator and alike. While it is true that Slackware users are usually technically well-versed, it is not true that it is as hard as portrayed. A lot of users sweat more with Gentoo, for example.

Stability

Slackware, due to its inherent simplicity, proven design, and the frequency of its releases, is a very stable distro. It follows a standard release model and there is no fixed schedule; a new release appears when is ready, well tested and stable enough (it's very well known for this). As a result, the current release, the 14.1, is from 2013 and the packages are outdated. A beta version of the future 14.2 release is currently in progress, and it will have fresh packages. Some people prefer to use the "current" development version as their daily system because of this. And for the support part, there is no official policy; from time to time, support is suddenly dropped from older releases.

Hardware Support

As with Arch and Gentoo, you must be willing to learn how to support hardware not included on the kernel or official packages/drivers, and depend heavily on third party maintained builds/binaries.

Aesthetics

There is no focus on aesthetics, aside from the DVD packaging and merchandising of the Slackware store.

Desktop Environment

The de facto official desktop of Slackware is KDE but it also gives you the option to install Xfce and several window managers (Fluxbox, Blackbox, WindowMaker, FVM2, and TWM).

Init System

The init system Slackware uses is a BSD-style one based on rc files, which is also compatible with traditional System V (sysv) scripts. There is no plan in the future to make a migration to systemd; a great part of the community is against a future systemd implementation in the distro.

Package Management System

Slackware has a very particular way of managing packages. These packages are simple tarballs (compressed or not), in which files are usually simply extracted in their respective paths followed by the execution of several `doinst.sh` scripts. There is no management of dependencies at all; you are supposed to take care of it. There are two official tools for managing packages in Slackware: `pkgtools` and `slackpkg`. The latter is the most recent one and it can manage network packages, not just local packages like `pkgtools`.

There are several unofficial package repositories for Slackware; SlackBuilds.org at `http://slackbuilds.org` is probably the most popular. There are also several unofficial tools to manage packages, like `swaret` or `slapt-get`, which both have dependency resolution. The packages from these repositories can be built from

source; compiling them, and also allows you to create your own packages. In fact, these packages come with a recipe (similar to Gentoo's ebuilds or Arch's pkgbuild) to build them, and the SlackBuilds repository is based on this.

Architecture

Slackware officially supports the Intel/AMD architectures in 32/64 bits, but there are also ports to the ARM and S/390 architectures.

Security/Anonymity

As with Arch or Gentoo, security depends more on the user than in other distros (although the user is always the weakest link in any OS), but the small official package base, and the stability and age of the packages helps a little to control vulnerabilities. Also, the packages are always updated to fix a security vulnerability; they are announced through the security mailing list. And a big part of the Slackware community is technically adept so they can handle these issues better than most.

Principles and Ethics

There is nothing written down about principles and ethics; you are expected to manage them for yourself as you wish. There are non-free binary blobs in the kernel, some non-free software, and nothing to stop you from installing third party packages in your distro.

Live CD

There is no official Live CD image for this distro, but Eric Hameelers (also known as AlienBOB, and one of the greatest contributors to Slackware) is currently working on a project to have a Slackware Live CD Edition. It will probably be released with the Slackware 14.2 release, which is currently in beta status.

Professional Certification

There is no specific professional certification available for Slackware.

Installation

Installing Slackware is a bit awkward. It's not like Debian or a friendly distro like Ubuntu, and it's not like Arch or Gentoo. It's in a sort of middle ground between the two experiences. It uses a setup tool that helps to make great part of the installation process easy, but it requires the use of the command line in the beginning and at the end. It reminds me a little of the Arch installation when it uses an ncurses tool (a text-based graphical environment) to install it. And of course it reminds me of back when almost all of the Linux distros had a very similar way of installation. In other words, Slackware is easier to install than Gentoo or Arch, but it is still a vintage experience and one that is harsh for the beginner.

Start by obtaining the ISO image file to install the distro. Go to the Slackware web site's "Get Slack" section at `www.slackware.com/getslack/` (see Figure 13-1). Here you must choose between going to the store and purchasing a physical DVD/CD to install the distro or downloading an ISO image via BitTorrent or a Slackware mirror. I usually choose to go to the most popular path, which is to download an ISO image from a mirror. So do that and click the mirrors link to get redirected to the Mirrors page (shown in Figure 13-2).

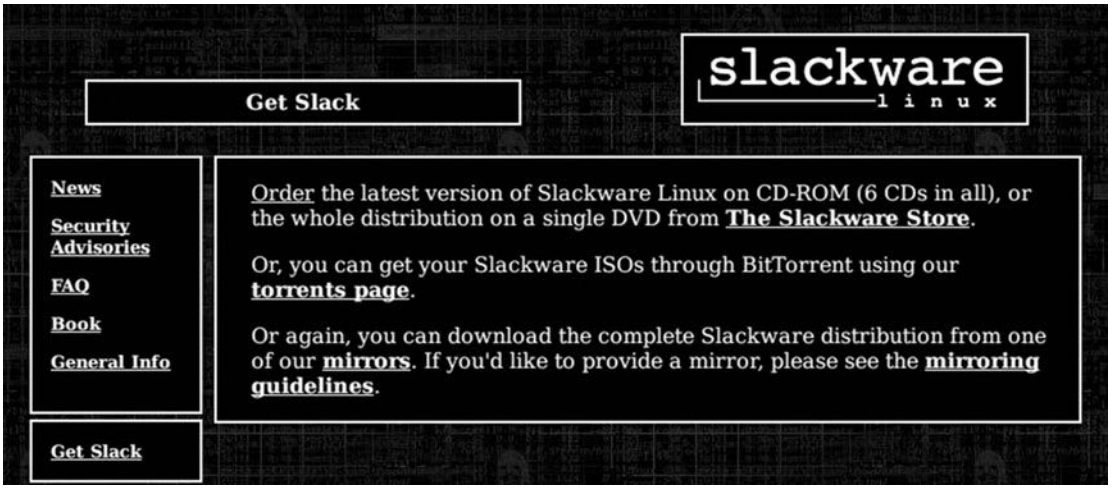
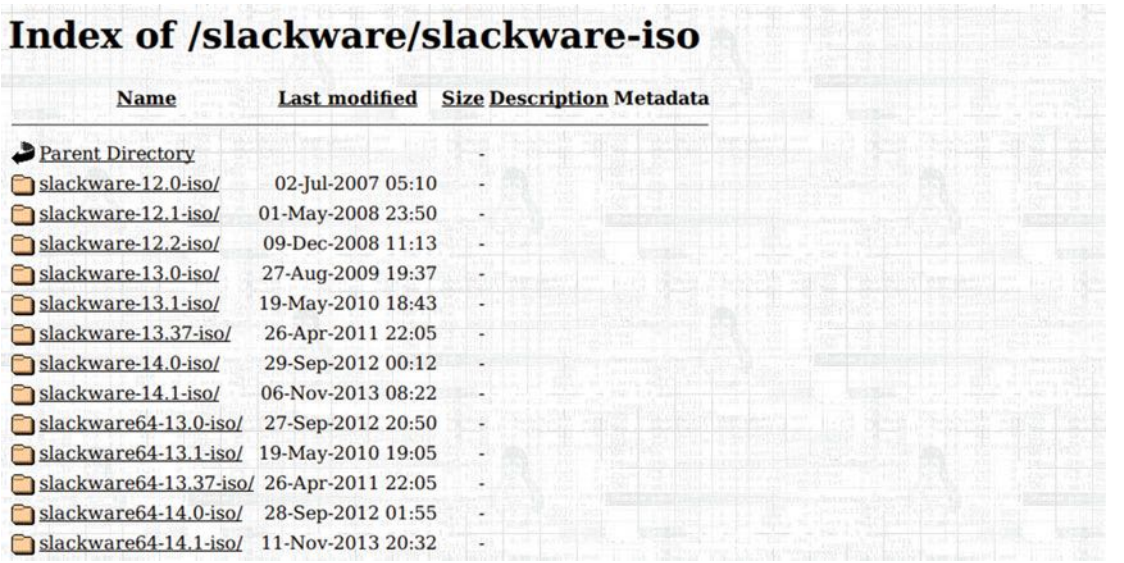


Figure 13-1. *The section of the Slackware web site where you can obtain the ISO image*

Figure 13-2. *The mirrors section of the Slackware web site*

Here you can select a mirror from “Available Mirrors” or get an ISO image directly from “ISO Images.” Both of these links are shown in the side menu. If you follow the latter link, you end up in the directory tree of the main mirror, as you can see in Figure 13-3. The last version is currently z4.1 from November 2013 and there are 32- and 64-bit versions. Go to the last folder where the 64-bit version is and download the DVD ISO image file that is inside.

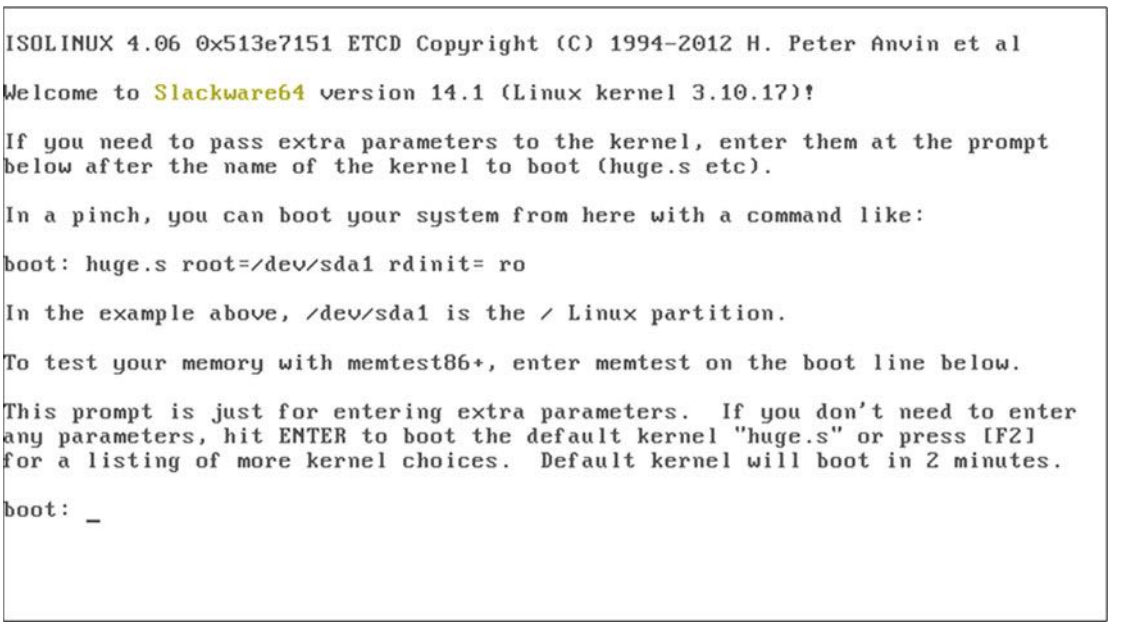


Index of /slackware/slackware-iso

Name	Last modified	Size	Description	Metadata
Parent Directory		-		
slackware-12.0-iso/	02-Jul-2007 05:10	-		
slackware-12.1-iso/	01-May-2008 23:50	-		
slackware-12.2-iso/	09-Dec-2008 11:13	-		
slackware-13.0-iso/	27-Aug-2009 19:37	-		
slackware-13.1-iso/	19-May-2010 18:43	-		
slackware-13.37-iso/	26-Apr-2011 22:05	-		
slackware-14.0-iso/	29-Sep-2012 00:12	-		
slackware-14.1-iso/	06-Nov-2013 08:22	-		
slackware64-13.0-iso/	27-Sep-2012 20:50	-		
slackware64-13.1-iso/	19-May-2010 19:05	-		
slackware64-13.37-iso/	26-Apr-2011 22:05	-		
slackware64-14.0-iso/	28-Sep-2012 01:55	-		
slackware64-14.1-iso/	11-Nov-2013 20:32	-		

Figure 13-3. *The directory tree in the main Slackware mirror*

When you boot for the first time from the ISO image, you will see the screen shown in Figure 13-4. It's a simple text screen with instructions on how to boot the system. If you do not want to use any option, you must press Enter and wait two minutes to boot.



```
ISOLINUX 4.06 0x513e7151 ETCD Copyright (C) 1994-2012 H. Peter Anvin et al

Welcome to Slackware64 version 14.1 (Linux kernel 3.10.17)!

If you need to pass extra parameters to the kernel, enter them at the prompt
below after the name of the kernel to boot (huge.s etc).

In a pinch, you can boot your system from here with a command like:

boot: huge.s root=/dev/sda1 rdinit= ro

In the example above, /dev/sda1 is the / Linux partition.

To test your memory with memtest86+, enter memtest on the boot line below.

This prompt is just for entering extra parameters.  If you don't need to enter
any parameters, hit ENTER to boot the default kernel "huge.s" or press [F2]
for a listing of more kernel choices.  Default kernel will boot in 2 minutes.

boot: _
```

Figure 13-4. *The first screen after the boot from the Slackware ISO image*

After a few seconds, it will ask you for the keyboard layout (shown in Figure 13-5). If you use a layout different from US ANSI, press 1 and follow the instructions (it will show you a menu where you can select a layout from various options). Just press Enter and continue.

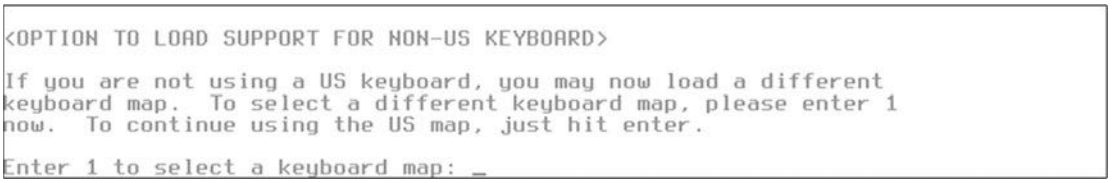


Figure 13-5. *The option to select a different keyboard layout*

After this you go directly to a terminal session where you have to log in as root without a password (shown in Figure 13-6). During this process, before login and after, you are provided with basic instructions and the steps you should follow. The first thing to do is prepare the disk before launching the setup tool. The instructions suggest using classic tools like `fdisk` or `gdisk`, but I'm going to use `parted`. I like to use this tool because is probably the most versatile one since it supports both MBR and GPT partitions. Again, I'm using a 2TB disk and making a simple partition scheme (`/` and `/home`) with a MBR partition table. Note that the kernel version is too old (3.10.17) and also the same happens with some tools like `lsblk` where you cannot use the option `--paths`.

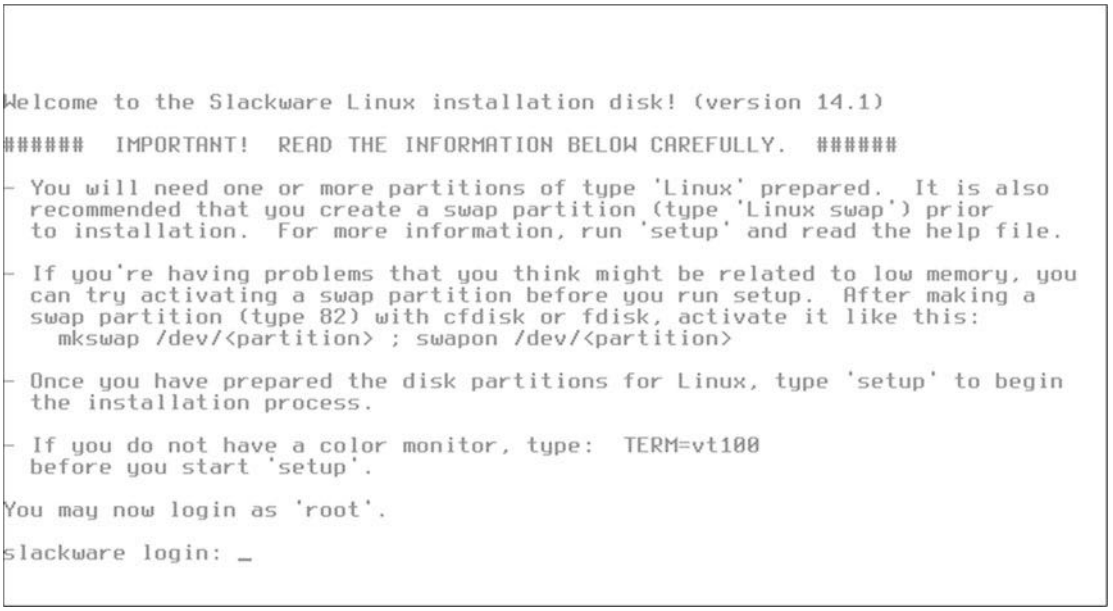


Figure 13-6. *The terminal login prompt where you end up after boot up*

```
# parted /dev/sda
(parted) mklabel msdos
(parted) mkpart pri ext4 1MB 50GB
(parted) mkpart pri ext4 50GB 100%
(parted) set 1 boot on
```

```
(parted) u GiB
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 2199GB
Sector size (logical/physical): 512B/512B
Partition table: msdos
Disk Flags:

Number   Start   End     Size    Type     File system  Flags
 1       1049kB  50.0GB  50.0GB  primary  ext4         boot
 2       50.0GB  2199GB  2149GB  primary  ext4
(parted) q
# lsblk
NAME       MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda          8:0    0     2T  0 disk
!—sda1       8:1    0    50G  0 part
`sda2        8:2    0     2T  0 part
sr0         11:0    1   2.3G  0 rom
```

After you finish partitioning the disk, you can run the `setup` program that will help you to almost complete the rest of the installation. A text-based interface (using the ncurses library) will appear, as you can see in Figure 13-7.

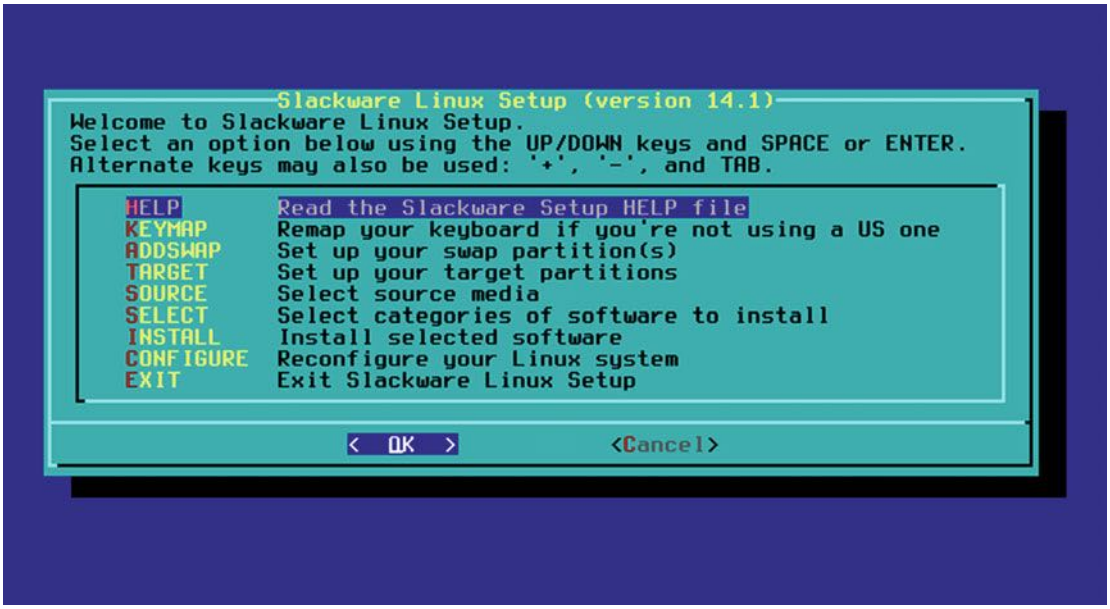


Figure 13-7. *The setup tool to help you with the installation*

It's a menu with multiple options, the Help entry being the first one. This help gives you instructions mostly about the first part of the installation (preparing the disk), which you just did. You can safely ignore the next two options if you already set your keyboard layout and if you are not using a swap partition, as in this case. So go directly to the Target entry and press Enter. Here you can see the available partitions (shown in Figure 13-8). Select each one of them and the tool will ask you to format it (Figure 13-9) and then select

one of the available file systems (Figure 13-10). Choose the quick format and the ext4 file systems for both partitions. For the first partition, the tool is going to assume that it is the / partition, and for the last one you must enter the mount point (in this case, /home). At the end, it shows a resume of how the disk partitions are formatted and mounted, which is shown in Figure 13-11.

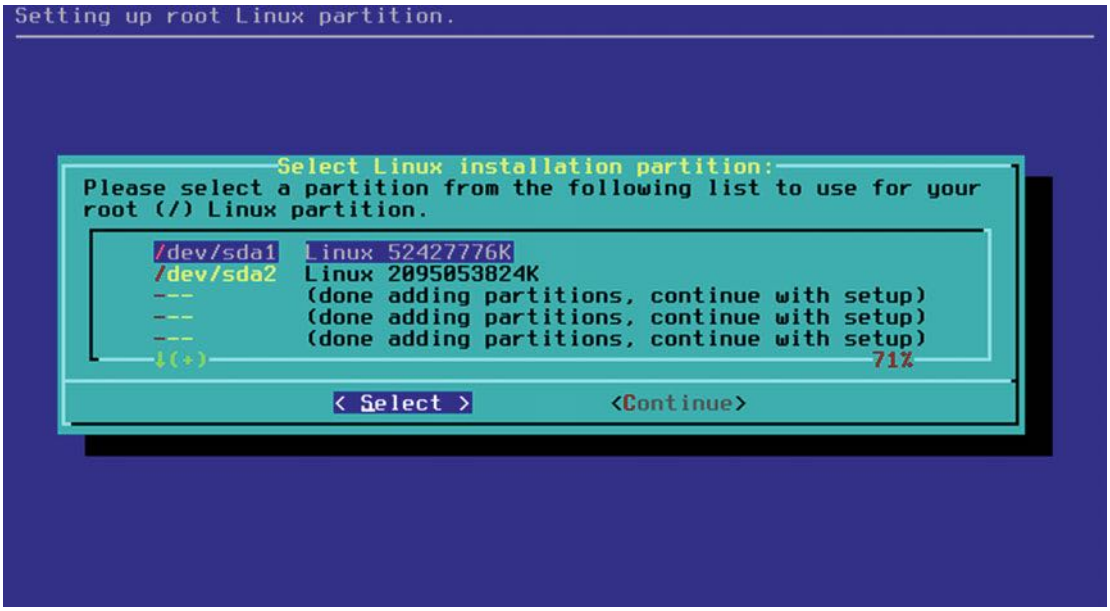


Figure 13-8. *The current partition on the disk*

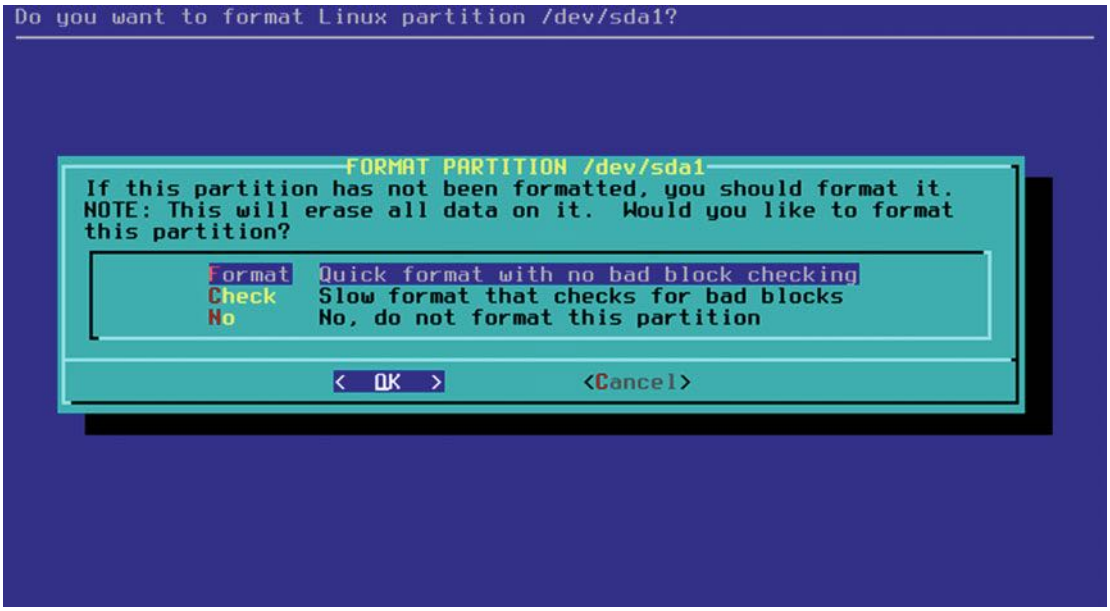


Figure 13-9. *Choose the type of formatting*

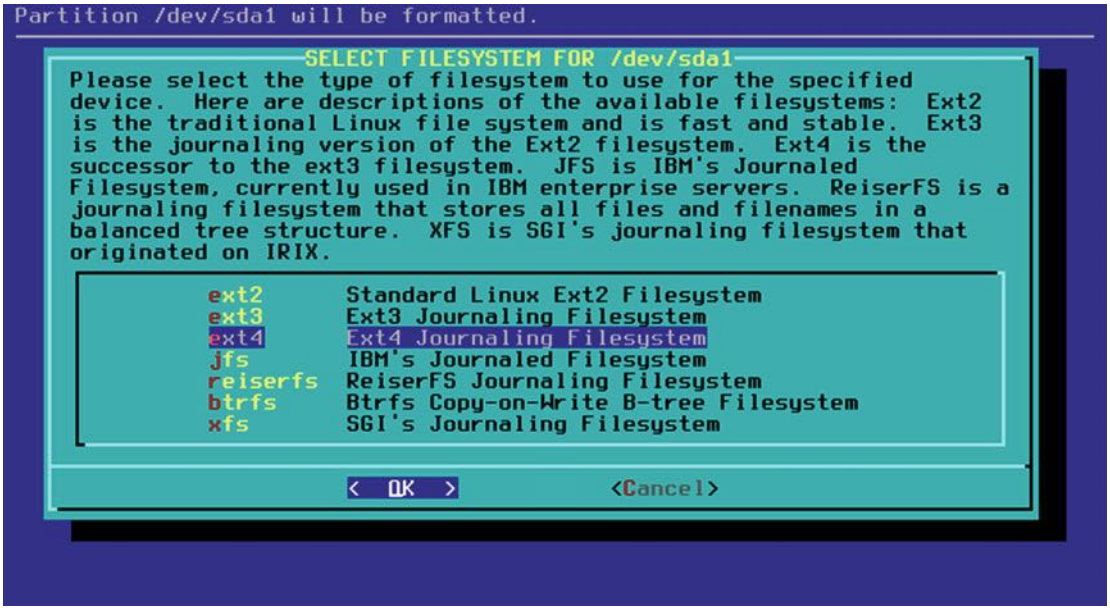


Figure 13-10. *Select the file system of the partition*

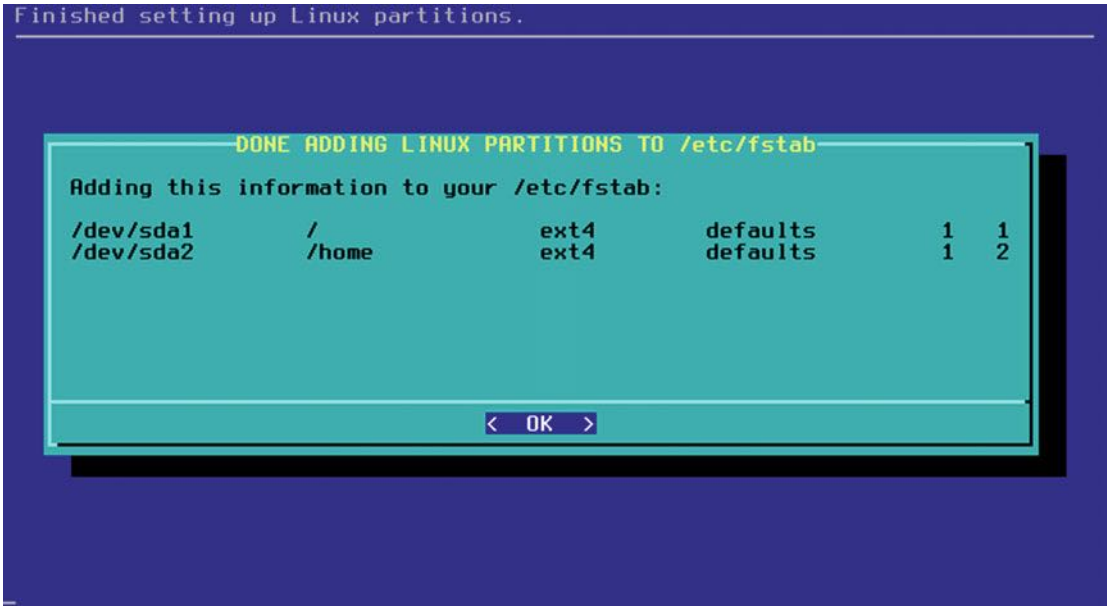


Figure 13-11. *The summary of the current status of the disk partitions*

In the next step, the tool will ask you what media you are using to install Slackware (Figure 13-12). In this case, select the first option. Once selected, it asks you to perform an auto or manual scanning of the media; select auto. This is going to take some time.

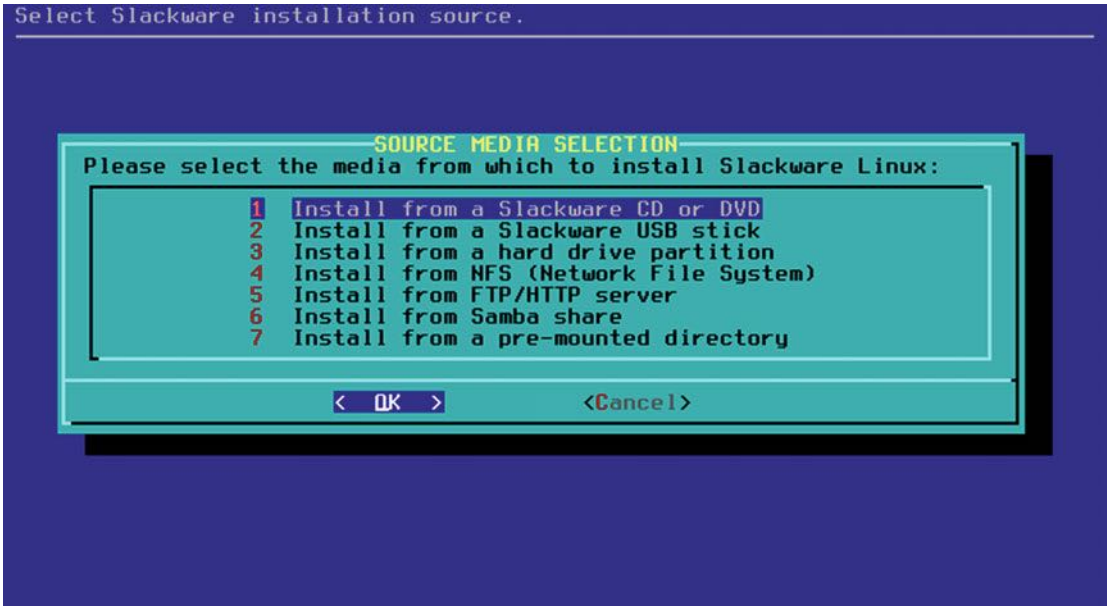


Figure 13-12. *Select the installation media*

Once it has scanned the media and knows the packages that you have available to install, you will be asked to select which ones you want to install. If you have doubts, it is better to use the ones selected by default or select all of them. In any case, the partition is big enough that you don't need to worry too much about this. You can see this in Figure 13-13. Press OK.

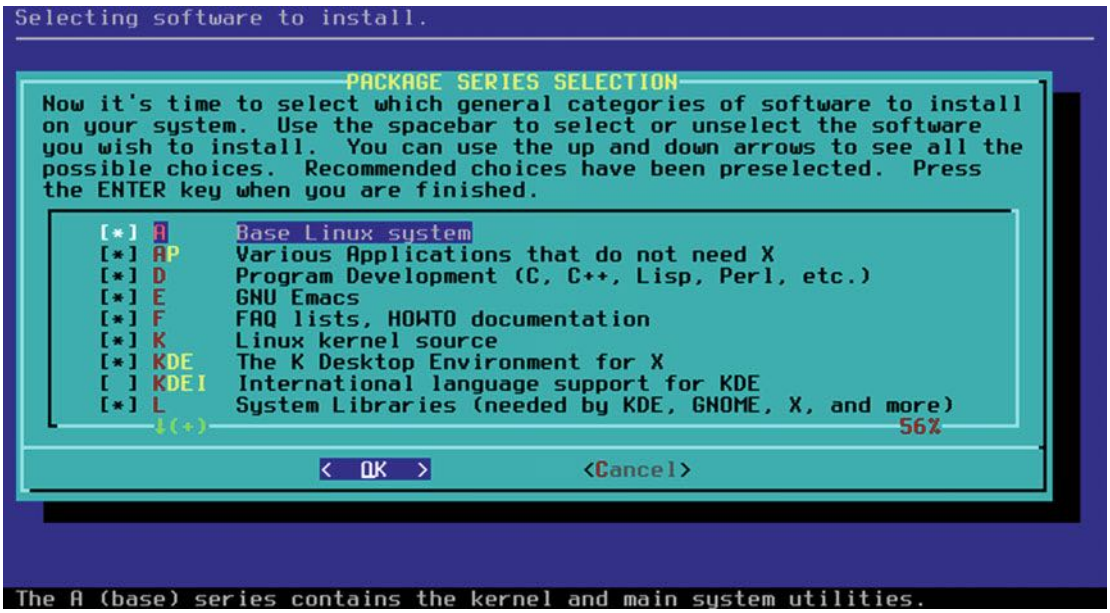


Figure 13-13. *Select the packages that you want to install*

The next screen asks you to choose how you want to install the packages (Figure 13-14). The first two entries are totally automatic and do not require your intervention; the rest are for choosing which packages you want to install. If you are not sure which packages install, I suggest you choose the first option, which is also the fastest one. Then press OK. The installation of the packages is going to take several minutes (about 9 minutes in my system).

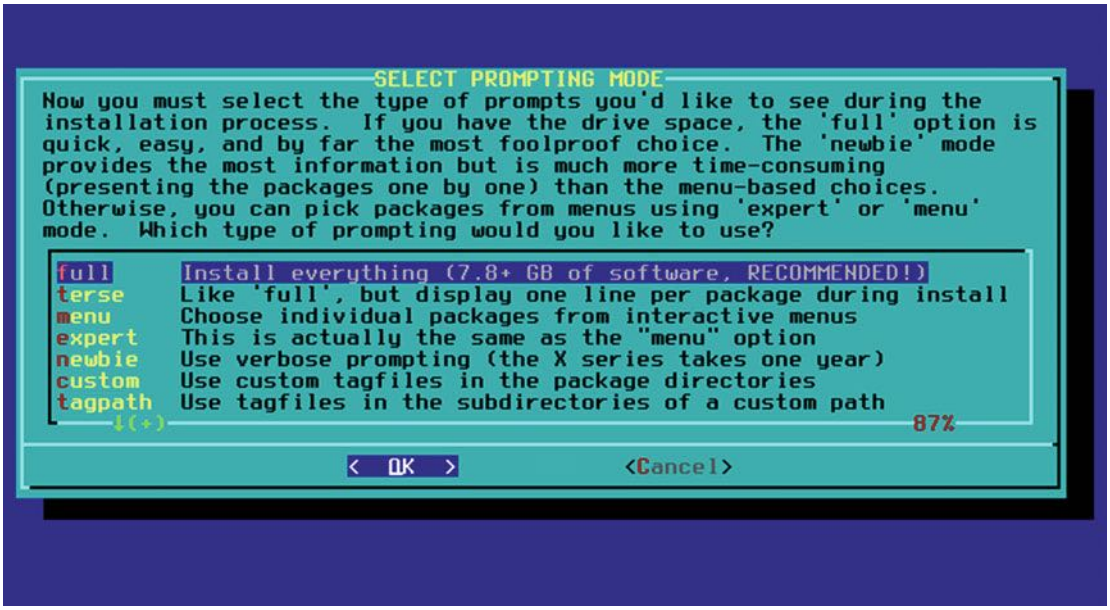


Figure 13-14. *The packages installation mode selection*

After all packages are installed, it will suggest that you create a USB stick to boot the system from (if your system is capable of that). See Figure 13-15. This could be very useful if the process to write the boot loader to the disk fails or simply if you do not want to install a boot loader on your disk and want to boot from the USB stick. It's your choice; in my case, I skipped this step.

Figure 13-15. *Create an optional USB stick to boot the system*

The next step is to install the boot loader on your hard disk (Figure 13-16). The tool to install is LILO, a traditional and old boot loader that today is discontinued; almost no other distro continues to use it. (But we have to take into account that Slackware 14.1 is from 2013, and LILO was discontinued in December 2015, so given the conservative nature of Slackware it's understandable.) Anyway, LILO does its job without any trouble, so let's install it on the hard disk. I choose the automatic method because it works well most of the time. Next, you can choose between several screen resolutions (Figure 13-17); in almost all modern systems, the last one, 1024x768x256, should work well. It will ask you one more thing about LILO, which is if you want to add any optional parameters to be passed to the kernel at boot time, but leave that blank and continue.

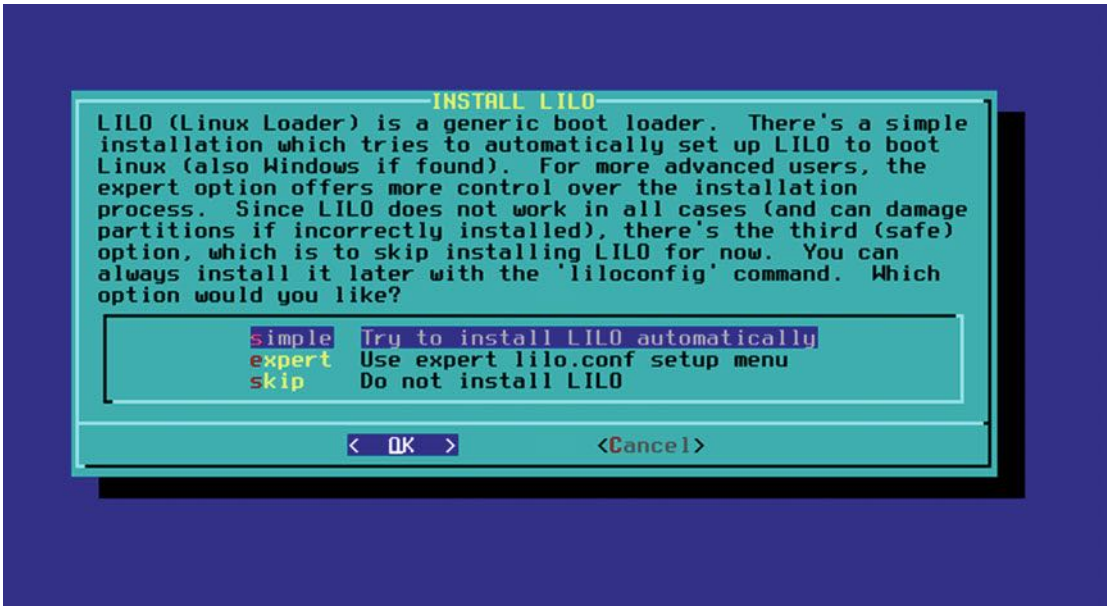


Figure 13-16. *The LILO boot loader installation step*

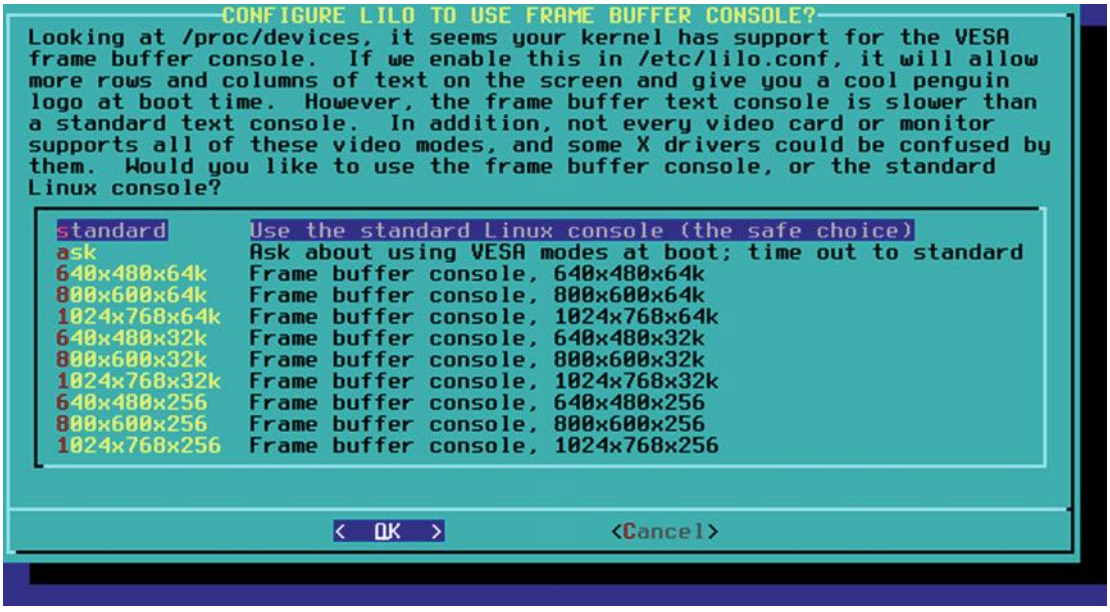


Figure 13-17. *Choose a screen resolution to use with LILO*

Next, it will ask if you want to use the UTF-8 mode in the text consoles (the command line), and it will warn you about possible conflicts. Choose no if you are not sure, but I tested this with UTF-8 and I didn't find any problems so far. So select Yes and continue (Figure 13-18).

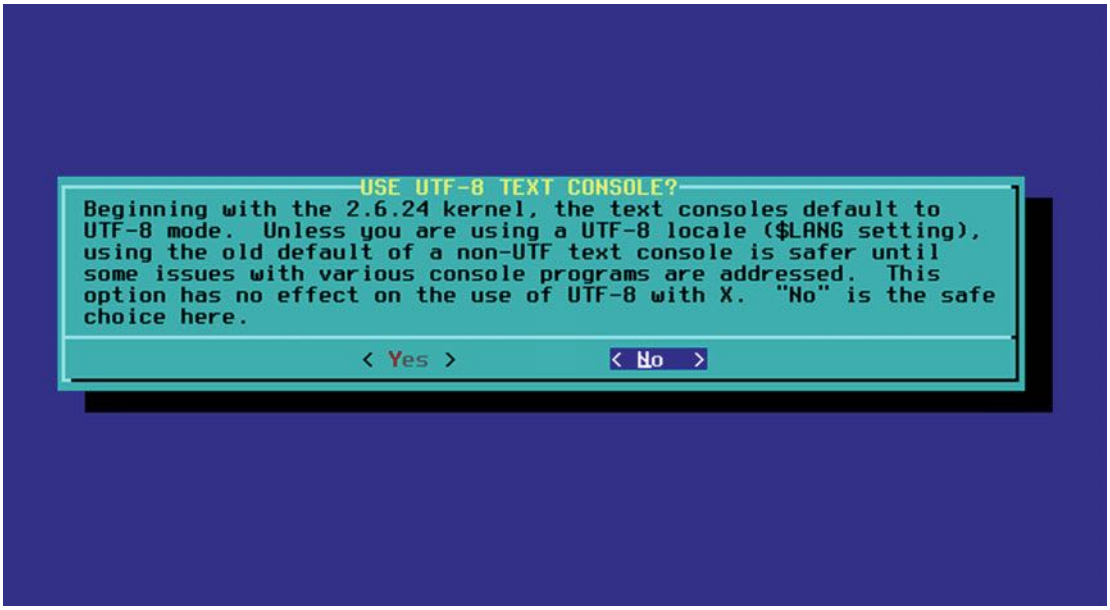


Figure 13-18. *Choose if you want to use UTF-8 in the text console*

Finally, it will ask you where to install LILO. Choose MBR to install it in the Master Boot Record of your hard drive and continue. The next screen sets up the mouse that you are going to use. Most people have an USB mouse, so this is a safe choice; if you are still using a PS/2 mouse and yours is not on the list (which is likely), the "ps2" entry is a safe option here. I selected "usb" (see Figure 13-19) and pressed OK. Now it will ask you if you want to install the `gpm` program to be able to copy and paste in the console with a mouse; choose Yes and continue.

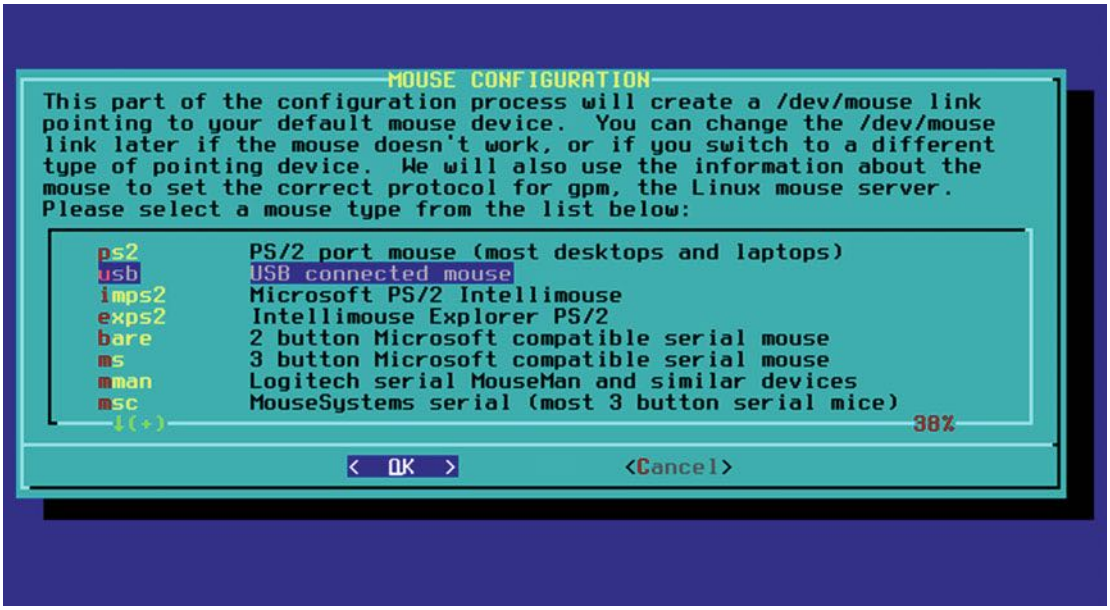


Figure 13-19. *Select which mouse type you want to use*

Then it will ask if you want to configure your network. Choose Yes, and it will ask you to set a hostname (Figure 13-20) and a domain (Figure 13-21). Introduce the hostname that you want to use as the name of your system, and a space if you do not want to setup a domain (you probably don't).

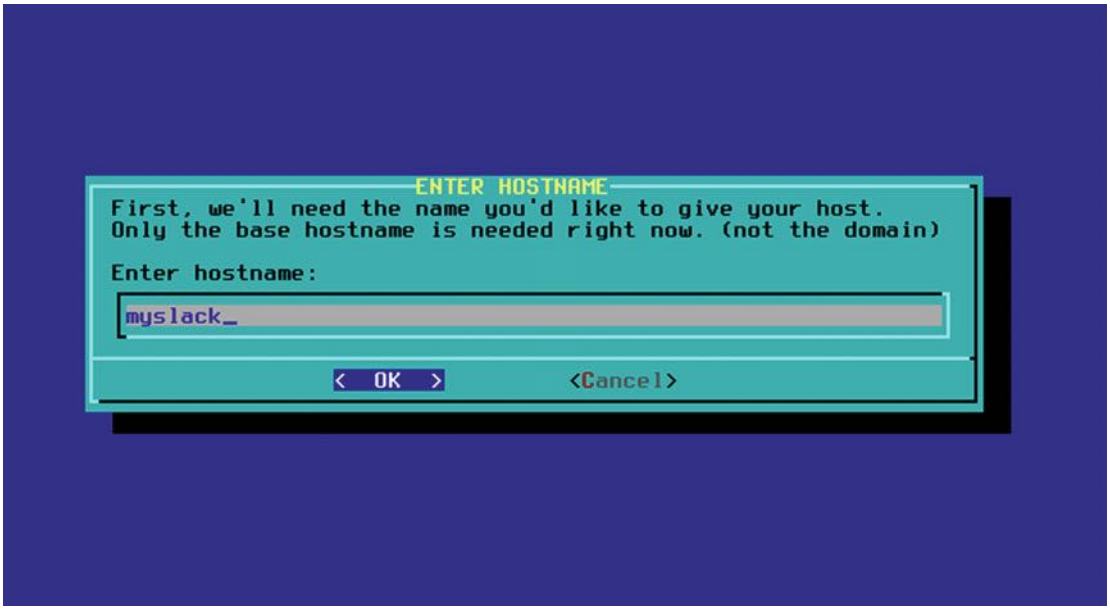


Figure 13-20. *Set the hostname of your system*

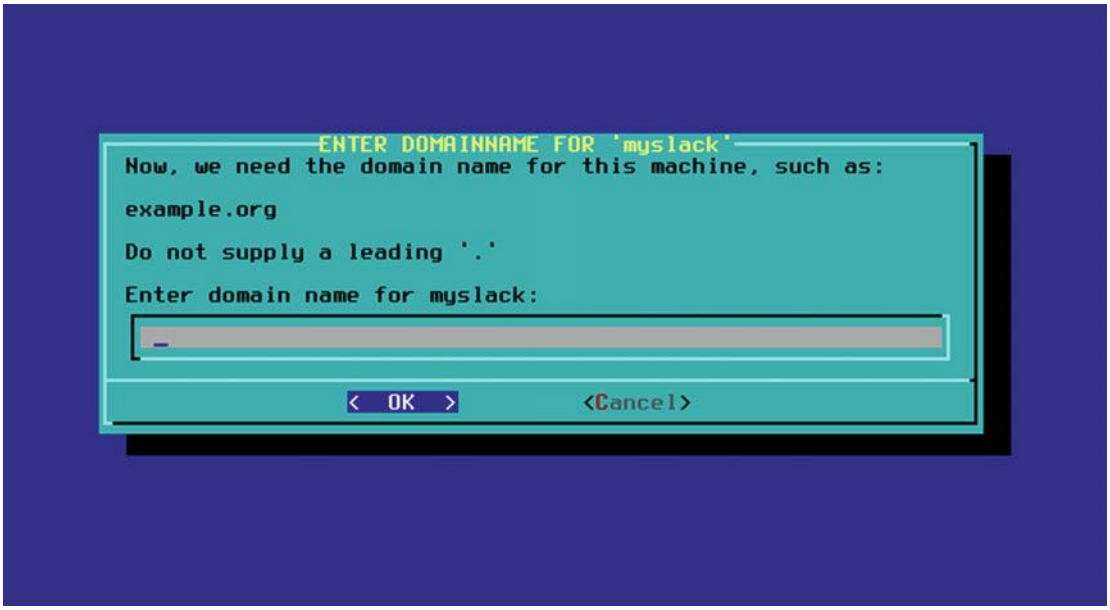


Figure 13-21. *Set the domain of your system*

Now it is time to set the IP address of your system on the network (see Figure 13-22). If you want a fixed IP address, use the first option. If you have a DHCP server, use the second one (the default). Finally if you have a wireless adapter, use the last one. I choose the default. Then it will ask you for a DHCP hostname; you usually don't need this, so leave it blank and continue.

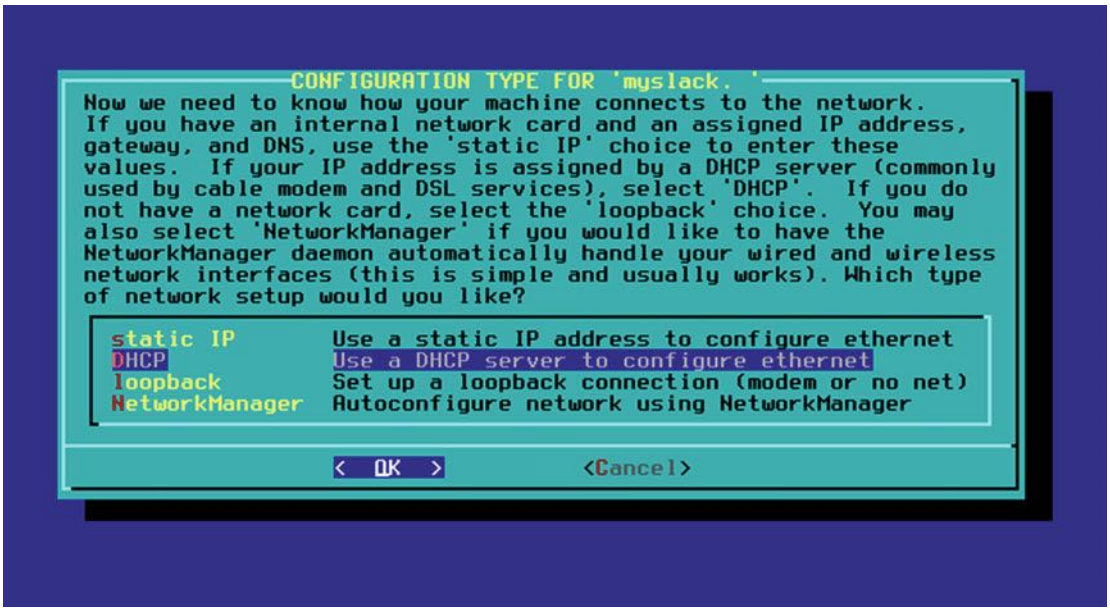


Figure 13-22. *Configure the IP address of your system*

At the end it shows a summary of your network configuration (see Figure 13-23). If everything is OK, go to the next step.

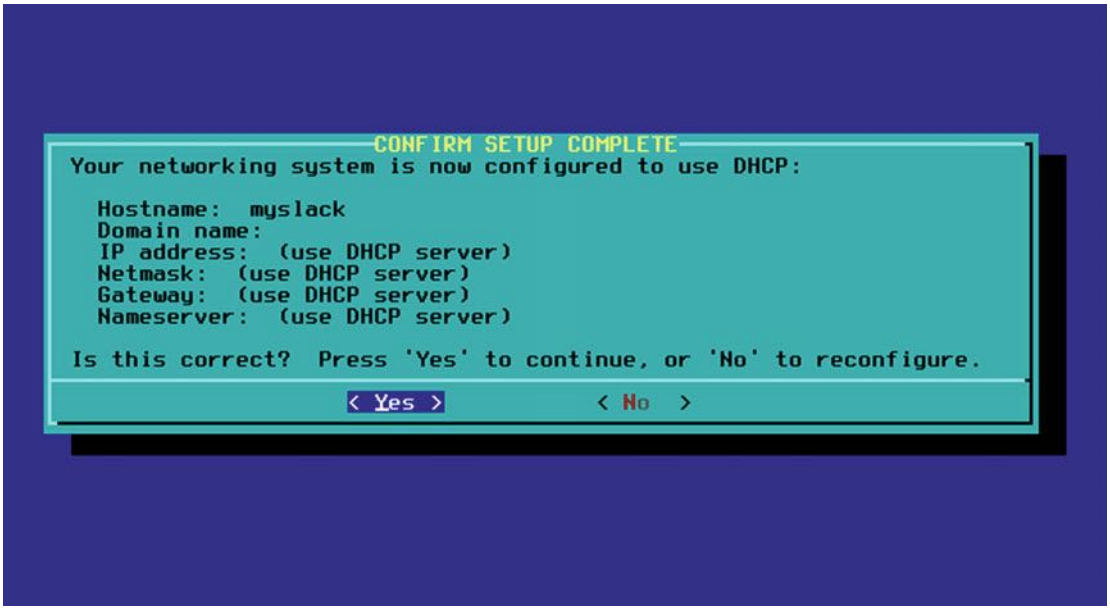


Figure 13-23. *The network configuration summary*

Now you must select what services you want to be started at boot time (see Figure 13-24). Usually, if you do not want to use any server in your system (such as a desktop system), the default choices are fine. But if you want to start `bind` to have a local DNS server as a local cache to navigate faster or a CUPS server for printers, etc., make your choice here. If you are not sure, go with the defaults; you can always set this later. To the question asking if you want to set up a console font configuration, choose No and continue.

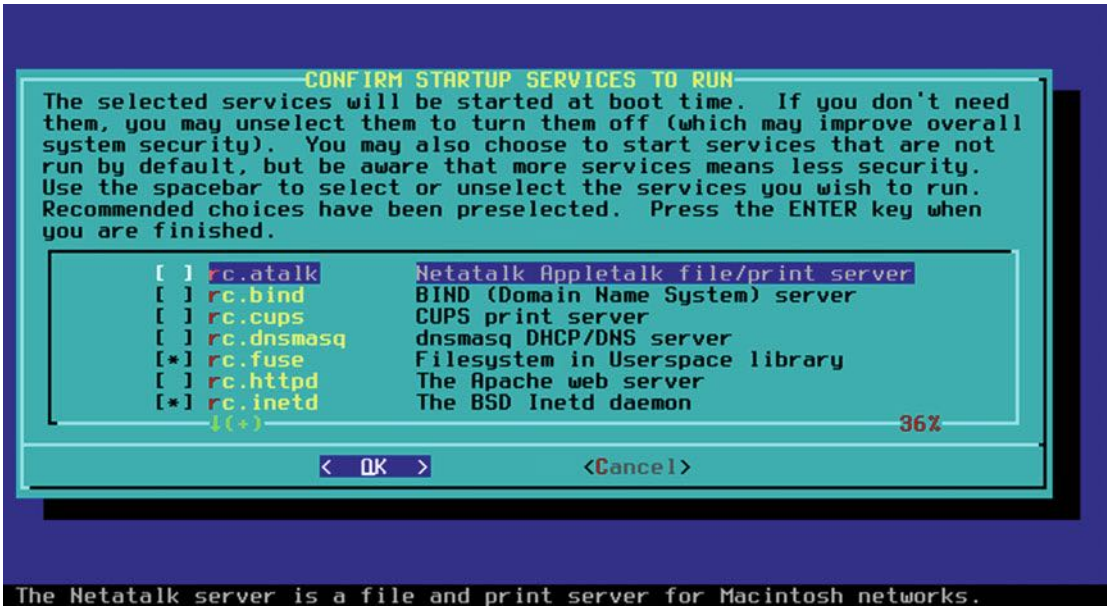


Figure 13-24. *Select what services to start at boot time*

Now you have to configure the clock (Figure 13-25) and the time zone settings (Figure 13-26). I'm going to give you one piece of advice here, but first you have to know that you have two clocks in your system: the hardware clock and the software clock. These were the same thing a long, long time ago, but nowadays they are separate. The first is set in the BIOS/UEFI (which can also be set from the OS) and the second one is in your OS. The hardware clock is the reference that the OS takes when the system is booted, then the OS uses the time zone information to present you the right local time. If your OS is properly configured, it will also synchronize the time periodically with the time on Internet servers (via the NTP, network time protocol). Thus you can set your hardware clock either to your local time or the UTC (Coordinated Universal Time). My advice is that you always set your hardware clock to UTC time, and then set your OS time function to be the time zone where you live. Doing it this way is going to save you some headaches in the future (like forgetting to change the clock for Daylight Saving Time events twice a year) or if you use more than one non-Window OS or distribution on the same machine. Windows is very particular about this topic, and it follows a different approach from almost all other OSes; Windows always assumes that your BIOS is set as your local time. If this is the case, that you have Windows and Linux at the same time in your PC, I suggest you use your local time for the hardware clock.

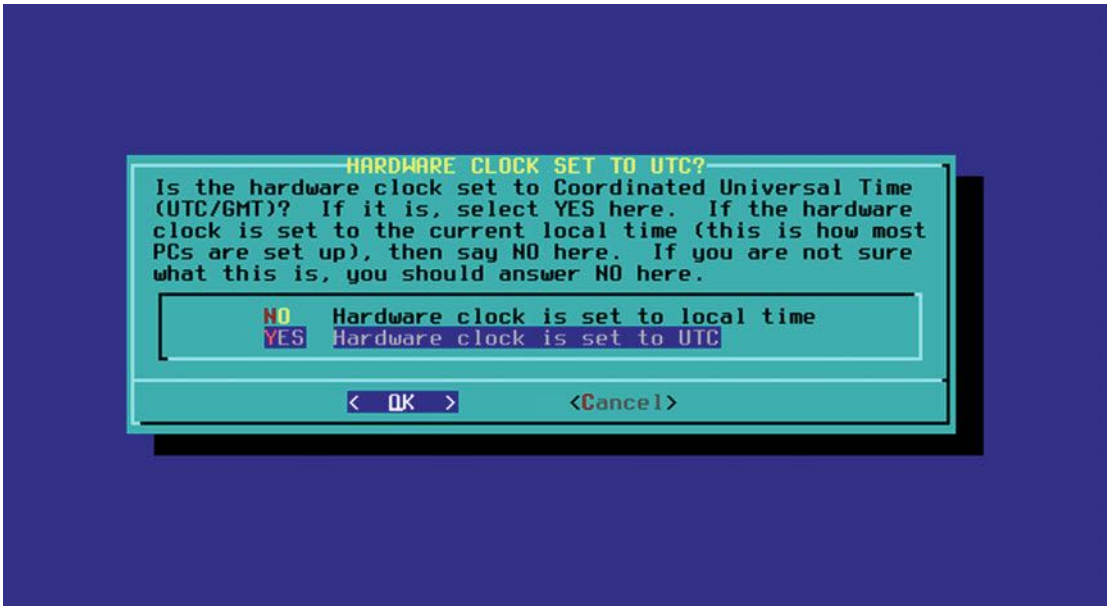


Figure 13-25. *The clock setting (use UTC)*

Figure 13-26. *Select your time zone*

The next step is to choose a desktop environment or a window manager as you default graphic environment (Figure 13-27). You probably want to select the KDE desktop environment, but I will select Fluxbox to show you a window manager that is different from the others you've already seen in this book.

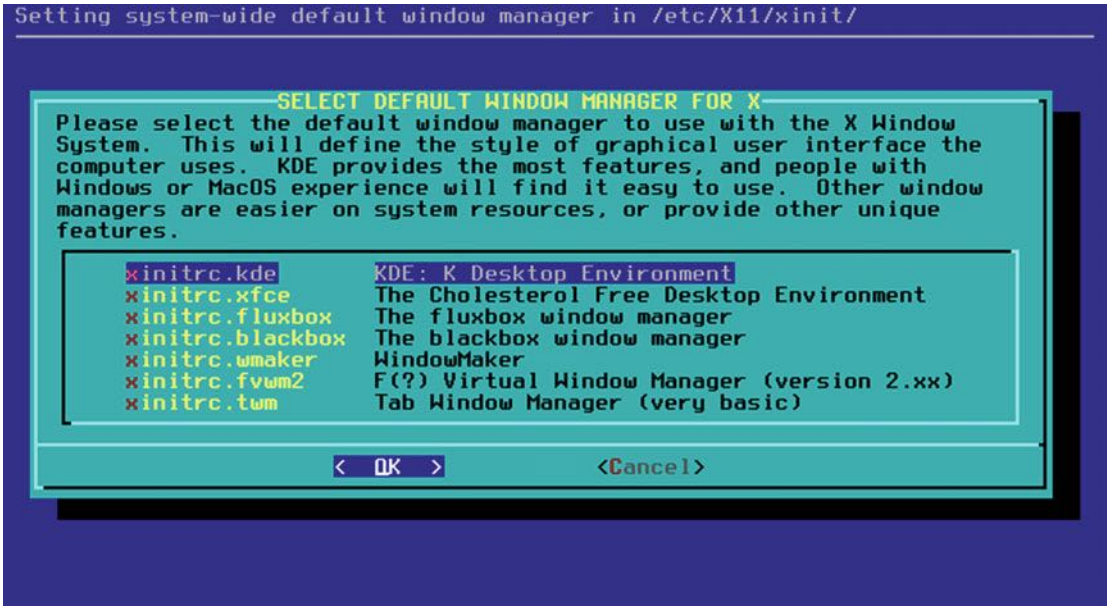


Figure 13-27. *Select a default DE or WM as your graphic environment*

You have to set a root password now. Remember to always create a very strong password with this account. This is shown in Figure 13-28.

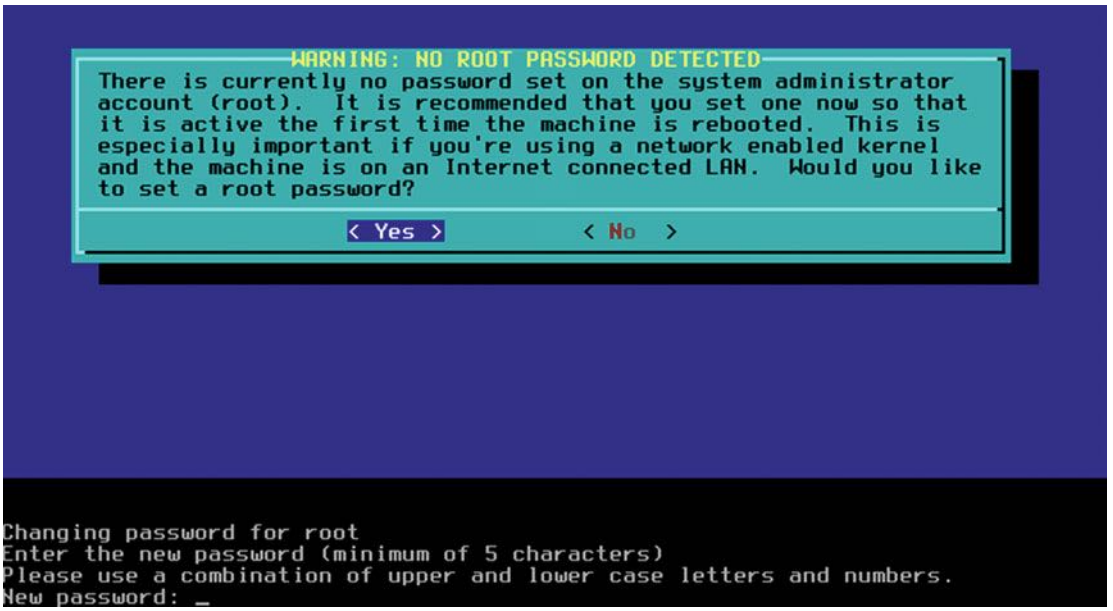


Figure 13-28. *Set a strong password for the root account*

That's it! You finished your job with the setup tool. The tool will show you a message telling you this. Press OK and it will take you to the main menu of the tool (shown back in Figure 13-7). Select Exit, press OK, and then enter the proper command in the console to reboot your system:

```
# reboot
```

This first time you start your new Slackware system, you will get the screen shown in Figure 13-29.

Figure 13-29. *The LILO boot loader of your new Slackware system*

You can wait two minutes to boot automatically or you can press Enter to start to boot your system. Either way you are going to end up in a terminal session where you have to log in as root. After showing you a cookie fortune message (a package that used to be installed in many Linux distros by default a long time ago), you will get a root prompt (Figure 13-30).

Figure 13-30. *The first prompt in a fresh installed Slackware system*

Notice that you do not start in your windows environment by default; you could do it now with the command `startx`, but you should do something else first. Maybe you noticed that you were never asked to create a user account. You should create a user account now so that you don't have to use the root account by default (never, never do that). Use the `adduser` script to do this:

```
# adduser
Login name for new user []: johndoe
User ID ('UID') [defaults to next available]:
Initial group [users]:
Additional UNIX groups:
...
Press ENTER to continue without adding any additional groups
Or press the UP arrow to add/select/edit additional groups
: audio cdrom plugdev video power
Home directory [ /home/johndoe]:
Shell [/bin/bash]:
Expiry date (YYYY-MM-DD) []:
...
Creating new account:
...
Full Name []: John Doe
Room Number []:
Work Phone []:
Home Phone []:
Other []:
...
New password:
Re-enter new password:
...
Account setup complete
# exit
myslack login: johndoe
password:
$
```

Finally, after you log in with your new user account, you can start the window manager with the command `$ startx`. Fluxbox is a very simple window manager, but it's usable with modern apps like Firefox or older ones like Emacs (see Figure 13-31). You can, of course, configure it to make it more elegant.

Figure 13-31. *The Fluxbox window manger with Firefox and Emacs*

This is a very basic system and you may still have to install some packages and do some configuration to have a reasonable system to work with every day.

Maintenance

The maintenance tasks in a distro like Slackware are made via the command line. Basically you use the same tools to perform all the operations: `pkgtools` or `slackpkg`.

Managing Apps and Updating

Three are two official tools to manage the packages. I'll show you how do it with `slackpkg`. The traditional tool, `pkgtool`, can also be used through an interactive menu (see Figure 13-32) or via commands.

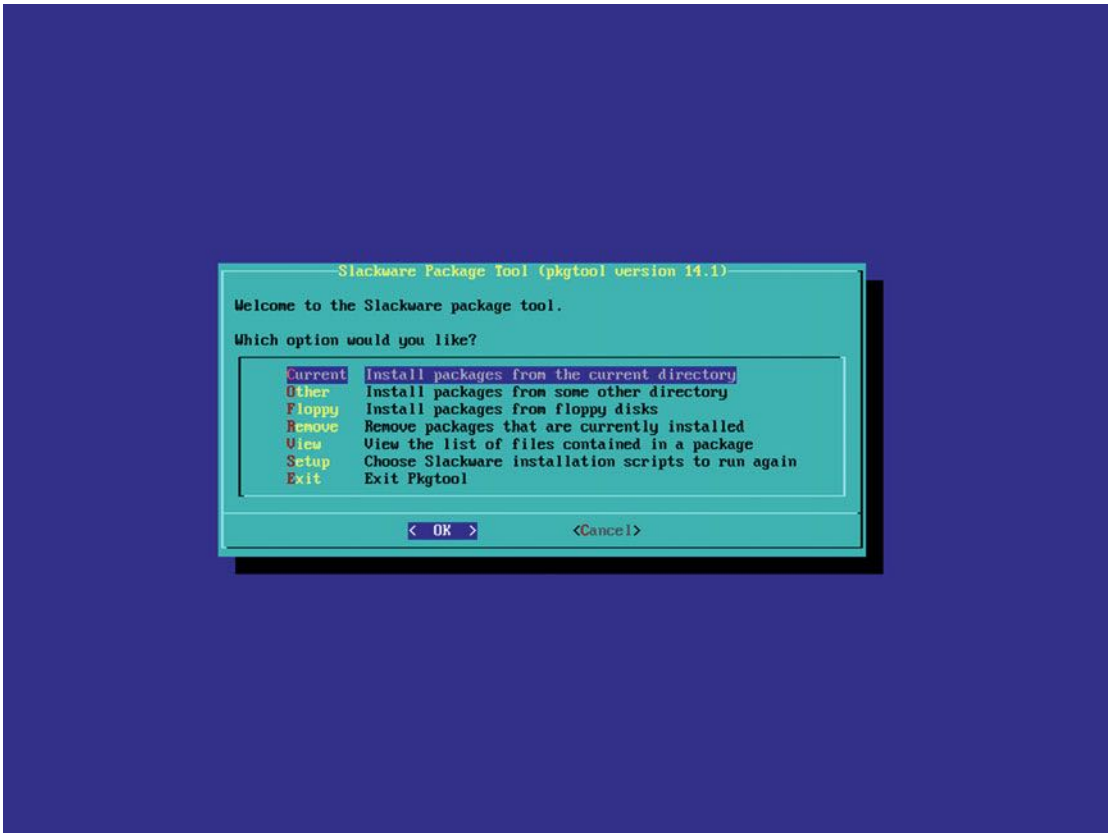


Figure 13-32. *The pkgtool interactive menu*

To install a package, use this code (don't forget to select a mirror by uncommenting it in `/etc/slackpkg/mirrors`):

```
# slackpkg update
# slackpkg search firefox
# slackpkg install mozilla-firefox-24.1.0esr-x86_64_1
```

To remove it:

```
# slackpkg remove mozilla-firefox-24.1.0esr-x86_64_1
```

And to update a package:

```
# slackpkg upgrade mozilla-firefox-24.1.0esr-x86_64_1
```

Don't forget that this tool does not manage dependencies, and it's up to you to ensure that they are installed before using an installed package.

Upgrading

Slackware needs to be upgraded each time a new release is launched (use the current branch or opt for a fresh install) because it does not follow a rolling release scheme. You can do this operation with `slackpkg`:

```
# slackpkg update
# slackpkg install-new
# slackpkg upgrade-all
# slackpkg clean-system
```

Upgrade Slackware can vary from release to release, so the best way to perform this operation is to follow the instructions that are shipped in every DVD/CD in the file UPGRADE.TXT in the root directory. Here are a series of commands that can help you to do this easily. I suggest you do it in another console. Press CTRL+ALT+F2 to be able to watch while you perform the steps in the first console.

```
# mkdir /mnt/cdrom
# mnt /dev/sr0 /mnt/cdrom
# cd /mnt/cdrom
# less UPGRADE.txt
```

Pros and Cons

Here is a list of things that I personally see as pros and cons of the Slackware distro. There is always room for discussion in this matter, but I've tried to be objective as possible.

Pros

- Slackware is a very stable distribution.
- It offers a very simple UNIX-like design.
- It is an original distribution.
- It's highly configurable.
- It offers very simple configuration trough scripts and text files.
- The packages are almost pure upstream ones, without modifications.

Cons

- The package managers do not resolve dependencies.
- It is not a user-friendly distro.
- There's a long time between releases, so it has outdated packages.
- If you want additional packages, you have to use third party repositories.
- It has no commercial support and a very small community.
- It has poor documentation and it's outdated.
- No Gnome support.
- There is only the outdated LILO as boot loader.

Summary

Slackware is the last of the distros suited to advanced users, at least in a traditional way. Slackware is a veteran distro: conservative, a little harsh, and yet after all this time, against all odds, still surviving and attracting new users while keeping a number of very enthusiast and loyal ones.

In the next chapter, you will see a distro that ushers in a new paradigm. It may be the future of Linux distros. NixOS offers some interesting new ideas of how a distro could be, and it's an example of the bleeding edge design of Linux distros.

CHAPTER 14

NixOS

NixOS is a very different Linux distro from those you've seen up to now. In fact, it is a state of the art distribution, ahead of everything. It's still not mature, it's continuing to evolve, it has still its quirks, and nobody can say if its bleeding edge technology is going to be adopted as the future of Linux. There are a few other proposals (like GuixSD, GoboLinux, and Qubes) that are also trying to make a difference and exploring new horizons. I include NixOS because it has very interesting concepts. It has already been adopted by about 5,000 professionals as their daily working distro; however, give it a wide when comparing it to others.

History

In 2003, Eelco Dolstra was a Dutch student at the Utrecht University, collaborating on the TraCE (Transparent Configuration Environments) project as part of his PhD research. He started to develop the Nix package manager, which later would become the core of his PhD thesis[1] and the core of the NixOS distro itself. Under the umbrella of the same TraCE project, another collaborator and student named Armijn Hemel would later develop the first prototype of NixOS, the Linux distro based on the Nix package manager, as part of his Master's thesis[2].

Thus Nix was created as part of academic research and NixOS was created as a proof of concept that Nix could be used to manage a whole Linux distro. Dolstra also would develop other tools like Hydra, which is a Nix-based continuous integration tool, and NixOps, which is a tool for provisioning and deploying NixOS machines.

In 2015, as a strategy to support the development of NixOS, the cost of the infrastructure (Hydra hardware, AWS costs), and another related projects, the NixOS Foundation was created. This foundation is supported mainly by individual donations.

Today a few developers, researchers, users, and professionals are part of the still-little community of the NixOS distribution, but it is growing steadily. Perhaps the most receptive users are the Haskell (a purely functional programming language) developers; they also like the idea of a functional-driven Linux distro.

[1] https://nixos.org/~eelco/pubs/phd-thesis.pdf
[2] https://nixos.org/docs/SCR-2005-091.pdf

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_14

Philosophy

NixOS is a distribution built around its package manager, Nix, which is a proof of concept of how to achieve a model of software deployment that is purely functional. What does that mean? The best way to explain it is to use the authors' own words:

> *Existing package and system configuration management tools suffer from an imperative model, where system administration actions such as package upgrades or changes to system configuration files are stateful: they destructively update the state of the system. This leads to many problems, such as the inability to roll back changes easily, to deploy multiple versions of a package side-by-side, to reproduce a configuration deterministically on another machine, or to reliably upgrade a system. In this article we show that we can overcome these problems by moving to a purely functional system configuration model. This means that all static parts of a system (such as software packages, configuration files and system startup scripts) are built by pure functions and are immutable, stored in a way analogous to a heap in a purely functional language. We have implemented this model in NixOS, a non-trivial Linux distribution that uses the Nix package manager to build the entire system configuration from a modular, purely functional specification.*
>
> —Eelco Dolstra, Andres Loh, and Nicolas Pierron,
> "NixOS: A Purely Functional Linux Distribution," Cambridge University Press, 2010

Thus, the philosophy of NixOS is to prove that this goal can be achieved, to be a live laboratory of the Nix development and concepts. Nix brings a series of advantages that also defines this distribution and makes it unique; this is explained further in the "Package Management System" section.

Distro Selection Criteria

Now that you know a little history, let's see how NixOS fares on the selection criteria listed in Chapter 2.

Purpose and Environment

NixOS is a general purpose distribution with a general version for all environments. There are two branches of this version, the "stable" and the "unstable," the latter of which is used for development. There also ISO images for virtualization/cloud environments, specifically for VirtualBox and Amazon EC2.

Support

Although NixOS has a very small community, you can get support for it through diverse channels:

- **NixOS documentation**: `http://nixos.org/nixos/manual`
- **Nix documentation**: `http://nixos.org/nix/manual`
- **Wiki**: `https://nixos.org/wiki`
- **Mailing list**: `http://lists.science.uu.nl/mailman/listinfo/nix-dev`
- **IRC**: `#nixos` at `irc.freenode.net`

One thing about the NixOS distro is the proportion of advanced Linux users and professional developers is much higher than in other distro communities, mostly because these folks are usually the early adopters of new advances, like the ones offered in this distro. Due of the immaturity of the distro, the best help should come from the mailing lists and IRC; not everything is covered in the documentation and wiki, and not everything is updated.

User Friendliness

NixOS is not a user-friendly distro; you must know how to deal with the command line and the peculiarities of its package manager, and you must learn how to write the declarative configuration file to even install it. In fact, a lot of new users start using this distribution by copying and modifying another configuration file from another user. I'll go a step further: I don't think it's even a friendly distro for advanced Linux users because they have to change the way they think about Linux and learn how to follow the NixOS guidelines.

Stability

Is NixOS stable or not? Well, that's a tricky question to answer. On one hand, it's an immature, still-evolving distro with some bugs and a small community far from reaching the necessary critical mass, which implies that the main developers are overloaded with too much work. On the other hand, it's a distro that allows multiple configurations, multiple package versions, and the ability to make rollbacks, so if something goes wrong with an update or setting, you always can come back to a more safe and stable previous configuration. Also, you can update or install packages only in your user environment without affect the global one (managed by root). Thus, you have a tremendous flexibility to achieve a very stable system, aside from the inevitable bugs and little inconvenience. And finally you can even opt between the stable and unstable branches.

NixOS follows a standard release model, and the new releases usually appear when they are tested and stable enough, without a fixed schedule. However, the goal is now to follow a release model more similar to the one for Ubuntu: a new release every six months. At this time, NixOS has released five stable releases, which follow a numbering scheme similar to Ubuntu's: year.month codename. The last release was 16.03 Emu.

Hardware Support

NixOS offers hardware detection. It supports the hardware that the kernel supports, it supports some wireless cards, and you can use CUPS for printers, but other things may not be supported. NixOS is relatively new, it's in development, and it has a small community (and a small install base), so users will need to fend for themselves when it comes to certain hardware support.

Aesthetics

Given the size of the community around this distribution and its blank-slate nature, it is understandable that there is not any effort to focus on the aesthetics of the distro, apart from the usual logos and backgrounds in Grub, desktop managers, and desktop environments.

Desktop Environment

Like Arch or Gentoo, you have plenty of options when it comes to desktop environments. You can choose a traditional desktop environments like KDE, Gnome, or Xfce or window managers like Awesome, Xmonad, i3, IceWM, etc. The default "official" one is KDE, because like Nix it is cross-platform.

Init System

NixOS has used systemd as its init system since December of 2014. Previously, it used System V.

Package Management System

In a distribution like this, which was built only to be the testing ground of a package manager, everything begins and ends in Nix. And Nix is a very special package manager and a very unique concept. There is nothing out there similar to it (the closest one, GNU Guix, is actually based on Nix). The new paradigm that Nix introduces has multiple advantages but some drawbacks, so you must change the way you think about some things or you are going to hit the wall so many times that you will end up frustrated.

First of all, in NixOS, everything is managed by Nix: the kernel, system packages, user packages, system-wide configuration files, and so on. Nix uses a lazy pure functional language designed especially for it, which is used in a declarative mode based on a configuration file(s) that defines how the system has to be built. No matter how many times you build up a system using the same configuration, you always get the same system state.

NixOS stores every package isolated from others in the Nix store (`/nix/store/`) using a cryptographic hash (obtained from the input used to build it) as a unique identifier used in the store path and for internal references. Every change in the Nix expression used to build the package generates another hash and therefore is stored in another different path; this allows different releases and customized versions of the same package to coexist. The same happens with the configuration files of packages/services, and as result you can keep several system configurations or states simultaneously on the same machine.

Nix repositories are called channels, and they contain the recipes to build the packages, called nix expressions (usually a .nix file, some scripts, and some auxiliary files) and also the binaries (however, some packages are built from source). Nixpkgs, which is essentially a git repository, contains all of the nix expressions to build all of the official packages. The prebuilt packages in the channels are generated by Hydra, a Nix-based continuous build system, which when a new release of a package is available, builds it, tests it, and if it passes, releases it. (Actually, it's not quite like that because all of the new packages have to pass the test to release a new channel version). There is a farm of machines to host this service. Official channels are usually a tag or branch of the Nixpkgs git repository. The stable channel is called nixos (few weeks behind Nixpkgs HEAD) and the unstable is called nixos-unstable (a delay of few days). You can subscribe to the channel that you want to use.

Thanks to this package/system management model, Nix provide a series of advantages to NixOS over the ordinary Linux distros:

- **Reliable upgrades**: While the configuration remains the same, you can upgrade your system safely to always obtain the same result, as if it were a new, fresh installation.
- **Atomic upgrades**: Changes in a configuration are made in a transactional way, thus they are atomic, and changes are only applied when the transaction is finished. For instance, if an upgrade is interrupted or fails, the previous state continues to work. If you keep on your system one configuration that works (such as the initial one), your system will always boot up.
- **Rollbacks**: You can always go back (rollback) to a previous state, make a global configuration change or a local user one, install/remove a package, etc. This is possible because a new configuration never overwrites a previous one. In fact, you can choose the configuration you want to use from the Grub menu when booting the system (you can do the same without boot, of course). Because the configurations are kept simultaneously on the disk, this works almost instantaneously, without have to restore anything from disk or overwrite anything. A rough comparison would be to say that they work as snapshots instead of backups (in reality, they work with symbolic links).

- **Test changes safely**: Thanks to the rollback capabilities and simultaneous configurations, NixOS allows you to test (as an option) a configuration, and if doesn't work, it doesn't make it the default one. Even better, NixOS has an option that allows you to test this configuration on a virtual machine that works as a sandbox and contains your previous system state and the new changes that you want to test. This process is very efficient (it uses QEMU but does not clone your current system in a disk image). The only drawback is that your data, your home partition, is not present in this VM. This also helps you to avoid populating your Grub entries with a lot of minor changes.
- **Reproducible system configurations**: You can copy a configuration from a machine in a similar machine and get the same system (except user data, obviously, and "mutable state" like `/var` contents). This is ideal for making changes on test machines before applying changes on production machines or for deployments (in fact, there is a dedicated tool, NixOps, to do that with NixOS).
- **Mixed model with source and binaries**: Nix builds packages from source by default, but since compiling is a slow process, there are prebuilt binaries available to download from a cache server (the URL for which is included the channels) when they are available. In the stable branch, this lets you build only a few packages, so upgrading or installing a system is fast enough. But in the current branch, the packages are more recent ones and frequently you must build too many packages.
- **Consistency**: When a package or configuration changes, all the necessary packages or dependencies are rebuilt too. The same happens with the kernel and the modules (this is not something new; you can have this in other distros with DKMS). Also, when a library is updated, all the packages that use it are linked to the new version.
- **Multi-user package management**: In NixOS, the packages installed with the root user are available for all the users, but each user can also install their own packages in their profile. The packages are still stored and managed in the Nix store, and different users can have different versions of the same package. Still, if two users install the same version of the same package, only one copy is stored and shared among them. Also, there are security measures to avoid potential vulnerabilities, like not allowing `setuid` binaries.

But the Nix model also imposes some drawbacks and disadvantages. Some of them relate to the willingness of the user to adapt to NixOS, which is a significant issue.

- **The Unix FHS (Filesystem Hierarchy Standard) is broken**. Directories like `/bin`, `/sbin`, `/lib`, `/usr`, or `/opt` either do not exist or simply contain links to some point of the `/nix/store` directory (a read-only one). This makes some administrative tasks and problem solving more complex; you must rely on Nix. Also, because it breaks compatibility with standard FHS, it's impossible to install any package directly compiled from source (e.g. with `$ make install`); you have to "nixify" it first. Usually it's not that hard, but it's a little inconvenient.

- **The broken FHS has annoying side effects**. Apps expect to find their dependencies in the usual places. Those that depend on path variables that define environments or that have package managers of their own, such as programming languages like Python, Ruby, etc. have problems with this. There a lot of tricks and workarounds made by package maintainers that are available in the wiki, but increases the room for errors. You can also use **nix-shell**, a tool used to build sandbox environments. As the distro base grows, this will happen more often, and it's a crucial challenge to address in order for this distro to be adopted by more people. You can avoid some of these problems via isolated VMs or containers per project or dealing with different NixOS configurations, but at the end you are going to have to fight this in other situations, too. For instance, I tried to use my Emacs config. Most of it worked, but it needed some extensions to be compiled in order to work, which was not a fun experience.
- **Hashes make readability difficult**. Using the hash at the beginning of every package/file name in the `/nix` directory hierarchy makes it harder to find/sort your searches. For instance, you can have a Firefox package stored in the nix store at `/nix/store/5rnfzla9kcx4mj5zdc7nlnv8na1navg-firefox-3.5.4/`. Fortunately, you can still use the `whereis, which`, and `locate` commands to find many things. But you only need to run a `printenv` command to view the environment variables to get a headache again.
- **Potential heavy disk usage**. The disadvantage of having many different versions of a package and many different configurations is the cost of the disk space used, which can be considerable, especially if there are multiple users on the system with very different profiles. And thanks to the rollback functionality, removed or obsolete packages are not removed physically from the disk. You still can perform Nix's garbage collector by hand or periodically trough a systemd service to remove old, unreferenced packages. And you can also opt to use the Nix optimization that uses hard links for identical files; this saves a lot of space.
- **Changed packages can suffer big delays before they are available**. The Hydra farm that built the packages only updates a channel when the rebuild is entirely done (all the packages that needed to be rebuilt). When a change affects many packages (such as a security one or an important library like gcc), this rebuild can take few days. And this matter will get worse as the package base grows over time. Of course, if you have the resources, there's nothing to stop you from making your own Hydra setup to build packages for you.
- **The package base is still behind another distros**. Nixpkgs has around 30,000 packages available currently, whereas other distros have around 50,000 packages in their repositories. This means that if you want to use some of those packages, you have to make a Nix recipe for them. The good news is that you can commit that recipe to the Nixpkgs GitHub repository and make it available to the rest of the community very easily and thereby help maintain the distro.

In summary, if you are capable of adopting the Nix way of doing things for everything, you can avoid many of the aforementioned problems and use it safely for some tasks. I still have serious doubts about NixOS as a multi-purpose desktop or even a server distro. It might make a good way to build a minimal server with a reliable configuration, but once problems appear, it would be unpleasant to debug that machine. Regardless, I have to say that I want to see this distro mature in order to see what it will achieve. Ultimately, I would like to see all packages adapt to the way in which NixOS works, and to that end, administrating a machine would end up like programming the state of it. In an ideal world, a NixOS paradigm (or something similar with the same benefits), would be the perfect solution for so many current problems, which usually are poorly solved with many workarounds.

Architecture

Currently NixOS only supports the Intel/AMD 32-/64-bit architectures. There also unofficial ports to ARM.

Security/Anonymity

Due to the particular nature of this distro, as with Arch or Gentoo security and anonymity depended heavily on the user. There are some options to help harden NixOS, like using the grsecurity kernel or AppArmor, but they are currently very unstable or depend too much on how the user configures them. For example, I use the grsecurity kernel by default in my Arch installations, and you can do the same with Gentoo without trouble, but if you try that now with NixOS, many packages won't work, e.g. some browsers.

Principles and Ethics

There is nothing explicitly said about principles and ethics. The distro has some proprietary software included, like binary blobs and drivers, and nothing stops you from creating ("nixifying") your own package from a proprietary software as binary. Otherwise, you must explicitly set that you want to use "unfree" packages in your configuration to be able to use the available ones (e.g. Nvidia drivers, Adobe Flash).

Live CD

NixOS has a KDE-based Live CD, which gives you a general idea about NixOS. You can also use it to install the distribution.

Professional Certification

Obviously, no professional certification covers this distribution.

Installation

Installing this distribution is a unique experience, as is maintaining it. To those people introduced to the DevOps tools, it would be like to use a tool like Ansible, Puppet, Salt or Vagrant to instead of define the final state from a previous VM/Docker Linux image, define the final state without an image. Thus, you no longer need to depend on a previous image to create what you want; you create the system that you want directly from the beginning and it stays like that even after upgrades. This operation is more natural because you do not have to rely on immutable images (in DevOps, if you change your base image, you usually have to change the operations), because you are creating the image itself in its desirable final state. Now imagine the possibilities of this combined with a DevOps tool like NixOS. Take this with a grain of salt; it's an oversimplification and a gross approximation.

The first step is to go to the NixOS web site and get the ISO image at `http://nixos.org/nixos/download.html`. You can see this web page in Figure 14-1. The ISO images to download are in two formats: minimal and graphical. You also have images for VirtualBox and Amazon EC2. The obvious choice for most users is the graphical Live CD (notice that is only available for 64-bit architecture). The major difference between the two is that in the graphical image you can use graphic tools to do the installation like a partition tool or a text editor, while in the minimal one you must use the command line tools. But I think that given the nature of this distro and the operations to perform, you need to use the terminal anyway; and if you feel comfortable with that, you should prefer the minimal installation CD (you still can choose the graphical one and simply not start the windows session). So click the graphical live CD option to download it.

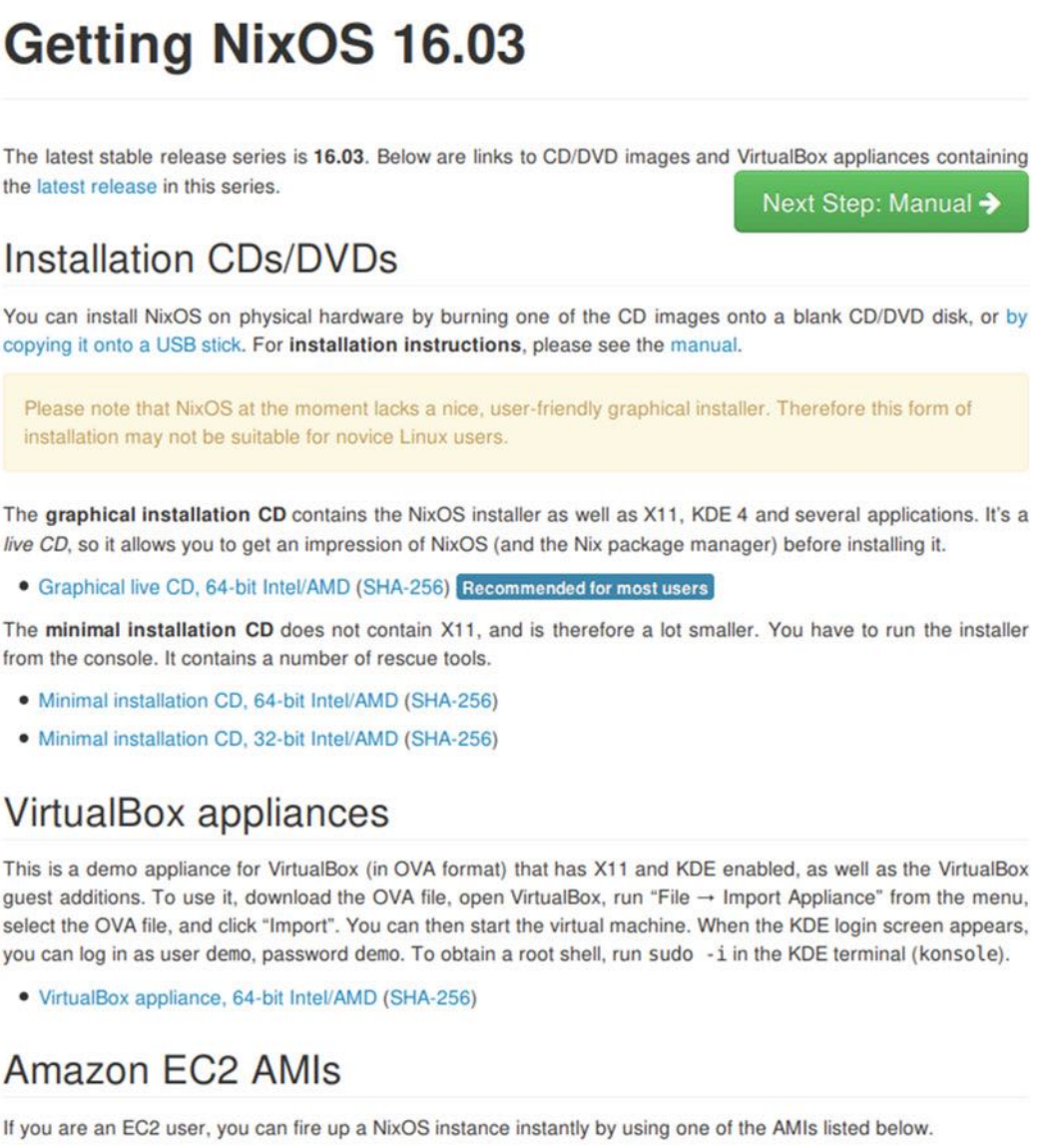


Figure 14-1. *The downloads section of the NixOS web site*

Note that I'm using a traditional BIOS system with a 2TB hard disk. I use the ISO image to boot the system and so the first screen is the typical one shown in Figure 14-2. Usually the first option should work in the majority of systems; if you have trouble with your graphics card, choose the second one instead. After finishing the boot up, you end in a terminal root prompt, like the one in Figure 14-3.

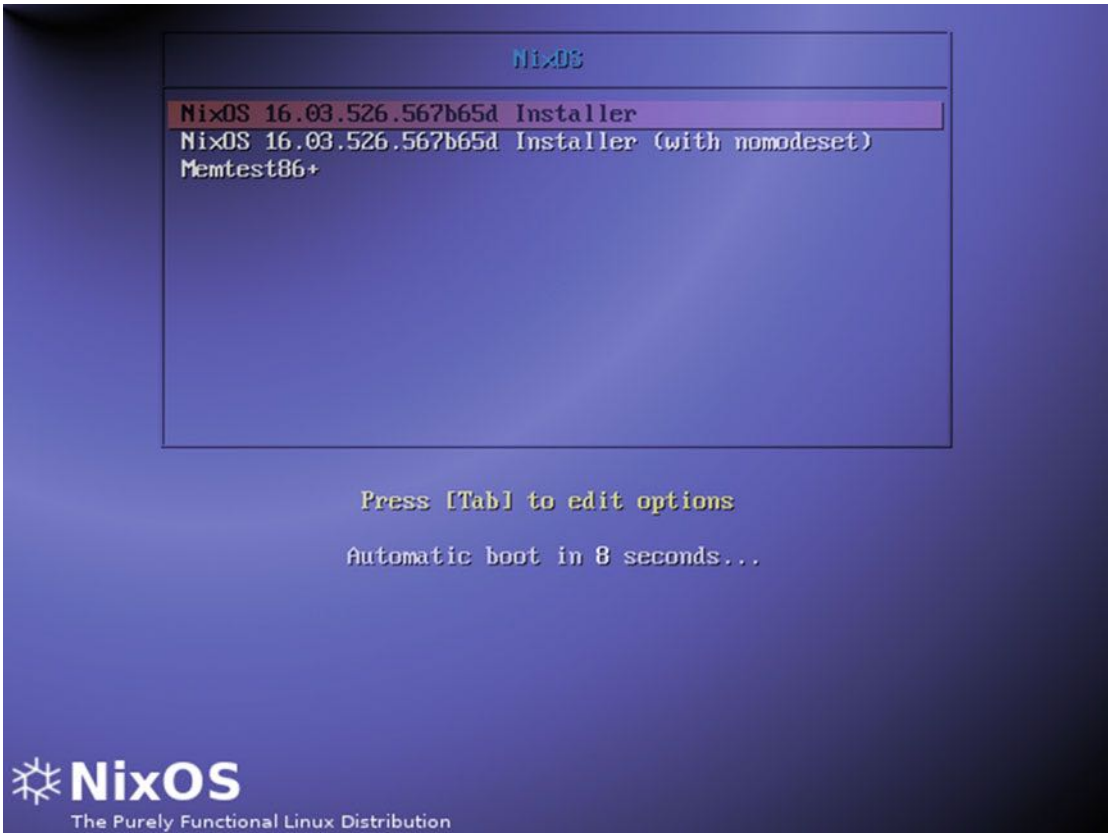


Figure 14-2. *The first screen after booting up from the NixOS ISO image*

```
<<< Welcome to NixOS 16.03.526.567b65d (x86_64) - tty1 >>>

The "root" account has an empty password.  Type `start display-manager' to
start the graphical user interface.


Press <Alt-F8> for the NixOS manual.

nixos login: root (automatic login)

[root@nixos:~]# _
```

Figure 14-3. *The NixOS root console that you get at the end of the boot up*

In order to start the graphical session, you must enter a command, but as I said previously, if you are OK with working in the command line, you can continue the installation from here and use terminal tools. Help is always available in the eighth console, as you can see in Figure 14-4, and it uses the same HTML manual on the Web via the `w3m` text browser. I would usually install it in this way, but let's try it the other way: start the graphical session by typing the `# start display-manager` command.

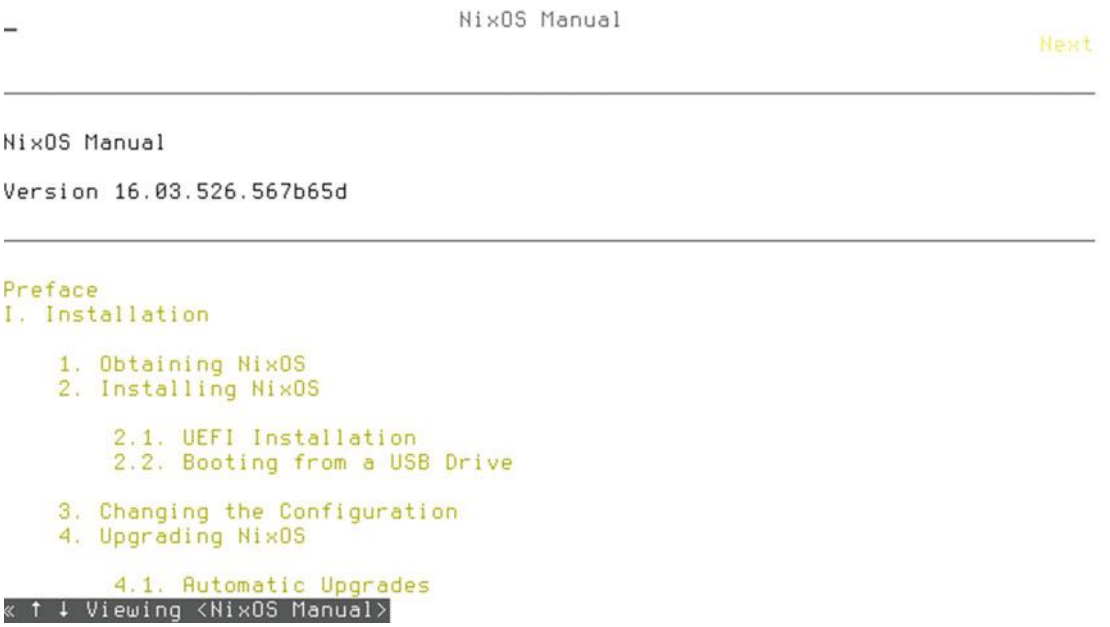


Figure 14-4. *The NixOS manual on the terminal*

A KDE session is opened; in a folder on the desktop are the three essentials tools that you are going to need to install the distro: a partition tool (GParted), a terminal (Konsole), and the NixOS manual (see Figure 14-5).

Figure 14-5. *The default KDE session of the NixOS Live CD*

Let's assume that the network was well detected and is working perfectly. The first thing you need to do is partition the disk, and for that, use the GParted tool (shown in Figure 14-6) and a very simple partition scheme, with only the `/` and `/home` partitions. You start creating the partition table by selecting the Device ➤ Create Partition Table menu entry. By default an `msdos` type should be selected and if not, select it as shown in Figure 14-7 and click Apply (ignore the warning; in my case, the disk is empty).

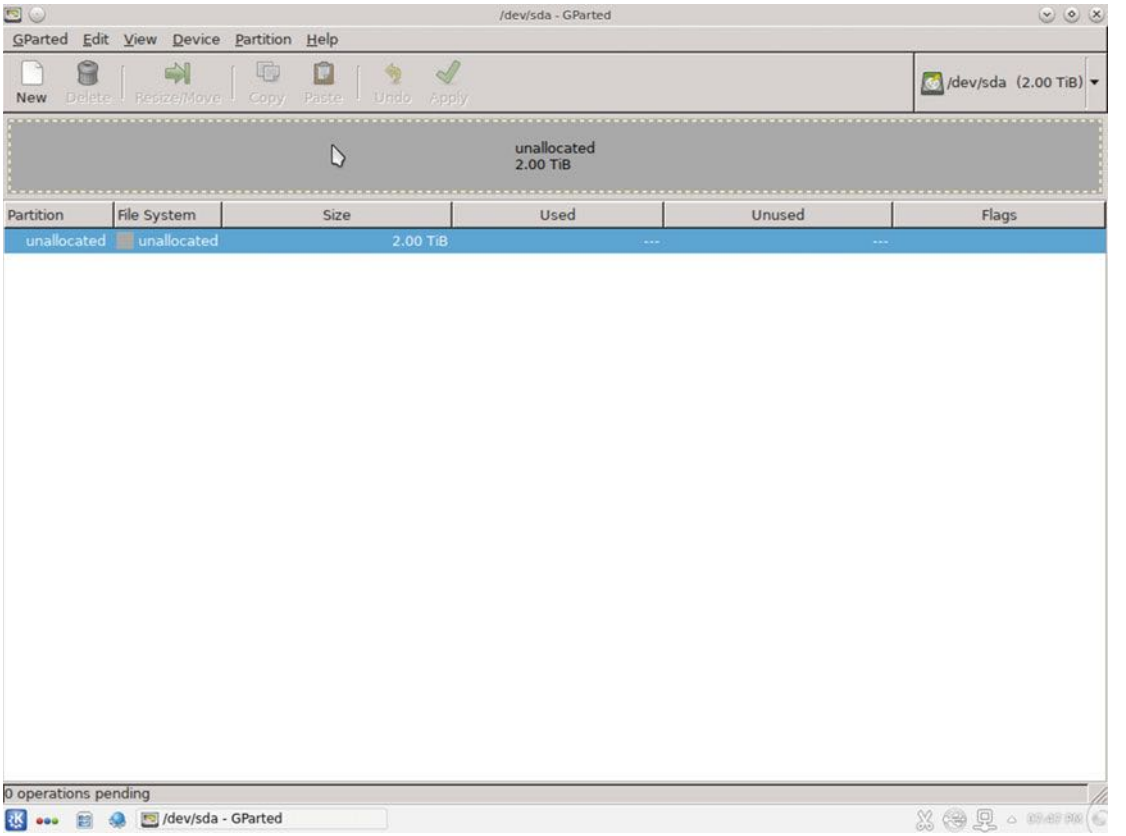


Figure 14-6. *The GParted tool to partition the disk*

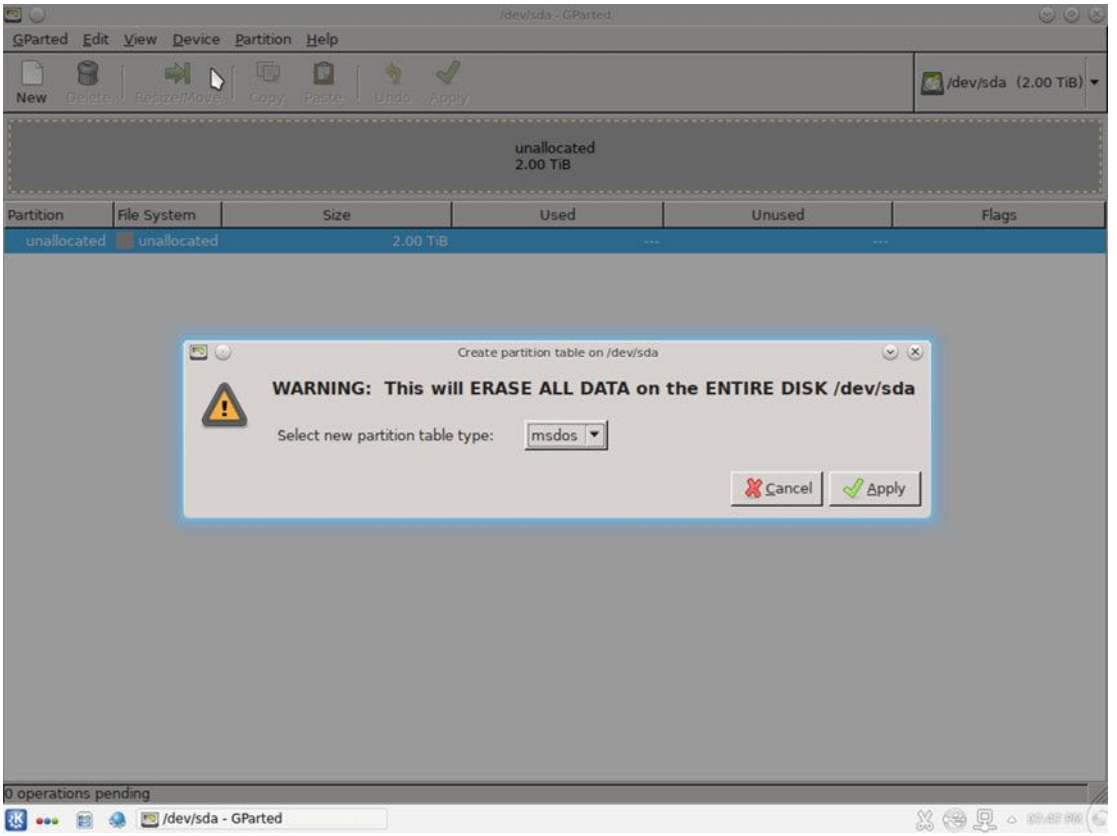


Figure 14-7. *Creating the partition table on the disk*

Create your first partition by clicking the New icon. Create an ext4 100GiB partition with a label of nixos (as in Figure 14-8). This may seem like a very big partition for just the system files, but in NixOS you can store several configurations and several versions of the same packages, and depending on the packages, this can take up a lot of disk space. The idea of putting labels on your partitions is very helpful in a distro like this, where if the device changes, the configuration can be reapplied via the labels. You can also use the UUIDs, but this way is easier (in the hardware configuration file, the UUID is added to the partition through the name when the configuration files are generated automatically for the first time). Next, create another ext4 partition with the rest of the disk in the same way, but with home as the label. Finally, click Apply. You should have something similar to the partition scheme shown in Figure 14-9.

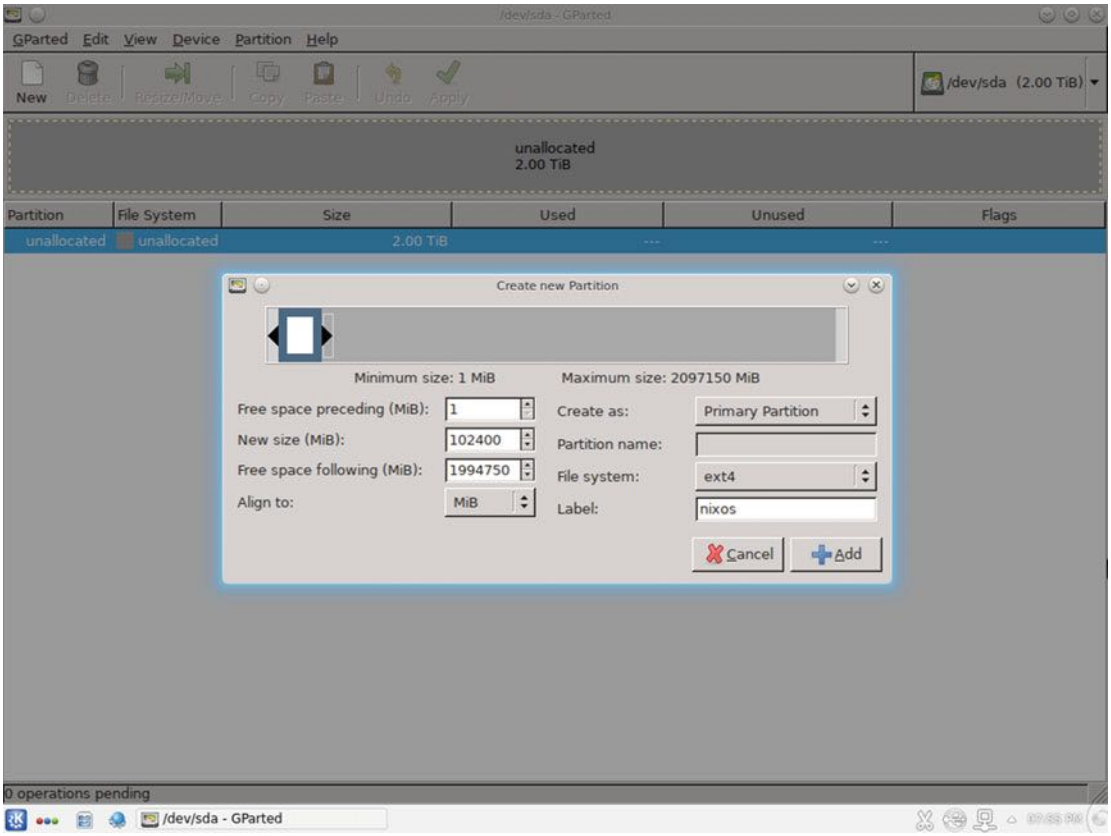


Figure 14-8. *Creation of the first partition*

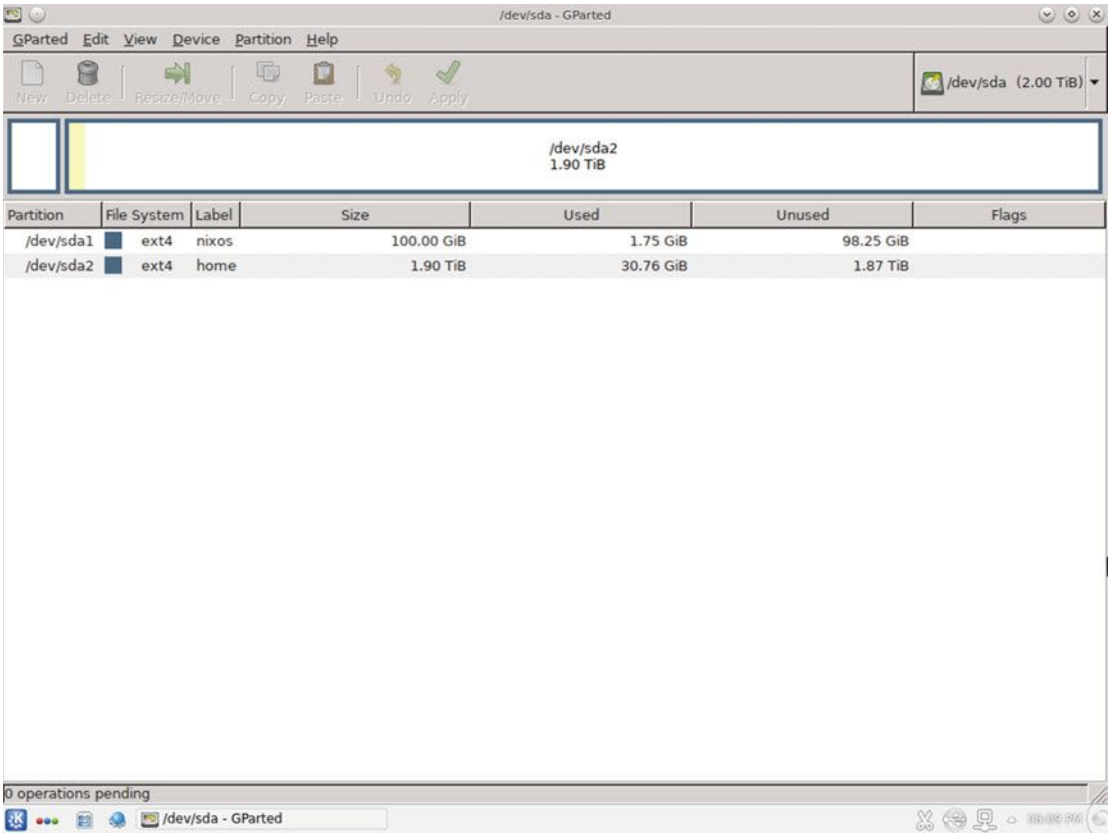


Figure 14-9. *The final partition scheme*

The next step is to mount the partitions. To do so, open the Konsole terminal and enter the following commands. In Figure 14-10, notice the partition labels and the UUIDs.

```
# lsblk -pf
# mount /dev/disk/by-label/nixos /mnt
# mkdir /mnt/home
# mount /dev/disk/by-label/home /mnt/home
```

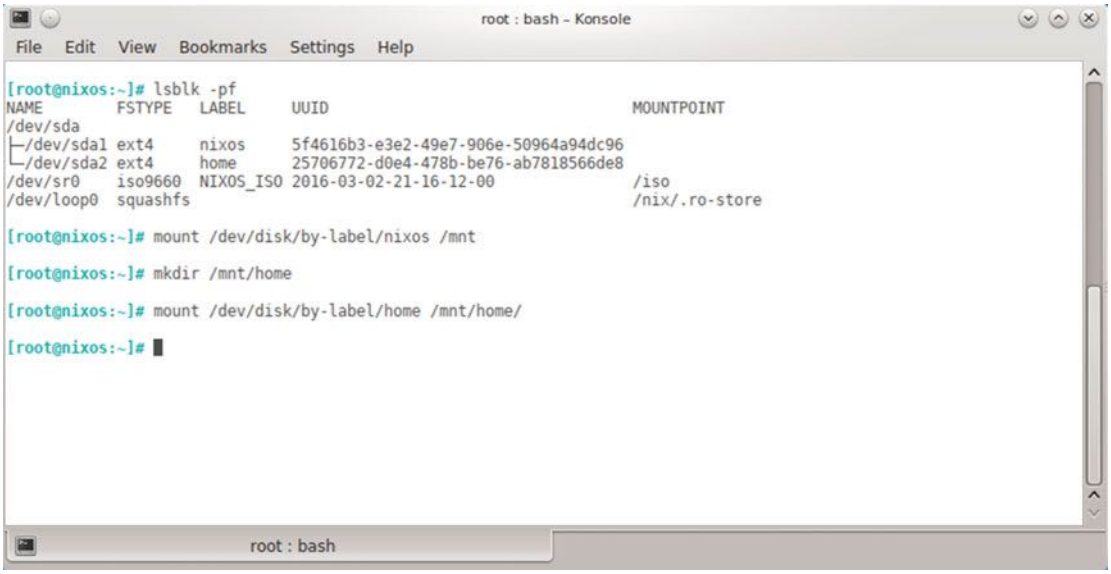


Figure 14-10. *Mounting the partitions*

Now you must generate the initial configuration file of NixOS and then edit it to suit your needs. The configuration file will be generated in the path `/mnt/etc/nixos/configurations.nix` and you will use the graphical editor Kate to edit it. This is shown in Figure 14-11.

```
# nixos-generate-config --root /mnt
# kate /mnt/etc/nixos/configuration.nix &>/dev/null
```

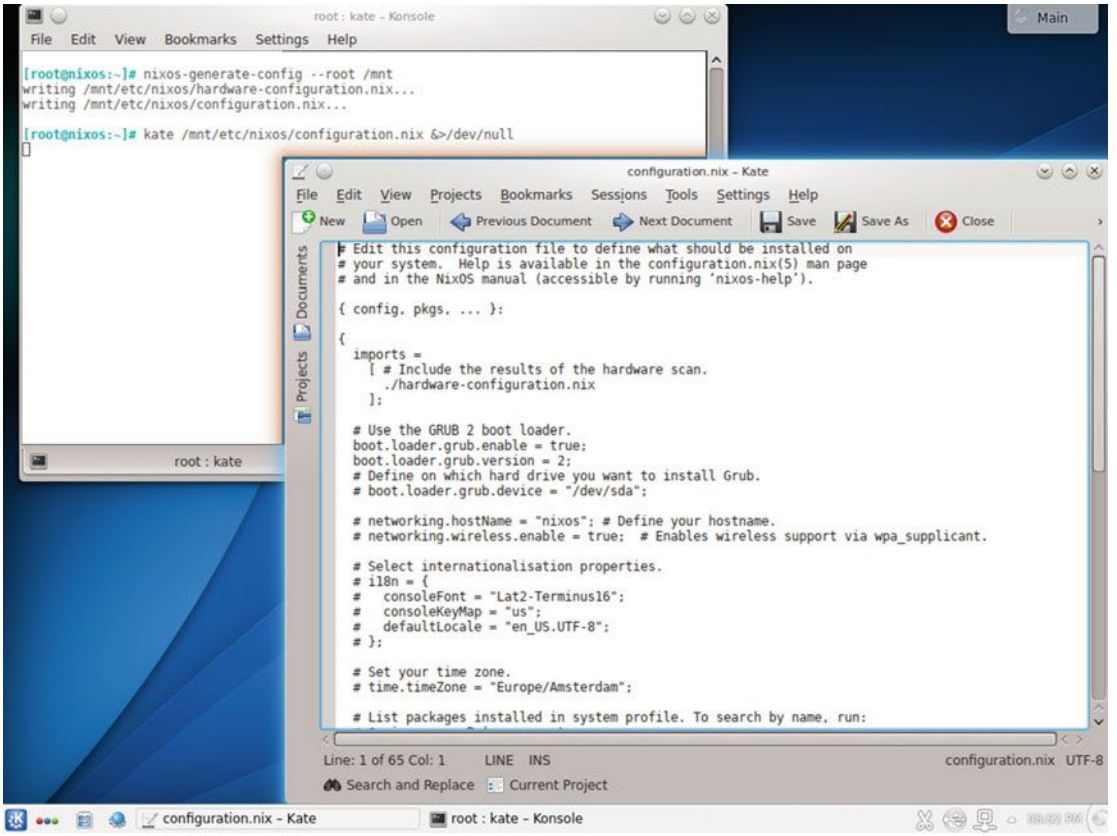


Figure 14-11. *Generating the configuration and opening it in Kate*

Let's take a moment for a sidebar. I'm going to tell you the easiest way to install NixOS and get a very minimal system going, quite similar to the one you get when installing Arch, Gentoo, or Slackware, for example. The only thing you need to do is very simple: first, edit the configuration file to uncomment the line that contains the option `boot.loader.grub.device` and execute the command `# nixos-install`. The system will be installed automatically. At the end, it will ask you for the root's password, and you only have to reboot the system to get a minimally functional NixOS system. This process will take a while because certain packages must be downloaded and compiled (if you choose the unstable version, then a lot of packages will be compiled from source). However, doing it this way means that you need to create a user, and configure the time zone, network settings, etc. after the installation is done. And worse, you have to install the packages you want after that, as with other distros. This is a waste of the advantages of this distribution. Remember, for example, you can install packages in a global way so they are available for all users, and/or packages for each user individually. Some packages, like a desktop environment or a window manager, should be easy if you install them directly and automatically from the configuration.

I strongly recommend that you read the manual (accessed via the desktop icon, which points to a local version of the web manual at `http://nixos.org/nixos/manual/index.html`) to have a more general impression of what you can do and how. Also, I strongly recommend that you check out the different options available for the configuration at `http://nixos.org/nixos/options.html`.

Let's start to edit the initial file to get a minimal configuration:

```
# kate /mnt/etc/nixos/configuration.nix &> /dev/null
```

The contents of the file after the editing should be something like the following :

```
# Edit this configuration file to define what should be installed on
# your system. Help is available in the configuration.nix(5) man page
# and in the NixOS manual (accessible by running 'nixos-help').

{ config, pkgs, ... }:

{
  imports =
    [ # Include the results of the hardware scan.
      ./hardware-configuration.nix
    ];

  # Use the GRUB 2 boot loader.
  boot.loader.grub.enable = true;
  boot.loader.grub.version = 2;
  # Define on which hard drive you want to install Grub.
  boot.loader.grub.device = "/dev/sda";

  networking.hostName = "nixos"; # Define your hostname.
  # networking.wireless.enable = true;  # Enables wireless support via wpa_supplicant.

  # Select internationalisation properties.
  i18n = {
     consoleFont = "Lat2-Terminus16";
     consoleKeyMap = "us";
     defaultLocale = "en_US.UTF-8";
  };

  # Set your time zone.
  time.timeZone = "America/New_York";

  # List packages installed in system profile. To search by name, run:
  # $ nix-env -qaP | grep wget
  environment.systemPackages = with pkgs; [
     wget
     firefox
     htop
  ];

  # List services that you want to enable:

  # Enable the OpenSSH daemon.
  # services.openssh.enable = true;

  # Enable CUPS to print documents.
  services.printing.enable = true;
```

```
  # Enable the X11 windowing system.
  services.xserver.enable = true;
  services.xserver.layout = "us";
  # services.xserver.xkbOptions = "eurosign:e";

  # Enable the KDE Desktop Environment.
  #services.xserver.displayManager.kdm.enable = true;
  services.xserver.desktopManager.kde4.enable = true;

  # Define a user account. Don't forget to set a password with 'passwd'.
  users.extraUsers.johndoe = {
    isNormalUser = true;
    uid = 1000;
    description = "John Doe";
    extraGroups = ["wheel" "audio"];
  };

  # The NixOS release to be compatible with for stateful data such as databases.
  system.stateVersion = "16.03";

}
```

When you have the configuration ready, introduce the following command in the console to automatically install the system:

```
# nixos-install
```

After installing all of the packages, the installation program will ask you to introduce the root password, so introduce a strong one. The program exits and the installation is done. Reboot the system and set a password for the user that you specified in the configuration, in this case johndoe. To do that, the nixos-install program allows you to jail root in the newly installed environment (because obviously the new user does not exist in the Live CD environment). Now you can reboot the system again.

```
# nixos-install --chroot
# passwd johndoe
# exit
# reboot
```

When the machine restarts, the first screen that you will see is the Grub menu (Figure 14-12). Notice that the second entry says “NixOS - All configurations.” Here you should see the configurations stored automatically (the ones that you did not add manually to the boot menu).

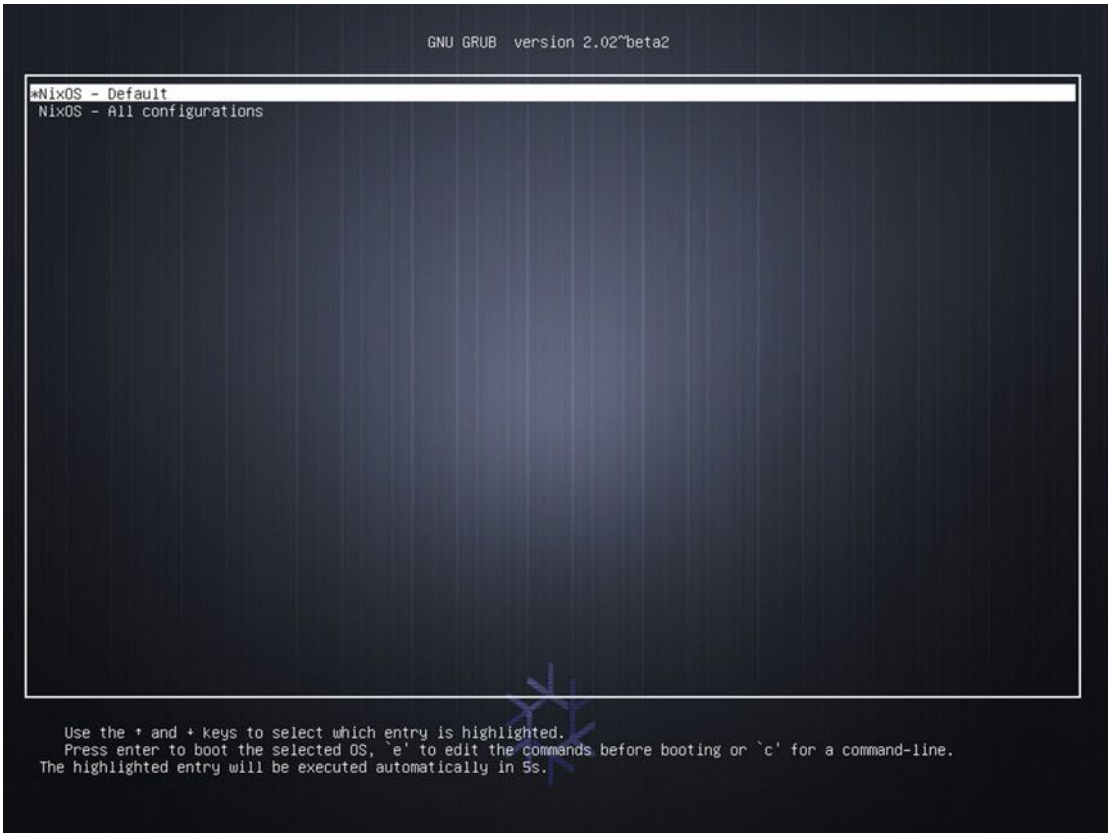


Figure 14-12. *The Grub menu of a fresh installation of NixOS*

Select the first entry to boot NixOS for the first time. After a few seconds, you will see the Slim desktop manager, which is the default one used in NixOS (see Figure 14-13). Enter your user and password.

Figure 14-13. *The desktop manager, Slim, is the default one for the NixOS*

Again you land up in a KDE environment like the Live CD I show you here, with the apps that I told you to specifically install (Figure 14-14). I'm afraid that the capture is too small to see clearly, but if you could, you would see that the binaries paths in the `htop` tool are awkward because all of them are in the `/nix/store directory`. Anyway, you can see how the KDE desktop manager and the htop and Firefox packages were installed as established in the configuration file. This is a powerful way to configure and install a system once you know how to do it, but the first time is a little awkward. If you want to make something more elaborate, you will have to navigate through the nix manual, and maybe even the mailing list and IRC too. This is not unusual with other advanced distributions, but even advanced user must learn a lot of things with NixOS if they want to use it continuously. That's it! You have a NixOS installed in your system.

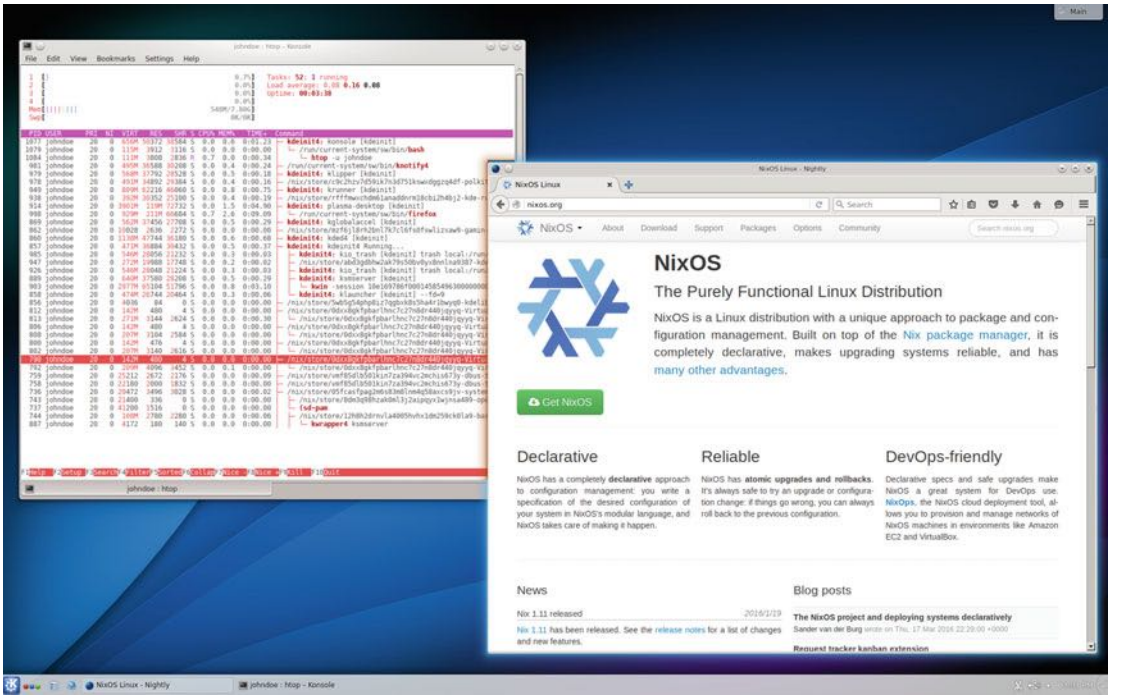


Figure 14-14. *The NixOS KDE with Firefox and htop (inside Konsole) running in the desktop*

Now I'll show you how easy it is to change your configuration and the default desktop manager. You must again edit the configuration file that is now at `/etc/nixos/configuration.nix` and uncomment the line (remove the preceding #) `#services.xserver.displayManager.kdm.enable = true;` and then execute this command into the Konsole terminal:

```
$ sudo nixos-rebuild test
```

If all goes well, you can log out from KDE and you will now have now KDM as your new desktop manager, as you can see in Figure 14-15. Log in. You can make this configuration the default one by running this command:

```
$ sudo nixos-rebuild switch
```

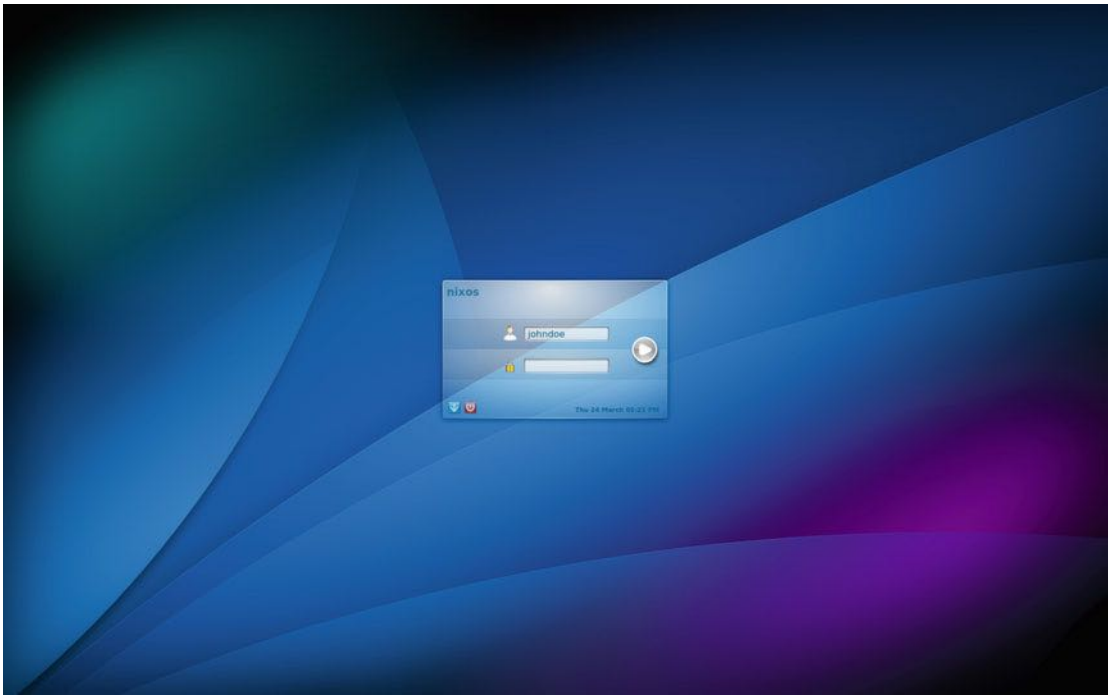


Figure 14-15. *You succesfully changed Slim for KDM by changing a line in the configuration*

But you make bigger changes, like using GDM and Gnome 3 instead of KDM and KDE, with only a few changes in the file; with other distros, this can be a big headache. You only have to change the following section of the configuration file:

```
# Enable the KDE Desktop Environment.
# services.xserver.displayManager.kdm.enable = true;
services.xserver.displayManager.gdm.enable = true;
# services.xserver.desktopManager.kde4.enable = true;
services.xserver.desktopManager.gnome3.enable = true;
```

Perform the same operations as before. If you restart the system, you will now have GDM as your login manager and a completely functional Gnome 3 as your desktop manager (see Figure 14-16).

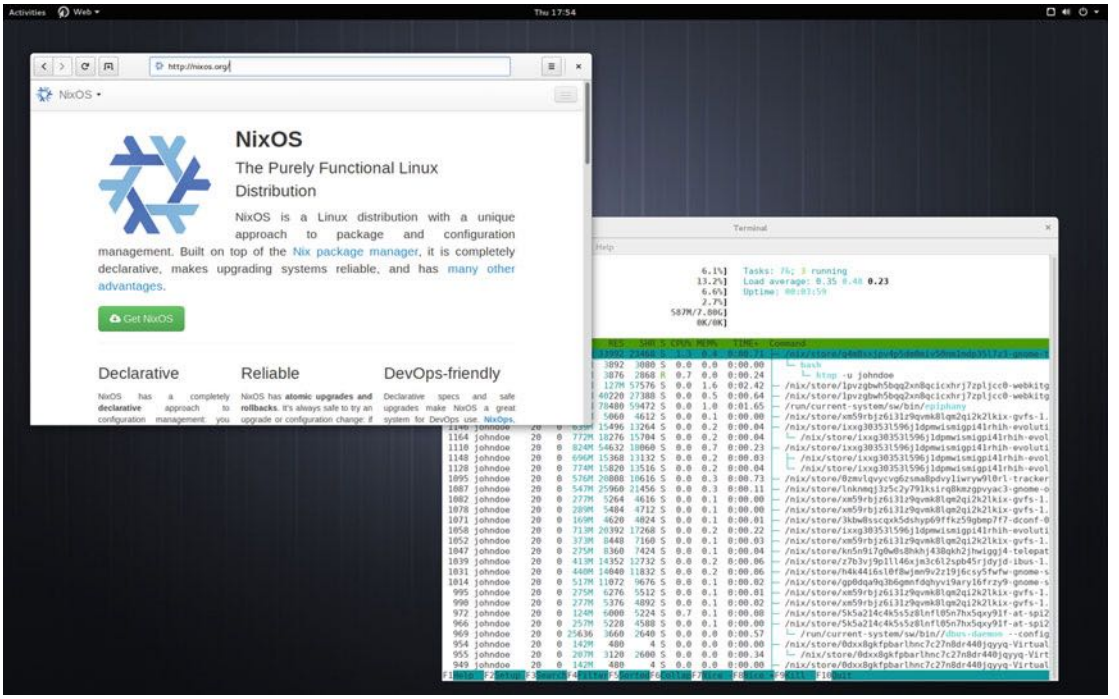


Figure 14-16. *A Gnome Shell session from changing two lines*

I think you can see how amazing this distro can be. All of the "mutable state" is kept between changes, of course; you can see that all of the operations you already performed in the terminal are present in the history.

And finally, when you restart the system, if you choose the second menu option, you will see something similar to Figure 14-17, which shows all of the configurations that you made. There's nothing to stop you from booting in the previous configuration, returning to KDE, and with `$ sudo nixos-rebuild switch`, making it your default configuration again. And if you want to switch frequently between configurations, you even can give them a name and add them as a new menu entry in the main Grub menu.

Figure 14-17. *In the Grub menu, you can choose between all of your previous configurations*

Many Linux users would really like to have something like this in their current distributions, myself included. I look forward to seeing how NixOS evolves.

Maintenance

All the maintenance in this distro is Nix related, and you should use Nix tools to perform the tasks to make sure they go as smoothly as possible.

Managing Apps

There are two different ways to install/remove applications in NixOS, either in a declarative way through the configuration file(s) or in an imperative way, as with traditional package managers. If I were managing a multi-user system I would use the declarative way to install all of the global packages available for all users, and I would let the users install their own packages in an imperative way. In the case of a one-person machine, I would install all the base packages from the configuration and I would only use the command line to install a few packages, such as those I only needed temporarily.

Installing packages in NixOS is very easy. Use this simple command:

```
$ nix-env -i firefox
```

To remove it, there are two options: make a rollback (I would only use this option if there are no other operations later that would change the state, like modifying the configuration or installing/removing/updating packages) or remove it. Use this code:

```
# This would remove the package (remember, not phisically from disk)
$ nix-env -e firefox

# This would perform a rollback
$ nix-env rollback
```

Updating

Again, there are two ways of updating packages in NixOS. You can update them using the imperative way with Nix or using the declarative way by rebuilding the entire system. With the first way, you can even upgrade packages individually.

To update something in an imperative way, first you must update the channel information (using the `nixos` channel):

```
$ nix-channel --update nixos
```

Now it's time to update a package or all of the available updates. First, here's how to update an individual package:

```
$ nix-env -u firefox
```

To update all of the packages that have a new version available, use the following code:

```
$ nix-env -u '*'
```

And for updating all of the packages in the declarative way, you only have to rebuild your configuration, and it will perform all the needed updates:

```
$ nixos-rebuild switch
```

Upgrading

One of the advantages of NixOS is precisely this, to do reliable distro upgrades. These upgrades are made via the NixOS official channels (a sort of repository). When you install NixOS from a current release (16.03 in this case), you are usually subscribed to this channel, and the system updates are made through this channel. When a new release appears, you only have to subscribe to that release channel and do an upgrade via `nixos-rebuild` to upgrade your entire system. If you installed from the unstable branch or you subscribed your system to the unstable channel, then the distribution would follow the rolling release model and would always be updated.

Let's see how to do this. First of all, you need to know that official channels are available:

```
$ nix-channel --list | grep nixos
```

When a new release appears, you switch to the new channel release. Say that a new release appears this year in September (as scheduled):

```
$ nix-channel --list | grep nixos
nixos https://nixos.org/channels/nixos-16.09
$ nix-channel --add https://nixos.org/channels/nixos-16.09 nixos
```

Finally, you upgrade the entire system:

```
$ nixos-rebuild switch --upgrade
```

Once this process is finished, you end up with a final state that is the same as if you made a fresh install with your current configuration.

Pros and Cons

The following is a list of some of the things that I personally see as pros and cons of the NixOS distribution. There is always room for discussion in this matter, but I tried to be as objective as possible.

Pros

- You can build your system automatically from configuration file(s) that can be version controlled with a reliable final state.
- NixOS offers bleeding-edge technology that has so many advantages: reliability, consistency, rollbacks, atomicity, reproducibility, etc.
- NixOS offers a state-of-the-art package manager that can be used apart from this distro; it's also multiplatform.
- Together with NixOps and Disnix, it makes a very promising tool for DevOps.
- It's an original distro and package manager, not a fork.

Cons

- It's not a user-friendly distro. Even advanced Linux users must learn the NixOS way of doing things.
- The documentation is incomplete, sometimes outdated, and you must rely on other support channels when you try to do something non-trivial.
- It's still in development and has some challenges to solve in order to become a real alternative.
- For non-trivial things, like building your own packages, you must learn the Nix functional language.
- There are serious drawbacks and bugs if you try to do something outside the Nix way. You have to follow the Nix way for everything, even for your dotfiles if you want a nice experience.
- It has small community, and a far way to go to reach the critical mass necessary to be a real alternative and evolve faster.
- It has a small install base.
- The Nix interface could be improved (for things like searching packages).

Summary

In this chapter, you got a glimpse of a possible and promising future Linux scenario. Although it still has a long way to go, NixOS is the most advanced, and with a better theoretical foundation, of all the existing experimental Linux distros at the present. There are already a considerable number of people (about 5,000) who are using this distro as their daily Linux OS. Together with systemd, this distro could trace the path to follow in the future.

In the following chapter I cover more alternative distros (like Qubes). Unfortunately, I only have room to give you a brief summary of each of them, but I hope it helps you get a broader vision of the Linux distros panorama.

CHAPTER 15

Other Alternatives

In this chapter, I show you some general purpose distros that I want to analyze in detail, because they either offer a good alternative or have peculiarity that is remarkable enough to draw your attention. There are notable absences like Red Hat, SUSE or CentOS, but the type of user who is going to choose to install and manage those corporate-oriented distros mostly likely won't need to read a book like this to make that decision. Also, because of that, I focused this book on desktop distros and skipped environments like the cloud or servers, and other interesting and promising desktop distros like Papyros and Apricity OS are not covered here.

There are flavors of popular distros that differ only in the desktop environment. For example, in the case of Ubuntu you can find Kubuntu, Xubuntu, Lubuntu, Ubuntu Gnome, Ubuntu MATE, Ubuntu Kylin, Mythbuntu, and Edubuntu (and those are only the official ones). I won't cover these flavors, but maybe one of them will suite your needs. If you want to explore the wide variety of Linux distros available, go to http://distrowatch.com. This web site makes a big effort to keep up with the highly volatile distro scene and offers the best compilation available.

Zorin OS

The Irish Zorin OS (first released in 2009) is another distribution based on Ubuntu (as are Linux Mint and elementary OS). Zorin OS provides the closest experience to the Microsoft Windows OS. To do so, it provides a similar look for several Windows versions (or OS X), and it installs Wine by default. Wine is a tool that allows you to execute native Windows applications in Linux. But not all Windows applications can be executed with Wine, so you may often experience apps that hang or have some functionality disabled.

Zorin OS is a good option for long-time Windows users reluctant to use Linux because of the unfamiliarity of the interface. It's also a good option in a mixed corporate environment where you need to run Linux but still need to run some Windows apps. You can also install Wine in other distributions (some even have it installed by default or its alternative, PlayOnLinux), but Zorin is a ready solution out of the box with this kind of user in mind. It's currently a popular distro but with a smallish install size.

It offers free and commercial versions. The latter come with a very low price for additional looks, additional software, and more support directly from the developers. The support for the commercial versions is basically to help in the installation of the distro, but I have serious doubts about the need for this kind of help in an Ubuntu-based distro. The distro has a default Windows 7-based theme, which can be changed for Windows XP or Gnome. The commercial versions have additional themes to look like Windows 2000, OS X, and Unity.

The distro selection criteria for Zorin OS are summarized in Table 15-1. You can learn more about Zorin OS at http://zorinos.com.

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_15

Table 15-1. *Distro Selection Criteria for Zorin OS*

Purpose and Environment	Two free versions: Core and Lite. Two commercial versions: Business and Ultimate.
Support	Commercial and community
User Friendliness	A user-friendly distro, especially for former Windows and OS X users.
Stability	Stable as Ubuntu at first. Same release scheme as Ubuntu with an added delay.
Hardware Support	Same as Ubuntu.
Aesthetics	Windows 7, Windows XP, and Gnome. Commercial: Windows 2000, OS X, and Unity.
Desktop Environment	Gnome or LXDE. The commercial version adds Unity.
Init System	systemd for the current version, Upstart in older ones.
Package Management System	dpkg
Architecture	Intel/AMD 32-/64-bits
Security/Anonymity	Same as Ubuntu.
Principles and Ethics	Nothing in particular.
Live CD	Yes
Professional Certification	No

Trisquel

Released in 2007, Trisquel is another distro that uses Ubuntu as its base. In this case, this Spanish distro is notable for being one of the few distros that the Free Software Foundation recognizes and endorses as a Free GNU/Linux distribution. Of these recognized distros, Trisquel is the most popular and active. It uses the Linux-libre kernel and removes all the proprietary software that Ubuntu originally supports, as well as any kernel firmware binary blobs and proprietary drivers.

It is a community distro supported only by donations and by a small community. As a result, even though is it a friendly distro, due to the restriction of only offering free products, there is limited support for hardware. You will probably have problems with graphic cards, wireless devices and so on, so I recommend researching your hardware options before choose this distribution.

It has a special version called Sugar TOAST, which is aimed at learning environments that use the Sugar desktop interface designed for teachers and children. This desktop environment was originally developed for the OLPC (One Laptop per Children) project.

Trisquel is the Spanish word for triskelion, a motif consisting of three interlocked spirals; it's from the Celtic culture, which one flourished in Galicia (where Trisquel and I were born). The distro's logo is a triskelion, a tribute to the Debian distro.

The distro selection criteria for Trisquel are summarized in Table 15-2. You can learn more Trisquel at `https://trisquel.info`.

Table 15-2. *Distro Selection Criteria for Trisquel*

Purpose and Environment	A main version, a "mini" one (only English/Spanish), and a Sugar TOAST version
Support	Community
User Friendliness	A user-friendly distro, but with hardware restrictions.
Stability	Based on Ubuntu LTS releases, therefore it's very stable.
Hardware Support	Less than regular distros, because it only supports free software.
Aesthetics	Aside from custom logos and themes, nothing more.
Desktop Environment	Gnome or LXDE. Sugar desktop for Sugar TOAST version.
Init System	Currently Upstart because it's based on Ubuntu 14.04. In the future, it will be systemd.
Package Management System	dpkg
Architecture	Intel/AMD 32-/64-bits
Security/Anonymity	Same as Ubuntu, but provides tools/docs to preserve anonymity/privacy.
Principles and Ethics	Only Free Software. Endorsed by the Free Software Foundation.
Live CD	Yes
Professional Certification	No

PCLinuxOS

The American distro PCLinuxOS is another descendant of Mandrake (later Mandriva), like Mageia. However, while Mageia tries maintain the legacy of Mandriva, PCLinuxOS started down its own path a long time ago. PCLinuxOS was forked from Mandrake in 2003, and that in part explains the divergence with Mageia, because it was created 8 years prior. One of the peculiarities of this distro is the Full Monty edition, which comes with a plethora of preinstalled applications to suit several needs, each with its own predefined KDE virtual desktop.

One of the big differences with Mageia is that PCLinuxOS is a rolling release distribution, with major releases from time to time. Even as a rolling releases distro, the packages are not always the latest versions but are instead stable packages that are merged into their repositories. Another big difference is that it is one of the few distros that still uses System V as its init system, since its community strongly opposed the change to systemd.

Another remarkable difference is the package management system, APT-RPM, which is a customization of the original Debian app tool; APT-RPM works with rpm packages instead of dpkg ones. PCLinuxOS also uses a traditional dpkg graphical tool, Synaptic, as its default graphical package management tool. Only a few distros currently use this package system.

PCLinuxOS is the perfect choice for users who like the Mageia/Mandriva style and rpm, need a friendly distro, do not like systemd, and prefer a rolling release scheme.

The distro selection criteria for PCLinuxOS are summarized in Table 15-3. You can learn more about PCLinuxOS at `www.pclinuxos.com`.

Table 15-3. *Distro Selection Criteria for PCLinuxOS*

Purpose and Environment	Unique version with an ISO image for each flavor: KDE, MATE, and Full Monty.
Support	Community
User Friendliness	A user-friendly distro
Stability	Rolling release scheme, but packages are only released if they are stable enough.
Hardware Support	Same as Mageia: very good.
Aesthetics	Its own layout, customization, and design in the Full Monty edition.
Desktop Environment	KDE and MATE. The Full Monty edition has a customized KDE.
Init System	System V (sysv), by community decision to not switch to systemd.
Package Management System	APT-RPM
Architecture	Intel/AMD 32-/64-bits
Security/Anonymity	Reasonable security out of the box.
Principles and Ethics	Pragmatic approach: leaves these decisions to the user.
Live CD	Yes
Professional Certification	No

Manjaro

Manjaro is an Austrian/German/French distribution that was first released in 2011. It's an Arch Linux-based, friendly alternative for users who want to have the advantages of Arch but want to avoid the non-friendly process of installing it. Since it is based on Arch Linux, Manjaro can use its repositories and its package manager, but Manjaro provides a friendly installer and tools to manage the packages, repositories, and the basic system settings.

By default, Manjaro has its own repositories and documentation, but you can use the Arch counterparts, which is an additional advantage. One known disadvantage, and a source of frequent criticism, is that it is still a slightly immature distro, with some fiascos (some of them serious security ones) in the past. In spite of being a controversial distro with many detractors, it is still a very popular distro, probably because there are many people who love the goods of Arch but not its problems, and Manjaro is like a breath of fresh air for them.

The official desktop managers are KDE and Xfce, but the community also maintain versions like Cinnamon, Gnome, MATE, Enlightenment, LXDE, LXQT, Fluxbox, Openbox, Netbook, and PekWM.

The Manjaro installer tool, Calamares (an independent Linux installer used by several distros), is very similar to the one used in Ubuntu. There were several attempts in the past to develop a friendly Arch Linux distro, and Manjaro seems to be the first that did it well. These days it is more popular than Arch itself.

The distro selection criteria for Manjaro are summarized in Table 15-4. You can learn more about Manjaro at `https://manjaro.github.io`.

Table 15-4. *Distro Selection Criteria for Manjaro*

Purpose and Environment	One version with two flavors: KDE and Xfce. A net edition is available.
Support	Community support. Documentation is not as good as the original from Arch.
User Friendliness	A user-friendly distro
Stability	Rolling release scheme, and still a bit unreliable.
Hardware Support	Better than Arch, with hardware automatic detection.
Aesthetics	Customized themes, installer, and general look.
Desktop Environment	KDE and Xfce. Several others supported by the community: Gnome, MATE, etc.
Init System	systemd
Package Management System	Pacman with its own repositories, plus the ability to use the Arch ones.
Architecture	Intel/AMD 32-/64-bits
Security/Anonymity	Regular out-of-the-box offerings. It should take these topics more seriously.
Principles and Ethics	Same as Arch: the user has the last word.
Live CD	Yes
Professional Certification	No

Antergos

The Spanish Antergos distro (originally named Cinnarch because Cinnamon was the original desktop environment of the distro) is one of the latest Arch Linux derivatives (circa 2012) oriented to be user friendly, like Manjaro. It basically adds a graphical installer tool, Cnchi (originally developed from scratch and inspired by the Ubuntu one) to the Arch Linux base, keeping almost all the other stuff intact. It also offers a graphical package manager, pamac, that was developed on Manjaro, and it uses the original Arch Linux repositories plus its own one. In a few words, Antergos is Arch Linux made easy.

During the installation process you must select which desktop environment you want to use, from a selection of the most common ones. You can choose a browser or an office suite, but any choice will still install some common applications, in a similar way to other user-friendly distros like Ubuntu.

Antergos has become quite popular lately because it's a vanilla Arch Linux-based distro with a friendly graphical installer. Some Arch users choose it to install a new "pseudo" Arch Linux system without needing to do it manually.

Several companies support the maintaining of the distribution, but it is still a community-developed one. Antergos is a Galician word (the original and main developer is from Galicia, Spain) that means ancestors; picking this word suggests homage to its ancestor, Arch.

The distro selection criteria for Antergos are summarized in Table 15-5. You can learn more about Antergos at `https://antergos.com`.

Table 15-5. *Distro Selection Criteria for Antergos*

Purpose and Environment	One main version available as Live or minimal ISO.
Support	Community support, plus the possibility to use Arch Linux docs.
User Friendliness	A user-friendly distro
Stability	Rolling release scheme, as stable as Arch Linux.
Hardware Support	Better than Arch Linux, with automatic hardware detection.
Aesthetics	Theme and icons made in collaboration with the Numix project.
Desktop Environment	Gnome, Cinnamon, MATE, KDE, Xfce, and OpenBox
Init System	systemd
Package Management System	Pacman using the Arch Linux repositories.
Architecture	Intel/AMD 32-/64-bits
Security/Anonymity	The same as Arch Linux.
Principles and Ethics	Pragmatism: the user gets the last word.
Live CD	Yes
Professional Certification	No

Sabayon

The Italian Sabayon (circa 2005, formerly RR4 Linux) works as an amicable alternative to Gentoo, upon which it is based. It's a user-friendly distro that follows the Gentoo philosophy but with a few differences that allow novice users to use it, like a graphical package manager that manages binaries instead of source. Advanced users can have an almost similar experience to Gentoo. It uses the same graphical installer as Manjaro, Calamares, and supports several desktop environments by default by selecting the corresponding ISO image. It can be a good distro for users who want to start with Gentoo but do not want to do so in one big step, instead preferring to learn at their own pace while still being able to enjoy the advantages.

The main differences from Gentoo are the package management and the init system. Sabayon uses systemd instead of Gentoo's OpenRC as its init system. More interesting yet is that Sabayon manages packages in a very particular way. It has two package managers: Entropy and Portage. The first one was developed by Sabayon and works with binary packages from its own repositories; it's aimed at novices and it has a graphical interface, Rigo, to be friendlier. Advanced users still can use Gentoo's Portage to build packages from source. The binary packages are usually more outdated and stable than the source code, from where you can build your packages with Portage.

The name Sabayon comes from the French word for a traditional Italian dessert, Zabaglione or Zabaione. The distro selection criteria for Sabayon are summarized in Table 15-6. You can learn more about Sabayon at `https://sabayon.org`.

Table 15-6. *Distro Selection Criteria for Sabayon*

Purpose and Environment	Versions for Desktop, Server, ARM, Docker, and Vagrant.
Support	Community
User Friendliness	User friendly, but suitable for advanced users.
Stability	Rolling release scheme.
Hardware Support	Better than Gentoo because it supports hardware auto-detection.
Aesthetics	The default ones of each desktop environment.
Desktop Environment	KDE, Gnome, MATE, Xfce, and Enlightenment
Init System	systemd
Package Management System	Entropy and Portage
Architecture	Intel/AMD 32-/64-bits. Also ARM (currently only Raspberry Pi 2).
Security/Anonymity	Normal. Less focused that Gentoo.
Principles and Ethics	Nothing in particular.
Live CD	Yes
Professional Certification	No

GoboLinux

Like NixOS, the Brazilian GoboLinux distro tries to innovate the way a Linux distro works. Started in 2003, GoboLinux has a different approach than NixOS; its focus is not on the package management, it is on the file system. GoboLinux does not use the Filesystem Hierarchy Standard (FHS); it uses its own directory hierarchy, storing all the programs under the same root directory (/Programs), classified by categories and using symbolic links to another directory that works as an index (/System/Index). The rest of the root directories are /Users, /Files, /Mount, and /Depot. To maintain compatibility with standard Linux programs, it uses several hidden (by default) symbolic links that recreate the standard FHS. This has some advantages, like the possibility of have more than one version of the same package available at the same time.

The packages are compiled from source with the Compile tool using recipes (scripts that defines how a package has to be build and installed), and it uses its original filesystem hierarchy as a sort of a native database of packages (thanks to the index). A graphical tool is also available to manage the packages. GoboLinux also uses its own simplistic init system based on a few simple scripts. For a long time, the root user was named Gobo, another peculiarity of this distro.

It's a small distro with a very small community and it has not get received much recognition. Also, the different root directory structure is controversial. The distro selection criteria for GoboLinux are summarized in Table 15-7. You can learn more about GoboLinux at `www.gobolinux.org`.

Table 15-7. *Distro Selection Criteria for GoboLinux*

Purpose and Environment	Unique version available as a Live ISO image.
Support	Community
User Friendliness	Reasonably easily to install and maintain, but still not user friendly.
Stability	Rolling release alike, based on recipes.
Hardware Support	Reasonable, with hardware auto-detection.
Aesthetics	Basically the default of Enlightenment.
Desktop Environment	Enlightenment
Init System	Its own simple system based on a few scripts.
Package Management System	Compile, but it also uses the filesystem hierarchy to manage the packages.
Architecture	Intel/AMD 32-bit newer than Pentium IV (i686)
Security/Anonymity	Reasonable, plus the different file hierarchy.
Principles and Ethics	Nothing in particular.
Live CD	Yes
Professional Certification	No

Qubes OS

Like Android, there is some controversy over whether Qubes OS is a Linux distribution or not. Qubes OS defines itself as an OS instead of a Linux distro, and in fact it is not a Linux distro in a strict sense. Its co-creator, Joanna Rutkowska (a well-known computer security researcher), defines it not as Linux distro or as a hypervisor, but as a hypervisor user. I cover it here because a lot of people refer to it as a Linux distribution. Qubes OS is an American OS released in 2012, and it tries to be the most secure desktop OS possible using "security by isolation." To achieve this, it uses several virtual machines hypervisor, in which processes are completely isolated from the others.

Qubes OS runs a Xen hypervisor under the hood, which uses a customized Fedora as its "dom0" domain (the most privileged domain that manages the hypervisor). The user domains (also known as Qubes) usually are a Fedora, Debian, or Whonix Linux distro, but can also be a Windows OS. There are system and user domains; the system domains are separated into Secure GUI & Administration (dom0), Network, and Storage. The user domains, also known as AppVMs (they are actually lightweight virtual machines), are usually separated by tasks or security level: personal, work, untrusted, shopping, banking, etc. There are templates that can be used as a base for those user domains with a common base software but separate memory, storage, and networks. There is a mechanism to securely share files and copy-and-paste between those domains.

In order to use all the capabilities of the OS, it is necessary that the hardware supports certain constraints, like VT-x and VT-d CPU virtualization capabilities, TPM secure platform, UEFI, and preferably a SSD and an Intel GPU. The company provides a hardware compatibility list and even a Qubes-certified laptop provided by Purism.

Qubes is developed by a company called Invisible Things Lab, but is released as open source and free software. The support comes from the company and the community; in other words, ITL supports the official templates for Fedora and Debian, but the community also supports Whonix, Ubuntu, and Arch Linux.

The distro selection criteria for Qubes OS are summarized in Table 15-8. You can learn more about Qubes OS at `www.qubes-os.org`.

Table 15-8. *Distro Selection Criteria for Qubes OS*

Purpose and Environment	Unique version available as an ISO image and a Live USB image.
Support	Community and ITL
User Friendliness	Once you understand how it works, it's a reasonably user-friendly OS.
Stability	Standard release scheme. It's still a little immature.
Hardware Support	Limited if you want to use all of the OS features
Aesthetics	The desktop environment defaults
Desktop Environment	KDE and Xfce
Init System	systemd in the Fedora domain. Actually, it boots the Xen hypervisor.
Package Management System	RPM
Architecture	Intel/AMD 64-bits. Preferably with VT-x, VT-d, and TPM technologies
Security/Anonymity	Bleeding-edge technology; it's very secure and offers integrated anonymity features.
Principles and Ethics	Allows you run proprietary OSes like Windows. A pragmatic approach: the user decides.
Live CD	Yes, a Live USB
Professional Certification	No

Solus

An Irish distribution that originated at 2013 as a Chrome OS alternative, Solus was discontinued, later renamed twice, and finally completely retooled and released in the current form as Solus in 2015. It's an original distro that focuses on being user-friendly Linux. It has gained popularity lately for its original desktop environment called Budgie, which is not only elegant, functional, and light, but also intuitive and fresh.

Solus also has its own package manager, eopkg, a fork of PiSi (the former package manager from the Pardus distro). It has a small package base, and although its internal design is very simple, it is not precisely user friendly (it lacks a GUI). It's still an immature distro in the early stages so it has room to improve.

The distro selection criteria for Solus are summarized in Table 15-9. You can learn more about Solus at `https://solus-project.com`.

Table 15-9. *Distro Selection Criteria for Solus*

Purpose and Environment	Unique version oriented to the desktop.
Support	Community
User Friendliness	For installation and normal use, yes. For maintenance, no.
Stability	Standard release scheme. One major release a year, two years of support.
Hardware Support	Regular, with hardware auto-detection.
Aesthetics	As one of the core values of the distro, it's pleasant to the eye.
Desktop Environment	Budgie
Init System	systemd
Package Management System	eopkg and ypkg
Architecture	Intel/AMD 64-bits
Security/Anonymity	Regular
Principles and Ethics	Pragmatic approach: the user has the last word.
Live CD	Yes
Professional Certification	No

Void Linux

Another independent distro that originated in Spain in 2008, Void Linux was influenced by NetBSD (where the original developer was a maintainer). In fact, the most remarkable feature of this distro, the package manager, was originally designed for BSD and later ported to Linux. As in the case of NixOS, Void was created as a testing platform for the XBPS (X Binary Package System) package manager. There is also Xpbx-src to build binary packages from source inside containers without using the root.

Another unique feature is the init system, runit, which is a very simple system based on a directory tree. Another feature inherited from BSD is the replacement of the OpenSSL library for LibreSSL by default, and more secure and reliable alternative ported from OpenBSD.

The distro selection criteria for Void Linux are summarized in Table 15-10. You can learn more about Void Linux at `http://voidlinux.eu`.

Table 15-10. *Distro Selection Criteria for Void Linux*

Purpose and Environment	A unique version with several flavors for each desktop environment and ARM.
Support	Community
User Friendliness	Not a user-friendly distro
Stability	Rolling release scheme
Hardware Support	Regular
Aesthetics	The default of each desktop environment
Desktop Environment	Enlightenment, Cinnamon, LXDE, MATE, Xfce
Init System	Runit
Package Management System	XBPS
Architecture	Intel/AMD 32-/64-bits and ARM
Security/Anonymity	Reasonable security plus the use of LibreSSL
Principles and Ethics	By default, only free software, aside from kernel binary blobs
Live CD	Yes
Professional Certification	No

Alpine

An original Norwegian distribution that proclaims itself as a general purpose distribution, but almost nobody use it in that way, Alpine is a security-centered and very lightweight distro that is usually employed to build firewalls, routers, VPN, VoIP, servers, and set-top boxes. Lately, it has gained increasing popularity as a minimal Docker image (only 5MB).

It's built around musl (a C library alternative to glibc) and BusyBox (an alternative to the several Linux core utilities in one executable), both very lightweight alternatives. The security features include a Linux kernel with PaX and grsecurity patches and stack-smashing protection (SSP) in all of the packages. Also, Alpine can be loaded and run from memory RAM, one of the reasons why it is used in embedded devices.

Alpine use its own package manager, APK, with a considerable package base given its lightweight nature.

The distro selection criteria for Alpine are summarized in Table 15-11. You can learn more about Alpine at `www.alpinelinux.org`.

Table 15-11. *Distro Selection Criteria for Alpine*

Purpose and Environment	Several versions and additional images for Xen and ARM platforms.
Support	Community
User Friendliness	Not user-friendly
Stability	Stable version with standard release scheme, but unstable as a rolling release.
Hardware Support	Regular
Aesthetics	The default of each desktop environment
Desktop Environment	Gnome, Xfce, MATE
Init System	OpenRC
Package Management System	APK
Architecture	Intel/AMD 32-/64-bits and ARM
Security/Anonymity	Solid security
Principles and Ethics	Nothing in particular.
Live CD	No
Professional Certification	No

Stali

Stali is a German distribution that was released for first time in 2016. It brings another radical approach to the Linux world. The distribution was developed by the suckless.org community, a well-known group of programmers, and creators of some popular tools like the window manager DWM, the surf browser, and dmenu. This community tries to develop quality software with a focus on simplicity, clarity, and frugality. This group has strong principles; their name is a clear statement of such.

This minimalism is apparent in all aspects of the distribution, from its size (an ISO of only 34MB) to the packages available (only a handful of carefully selected ones). Also there is no package manager available; the distro itself is a git repository and you only need to make a `# git pull` to do an upgrade. This also allows you to downgrade to a specific release with a `# git checkout 0.2`.

As recognized systemd haters, they do not implement this init system in Stali. They developed their own init system, sinit (stands for suckless init). Another notorious difference is that they not follow the FHS (File Hierarchy Standard); they use a simpler filesystem structure instead.

But the main characteristic of this distro, and the one that gave it the name (Stali stands for Static Linux), is that all the binaries are static linked, so there are no dynamic libraries at all. The C library used is musl, and as a result, Kernel modules are not supported.

The distro selection criteria for Stali are summarized in Table 15-12. You can learn more about Stali at `http://sta.li`.

Table 15-12. *Distro Selection Criteria for Stali*

Purpose and Environment	Unique version available as ISO image
Support	Community
User Friendliness	Not user-friendly
Stability	Still in the early stages
Hardware Support	Poor
Aesthetics	The default of each desktop environment
Desktop Environment	DWM (window manager)
Init System	Sinit
Package Management System	None. Upgrades and installation using Git.
Architecture	Intel/AMD 64-bits
Security/Anonymity	Good. Its unique nature probably makes it less prone to common vulnerabilities.
Principles and Ethics	Nothing in particular.
Live CD	No
Professional Certification	No

LFS

Although it is not very popular, I think that this distro is the inspiration for many little distros over the past 15 years, because many Linux developers learned from this Canadian project, which started in 1999. The Linux From Scratch (LFS) project is not a traditional Linux distro (but it is a Linux distro); it consists of a book and a collection of source code packages to build your own Linux system from scratch. Obviously the book is the soul of the distribution, which intends to be a learning tool about how Linux works internally and the process of building a minimal but functional Linux system. There are several servers where you can download the collection of packages for each release of the book, which is continuously updated.

If you follow the instructions of the book, it is a tedious and long process to achieve what you can do in five minutes with any old, minimal distro. But at the end you will have a deeper knowledge about Linux that you could ever achieve using a friendly Linux distro like Ubuntu for 10 years. As a learning tool, LFS is priceless, but you need to have real interest in how Linux works to use this distribution.

The relative success of this project inspired other projects with the same philosophy, such as Beyond Linux From Scratch (BLFS) for building a more complete Linux system, Cross Linux From Scratch for alternative hardware architectures, Automated Linux From Scratch (ALFS) to automate the process, and Hardened Linux From Scratch to build a more secure Linux system. For example, you can use CLFS (or even LFS) to build a very minimal Linux system (under 10MB) or install it in embedded systems or obsolete platforms.

The distro selection criteria for LFS are summarized in Table 15-13. You can download the book and learn more about LFS at `www.linuxfromscratch.org`.

Table 15-13. *Distro Selection Criteria for LFS*

Purpose and Environment	Unique version available as a book and source code packages collection
Support	Community
User Friendliness	Obviously not, but at the same time it's a learning experience.
Stability	Standard release scheme with stable releases and betas
Hardware Support	Very limited and basic
Aesthetics	It's a minimal, compact, command line-based Linux system.
Desktop Environment	No. KDE, Gnome, Xfce, LXDE, and LXQt in BLFS.
Init System	System V or systemd
Package Management System	None. All packages are built from source.
Architecture	Intel/AMD 32-/64-bits. Other architectures covered with CLFS.
Security/Anonymity	Nothing in particular. The HLFS project covers this topic.
Principles and Ethics	All of the packages are free software.
Live CD	No, but there are several editions of older LFS releases.
Professional Certification	No

Summary

In this chapter, you saw a few distros that offer a more complete vision of what a Linux distro can be. Some of them are very new, and most likely nearly half of them will never achieve great popularity. However, they are still an example of the dynamic and active nature of the Linux community. Some of the more esoteric innovations of these distros may never be part of the mainstream, and some of them will probably not exist in the near future, but all of them are a testimony to the power of free software for developing new ideas.

In the next chapter, you are going to see a compilation of task-oriented distros, which were built to focus on one specific task or area for specialized work environments.

PART 3

Task-Oriented Distros

This is a short section consisting of one brief chapter. This is because covering the task-oriented Linux distros in the same way and at the same level of detail as the general purpose distros would require another book—not because there are as many task-oriented distros as general purpose ones, but because with task-oriented Linux distros there are so many different tasks to cover. So how could I select one task and one distro and not cover others? And how could I possibly know which areas would interest you?

The best solution is to present related tasks grouped by category and then provide a list and description of the relevant Linux distributions.

CHAPTER 16

Task-Oriented Distros

The same reasons behind the development of so many general purpose Linux distributions are behind the proliferation of task-oriented distros. The flexibility of Linux and the advantages of Free Software make it reasonably easy to make a Linux distribution that can fulfil a unique purpose. These distributions can be customized, but all of them were made with only one type of task in mind, as specific as building a firewall to protect a network from security breaches or as ambiguous as scientific research.

In this chapter, I list of several niche distros and briefly describe them. The list isn't comprehensive, but it is accurate. It only includes actively developed distros. (Why talk about distros that have been abandoned or are in an uncertain state?)

Mobility and IoT (Internet of Things)

These distros are to be installed on mobile devices, like smartphones and tablets, or on the Internet of Things (which refers to the interconnection through the Internet of all kinds of things outside the traditional concept of computing, like home electrical appliances, vehicles, etc.). In both cases, these distros have adapted to these new environments both in functionality and interface.

Android

The omnipresent Android system is the most extended mobile OS in the world. It currently has more than 50% of the market share on mobile devices. It was developed by Google, and it follows an open source model with some proprietary components (mostly drivers and firmware). There is some controversy over whether Android is really a Linux distro, but Chris DiBona (Google's open source chief), the Linux Foundation, and some popular journalists consider it one. www.android.com

Ubuntu Phone/Tablet/IoT

These Ubuntu alternatives, based on the Ubuntu distro for desktops, can be used on mobile phones and tablets. The ultimate goal of Ubuntu is to be a Linux system that allows you to use your phone/tablet as a PC. Ubuntu Core (also known as Snappy) is suited for the IoT. www.ubuntu.com/phone, www.ubuntu.com/tablet, and www.ubuntu.com/internet-of-things

Sailfish OS

Developed by Jolla, Sailfish OS evolved from the former MeeGo OS (by Nokia and Intel) that was derived from Maemo. It continues to be developed but its presence in the market is almost minimal (e.g. Fairphone 2). It's compatible with Android applications and hardware. http://sailfishos.org

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6_16

Firefox OS

Firefox OS is going to be discontinued soon, at least for smartphones, but it may continue to be developed for IoT (it's currently available for smart TVs). It basically consists of a Firefox browser over a Linux kernel with HTML5 applications. Like Android, there is room for discussion about whether it's a real Linux distro. http://mozilla.org/firefox/os

Cloud-Centered Operating Systems

These distros try to make real the Sun Microsystems motto from the 1990s that "The network is the computer," coined by John Gage in 1984. Nowadays, with the cloud available everywhere, we only have to bring this power to the desktop to close the cycle. And for that, these distros aim to replace local applications with web ones and store data on a cloud drive instead of on a local hard drive.

Chrome OS/Chromium OS

Chrome OS and Chromium OS are based in the Google cloud (and are made by Google), where all of the apps and storage reside (recently added support for Android Apps). They use the Linux kernel plus the Chrome browser, and they were originally based on the Gentoo distro plus the Ubuntu's Upstart init system. Chromium OS is the open source version. Chrome OS is only available as an OEM OS in laptops usually called Chromebooks or mini desktop PCs called Chromeboxes, which are made by several manufactures, Google included. http://google.com/chromebook

Cub Linux

Formerly Chromixium OS, Cub Linux is an Ubuntu-based distro that mimics the Chrome OS functionality and look. https://cublinux.com

Router Distros

Router distros are installed in network appliances, routers, or PCs to work as a network router. They can replace a previous manufacturer's firmware, or they can be used to build a router with a simple PC and several network cards. They usually provide several network services beyond the router functionality, like an included firewall.

OpenWrt

OpenWrt is a Linux distro for embedded devices focused on router network traffic. It is a popular and powerful alternative to the OEM firmware of many SoHo routers, but it can also be used on any Linux-compatible hardware. http://openwrt.org

Zeroshell

Zeroshell is suited for servers and embedded systems to provide network routing services. It is available as a Live CD or Compact Flash image and is managed from a web interface (hence its name). www.zeroshell.org

RouterOS

RouterOS is a router operating system based on Linux and developed by the computer networking manufacturer MikroTik to be used in its devices. It can also be installed in regular Intel/AMD computers. It's well known for being a solid, powerful, and secure OS. http://mikrotik.com/software

Embedded Systems and CNC

I include here two well-known Linux distros that can be installed in embedded devices (spanning a wide range) or in CNC (computer numerical control) industrial machines. While it is common to find Linux in embedded devices, Windows is the most common OS in CNC machines. In fact, I worked for several years in industries from different fields that had different kinds of CNC machines and I never saw Linux installed on any of them.

KaeilOS

KaeilOS is oriented to embedded devices (industrial devices, automation, medical equipment, automotive, etc.). Currently it is part of the Yocto project, a project by the Linux Foundation to produce distros suited for embedded devices. www.kaeilos.com, www.yoctoproject.org

LinuxCNC

LinuxCNC is a distro for controlling CNC machines like milling machines, lathes, 3D printers, plasma cutters, laser cutters, robot arms, etc. It can be installed as a normal package in Debian/Ubuntu or from a Live CD with a Debian distro included. www.linuxcnc.org

Storage / NAS

This type of Linux distro is suited for creating a NAS (network-attached storage) usually with several disks working in a RAID (redundant array of independent disks) setup. The most famous free software for this kind of task is FreeNAS, a BSD-based software that benefits from the adoption of the very reliable and powerful ZFS as its filesystem. Due to license conflicts, ZFS is not yet fully available in Linux; these distros instead work with BTRFS or other file systems to provide decent solutions.

Rockstor

Rockstor is an advanced NAS and cloud storage solution based on CentOS and BTRFS. Basically it is a friendly alternative to FreeNAS but it is based on Linux instead of BSD. http://rockstor.com

OpenMediaVault

OpenMediaVault is another FreeNAS alternative based on Debian and focused on the SoHo environment for building a NAS solution. It can easily be extended by plug-ins. www.openmediavault.org

Enterprise Server and Thin Client

Some of these distributions try to replace the functionality of a Windows server with Active Directory, Outlook, and the rest of the usual suspects in a mixed corporate environment. The thin client distro is a good software alternative to the usual hardware ones.

ClearOS

ClearOS is a web-managed commercial server platform that consolidates a lot of different services into one system (network, gateway, server, cloud, security, backup, mail, etc.). It is based on Red Hat. www.clearos.com

Zentyal Server

Formerly eBox, Zentyal Server is a commercial replacement of a Windows Server system, with compatibility with Outlook and Active Directory. It's based on Ubuntu. www.zentyal.com

Univention Corporate Server

Based on Debian, Univention Corporate Server works as a server for distributed heterogeneous and virtualized environments, working with Windows, OS X, and Linux systems. www.univention.com

NethServer

NethServer is a CentOS-based Linux server for small organizations that works as a central system with a model built on pre-configured modules. http://nethserver.org

Thinstation

Thinstation is a Linux-based thin client that can connect to Citrix, NoMachine, 2X ThinClient, MS Windows terminal services, VMWare, Cendio, Tarantella, X window systems, telnet, VMS, tn5250, and SSH. It can be booted from a network (Etherboot/PXE) or from a CD/USB/HD. http://thinstation.github.io/thinstation

Telephony

A hardware PBX (private branch exchange), also known as a business telephone system, can be very expensive, especially if it has complex functionalities like VoIP, an answering machine, call accounting, etc. A software PBX can use basic and cheaper hardware to achieve the same functions and is more flexible and easy to manage. Asterisk is a well-recognized software PBX, and these two distros are a good way to build a complete Asterisk system to suit your needs.

AsteriskNOW

AsteriskNOW is a CentOS-based distro that can be used to build an Asterisk system, which is a software-based PBX for managing voice calls, voice mail, VoIP, automatic call distribution, and more. www.asterisk.org/downloads/asterisknow

Elastix

A CentOS-based distro that can unify all enterprise communications in one solution, Elastix uses Asterisk to provide a PBX with fax, instant messaging, e-mail, VoIP, and video capabilities. www.elastix.org

System Troubleshooting

In an effort to make the lives of system administrators and computer technicians easier, a handful of distros focus on deployments, recoveries, and repairs of computers. Often running a distro off a Live CDs works, but if not, these distros can be very helpful. They're also a great choice if the software you need is not installed and there is no network connection available, or if you just like to have a set of lifesaving tools handy. In fact, several of these tools have saved my day more than once.

GParted Live

GParted Live is a Live CD distro focused on managing disk drives, using the popular graphical tool GParted and other related tools. http://gparted.sourceforge.net

SystemRescueCD

SystemRescueCD is a compilation of tools over a Gentoo distro oriented to rescue a failed system (Linux or not). It's shipped as a Live ISO image. www.system-rescue-cd.org

Grml

Of interest to system administrators, Grml is a Debian-based live CD distro that offers many tools (2GB compressed into less than 500MB) for rescue systems and deployment. It can also be installed as any Linux distro. http://grml.org

Rescatux

Rescatux is a Debian-based distro with a graphical interface that guides you through menus to easily resolve many common problems and rescue a system. www.supergrubdisk.org/rescatux

Clonezilla Live

Clonezilla is another Debian-based Live CD that is an imaging/cloning/partition tool similar to True Image or Norton Ghost. It's useful for deployment, backups, and recovery. www.clonezilla.org

Redo Backup and Recovery

Similar to Clonezilla but with a graphical interface that makes it user friendly, Redo Backup and Recovery is based on Ubuntu. http://redobackup.org

Security and Anonymity

Security is a vast field in computing, covering almost every specialty you can imagine. Wherever there is hardware, software, or a network, security matters, and it is becoming even more important as time goes on. You can approach security from different points of view: penetration testing to discover weaknesses of a system, forensics to collect evidence and analyze it, blocking and analyzing attacks from outside your system, securing your identity and communications from potential eavesdropping, analyzing malware to understand how it works, and so on. These distros cover one or more of these computer security approaches.

Kali Linux

Formerly known as Backtrack, Kali Linux is a Debian-based distribution that offers several penetration testing and digital forensics tools. It is the most recognized distro of this kind. www.kali.org

BackBox Linux

Based on Ubuntu, BackBox Linux is focused on penetration testing, network analysis, and computer forensics. It also offers ethical hacking tools. https://backbox.org

Fedora Security Lab

A Fedora Labs project, Fedora Security Lab offers security auditing, system rescue, and forensics, plus a safe environment in which to teach security testing. https://labs.fedoraproject.org/en/security

BlackArch

Another alternative to Kali and BackBox, Black Arch is based on Arch Linux. It offers 1,400 tools for penetration testing and security research. http://blackarch.org

Parrot Security OS

Parrot Security OS is a suite of tools that provide penetration testing, computer forensics, cryptography, and anonymity. It's based on Debian and is ready for cloud environments. https://parrotsec.org

Wifislax

A Slackware-based distro, Wifislax is oriented to network security. It's a very lightweight distro with a big focus on wireless networks, and it supports a great number of wireless adapters. http://wifislax.com

Tails

Tails stands for The Amnesic Incognito Live System and it is a Debian-based Live CD that is focused on providing complete Internet anonymity. It provides tools to navigate through Tor or I2P by default plus communication cryptographic tools. https://tails.boum.org

Whonix

A Debian-based distro focused on privacy, security, and anonymity on the Internet, Whonix uses the Tor network by default and uses two virtual machines to separate the desktop from the network. www.whonix.org

CAINE

CAINE (Computer Aided Investigative Environment) is an Ubuntu-based distro that offers a complete suite of tools for professional digital forensics analysis. www.caine-live.net

DEFT

A Debian-based Live CD distribution for computer forensics and incident response, DEFT stands for Digital Evidence & Forensics Tool. http://deftlinux.net

IPFire

An original and lightweight distribution to build firewalls to secure network traffic, IPFire is based on a modular design. It can also work as a proxy server, IDS, virus scanner, or a VPN gateway. It has an easy-to-use web interface. www.ipfire.org

Untangle NG Firewall

Untangle NG Firewall is a network gateway distribution based on Debian. It has a modular design that can work as a firewall, IDS, VPN, web filter, etc. It also has a very friendly web interface. www.untangle.com/untangle-ng-firewall

Endian Firewall

Endian Firewall is a firewall distribution based on RHEL (Red Hat Enterprise Linux). It can work as a firewall and offers IPS, antivirus, web, and e-mail security inspection. www.endian.com/community/overview

SELKS

Built upon Debian, SELKS is a specialized Live CD distro for running a suite of tools based on Suricata, which is a tool that provides a network IDS, IPS, and NSM engine. The other tools are Kibana for analyzing alerts and Scirius to configure the Suricata rules. https://stamus-networks.com/open-source/#selks

REMnux

REMnux is a very specialized Ubuntu-based distro that performs reverse engineering of Windows and Linux malware. It also provides tools to analyze Flash programs, obfuscated JavaScript, PDF files, and memory. https://remnux.org

Old Computers

People with few resources like those in developing countries often can't keep up with the pace of the computer technology race, and so they usually have outdated (and frequently second-hand) hardware. The following distros are good options to pair with older hardware.

Puppy Linux

Puppy Linux is a very popular Live CD distro that works well on old computers with little memory and CPU resources. It is an original distro and it is very, very lightweight; it loads into RAM (it needs 256MB at maximum, but it can work with only 48MB) to provide a fast but full-featured desktop. http://puppylinux.org

Tiny Core Linux

Tiny Core Linux is a minimal distro that is about 16MB but only needs 46MB of RAM. It has a very minimalistic desktop that is not installed by default, but you can easily install it. http://tinycorelinux.net

LXLE

LXLE is based on Ubuntu LTS (another lightweight distro) and it uses the lightweight LXDE desktop manager. It can work with only 512MB of RAM and 8GB of hard drive space. www.lxle.net

Science

Science is unthinkable without computing anymore. Sometimes you need tools suited for your field, and sometimes you need computing power to run complex algorithms that are very resource-hungry. The following distros were built by scientists for scientific needs.

Scientific Linux

A Red Hat-based distro originally co-developed by the Fermilab and CERN (now switched to CentOS), Scientific Linux address scientific computer needs with infrastructures and research tools. www.scientificlinux.org

Bio-Linux

Bio-Linux is based on Ubuntu and offers a full-featured bioinformatics workstation with a plethora of tools focused on that environment. http://environmentalomics.org/bio-linux

Fedora Scientific

Fedora Scientific is a Fedora Labs project that offers a collection of the most used open source scientific and numerical tools preinstalled by default. http://labs.fedoraproject.org/en/scientific

Education

These education distributions were built to be an alternative to the proprietary OSes and software so widespread in schools and colleges around the world. These options range from being a cheaper way to build a computer lab to supporting all of the aspect of an educational institution.

Edubuntu

Edubuntu is a special flavor of Ubuntu dedicated to education, and it was created to be used in the classroom. The main focus is to help teachers with limited computer skills create a computer lab. www.edubuntu.org

UberStudent

UberStudent is another distro based on Ubuntu for educational purposes but it is focused on students, teachers, and schools of secondary and higher education. http://uberstudent.org

DebianEdu

DebianEdu, also known as Skolelinux, is a Debian-based distro focused on providing a free software alternative to proprietary educational software. https://wiki.debian.org/DebianEdu

Home Theater and Audiophile Systems

If you want to build a home theater system for your living room or you want to enjoy your music at an audiophile level, these distributions can do that for you. They can help you achieve a cheaper but yet very powerful alternative to the usual commercial ones. You can create an audiophile system with a simple Raspberry Pi, a DAC, an amplifier, and a pair of speakers. As for home theaters, the following distros make these systems more flexible than any commercial alternative.

Mythbuntu

Mythbuntu is a derivative of Ubuntu with MythTV preinstalled, which is media center software for building a home theater PC. It can work as standalone system, as a server for several clients in the home, or in a mixed configuration. www.mythbuntu.org

OpenELEC

OpenELEC lets you make a media center out of an embedded device or single-board computer (like the Raspberry Pi family). It uses the popular and powerful Kodi Entertainment Center software as an HTPC inside a minimal Linux system. Several manufacturers use it as an OEM OS in set-top boxes, hardware media players, and media center systems. http://openelec.tv

OSMC

OSMC, formerly Raspbmc, is another media center for Raspberry Pi and Apple TV, a distribution that also uses the Kodi Entertainment Center software with a minimal Debian distro underneath. http://osmc.tv

Rune Audio

Rune Audio is the distro installed in Raspberry Pi and similar devices, which you can convert to a HiFi digital audio player (usually connected to a DAC). It is based on Arch Linux and can be controlled remotely from various clients (Android, iOS Web, Windows, and Linux). www.runeaudio.com

Volumio

Volumio is a Debian-based distro for audiophiles, and it can be installed on embedded devices like Raspberry Pi and similar. It can play music from several sources like Mp3, MPD, Spotify, SoundCloud, Last.FM, etc. It can be remote controlled from a web interface. https://volumio.org

Gaming

The following distros can make your computer a gaming system, something traditionally reserved for Windows machines and game consoles. Linux is not at the same level as those systems, but these distros will convince you that it's possible to enjoy games in Linux.

SteamOS

SteamOS is a Debian-based distro that works as the gaming platform for the Steam Machine video games. It was initially intended to be the OEM OS for the Stem Machines, but you can also build your own system and install Steam OS. http://store.steampowered.com/steamos

Fedora Games Spin

A flavor (Spin) of Fedora that works as a showcase for the Fedora distribution as a gaming platform, Fedora Games Spin includes several preinstalled games. https://labs.fedoraproject.org/en/games

Multimedia and Arts

Artists and creators are the intended audiences for these distros, which offer tools suited for their needs. OS X and Windows may have better tools in several of these fields, but they are also usually expensive, while Linux distros are cheaper or free and still let you achieve professional results. Some tools, like Blender, are very powerful and industrial grade.

Ubuntu Studio

Ubuntu Studio is a based on Ubuntu, obviously, but it adds creative tools for audio, video, graphics, photography, and publishing tasks. http://ubuntustudio.org

Fedora Design Suite

Fedora Design Suite provides several multimedia production and publishing tools by default.
https://labs.fedoraproject.org/en/design-suite

Summary

This chapter covered a wide range of tasks and distros for each of them. This compilation could be more extensive, both in tasks and in distros, but I think that this is enough for you to get an idea of what you can expect. Note, however, that these distros are usually more ephemeral than the traditional ones, and they are usually abandoned or replaced by others in a matter of a few years. If you are interested in a task-oriented distro, you should search beyond the distros and tasks mentioned in this chapter.

A bonus chapter awaits you online! In it I show you several operating systems that are not Linux but have enough in common with Linux to deserve your attention. If you are willing to learn more about them, go to

Index

J. Dieguez Castro, *Introducing Linux Distros*, DOI 10.1007/978-1-4842-1392-6

E

F

G

H

I, J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z