Arch Linux Environment Setup How-to

Get started with Arch Linux as a blank canvas and build the simple and elegant environment you want

Ike Devolder

Arch Linux Environment Setup How-to

Get started with Arch Linux as a blank canvas and build the simple and elegant environment you want

Ike Devolder

Arch Linux Environment Setup How-to

Copyright © 2012 Packt Publishing

Credits

Author

Ike Devolder

Reviewers

David Couzelis

Michael Driscoll

Acquisition Editor

Joanna Finchen

Commissioning Editors

Maria D'souza

Shreerang Deshpande

Technical Editor

Manmeet Singh Vasir

Copy Editor

Alfida Paiva

Project Coordinator

Shraddha Bagadia

Proofreaders

Maria Gould

Aaron Nash

Production Coordinator

Prachali Bhiwandkar

Cover Work

Prachali Bhiwandkar

Cover Image

Aditi Gajjar

About the Author

Ike Devolder started playing around with Linux in 1999 and became a full-time Linux user in 2002. At that time his preferred distribution was Slackware, which he used for many years without complaints. In 2005 he rediscovered Arch Linux (he had already played around with it in the past) and started considering switching all his desktops from Slackware to Arch Linux. In 2006 it was done, with his laptop and desktop both running smoothly on Arch Linux, and since then he has gradually stopped being a distrohopper.

These days Ike is a Web Developer for Studio Emma in Belgium. He uses Arch Linux in his day job too.

I would like to thank my wife for her patience, moral support, and love. I would also like to thank the whole Arch Linux community for bringing us a superb Linux distribution. In the end, I would like to thank my parents for supporting me all the way in whatever choice I have ever made.

About the Reviewers

David Couzelis is a professional Software Developer who fell in love with GNU/Linux over a decade ago. As a hobby he makes video games, and hopes to someday make one that's actually fun. He's also one of those free and open source software advocates, the type who teaches people about software freedoms even if they didn't ask.

Michael Driscoll has been programming in Python since spring 2006. He enjoys writing about Python on his blog at `http://www.blog.pythonlibrary.org/`. Mike also occasionally writes for the Python Software Foundation (PSF), IProgrammer, and Developer Zone. He enjoys photography and sitting down with a good book.

> I would like to thank my wife, Evangeline, for always supporting me. I would also like to thank my friends and family for all that they do to help me. Lastly, I would like to thank Jesus Christ for saving me.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- ▶ Fully searchable across every book published by Packt
- ▶ Copy and paste, print and bookmark content
- ▶ On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

I would like to dedicate this book to my daughter Moyra.

Table of Contents

Preface

Arch Linux is an independently developed, general purpose GNU/Linux distribution, optimized for i686/x86-64 systems. The distribution is versatile enough to suit you in any role/need. It has been designed focusing on simplicity, code elegance, and "do-it-yourself" principles. The basic installation of Arch Linux is a very minimal base system. From the base system, everything can and will be configured by the user to suit their ideal environment, suited for their own unique purposes. The supported method of configuration is from the shell editing simple text files. Being a rolling-release distribution, there are no fixed releases. From time to time there are new install images provided by the Release Engineering Team, so that the installation media suits the new features introduced over time. Because of this rolling-release model, Arch Linux provides you with bleeding-edge software, typically the latest stable versions available. Pacman is the package manager of Arch Linux, which is designed to be an easy to use binary package manager.

History

Arch Linux was founded in 2002 by Judd Vinet who was inspired by the elegance and simplicity of some other Linux distributions such as Slackware, Polish Linux Distribution, and CRUX. But this simplicity came without a package manager, which was a big disappointment. Basically, to make his own life easier, Judd Vinet started Arch Linux with pacman as the package manager that can automatically handle the installation, upgrade, or removal of packages. Over the years Arch Linux kept on gaining more users, more developers, and has now been in the top 10 used distributions on DistroWatch for a pretty long time. Arch Linux is still being developed by volunteers and is not backed by some big company; the goal is to stay *free* in every sense of the word. In 2007 Judd Vinet passed the Project Lead to Aaron Griffin, who remains the Lead Arch Linux Developer to this day.

The Arch Way

The Arch Linux philosophy, also described as "The Arch Way", is in general summarized as being KISS (keep it simple stupid). There are five core principles defined in this core: philosophy, simplicity, code-correctness over convenience, user-centric, openness, and freedom.

Simplicity is definitely the base objective for Arch development, with the idea that a lightweight base structure with high quality code will have lower system resource demands. The base system found in Arch Linux has no clutter that might hide parts of the system, or make it difficult to access parts of the system. All configuration files are simple, nicely documented, easy to read, and nicely arranged for quick editing. There are no special configuration tools that might hide possibilities from the user, which leads to a system configurable to the very last detail. The Arch Linux Developers believe that trying to hide the complexity of a system results in an even more complex system, and this should always be avoided.

> *Arch Linux defines simplicity as without unnecessary additions, modifications, or complications, and provides a lightweight UNIX-like base structure that allows an individual user to shape the system according to their own needs. In short: an elegant, minimalist approach.*

Code-correctness over convenience implies clean, correct, simple code and not unneeded patching, automation, eye candy or "newbie-friendliness". Software patches are only introduced when needed. Packages found in Arch Linux are really the way the Developers created them, nothing more, nothing less.

> *Simplicity of implementation, code-elegance, and minimalism shall always remain the reigning priorities of Arch development.*

Being **user-centric** means that the users fully manage the system on their own. The system will offer no, to little assistance. There is a simple set of maintenance tools that are simply relaying the commands given by the user. Arch Linux is founded on simple, sensible design, and excellent documentation. As a user, you need more of a "do-it-yourself" approach rather than asking the Developers to implement a new feature. Most Arch users have indeed the tendency to solve their problems and share them with the entire community, which also leads to a friendly and helpful community of developers and users.

> *Arch Linux targets and accommodates competent GNU/Linux users by giving them complete control and responsibility over the system.*

Openness is also briefly touched upon in the just discussed principle, saying most Arch users share the solutions to the problems they have encountered. Also, the Arch Linux Developers strive for an open system, which goes hand in hand with simplicity. Openness is intended to make things simple. It also removes the abstraction of things, which might lead to a steeper learning curve, but in the end it leads to an easily controllable and maintainable system. More experienced Arch users find some of the helper tools provided by other distributions cumbersome, and things that are sitting in the way of easy and fast configuration. The Arch Linux community is also known to be very open and willing to give advice.

> *Arch Linux uses simple tools, that are selected or built with openness of the sources and their output in mind.*

Freedom is probably one of the most important things you get when you start using a Linux-based operating system. This is where Arch Linux is one of the leading distributions where all configurations and decisions regarding the system are taken by the user. In the end, the user defines the system. Arch Linux is built in such a way that you can—if you really want to—rebuild the entire system. It even provides a simple tool to do so. The entire system and all of the components are 100 percent transparent, so you can replace everything with something else providing the same functionality. In addition to this, Arch Linux also provides the freedom to use exclusively open source software, but you can also without hassle use proprietary software packages.

> *By keeping the system simple, Arch Linux provides the freedom to make any choice about the system.*

Ending with a final quote from Judd Vinet:

> *[Arch Linux] is what you make it.*

Bleeding edge

Arch Linux strives to maintain and have the latest and greatest software available in its repositories. Being a rolling-release distribution, it allows a one-time install with continuous updates without ever having to reinstall or having to go to massive procedures for getting the system upgraded. The goal of bleeding edge is also to provide for the newest features popping up in the Linux world—be it in the field of filesystems (ext4, ReiserFS, XFS, JFS, Btrfs, NILFS, and so on), software RAID, or boot scripts (systemd)—along with supporting the latest features of the newest hardware because of the immediate availability of the newest kernels.

What this book covers

Installing Arch Linux using the official ISO (Should know), explains the procedure to get Arch Linux installed on your system using the official installation media.

Installing Arch Linux using the Archboot ISO (Should know), explains the procedure to get Arch Linux installed on your system using the Archboot installation media.

Configuring your system (Should know), explains where to set your hostname, load special modules, and have a different keyboard layout than QUERTY.

Installing and removing packages (Must know), explains how to use the package manager (pacman) for adding and removing packages.

Booting and managing services with systemd (Should know), explains configuring and managing how your system starts up and what services are being started automatically, but using systemd and not sysvinit.

Booting and managing services using initscripts (Should know), explains configuring and managing how your system starts up and what services are being started automatically using initscripts.

Configuring GUI using Xorg (Should know), briefly explains how to get your GUI up and running.

What you need for this book

In general when reading this book, we assume you already have a basic knowledge of how to install and use a Linux distribution such as Ubutnu, openSUSE, Fedora, and so on.

Arch Linux in general aims at more experienced users and also users with a big "do-it-yourself" mentality.

Who this book is for

For those who love free software and want to use a super-customizable Linux distribution, all those wanting the latest software when it is released, and everyone wanting to end up with a system customized to their needs and wishes, Arch Linux is the way to go.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "Add the `nomodeset` parameter to the kernel command line to make sure that the open source drivers will not kick in."

A block of code is set as follows:

```
Section "InputClass"
    Identifier "evdev keyboard catchall"
    MatchIsKeyboard "on"
    MatchDevicePath "/dev/input/event*"
    Driver "evdev"
    Option "XkbModel" "pc105"
    Option "XkbLayout" "be"
EndSection
```
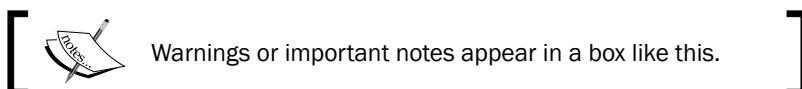
When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
DAEMONS=(syslog-ng !network crond)
```

Any command-line input or output is written as follows:

```
systemctl list-units --type=service
systemctl list-units -a --type=service
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "The **Auto-Prepare** option will guide you with the creation for a default partition scheme."

> Warnings or important notes appear in a box like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the **Errata** section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

Arch Linux Environment Setup How-to

Welcome to Arch Linux Environment Setup How-to. Arch Linux is a very flexible distribution and this book will guide you to the point where you can get a basic system in place. From there on you can go in any direction you want. A simple server, a full-blown desktop system with all the bells and whistles. In the end, Arch Linux is always what you make of it.

Installing Arch Linux using the official ISO (Should know)

These days the installation of Arch Linux might look like a work of insanity, as there is no installer available in the official media, just a guideline to follow. The installation without an installer is as easy as it can be. For experienced users, installing without an installer is even more convenient. The newest ISOs require that you have your machine connected to the Internet, as there are no longer packages available on the installation media.

Getting ready

You can get the official ISO image file from `https://www.archlinux.org/download/`. On this page you will find a download link to the latest release. Depending on your preference, download the torrent file or the ISO image file immediately.

The following list describes the main tasks that we will perform in this recipe:

> ▸ **Preparing, booting, and setting keyboard layout**: We are going to get the ISO file from the download page of the Arch Linux website and store it on the preferred media of our choice. At the time of writing this book, there is a dual ISO image file that contains both i686 and x86-64 architectures on one disk. Start your PC with your preferred installation media (CD or USB stick). On most PC systems, you can access the boot menu by pressing one of the function keys, usually between *F8* and *F12* depending on the motherboard manufacturer. On older machines where you do not yet have a boot menu, you might need to change the boot order in the BIOS where the CD-ROM (or DVD/Blu-ray) has to be chosen as the first device to try booting from. We'll also explain how to use a different keyboard layout than the default one in this recipe.

> ▸ **Creating, formatting, and mounting partitions**: You can partition the disks the way you want using cfdisk (for MBR disk partitioning) or cgdisk (for GUID disk partitioning). After creating the partitions, we can choose to format our created partitions with specific filesystems. When all partitions are formatted, we need to mount the partitions. First we will mount the root partition to `/mnt`. The other partitions will be mounted later on after you have created the specific folders. We'll designate our device with `/dev/sdX`; in your case this can be `/dev/sda`, and so on.

> ▸ **Connecting to the Internet**: To be able to continue installing the ISO you need to connect to the Internet, because there are no packages available for installation on the ISO. For a wireless network you will need to use netcfg. When connected to a wired network, just use dhcpcd or dhclient.

> ▸ **Installing the base system and boot loader**: These days the base system gets installed by running a simple script **pacstrap**. Pacstrap takes multiple parameters, the target location, and the packages or groups you want to install. For people who want to develop on their machines, the best `base` install is adding `base-devel` to the default installation. For normal end users, just `base` will be sufficient to start.

> ▸ **Configuring the system**: In this recipe, we'll describe the flow of what to do during the configuration. For more extensive information on how to configure your system, refer to the *Configuring your system* recipe.

How to do it...

The following steps will guide you in preparing, booting, and setting keyboard layout:
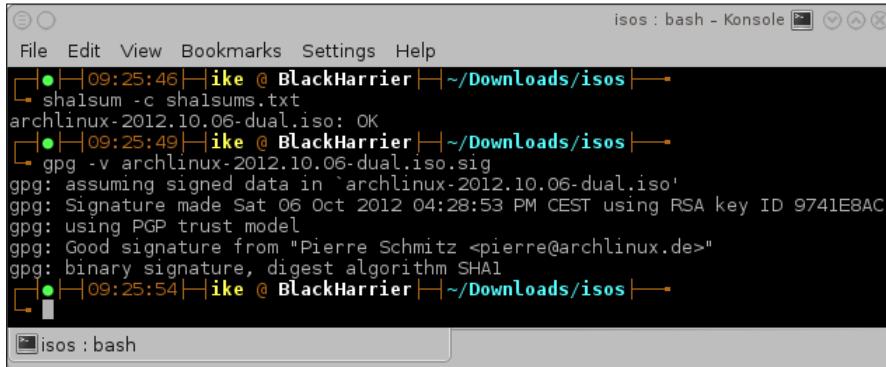
1. Once you have downloaded the ISO image file, you should also verify its integrity by downloading the `sha1sums.txt` file from the download page.

   > These days you can also check if the ISO is completely valid by verifying the signature of the ISO.

2. Verify the integrity by issuing the `sha1sum -c sha1sums.txt` command and you'll see whether your download was successful or not. Also check if the signature of the ISO is correct by running `gpg -v archlinux-...iso.sig`:

   ```
   sha1sum -c sha1sums.txt

   gpg -v archlinux-2012-08-04-dual.iso.sig
   ```

   The following screenshot shows the execution of this step:
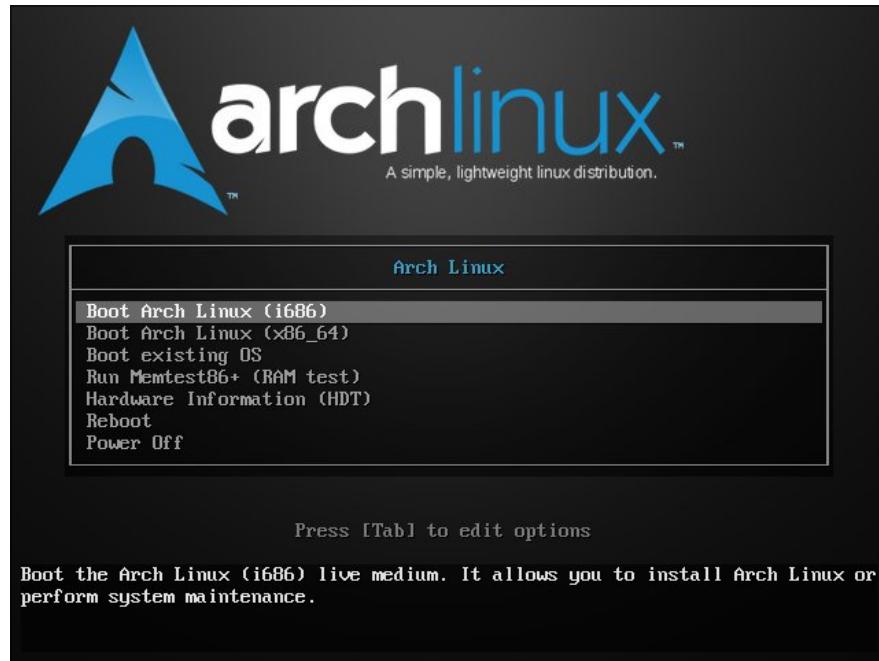
   ```
                                                      isos : bash - Konsole
   File   Edit   View   Bookmarks   Settings   Help
   ●  09:25:46  ike @ BlackHarrier  ~/Downloads/isos
    sha1sum -c sha1sums.txt
   archlinux-2012.10.06-dual.iso: OK
   ●  09:25:49  ike @ BlackHarrier  ~/Downloads/isos
    gpg -v archlinux-2012.10.06-dual.iso.sig
   gpg: assuming signed data in `archlinux-2012.10.06-dual.iso'
   gpg: Signature made Sat 06 Oct 2012 04:28:53 PM CEST using RSA key ID 9741E8AC
   gpg: using PGP trust model
   gpg: Good signature from "Pierre Schmitz <pierre@archlinux.de>"
   gpg: binary signature, digest algorithm SHA1
   ●  09:25:54  ike @ BlackHarrier  ~/Downloads/isos

   isos : bash
   ```

3. As you can see in the previous screenshot, the ISO's checksum is ok and the signature is valid.

4. Now that we are sure our ISO is ok, we can burn this to a CD with our favorite burning program.

5. Insert the CD into the drive, or insert the USB stick into the USB port of your PC.

6. Enter the boot menu, or let your computer automatically boot from the inserted installation media.

7. If the previous steps are performed correctly, you will see the following screenshot:

8. Select the architecture you want and press *Enter*, and we'll be on our way.

9. Search the keyboard layout desired for your region. The available keyboard layouts can be found at `/usr/share/kbd/keymaps/`.

10. Set the desired keyboard layout with `loadkeys keyboardlayout`.

Now let's perform the following steps to create, format, and mount partitions:

1. Start cfdisk or cgdisk, having the first parameter as the device you want to partition:

   cfdisk /dev/sdX

   cgdisk /dev/sdX

2. Create your partition scheme.

3. Store the partition scheme.

4. Use the `mkfs` command to create a filesystem on a specific partition:

   mkfs -t vfat /dev/sdX

   mkfs.ext4 -L root /dev/sdX

5. Mount your root partition to `/mnt`:

   mount /dev/sdX3 /mnt

6. Make directories under `mount` for your other partitions:

   `mkdir -p /mnt/boot`

7. Mount the other partitions:

   `mount /dev/sdX1 /mnt/boot`

The following steps are needed to connect to the Internet:

1. When we need a wireless network, create a netcfg profile and run `netcfg mywireless`.

2. Use dhclient or dhcpcd to get an IP address.

The following steps should be performed for installing the base system and boot loader:

1. Run pacstrap with the desired parameters:

   `pacstrap /mnt base base-devel`

2. Install the desired boot loader: the best choice at this moment is Syslinux.

3. The final installation of the boot loader will be done in a chroot during the initial configuration (discussed later in the book).

We'll now list the steps to do during the configuration:

1. Generate `fstab` with `genfstab`:

   `genfstab -p /mnt >> /mnt/etc/fstab`

2. Change the root into the system location:

   `arch-chroot /mnt`

3. Set your hostname in `/etc/hostname`.

4. Create `/etc/localtime symlink`.

5. Set your locale in `/etc/locale.conf`.

6. Uncomment the configured locale in `/etc/locale.gen`.

7. Run `locale-gen`.

8. Configure `/etc/mkinitcpio.conf`.

9. Generate your initial ramdisk:

   `mkinitcpio -p linux`

10. Finish installation of your boot loader.

11. Set the root password with `passwd`.

12. Leave the chroot environment (`exit`).

> For more extensive information on how to configure your system, refer to the *Configuring your system* recipe.

> For a detailed description of the main tasks performed, refer to the *Getting ready* section of this recipe.

How it works...

We downloaded the ISO image file via torrent, or via HTTP from the mirror sites listed on the download page. The `sha1sum` command lets us verify the integrity of the downloaded ISO. On top of the checksum, we can also check the integrity by verifying the signature available for the ISO. So now, we can rest assured that the downloaded file is the real one. The ISO contains a fully working operating system. It also contains all the necessary tools to perform system recovery and installation.

The keyboard configuration set with `loadkeys` will make sure that the key you press on your keyboard will be translated to the correct letter on your screen. Using a different keyboard layout from the one on your physical keyboard might be confusing.

We then created a partition scheme on the selected disk with the appropriate tool (cfdisk or cgdisk). **Make Filesystem** (**mkfs**) is a unified frontend to create a filesystem. Using it we created our filesystem layout manually under `/mnt` by creating our default partition layout in our root, and mounting the specific partitions accordingly.

You can make a connection with your wireless network (if needed), and then use dhcpcd or dhclient to obtain an IP address that enables you to access the Internet.

Pacstrap will run **pacman** with a modified root location to install the desired packages into the newly created system.

For example, installing Syslinux:

```
pacstrap /mnt syslinux
```

The specific configuration files will ensure we don't have to do all those steps over and over again on every boot.
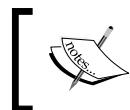
There's more...

You can use the official ISO directly from the USB stick if you prefer. There might be some issues with the verification of the ISO. The next two sections discuss them and provide you with the solution.

Using the ISO from a USB drive

The ISOs downloaded for Arch Linux are all 'hybrid' images, which means you can put them on a USB drive and they will be bootable. So installing from a USB drive is also very simple. Place a USB drive in your machine (*warning: it will lose all its data*) and issue the following command:

```
dd if=archlinux-2012.08.04-dual.iso of=/dev/sdX bs=1M
```

> Make sure you have `if=the correct ISO filename` and `of=/dev/sdX`, where you don't use a *partition* of the USB drive like `/dev/sdX1`, but the full USB drive. So use only `/dev/sdX`.

Having problems verifying the ISO signature?

When you don't have the signer's public key in your `gpg keyring` you will get an error like **gpg: Can't check signature: No public key**. This means you will first have to import the signer's public key before verification of the signature is possible:

```
gpg --keyserver wwwkeys.pgp.net  --recv-keys 0x9741E8AC
```

Import the public key, in this case the public key of Pierre Schmitz. Then you can run the verification of the ISO again. The verification should now give you **gpg: Good signature from "Pierre Schmitz <pierre@archlinux.de>"**. When you have done the steps described here, you will get a warning that the key is not certified with a trusted signature. In the case of verifying the integrity of the ISO, this is of no importance. For more information about GPG and signatures, see `http://www.gnupg.org/`.

The next section talks about a common sample of a good desktop partition scheme.

A good partition scheme for desktops

On a desktop system, especially Arch Linux, I personally suggest having a separate `/var` partition. Depending on the other goals you might have for this partition (for example, running a huge MySQL database, other databases, and so on), a sensible value would be 5 GB and above. Don't overdo it or you will have a lot of empty space in the `/var` partition. Why make `/var` so big? Pacman keeps its cache in `/var`, and you don't really want your root filesystem being deadlocked by a filled up disk with package cache.

- **Boot partition**: 50 MB
- **Swap partition**:
  - When your RAM is less than 4 GB: RAM + one third of RAM
  - When your RAM is greater than 4 GB: Fix it on 4 GB (there is actually no real need to make it bigger)
- **Root partition**: 10 GB (gamers might want to go to 50 GB here)

▶ **Var partition**: 5 GB (if only used for cache) when having some data from databases stored on it; I would go up depending on the need

▶ **Home partition**: These days you'll end up having 300 to 400 GB and even more

The netcfg tool originally developed by Arch Linux provides us with tons of options, which we will discuss in the next section.

Netcfg sample configurations

The sample configurations can be found in the `/etc/network.d/examples` folder. In the following table, we have given a list of sample configurations provided by the netcfg package:

Connection type	Example profile
Wireless/WEP hex key	`wireless-wep`
Wireless/WEP string key	`wireless-wep-string-key`
Wireless/WPA-Personal (passphrase/preshared key)	`wireless-wpa`
Wireless/WPA-Enterprise	▶ `wireless-wpa-config`(`wpa_supplicant` configuration is external) ▶ `wireless-wpa-configsection` (`wpa_supplicant` configuration stored as a string)
Wired/DHCP	`ethernet-dhcp`
Wired/static IP	`ethernet-static`
Wired/iproute configuration	`ethernet-iproute`

> For the more extended explanations of some parts to follow, I would refer you to the *Configuring your system* recipe.

Genfstab extra options

If you prefer the use of UUID or label in your `fstab` file, you can pass an extra parameter to the `genfstab` script: `U` for UUID or `L` for labels.

Final installation of Syslinux

The final installation of Syslinux has to be done from within the chrooted environment.

`/usr/sbin/syslinux-install_update -iam`

Should the previous command fail while trying to set the boot flag, use the following command:

`/usr/sbin/syslinux-install_update -im`

After a successful installation of Syslinux, configure the way your system will boot by editing `/boot/syslinux/syslinux.cfg`.

Installing Arch Linux using the Archboot ISO (Should know)

In this recipe we'll learn about the installation of Arch Linux. It might seem a little scary because of the manual installation process, but it is not difficult at all. The installation script will practically guide you all the way through the installation, and you can have a system capable of booting in even less than 10 minutes. In the Arch Linux ecosystem, the installation is something that is usually done once and then never again because of the rolling release of packages. Also see the *Beginner's Guide* at `https://wiki.archlinux.org/index.php/Beginners'_Guide` to get a grasp of what Arch Linux is all about. As Arch Linux is an ever moving target, the preferred method of installation is when you are connected to the Internet so you can fetch all the latest and greatest software available. Even before downloading, it is a must to read the latest news at `http://www.archlinux.org/news/` about any new developments.

When we only want a taste of Arch Linux and are probably far from sure that this is what we want for all day use, consider learning how the system works by installing it in a virtual machine. In the *Beginner's Guide* there is a useful section about it at `https://wiki.archlinux.org/index.php/Beginners'_Guide#Install_on_a_virtual_machine`.
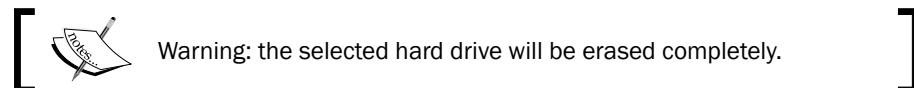
Getting ready

We can get the Archboot ISO from `http://wiki.archlinux.org/index.php/Archboot`. On this page we will find a download link to the latest release. We can select to download the torrent files or the ISO immediately.

The following list describes the main tasks that we will perform in this recipe:

- **Preparing the installation media**: We are going to download the ISO file from the Archboot wiki page and save it on the preferred media of our choice. At the time of writing, there is a choice to download an architecture-specific ISO or a dual ISO that contains both i686 and x86-64 architectures on one disk. I would suggest downloading the dual ISO, which makes installing on any PC and any of these architectures possible without any hassle.

- **Booting the install media and starting installation**: Start your PC using your preferred installation media (CD or USB drive). On most PC systems, you can get into the boot menu by pressing one of the function keys, usually between *F8* and *F12* depending on the motherboard manufacturer. On older machines where you do not yet have a boot menu, you might need to change the boot order in the BIOS where the CD-ROM (or DVD/Blu-ray) has to be chosen as the first device to try booting from.

▶ **Setting keyboard and console font**: When using a different keyboard layout than the default one, you will for sure need this recipe to configure your specific keyboard layout and optionally the console font you like. If you have configured both, the installer will also bring these settings into the configuration of our installed system.

▶ **Setting date and time**: In this recipe we will also configure three small parts—we'll set our time zone, the current time, and the current date.

▶ **Auto preparing hard drive**: The **Auto-Prepare** option will guide you with the creation of a default partition scheme. When we can't have our entire drive erased, or we want to differ from the default partition scheme chosen by the installer, we should skip to the steps for *Manually preparing hard drive*.

Warning: the selected hard drive will be erased completely.
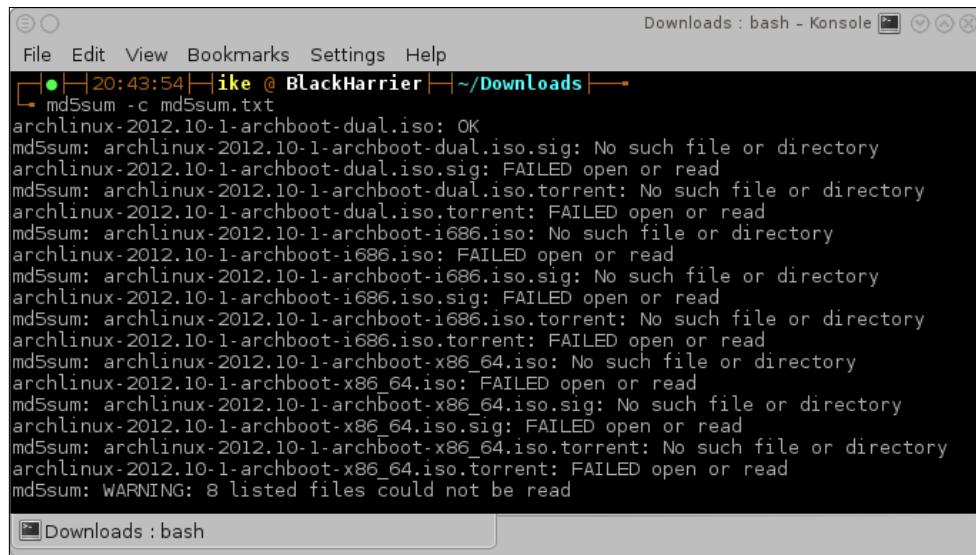
▶ **Manually preparing hard drive**: Manual partition gives you full control over how many partitions you create. This can also concern multiple disks, so there is total freedom. Use the manual preparation when you are used to partitioning.

▶ **Selecting source**: We can select if we want to install from the repositories available on the Internet, or to install immediately from the packages available on the installation media. It would be best to opt for FTP/HTTP as this would ensure we have the latest packages available.

▶ **Selecting packages**: From my point of view the best way to select your packages is by first installing the base system. And when we are done and the system is booting standalone, add packages along the way as we need them. The installer will ask if you want to add the extra repository so you will be able to install everything at once. I would say the safe option here is not to include the extra repository and start with the base system. In general, it will be faster to install applications when you need them. After the installation, we should only check some automatically created configuration files just to make sure the installer created them correctly, and we will get a bootable system when rebooting after installing the boot loader. The more extensive configuration will be explained later on. If you want to change anything here during the installation procedure, I would suggest you skip forward to the *Configuring your system* recipe where the configuration is explained in depth.

▶ **Installing boot loader**: By the end of the recipe we'll have almost everything that we need. However, for our machine to be really usable, we'll have to install the boot loader as this piece of software will enable us to get into Arch Linux when we reboot.

How to do it...

Let's perform the following steps for preparing the installation media:

1. Once we have downloaded the ISO image file, we must verify its integrity by also downloading the `md5sum.txt` file from the `archboot` folder.

2. We will verify the integrity by issuing the `md5sum -c md5sum.txt` command and checking whether our download was successful:

```
                                                    Downloads : bash - Konsole

File  Edit  View  Bookmarks  Settings  Help

      ● | 20:43:54 | ike @ BlackHarrier | ~/Downloads |
      md5sum -c md5sum.txt
archlinux-2012.10-1-archboot-dual.iso: OK
md5sum: archlinux-2012.10-1-archboot-dual.iso.sig: No such file or directory
archlinux-2012.10-1-archboot-dual.iso.sig: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-dual.iso.torrent: No such file or directory
archlinux-2012.10-1-archboot-dual.iso.torrent: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-i686.iso: No such file or directory
archlinux-2012.10-1-archboot-i686.iso: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-i686.iso.sig: No such file or directory
archlinux-2012.10-1-archboot-i686.iso.sig: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-i686.iso.torrent: No such file or directory
archlinux-2012.10-1-archboot-i686.iso.torrent: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-x86_64.iso: No such file or directory
archlinux-2012.10-1-archboot-x86_64.iso: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-x86_64.iso.sig: No such file or directory
archlinux-2012.10-1-archboot-x86_64.iso.sig: FAILED open or read
md5sum: archlinux-2012.10-1-archboot-x86_64.iso.torrent: No such file or directory
archlinux-2012.10-1-archboot-x86_64.iso.torrent: FAILED open or read
md5sum: WARNING: 8 listed files could not be read

 Downloads : bash
```

3. As we can see, all the ISOs were downloaded but the torrent files were not. This leads to some successful checks and others state that the files are not found, but in the end we know the ISO file we downloaded is ok.

4. Now that we are sure our ISO is ok, we can burn this to a CD with our favorite burning program.

The following steps will guide you in booting the install media and starting the installation:

1. Insert the CD in the drive or insert the USB stick in the USB port of your PC.

2. Enter the boot menu or let your computer automatically boot from the inserted installation media.
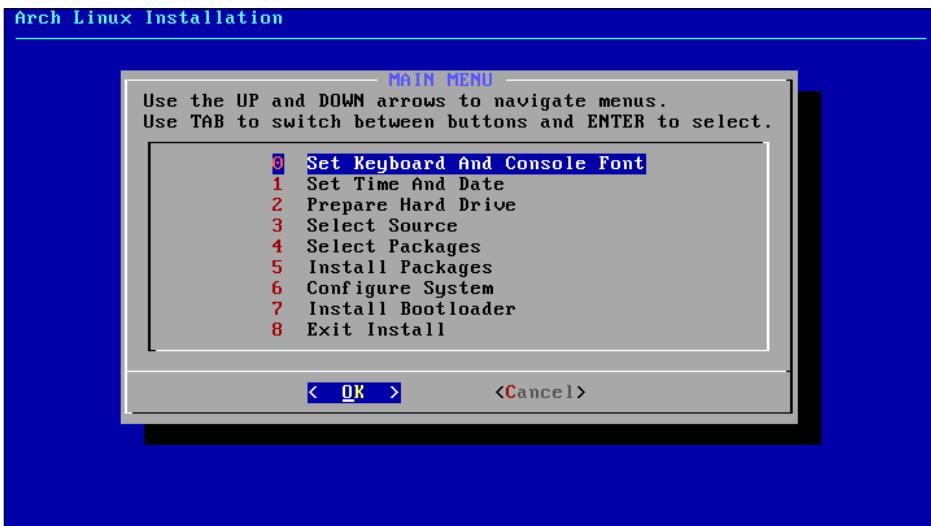
3. If the previous steps went fine, we should see the following screenshot:

4. Select the architecture depending on whether we want to boot the **Long Term Supported** (**LTS**) or the default kernel. The default kernel should be fine, but if we want to run the LTS kernel for stability reasons, we can select it. Press *Enter* and we'll be on our way.

5. When the installation media is completely started, we get some initial information about the Archboot environment. When we press *Enter*, the installation scripts will be started:

6. When ready to go, press *Enter* and we will be presented with the **MAIN MENU** screen. The menu contains several steps that are followed in a chronological order for a fresh installation:

```
Arch Linux Installation

                    MAIN MENU
    Use the UP and DOWN arrows to navigate menus.
    Use TAB to switch between buttons and ENTER to select.

            0  Set Keyboard And Console Font
            1  Set Time And Date
            2  Prepare Hard Drive
            3  Select Source
            4  Select Packages
            5  Install Packages
            6  Configure System
            7  Install Bootloader
            8  Exit Install


            <  OK  >          <Cancel>
```

Let's perform the following steps to set keyboard and console font:

1. We are presented with the keyboard layout choice, where you get a list with all the possible keyboard layouts that Arch Linux supports. Select your keyboard layout and continue.

2. Optionally, you can also select a console font of your liking. If you have no idea or don't want to do this, just go back to the main installer menu and continue with the next step. If you have a preference, then select the **Set Console Font option** and select your font.

Let's set our time zone, the current time, and the current date:
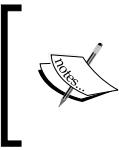
1. Select the time zone. This is something similar to Continent/Capital. For example, Europe/Brussels.

2. Select **UTC** for your hardware clock. This will make sure the daylight saving changes will be applied correctly.

3. Set your current system time. This is actually the time it is now in your time zone.

4. Set your current system date.

The following steps should be performed for auto preparing the hard drive:

1. First of all, you are presented with a choice of using the old MBR partition tables or the new GUID (GPT) partition tables. The choice is entirely yours; however, for older machines the safe choice is MBR. For newer machines and machines with UEFI bootups, it's recommended to use the GPT partition tables.

2. Select the size you want to be used for the boot (`/boot`) partition.

3. Select the size you want to be used for the swap (`/swap`) partition.

4. Select the size you want to be used for the root (`/root`) partition.

5. The remaining disk space will automatically be claimed for your home (`/home`) partition where all the users' data will be residing.

6. Select the filesystem you want to be used for the root partition and for the home partition. The safest choice here is ext4. If you are an adventurous spirit, you could go with Btrfs or NILFS. Do your research before making a choice for using a currently marked experimental filesystem, if it really is the thing you want to use.

7. Choose the naming system you want to be used in the configuration files.

8. When we are completely sure everything is ok, we continue and the installer will prepare the entire drive. If all goes well, the installer will state that the preparation was successful.

The following steps should be performed for manually preparing the hard drive:

1. Select the partition type (MBR or GPT). For extended information see step 1 performed for *Auto preparing hard drive*.

2. Select a disk to partition.

3. The installer presents you with cfdisk or cgdisk depending on whether you have selected to use MBR or GPT partitioning.

4. After the partitions are created, we need to go back to the partitioning menu and use the **Set Filesystem Mountpoints** option.

5. First you need to select what partition you will be using as swap.

6. The installer will then ask you to select the root partition for your system.

7. After these required partitions are selected, you can keep on selecting partitions and set their mount points to the location you want.

> As we're doing a manual partitioning layout, we can apply the hints from the Auto-Prepare section in the manual section. However, by using the manual approach, you gain a lot of extra freedom to create a partition scheme as you please.

The following steps are needed to select your source:

1. Select CD-ROM or FTP/HTTP.
2. Configure your network.
3. Select the mirror you want to use (this only has to be done when FTP/HTTP is selected).

The following steps are needed to select your packages:

1. Don't enable the extra repository.
2. Only select **base**; this contains all the packages needed to get a working system after the installation.
3. Select the packages you want to install.
4. Let the installation script do its job of installing the packages.

Finally, let's install the boot loader:

1. Choose the boot loader you like the best.

   The best known will be GRUB and Syslinux—those are probably also the best supported. There are also other boot loaders available such as LILO.

2. The installer will suggest you check the configuration file it has created. In most cases this configuration will be correct, but you should always check if it is correct. See if the correct devices are used in the configuration file. You can verify the correct devices by checking your filesystem's layout.

3. Install the boot loader.

   Now that you have taken all the steps needed to install a basic working Arch Linux system on your computer, the only thing left in the installation procedure is to reboot your system and remove the installation media. Then you can enjoy the first boot of your freshly installed Arch Linux.

4. Reboot the system.

> For a detailed description of the main tasks performed, refer to the *Getting ready* section of this recipe.

How it works...

We downloaded the Archboot ISO via the torrent files or directly using a web browser, and the `md5sum` command lets us verify the integrity of the downloaded ISO. So now, we can rest assured that the downloaded file is the real one.

Never use an ISO where you have not verified the checksums.

The Archboot ISO contains a fully working operating system so first we boot that. We are automatically logged into the live system on pressing the *Enter* key. Next, the installation scripts are called and we get dropped in the Archboot installation scripts.

The installation scripts will call the correct application to set the keyboard layout and the console font, and the selected date and time settings. The installation scripts will also keep track of the selected values to put them in the appropriate configuration files.

If we have selected the **Auto-Prepare** option, then based on the configuration we have made, the installer will run fdisk and create the partition scheme that we want.

If we have chosen to do a manual partition of the disk, then we can select any layout we want. Once we are satisfied with the partition scheme, the installer will ask some questions (for example: Where to mount? What filesystem?), and uses our answers to mount the partitions we have made to the correct mount points. The installation scripts will also keep track of the answers given so that they can be used later on for the generation of a configuration file.

The installation script will show you a list of packages that pacman has made available for you to install. When you have selected all the packages you want on your system, the installation script will pass these packages with some extra parameters back to pacman, which will do the actual installing.

At the end of the installation, a list of configuration files will be prepared for you. These configuration files are there for you to review. When you are satisfied with the configuration file, save it. The installation script will now use the configuration file to put the selected boot loader into place.

There's more...

When we want to divert from the default settings, we might need some extra knowledge. So let's discuss them one by one.

The ISOs can be used from a USB drive

The ISOs downloaded from Arch Linux are all "hybrid" images, which means you can save them on a USB drive and they will be bootable. Installing from a USB drive is also very simple. Just connect a USB drive to your machine and issue the following command:

```
dd if=archlinux-2012.04-2-archboot-dual.iso of=/dev/sdX bs=1M
```

All information on the USB drive will be overwritten.

We need to make sure that we set the correct input file, our ISO, to the `if` parameter. Also the output file parameter must be the device and not some partition of the device, such as `of=/dev/sdX`. The `X` stands for the letter assigned by the system to your USB drive.

Selecting local time versus UTC

There are some rare cases where the choice for local time on your hardware clock is the best choice. For example, when installing Arch Linux next to Windows XP, which is an operating system not capable of handling a different time on a hardware clock and system clock. More recent versions of the Windows operating system can handle the hardware clock being set to UTC.

## A safe partition size choice for desktop systems
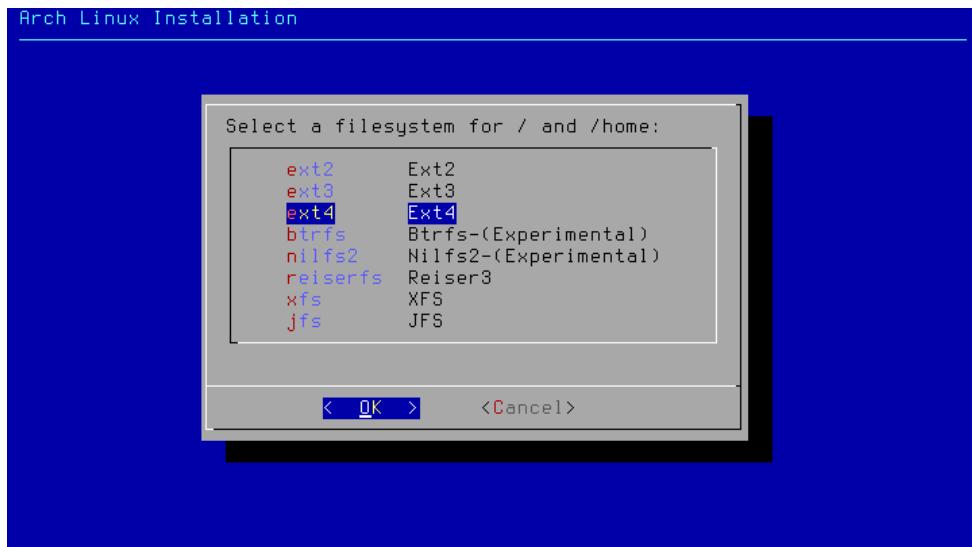
Partitioning is a very wide topic. It can be done in an endless number of combinations. We will now discuss some extra information required to understand the partitioning. Also some nice defaults to have a system up and running quickly.

The following list is a selection for normal desktop use. For people wanting to play a lot of games, these size selections will not fit your needs:

- **Boot partition**: 50 MB
- **Swap partition**:
    - When your RAM is less than 4 GB: RAM + one third of RAM
    - When your RAM is more than 4 GB: Fix it on 4 GB (there is actually no real need to make it bigger)
- **Root partition**: 10 GB (gamers might want to go to 50 GB here)
- **Home partition**: These days you'll end up having 300 to 400 GB and even more

Selecting the desired filesystem

When selecting which filesystem to use for your root and home partitions, you should be well informed about the possibilities of the filesystems. When you have no idea, the best choice is **ext4** as this is the default filesystem these days with modern features, nice speed, and robustness, so you don't lose any data.

```
Arch Linux Installation


        Select a filesystem for / and /home:

            ext2      Ext2
            ext3      Ext3
            ext4      Ext4
            btrfs     Btrfs-(Experimental)
            nilfs2    Nilfs2-(Experimental)
            reiserfs  Reiser3
            xfs       XFS
            jfs       JFS



           <   OK   >      <Cancel>
```

What about the filesystem on boot?

The boot partition will automatically be formatted with the ext2 filesystem. This is the safest choice, as all boot loaders you can find out there will be able to get your system to boot when you have a boot partition formatted as ext2.

Naming schemes for block devices

There are three ways to point to a block device (partition):

  ▸ **UUID scheme**: This is a unique ID that we can use to point to a block device

  ▸ **LABEL scheme**: Here we can point to a partition by using its label

  ▸ **KERNEL scheme**: This is the oldest method used to point to a block device by directly pointing to the device node

Using the UUID scheme might look ugly in your configuration but this is the most certain way you will always point to the correct device. Say you have some hardware changes and the devices are ordered in a new way; with this you will still have the correct block device selected.

The LABEL scheme is looking very elegant and simple but there might be some name collisions as multiple physical disk partitions can have the same name.

The KERNEL scheme is actually the oldest and here we are just pointing to some device node (such as `/dev/sda1`), but this could fail someday after some hardware changes, which could lead to a different order of the device nodes.

```
Arch Linux Installation



            ┌──────────────────────────────────────────┐
            │ Select the device name scheme you want to use │
            │ in config files (recommended is UUID).     │
            │                                            │
            │  FSUUID   /dev/disk/by-uuid/<uuid>         │
            │  FSLABEL  /dev/disk/by-label/<label>       │
            │  KERNEL   /dev/<kernelname>                │
            │                                            │
            ├──────────────────────────────────────────┤
            │      <  OK  >        <Cancel>              │
            └──────────────────────────────────────────┘
```
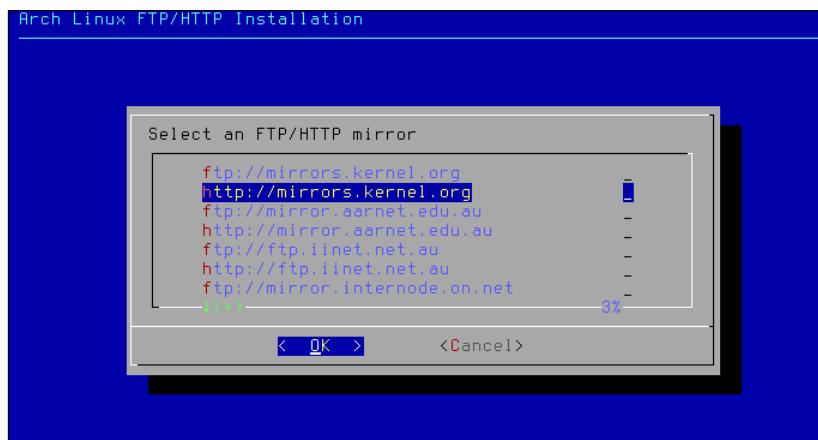
A good partition scheme for desktops

On a desktop system, especially on Arch Linux, I suggest having a separate `/var` partition. Depending on the other goals you have for this partition (for example, running a huge MySQL database, other databases, and so on), the appropriate value would be 5 GB and up. Don't overdo it or you will have a lot of empty space in the `/var` partition. Why so big? Pacman keeps its cache in `/var` and you don't really want your root filesystem being deadlocked by a disk that's filled up with package cache.

## Selecting a mirror geographically close to you
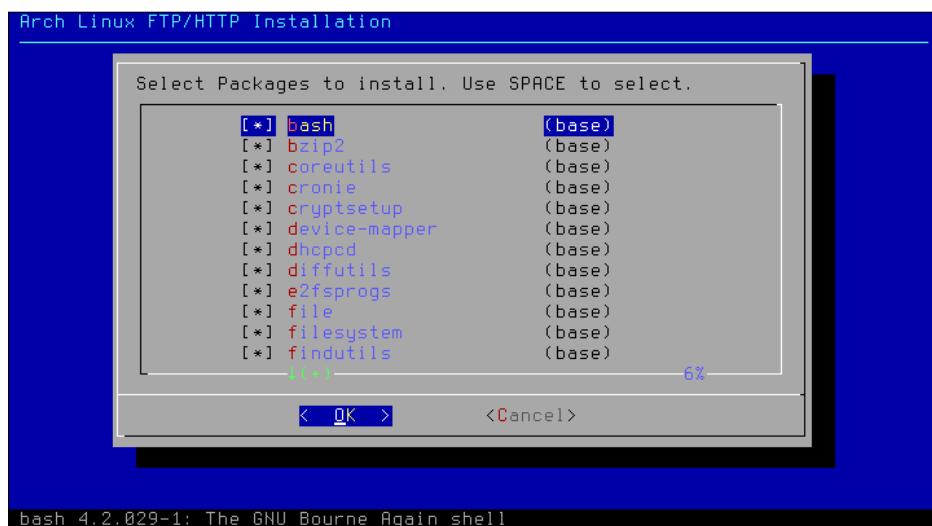
When installing Arch Linux from the Internet, it is best to choose a mirror close to our home for the best download speeds:

```
Arch Linux FTP/HTTP Installation



         ┌────────────────────────────────────────────┐
         │ Select an FTP/HTTP mirror                    │
         │                                              │
         │   ftp://mirrors.kernel.org                   │
         │   http://mirrors.kernel.org                  │
         │   ftp://mirror.aarnet.edu.au                 │
         │   http://mirror.aarnet.edu.au                │
         │   ftp://ftp.iinet.net.au                     │
         │   http://ftp.iinet.net.au                    │
         │   ftp://mirror.internode.on.net              │
         │                                         3%   │
         ├────────────────────────────────────────────┤
         │      <  OK  >        <Cancel>                │
         └────────────────────────────────────────────┘
```

When your network is up and running, you can select a mirror and select one that is as close as possible to your location. We will get the best performance this way. Should you be in doubt what the closest mirror is for you, you can always select a global mirror such as `mirrors.kernel.org`, which will automatically choose a server close to you.

What packages to select

During the installation of Arch Linux, we can select a list of packages to install. I will now share my own preferred way. When this is a first time installation, my personal preference would be to leave all the packages selected from the base group. If you really insist on removing some packages from the base group, go ahead and remove them, but you should really know what you are doing in that case.

Having the extra repository enabled

If this is not the first install of Arch Linux, you can definitely enable the use of the extra repository so that you can select a whole bunch of applications, which you know for sure you want to have installed on your system. For example, you can immediately install Xorg, GNOME, XFce, KDE, and so on. For a first time install, I would go step-by-step and leave the extra as it is for now.

Configuring your system (Should know)

In this recipe we'll explain the configuration files used to identify your system. Usually, these files only need to be configured once and stay the same for the entire life of the system.

Getting ready

The following list describes the main tasks that we will perform in this recipe:

- **Configuring hostname**: A **hostname** is a name we give to a machine that will make it easy to recognize which machine we are talking about. So if we are in a network, we can easily separate the different machines by their hostnames.

- **Configuring console**: We'll also configure the virtual console, what keyboard layout to use, and maybe some special font and mapping for these.

- **Configuring the localization**: In this task we will set up our machine with the correct localization. This can be done very extensively in the `locale.conf` file. In the most common cases, we only set `LANG` and `LC_COLLATE` in this file. If you want to narrow it down, you can get more information by typing `man locale.conf`. However, `LC_COLLATE` is the fallback when everything else fails.

- **Configuring time zone**: Setting your time zone will make sure that your system clock is correct. This has to be used in combination with `/etc/localtime`. Actually, they have to be changed together so for your own safety change the `/etc/localtime` symlink, and then immediately add the new time zone to `/etc/timezone`.

- **Configuring module handling**: Do you want to load extra modules that are not getting loaded by default? In the case of "fancy" hardware or some third-party software such as VMware or VirtualBox, we might need this. You can add a configuration file in `/etc/modules-load.d/` with a list of modules. The modules have to be separated by newlines. When you want to add some comments in those files, you can do that by starting your line with `#` or `;`. The files placed in the directory only need one extra requirement; the name has to end with `*.conf`.

How to do it...

The following step configures the hostname:

1. Edit the hostname with `vim /etc/hostname` to configure it.

Let's list the steps required to configure the console:

1. Edit `/etc/vconsole.conf`.

2. Add the keywords with their values. For example:

   ```
   KEYMAP=us
   FONT=lat9w-16
   FONT_MAP=8859-1_to_uni
   ```

Let's list the steps required to configure the localization:

1. To indicate what locales we want to support, edit `/etc/locale.gen`.

2. When we have changed the `locale.gen` file, run `locale-gen`.

3. To indicate what locale we want to use by default, edit `/etc/locale.conf`:

    ```
    LANG=en_US.UTF-8
    LC_COLLATE=C
    ```

Let's list the steps required to configure the time zone:

1. Create a symlink `/etc/localtime` pointing to your time zone:

    `ln -s /usr/share/zoneinfo/Europe/Brussels /etc/localtime`
    `vim /etc/timezone`

2. Copy the time zone's name into `/etc/timezone`.

Let's list the steps required to configure modules:

1. For loading modules, add a configuration file to `/etc/modules-load.d/`.
2. For blacklisting and passing special parameters to modules, add a configuration file to `/etc/modprobe.d/`.

> For a detailed description of the main tasks performed, refer to the *Getting ready section* of this recipe.

How it works...

On boot, the system will read the content of the `/etc/hostname` file. This content will be used to identify the system to us as the user, and also to identify the machine in a network. The hostname is mostly for user convenience because it's much easier to remember than a big list of numbers.

The terminal will allow you to type on your localized keyboard. The output on your screen will correspond with the button pressed on your keyboard. Also, the terminal will show you the text output in the selected font. Lastly, it will do a translation of text encoding where needed. In the example shown in the steps to configure the console, the output will be translated from ISO-8859-1 to Unicode.

All applications supporting the localization will bring you their output in American English, if `LANG=en_US`. When the application does not have the selected language, it will fall back to the "C language". The C language is the default language in computer systems, so this will also be in English.

The time shown on your system will be the one from your time zone. When you are in a time zone where you have daylight saving changes, your PC will automatically adjust to it.

All files residing in `/etc/modules-load.d/` ending with `*.conf` will be used to load extra modules. All `*.conf` files residing in `/etc/modprobe.d/` will be used to determine if the modules must be blacklisted, or used with some special options.

There's more...

Let's look at some tips and tricks for easier configuration of your desired keymap and console font and also, an easy way to find the available time zones.

KEYMAP

We can get a list of all the available keyboard layout mappings by listing the folder `/usr/share/kbd/keymaps`. From the following screenshot, we can see we have found the Belgian layout so we can add `KEYMAP=be-latin1` to our configuration file:

CONSOLEFONT

We can find a list of all the available fonts for the console by listing `/usr/share/kbd/consolefonts`. When we find the desired font, we can add this to our configuration file. For example, `CONSOLEFONT=Lat2-Terminus16.psfu.gz`.

CONSOLEMAP

To get a list of possible transformations, we can take a list of `/usr/share/kbd/consoletrans`. Optionally, we can add this to our configuration file, but this is not always needed. For example, `CONSOLEMAP=8859-1_to_uni`.

Finding your time zone

Finding your time zone is far from difficult as it is practically always Continent/Capital. We can run `ls` in the `/usr/share/zoneinfo/` folder and take it from there:

```
ls /usr/share/zoneinfo/
```

29

Installing and removing packages (Must know)

In this recipe we will see how package management in Arch Linux revolves around pacman. **Pacman** is just short for package manager, and not a joke that package management is a game. The complete guide to pacman can be found on the wiki at `https://wiki.archlinux.org/index.php/Pacman`. Arch Linux has three official repositories: [core], [extra], and [community]. Both [core] and [extra] are maintained by the Arch Linux Developers, while the [community] repository is maintained by the Trusted Users. There are also a lot of unofficial repositories maintained by the Arch Linux enthusiasts, which install specific types of software not found in the official repositories, thereby making your life much easier. The list of unofficial repositories can be found at `https://wiki.archlinux.org/index.php/Unofficial_User_Repositories`.

Arch Linux is using a complete open model, which means every piece of software provided in the official repositories can be rebuilt by the user. This is what we call the **Arch Build System** or **ABS**. Arch Linux also gives the users a very easy way to share buildscripts for other software not found in the repositories by means of the **Arch User Repository** or **AUR**. And finally, there is also a nice list of commonly used software on the wiki at `https://wiki.archlinux.org/index.php/Common_Applications`. This list might help you find the ultimate application to fit your needs.

Getting ready

The following list describes the main tasks that we will perform in this recipe:

- **Configuring pacman**: Firstly, the default configuration of pacman is done in `/etc/pacman.conf`. Secondly, you also have `/etc/pacman.d/mirrorlist` installed on your system by default, which contains the selected mirror used during the installation. The mirrorlist contains all the official mirrors of Arch Linux. The default configuration that you get after the installation works perfectly fine. You can find all the information related to pacman on the wiki page at `https://wiki.archlinux.org/index.php/Pacman`.
- **Full system upgrade**: This is probably the most used action of all, once you have your system installed with all the software that suits your needs. Often you will only be left doing a full system upgrade from time to time.

▸ **Installing a package from the repositories**: Installing packages with pacman is straightforward. We can pass multiple packages to the command to install more than one package, or we could even pass a groupname to install a whole group of packages.
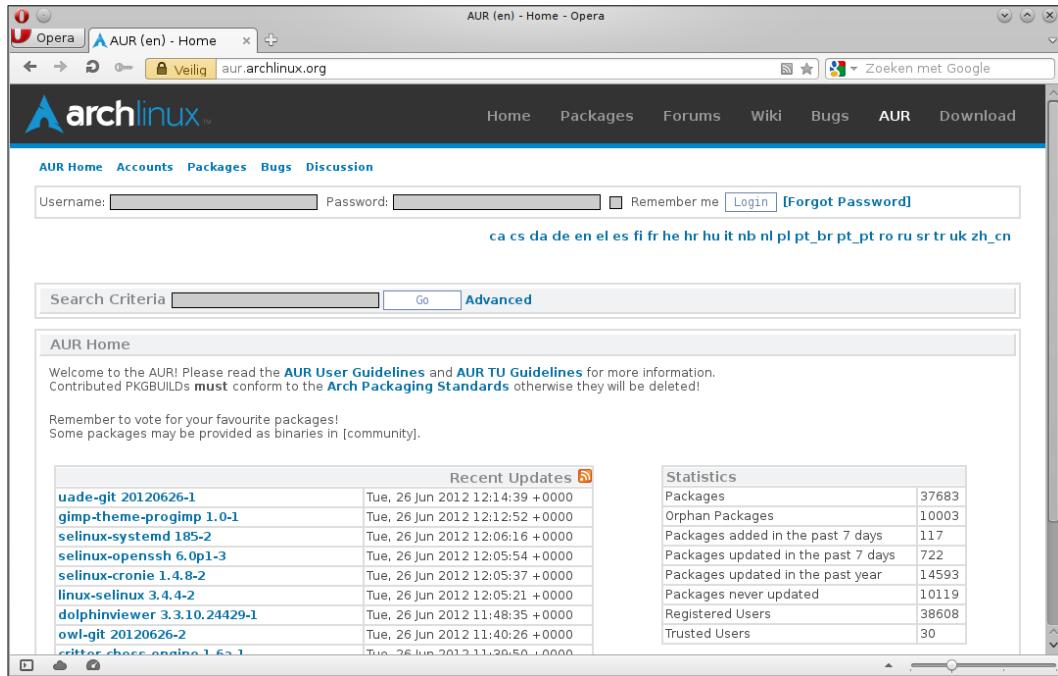
> We can also install a package through an update where we need to pass the package file to pacman in order to have some piece of software installed.

▸ **Searching packages in the repositories**: This will enable us to search if our favorite piece of software is available in the repositories so we can install it hassle free.

▸ **Installing a package from a disk**: This practically behaves in the same way as installing from the repositories with one exception: here we pass a file to the command and not a package name.

▸ **Removing a package**: When we get tired of a package, or we have found some new piece of software that does the same job but suits our needs better, we could remove a certain package or even group.

▸ **Cleaning package cache**: Over time, we don't want our disk being filled up with old package files, so from time to time it is a very good practice to clean the package cache.

▸ **Official and unofficial repositories**: Arch Linux provides us with a lot of packages by default and these can be found in the official repositories. The official repositories contain packages supported by the Arch Linux Developers and Trusted Users.

▸ **Using the Arch Build System**: The **Arch Build System** (**ABS**) is similar to the ports system you find in FreeBSD. All the buildscripts to create the packages found in the official repositories are available through the ABS. This enables you as a user to rebuild every package according to your own wishes, with your own compiler flags, and so on. If you want to fully use the ABS, you will need to install **base-devel** and **abs** with pacman. If you only want to check how things got done, you will only need to install abs.

▶ **Using the Arch User Repository**: The **Arch User Repository** (**AUR**) contains packages not found in the official repositories and are pure user contributed content. So everyone registered on the AUR website `https://aur.archlinux.org` can upload new packages, so other users might benefit from that work. To use the AUR you must install base-devel. For more details about the AUR, I will refer to the wiki at `https://wiki.archlinux.org/index.php/AUR`.

▶ Using makepkg: Makepkg is the tool used for building packages for Arch Linux. I will not go too deep into the usage, but will bring you up to speed to get your packages built from the ABS or AUR. Makepkg assumes that base-devel is installed. Also see `https://wiki.archlinux.org/index.php/Makepkg`. Makepkg has to be called in a directory with at least one file named `PKGBUILD`.

How to do it...

Let's list the steps required to configure pacman:

1. Edit `/etc/pacman.conf` to modify options or add/remove some repositories.

2. Edit `/etc/pacman.d/mirrorlist` to change or add a mirror with the official repositories in it.

3. Run `pacman -Syu` as root for the full system upgrade.

4. To install a package from the repositories, in a terminal run `pacman -S somepackage` as root.

5. As root, run `pacman -Ss somepackage` for searching packages in the repositories.

6. Run `pacman -U somepackage.pkg.tar.xz` as root in a terminal for installing a package from disk.

7. As root, run `pacman -R somepackage` for removing a package.

8. For cleaning the package cache, run `pacman -Sc` as root and answer the questions.

Let's list the steps required to configure official and unofficial repositories:

1. Open `/etc/pacman.conf`.

2. Disable or enable an official repository, or add an unofficial repository.

Let's list the steps required to use the ABS:
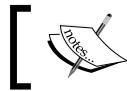
1. Install base-devel and abs:

   pacman -Syu base-devel abs

2. Optionally edit `/etc/abs.conf`.

3. Let abs synchronize the `abs` scripts. You must run abs as root.

Let's list the steps required to use the AUR:

1. Install base-devel:

   pacman -Syu base-devel

2. Search for your favorite application in the AUR.

3. Download the tarball and start building it.

Let's list the steps required to use makepkg:

1. Go to a folder with a buildscript in it (`PKGBUILD`).

2. Run `makepkg`.

3. This should get the package to build. If it is missing dependencies to build the package, you will need to install those first.
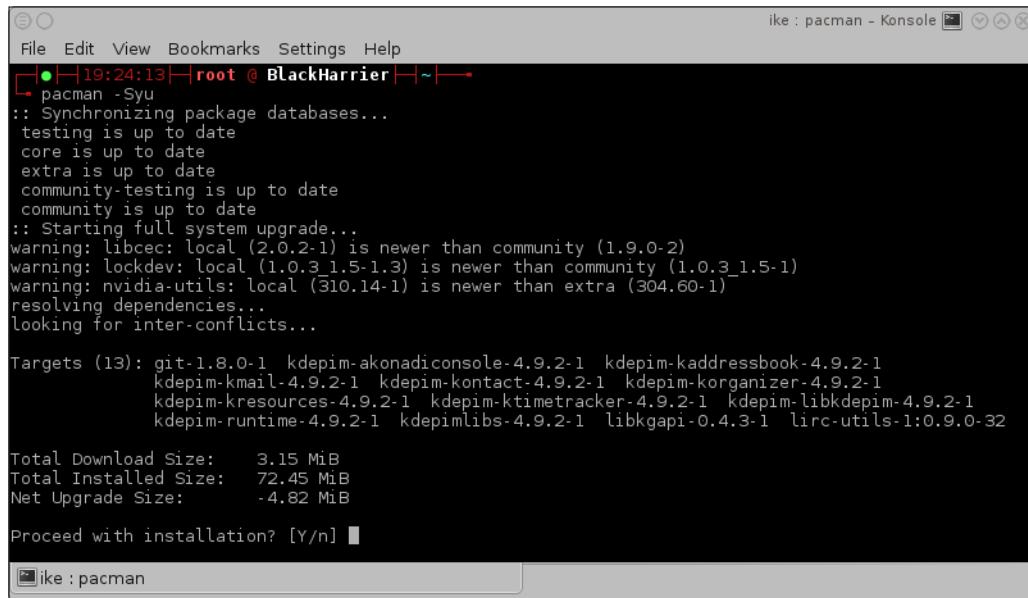
> For a detailed description of the main tasks performed, refer to the *Getting ready* section of this recipe.

How it works...

The options set in `pacman.conf` will determine the behavior of pacman. These vary from ignoring packages or groups, from updates to extra repositories. What we have defined in the mirrorlist will determine where our packages come from (from which server).

The full system upgrade command will synchronize your local package databases with the remote ones, and depending on the packages you have installed it will ask you if you want to continue with the installation of some upgraded packages.

The `pacman -S somepackage` command will look for the package name or group you have passed, and if it exists, it will continue to try and install it.
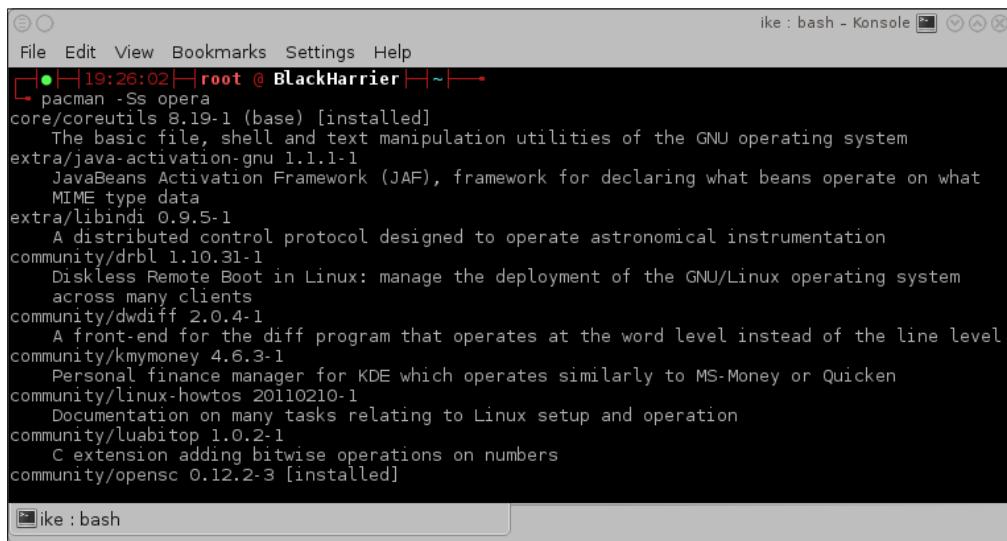
On executing the `pacman -Ss somepackage` command, pacman will search in the locally synced databases if the package we are looking for is available somewhere. If that is the case, we will see what version is available for us to install.

On executing the `pacman -U somepackage.pkg.tar.xz` command, pacman will check the dependencies of the package and try to install these alongside the package you want to install. When all goes well, it will install the package inside the tarball on the correct location.

The package will be removed from the system on executing the `pacman -R somepackage` command. The only thing left of the package will be an entry in the cache.
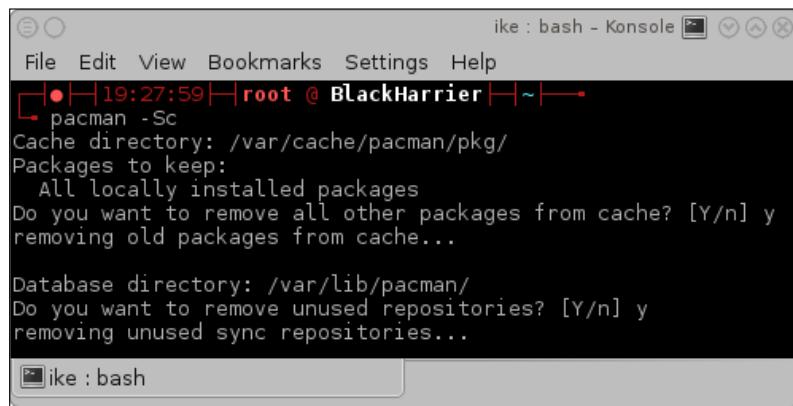
On executing the `pacman -Sc` command, pacman will look for old package files and ask you if you want those to be removed from the filesystem.
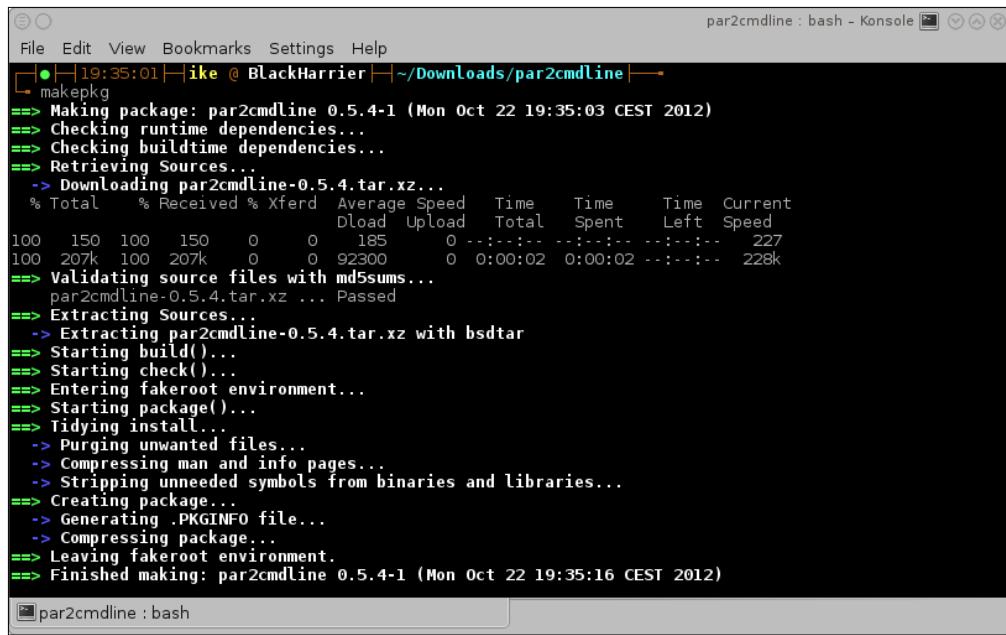
Pacman uses `pacman.conf` for the options and the repositories. All enabled repositories will be used for package installation now. Should we have some configuration mismatches, pacman will notify us.

Abs will use `rsync` to synchronize the buildscripts used in the official repositories to your local computer. This will facilitate you, for example, to build an officially supported package with other options enabled or disabled.

The AUR is actually just a website with a collection of user contributed buildscripts. If your favorite application is already available there, you can benefit from the work that someone else has already done. It is also very easy to contribute improvements via comments.

Makepkg will read the information described in the `PKGBUILD` file to get the package built correctly, and have it in the correct format for pacman to install it on your system.

```
                                             par2cmdline : bash - Konsole
File   Edit   View   Bookmarks   Settings   Help
  ● ┤19:35:01├ike @ BlackHarrier├~/Downloads/par2cmdline├─
  makepkg
==> Making package: par2cmdline 0.5.4-1 (Mon Oct 22 19:35:03 CEST 2012)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> Retrieving Sources...
 -> Downloading par2cmdline-0.5.4.tar.xz...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   150  100   150    0     0    185      0 --:--:-- --:--:-- --:--:--   227
100  207k  100  207k    0     0  92300      0  0:00:02  0:00:02 --:--:--  228k
==> Validating source files with md5sums...
   par2cmdline-0.5.4.tar.xz ... Passed
==> Extracting Sources...
 -> Extracting par2cmdline-0.5.4.tar.xz with bsdtar
==> Starting build()...
==> Starting check()...
==> Entering fakeroot environment...
==> Starting package()...
==> Tidying install...
 -> Purging unwanted files...
 -> Compressing man and info pages...
 -> Stripping unneeded symbols from binaries and libraries...
==> Creating package...
 -> Generating .PKGINFO file...
 -> Compressing package...
==> Leaving fakeroot environment.
==> Finished making: par2cmdline 0.5.4-1 (Mon Oct 22 19:35:16 CEST 2012)

 par2cmdline : bash
```

There's more...

The `pacman.conf` file has sections defined by `[section]`. These sections can have some options defined inside them. There is one special section called `[options]` where the global options can be configured for pacman. The other sections are the repositories defined by default or the user. In regards to the repositories, the order of the declaration is important. The repository nearest to the top of the file will take precedence, in descending order. This is of great importance for repositories that provide packages with the same name. The order is also important so as to understand why `[testing]` must be defined above `[core]`. Generally the `pacman.conf` file is very well documented by means of comments that explain the options provided, but here we'll try to explain them in depth.

The **mirrorlist** is a file with all official Arch Linux mirrors listed. Let's find a mirror closer to home so we can get the best download speeds. We can define more than one server here. Note that this will not give us the packages from the most up-to-date server, but when the first in the list is unreachable then the second can be used, so we are still able to update our system.

The pacman.conf options

The `pacman.conf` options can be found at `https://www.archlinux.org/pacman/pacman.conf.5.html`. By default, everything should be working out of the box. The default should also be sufficient for new users.

Some repository samples

The following is a sample of the `[core]` repository. We can see that signature checking is required for the packages, and we use the mirrorlist for the configured servers.

```
[core]
SigLevel = PackageRequired
Include = /etc/pacman.d/mirrorlist
```

The following sample will use the default `SigLevel` defined in the `[options]` section, and when used, it will first try to use FTP, and when this is not available it will fall back to HTTP:

```
[otherrepository]
Server = ftp://10.0.0.1/$repo/$arch
Server = http://10.0.0.1/$repo/$arch
```

In the following sample, we have a local repository where you can see all possible URLs that can be used to fill in the `Server` option:

```
[somelocalrepository]
Server = file:///home/packages/$repo/$arch
```

More information about official and unofficial repositories

By default, the official repositories are listed in `pacman.conf`. Not all of them are enabled by default. There is also a nice list of unofficial repositories where you can find some very high quality software.

For complete information about the official repositories and in what cases you should enable or disable them, check out `https://wiki.archlinux.org/index.php/The_Arch_Linux_Repositories`.

Due to the fact that it is easy to set up your own repository, there are a lot of users who are building a set of specific packages and providing them as unofficial repositories for everyone. The complete list of unofficial repositories can be found on `https://wiki.archlinux.org/index.php/Unofficial_User_Repositories`.

More information on the ABS

By using the ABS, we as users get a great deal of flexibility over our system. For first time users of ABS, you would want to check the ABS wiki page at `https://wiki.archlinux.org/index.php/ABS`. For a simplified introduction, you can also refer to `https://wiki.archlinux.org/index.php/ABS_FAQ`.

Booting and managing services with systemd (Should know)

Systemd provides us with a more modern approach of booting. It also benefits from the modern multicore processors and relies on very aggressive parallelization to get the job done very quickly. Arch Linux provides systemd by default. Systemd uses "the so-called service files" to define how and when a certain service or "a so-called daemon" has to be started.

Getting ready

The following list describes the main tasks that we will perform in this recipe:

- **Installing systemd**
- **Setting the default target**: The default target can be compared with the runlevel when using initscripts. It is not the same, but flows from the same idea that you want some stuff being started in certain cases. "A so-called runlevel" is called a **target** in the systemd world.
- **Starting a service manually**: Using the `systemctl` command, we can start and stop services on demand.
- **Enabling services during boot time**: Of course systemd can start services during boot time. It is very simple to enable or disable these services.

How to do it...

Let's list the steps required to install systemd:

1. Install both systemd and systemd-arch-units to make the installation work by running `pacman -S systemd systemd-arch-units`.

2. Add `init=/bin/systemd` to your kernel command line by editing `/boot/syslinux/syslinux.cfg`:

   `APPEND initrd=/initramfs-linux.img root=/dev/sda2 rootfstype=ext4 ro `**`init=/bin/systemd`**

Let's list the steps required to set the default target:

1. If you want to end up with a graphical system, enable the graphical target:

   `systemctl enable graphical.target`

2. When a terminal is sufficient, the multiuser target will suffice:

   `systemctl enable multi-user.target`

3. Start a service manually by running the following command:

   `systemctl start service`

4. Enable services during boot time by running the following command:

   `systemctl enable service`

> For a detailed description of the main tasks performed, refer to the *Getting ready* section of this recipe.

How it works...

We need the systemd-arch-units package, because not all packages providing services are providing the service files needed to be able to use them with systemd. Due to the fact that we added `init=/bin/systemd` to the kernel command line, the system will use systemd for startup.

These days only two targets can be set automatically as the default target, as only `graphical.target` and `multi-user.target` provide the default target installation.

In the `systemctl start service` command, `service` is actually the name of a file. For example, NetworkManager has a systemd service file called `NetworkManager.service`, and it is this full name we need to pass to the `systemctl` command. For example:

`systemctl start NetworkManager.service`

To be able to know what services to start during boot time, `systemctl` will create symlinks to the specific service files in the correct locations where systemd will search them during boot.

There's more...

Once we are satisfied with systemd booting our system, we can eventually fine-tune it a little more.

Systemd only initialization

The following command will provide symlinks for initscripts' compatibility:

`pacman -S systemd-sysvcompat`

The result is that you can omit the extra parameter `init=/bin/systemd` on your kernel command line.

We can easily change the desired target by changing the kernel command line in our boot loader configuration. This makes it easy to test if some specific target suits our needs.

Set the target on the kernel command line

Similar to the appending of a number to the kernel command line when using initscripts, we can also do this for systemd using the `systemd.unit` parameter.

For example, considering Syslinux as the boot loader, open `/boot/syslinux/syslinux.cfg` and add `systemd.unit=multi-user.target` to the kernel command line:

```
APPEND initrd=/initramfs-linux.img root=/dev/sda2 rootfstype=ext4
ro systemd.unit=multi-user.target
```

The previous example is valid for a system with only systemd, otherwise we also would require `init=/bin/systemd`.

List all available services

We might want to know what services we have available on our system, and of course we want to know what we can do with those services.

We can list all the used services or choose the list all the available services by running the following commands:

```
systemctl list-units --type=service
systemctl list-units -a --type=service
```

Default actions for services

By default systemd supports actions such as `start`, `stop`, `restart`, `reload`, and `status`. There are more actions available, which can be found by issuing `man systemctl`.

Check if a service will be started at boot

Before enabling a service, we might want to check if the service is not already enabled. We can check if a service is already enabled for startup during the boot process by using the `is-enabled` action:

```
systemctl is-enabled service
```

Disable a service from starting during boot

If we find some service we no longer need, we would want to disable it. Sometimes we no longer want some services being started during the boot process, so we need to disable them:

```
systemctl disable service
```

Booting and managing services using initscripts (Should know)

In this recipe we will learn about initscripts. **Initscripts** is a collection of scripts that make sure your computer starts up fine. It also provides the necessary functions and tools to manage services on your system. Initscripts' services are often referred to as **daemons** (`https://wiki.archlinux.org/index.php/Daemon`). These days, booting using initscripts can still be used in conjunction with systemd, but over time the use of initscripts will be discouraged and phased out. **Systemd** is a daemon that controls the startup of your system and also manages the services running on them. The initscripts are written in **Bash**, so they are easy to read and modify when needed. Extended information about the boot process of Arch Linux can be found at `https://wiki.archlinux.org/index.php/Arch_Boot_Process`.

Getting ready

The following list describes the main tasks that we will perform in this recipe:

- ▶ **Changing runlevel**: The definition of a runlevel is somewhat abstract. The **runlevel** will determine what applications will be started. Arch Linux uses some of the runlevels globally defined, and we can find exactly what they do on `https://wiki.archlinux.org/index.php/Runlevels`. While the system is running, you can change the runlevel on the fly, or maybe you want to do some administrative actions on the machine that require you to change the runlevel.

- ▶ **Setting the default runlevel**: By default the runlevel used is `3`. This is the multiuser runlevel without starting `X`.

- ▶ **Manually starting a service (daemon)**: With initscripts, all the files needed for starting and stopping some services are also just Bash scripts so they can be called directly.

- ▶ **Automatically starting a service (daemon)**: The list of daemons that get started during boot time are configured in the `rc.conf` file within the `DAEMONS` array.

How to do it...

The following step changes the runlevel:

1. Run `telinit runlevel` to change the runlevel, where `runlevel` is a number from `0` to `6`.

Let's list the steps required to set the default runlevel:

1. Edit `/etc/inittab`.

2. Change the line where you find `id:3:initdefault:`. Here you will already see that the default selected runlevel is `3`.

3. Test if your configuration was done correctly by running `telinit q`.

4. To manually start a service (daemon), begin by calling the script directly:

   `/etc/rc.d/somedaemon start`

Let's list the steps required to automatically start a service (daemon):

1. Edit `/etc/rc.conf`.

2. Add a new item to the `DAEMONS` array as you please:

   `DAEMONS=(syslog-ng network crond)`

3. Save the file.

> For a detailed description of the main tasks performed, refer to the *Getting ready* section of this recipe.

How it works...

The `telinit` command will change the runlevel and depending on what number you switch to, some running applications and daemons might get stopped or just get started.

The default selected runlevel set in the `inittab` file determines what scripts will be used during the startup of your PC. This is also the reason why you always need to check if the configuration is correct by running `telinit q`, because if the `inittab` file is corrupted somehow your system will become unbootable.

The Bash script sitting in the `/etc/rc.d/` folder will be executable. It will by default also provide three actions: `start`, `stop`, and `restart`.

During startup, the initscripts will read the `DAEMONS` array defined in the `rc.conf` file, and in the order they are defined all the daemons will be started during boot.

There's more...

Now let's talk about some general information that is relevant to this recipe.

The runlevels

The runlevels are just numbers, but for us humans it is easier to remember sentences. So we do some matching of an action versus the runlevel number.

43

The following list defines each runlevel number:

- ▸ `0`: Poweroff
- ▸ `1`: Single-user mode (rescue mode)
- ▸ `2` and `4`: These are user defined, but as on any other system they are the same as `3` by default
- ▸ `3`: Multi-user mode; users can log in via a terminal or network
- ▸ `5`: Multi-user graphical mode; this is runlevel `3 + X` (some display manager)
- ▸ `6`: Reboot
- ▸ `emergency`: Emergency shell (you will encounter this during boot failures)

Setting the default runlevel in the kernel command line

The modification of the `/etc/inittab` file can lead to an unbootable system. So there are other ways to configure the default runlevel.

We can set the default runlevel we want in the kernel command line configured in our boot loader configuration file. This will allow us to safely switch runlevels.

Lets perform the following step to set the default runlevel via the boot loader (Syslinux in this example):

- ▸ Edit `/boot/syslinux/syslinux.cfg`.

  `APPEND initrd=/initramfs-linux.img root=/dev/sda2 rooftfstype=ext4 ro `**`5`**

  We have set the default runlevel to `5` (graphical mode).

The default actions

Arch Linux provides a helper application that makes it easy to start multiple daemons in one command.

The scripts that facilitate the starting of the daemons usually provide three actions by default:

- ▸ `start`: Starts the daemon
- ▸ `stop`: Stops the daemon
- ▸ `restart`: Restarts the daemon

Another common action is `reload`, which facilitates the running daemon to actually reload its configuration without really stopping.

The rc.d helper

Arch Linux provides a helper that can start and stop multiple services in one command. The helper comes with a nice man page, which you can read by issuing `man rc.d`.

```
rc.d action daemon1 daemon2 …
```

The actions you can provide to `rc.d` are the same that you can pass to the scripts directly. So if a script provides the `reload` action, `rc.d` can use it.

Now, the default way of starting the daemons is sequential. The first has to be started correctly before the next can be started. We can improve our boot time by configuring the `DAEMONS` array a little differently.

Daemons can be started in the background

If we want some service to be started in parallel with the one's following it, you can add `@` in front of the entry in the `DAEMONS` array:

```
DAEMONS=(syslog-ng network @crond)
```

Leave a daemon in the array without being started

We can also leave a service in the `DAEMONS` array but still disable it from automatically being started. For this we need to add `!` in front of the service:

```
DAEMONS=(syslog-ng !network crond)
```

Get a list of the available daemons

We can get a list of all the available services by running `rc.d list`.

Configuring GUI using Xorg (Should know)

In this recipe we will learn how to configure GUI using **Xorg**. When we want to use our system as a desktop system, we will need Xorg one way or the other. These days Xorg is the de facto standard for displaying and using graphical interfaces on Linux-based systems. Also for most single-screen setups, you don't need to configure anything. Multi-screen setups and setups with proprietary drivers are the exceptions to this rule. Some distributions provide you with tools to install the correct video driver; Arch Linux does not. So, eventually we will need to find out which driver to install. In relation to the keyboard, mouse, and a lot of other input devices, Xorg can find them mostly automatically. In some cases the input devices need an additional Xorg driver installed.

Getting ready

The following list describes the main tasks that we will perform in this recipe:

- **Installing Xorg**: We'll install the basic required packages to be able to use the Xorg graphics system.

- **Changing keyboard layout**: For most people changing the keyboard layout is not needed, as most use QWERTY. But for some parts of the world where other layouts are being used, this might come in handy.

- **Installing input drivers**: When we have some special input hardware with us, we might need to install extra input drivers. These days everything should be detected automatically, but sometimes in order to meet the specific requirements of your input hardware we need to install input drivers. For example, laptop users might need to install the Synaptics driver for their touchpad.

- **Installing video drivers**: First we need to find out which graphics we have installed in our system and what will be the best driver to use for them.

- **Using the proprietary NVIDIA drivers**: The proprietary NVIDIA drivers are easy to install on Arch Linux, as they can be found in the official repositories. There is an article in the Arch Linux wiki (`https://wiki.archlinux.org/index.php/NVIDIA`) that covers all the bits and pieces.

- **Using the proprietary AMD drivers**: The reference for using the AMD (ATI) Catalyst drivers with Arch Linux is the following wiki page:

  `https://wiki.archlinux.org/index.php/ATI_Catalyst`

  On this page we find a whole bunch of information needed to get the Catalyst drivers working nicely on your hardware.

> A short while ago, the catalyst and catalyst-utils packages made it into the official repositories. So they become very easy to install.

How to do it...

1. Install xorg-server:

   ```
   pacman -S xorg-server
   ```

Let's change the keyboard layout now:

1. Run the `setxkbmap` command followed by the desired keyboard layout:

   ```
   setxkbmap be
   ```

2. Install the input driver:

   ```
   pacman -S xf86-input-synaptics
   ```

Let's list the steps required to install video drivers:

1. Find the graphics card used in your system.
2. Search if there is a driver available:

   `pacman -Ss xf86-video`
3. Install the driver:

   `pacman -S xf86-video-driver`

Let's list the steps required to use the proprietary NVIDIA drivers:

1. Install nvidia and nvidia-utils:

   `pacman -S nvidia nvidia-utils`
2. Create the configuration file `/etc/X11/xorg.conf.d/20-nvidia.conf`.
3. Restart and see if the driver is being used.

Let's list the steps required to use the proprietary AMD drivers:

1. Install catalyst and catalyst-utils by running the following command:

   `pacman -S catalyst-dkms catalyst-utils`
2. Add the `nomodeset` parameter to the kernel command line to make sure that the open source drivers will not kick in:

   ```
   APPEND initrd=/initramfs-linux.img root=/dev/sda3
   rootfstype=btrfs ro vga=773 nomodeset
   ```
3. Add a default configuration file `/etc/X11/xorg.conf.d/20-catalyst.conf` so that Xorg knows it has to use the proprietary driver.

   ```
   Section "Device"
     Identifier  "aticard"
     Driver      "fglrx"
   EndSection
   ```

> For a detailed description of the main tasks performed, refer to the *Getting ready* section of this recipe.

How it works...

Pacman will download and install the xorg-server package and its minimally needed dependencies.

The `setxkbmap` command will change the selected keyboard layout. For example, we had a US layout by default, and after running `setxkbmap be` it will be the Belgian AZERTY layout.

47

The specific drivers get installed to your system by pacman, which will provide Xorg with the means to understand the input from a touchpad and handle it correctly.

The NVIDIA driver should be automatically detected by Xorg, but to be sure you can add a file to `/etc/X11/xorg.conf.d/`, such as `/etc/X11/xorg.conf.d/20-nvidia.conf`:

```
Section "Device"
  Identifier  "NVIDIAcard"
  Driver      "nvidia"
EndSection
```

You could when needed have a default configuration created by issuing the following command:

```
nvidia-xconfig
```

When having two connected screens, you can also autogenerate a twinview default configuration:

```
nvidia-xconfig –twinview
```

When we are using the AMD proprietary drivers, setting the `nomodeset` parameter makes sure that the built-in kernel drivers don't start to conflict with the proprietary driver. The Xorg configuration file will make sure that Xorg will start without errors. Although not entirely necessary, we make sure that Xorg uses the driver and serves us well.

There's more...

We can opt to use our graphical environment directly, which will imply the installation of xorg-xinit, or we could use our Xorg environment with a display manager. A **display manager** is a graphical login screen so we can use Xorg all the way.

Directly using Xorg

In order to start Xorg directly from the terminal, we first need to install the xorg-xinit package to facilitate this:

```
pacman -S xorg-xinit
```

Then the `startx` command will get us into the X server. When using this directly without configuration, X will not start as we will have some missing applications requested by the default configuration. So let's go forward and install:

```
pacman -S xorg-twm xorg-xclock xterm
```

Using Xorg with a window manager or a desktop environment

For our own convenience it's best to install a **window manager** (**WM**) or a **desktop environment**. For a complete list of window managers go to `https://wiki.archlinux.org/index.php/Window_manager`. And for a full list of desktop environments go to `https://wiki.archlinux.org/index.php/Desktop_Environment`.

In this section we'll install Xfce (`https://wiki.archlinux.org/index.php/Xfce`):

```
pacman -S xfce
```

Pacman will ask if we want to install all the packages from the Xfce group. We say yes as this is most convenient.

As Xfce is installed now, we can issue `startxfce4` from a started Xorg session or modify our `~/.xinitrc`. In the `.xinitrc` file we can uncomment the line `# exec startxfce4` and save the file. When logged in, we can now issue `startx` and enjoy the Xfce desktop environment.

Using Xorg with a display manager

For desktop users this is the common usage as they need a graphical login screen and continue from there to the desired desktop environment or window manager. There are several display managers available in Arch Linux. In this section we'll only describe **LXDM**, which is a fairly simple display manager. For a list of other display managers available for Arch Linux you can read `https://wiki.archlinux.org/index.php/Display_Manager`.

To install LXDM we issue pacman with the following command:

```
pacman -S lxdm
```

When using initscripts, we can add LXDM at the end of our `DAEMONS` array. When we are using systemd, we can issue `systemctl enable lxdm.service` to enable the startup of LXDM on boot. When we have rebooted, the LXDM display manager shows up and we can select, for example, **Xfce Session** from the Desktop Session drop-down list.

Setting keyboard layout in the configuration files

As we want our keyboard and mouse automatically configured every time our computer starts, we can set the keyboard layout in a configuration file, so we don't have to run `setxkbmap` every time we come into Xorg. We already have xf86-input-evdev installed, so we can put our keyboard settings in the same file `10-evdev.conf`, such as `/etc/X11/xorg.conf.d/10-evdev.conf` (only the keyboard section):

```
Section "InputClass"
    Identifier "evdev keyboard catchall"
    MatchIsKeyboard "on"
    MatchDevicePath "/dev/input/event*"
```

```
        Driver "evdev"
        Option "XkbModel" "pc105"
        Option "XkbLayout" "be"
EndSection
```

The highlighted parts are added and do the following:

- `XkbModel`: What keyboard model are we using? In our example, `pc105`.

- `XkbLayout`: What keyboard layout are we using? In our example, `be` (Belgian AZERTY).

Finding the keyboard options

We can find all the possible models, layouts, and options in `/usr/share/X11/xkb/rules/evdev.lst`.

Finding what graphics card is used on your system

By default we have limited support for our graphics card. To enhance our Xorg experience, we must find what graphics card we have, and what drivers to install for them. With matching drivers for our hardware, we will get far better performance than before.

The easiest way to find which drivers to install is by using the application **lspci**:

```
lspci | grep VGA
```

Example output:

```
01:05.0 VGA compatible controller: Advanced Micro Devices [AMD] nee
ATI RS880M [Mobility Radeon HD 4200 Series]
```

Now we already know that we have an ATI card. In some cases we will need the driver used by the kernel to determine what Xorg driver to use exactly. In this example we already know it will be xf86-video-ati.

Installing the graphics driver
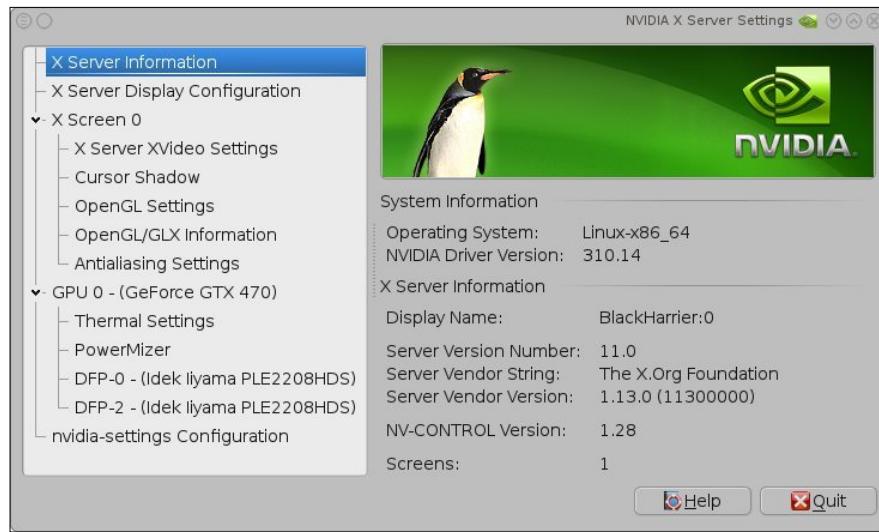
Now we can install the driver we need for our specific system. In this example it will be xf86-video-ati, but it can be any of the drivers available (replace `ati` with what you need in the following example).

```
pacman -S xf86-video-ati
```
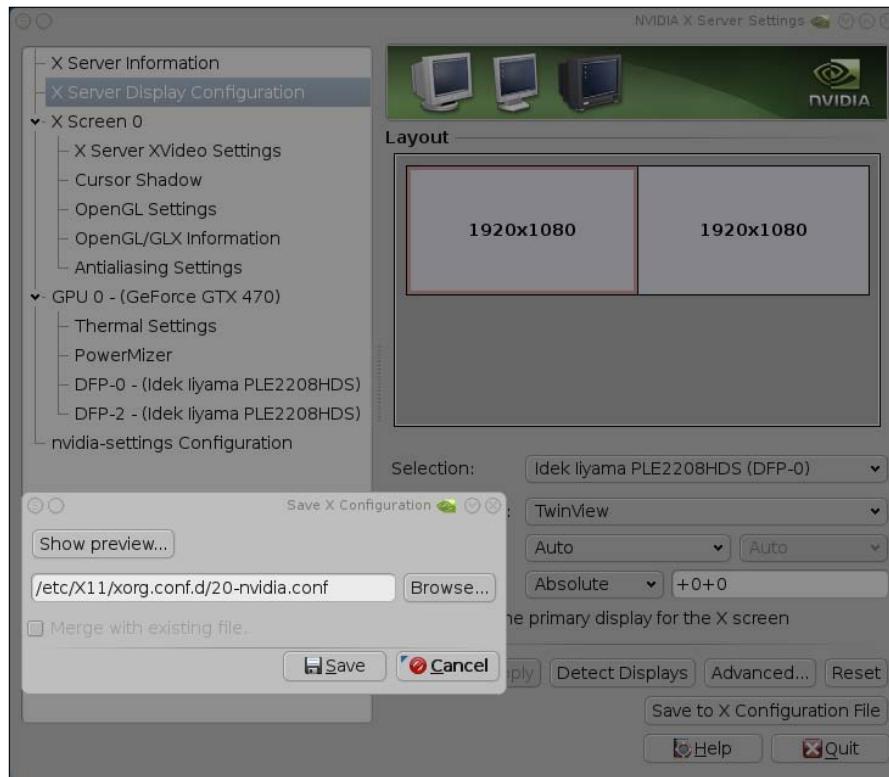
NVIDIA GUI configuration

NVIDIA does not really want us to struggle with the configuration of our graphics card. So they have provided us with a nice GUI to create a fine-tuned configuration.

The simplest way to configure your NVIDIA infrastructure is by configuring everything with **nvidia-settings**. You can issue this application as root so that you can write the global configuration file.

Using the GUI you also can store a configuration file. Do so by entering `/etc/X11/xorg.conf.d/20-nvidia.conf`:

AMD also provides us with some helpful tools to make the configuration very easy. Let's discuss them now.

## Autogenerate an Xorg configuration file with AMD
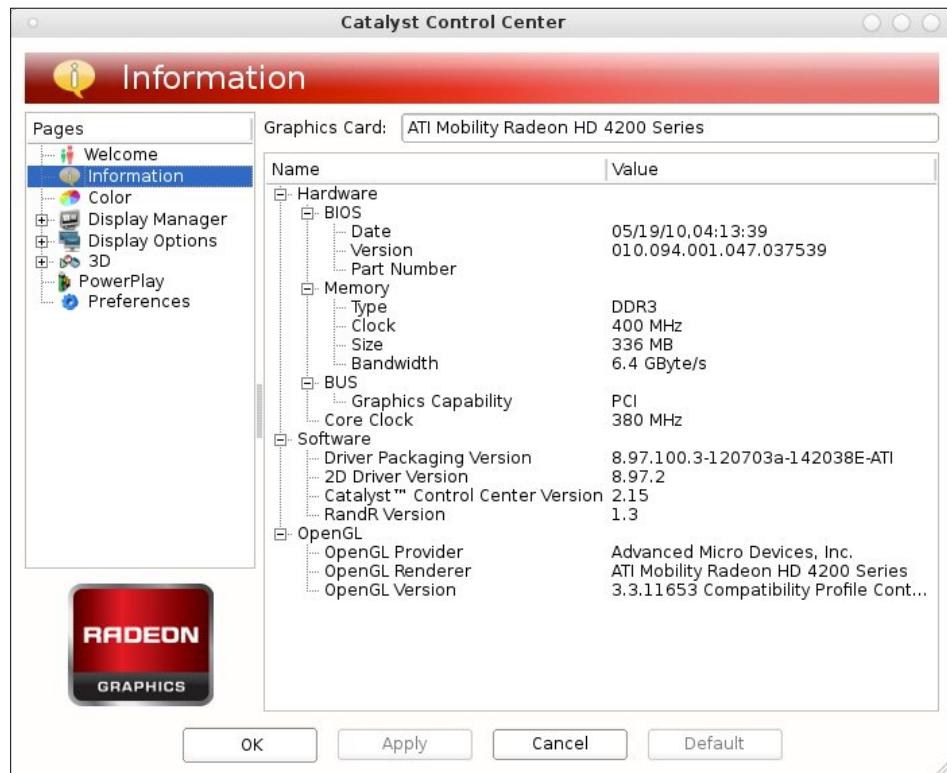
If you want to do some extended configuration, you can start a new configuration file by running:

```
aticonfig --initial
```

This will create a new configuration file, `/etc/X11/xorg.conf`.

AMD GUI configuration

You can further fine-tune the working of your ATI hardware by running the AMDCCCLE application.

[PACKT] open source *
PUBLISHING   community experience distilled

Thank you for buying
Arch Linux Environment Setup How-to

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.
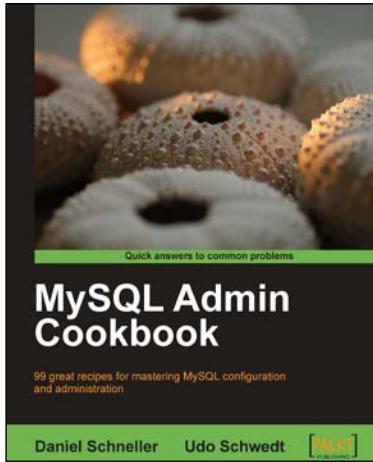
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.
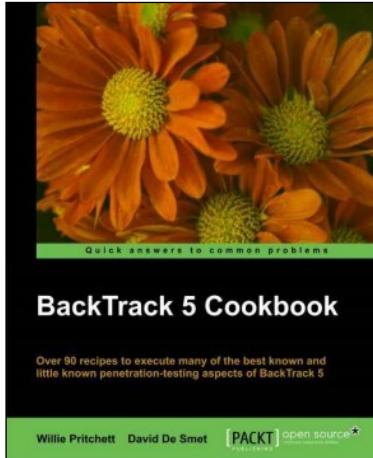
MySQL Admin Cookbook

ISBN: 978-1-84719-796-2   Paperback: 376 pages

99 great recipes for mastering MySQL configuration and administration

1. Set up MySQL to perform administrative tasks such as efficiently managing data and database schema, improving the performance of MySQL servers, and managing user credentials

2. Deal with typical performance bottlenecks and lock-contention problems

3. Restrict access sensibly and regain access to your database in case of loss of administrative user credentials
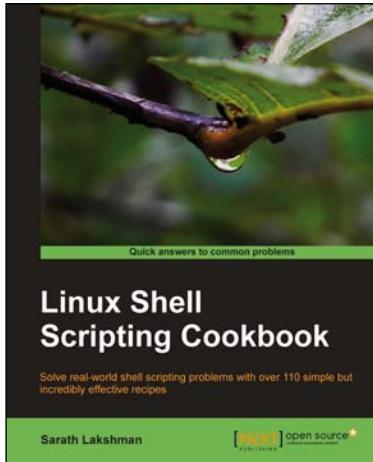
BackTrack 5 Cookbook

ISBN: 978-1-84951-738-6   Paperback: 236 pages

Over 90 recipes to execute many of the best known and little known penetration-testing aspects of BackTrack 5

1. Learn to perform penetration tests with BackTrack 5

2. Nearly 100 recipes designed to teach penetration testing principles and build knowledge of BackTrack 5 Tools

3. Provides detailed step-by-step instructions on the usage of many of BackTrack's popular and not-so-popular tools

Please check **www.PacktPub.com** for information on our titles
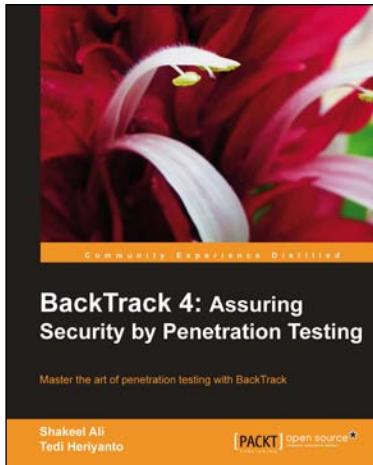
Linux Shell Scripting Cookbook

ISBN: 978-1-84951-376-0   Paperback: 360 pages

Solve real-world shell scripting problems with over 110 simple but incredibly effective recipes

1. Master the art of crafting one-liner command sequence to perform tasks such as text processing, digging data from files, and lot more

2. Practical problem solving techniques adherent to the latest Linux platform

3. Packed with easy-to-follow examples to exercise all the features of the Linux shell scripting language

BackTrack 4: Assuring Security by Penetration Testing

ISBN: 978-1-84951-394-4   Paperback: 392 pages

Master the art of penetration testing with BackTrack

1. Learn the black-art of penetration testing with in-depth coverage of BackTrack Linux distribution

2. Explore the insights and importance of testing your corporate network systems before hackers strike it

3. Understand the practical spectrum of security tools by their exemplary usage, configuration, and benefits

Please check **www.PacktPub.com** for information on our titles