Quick answers to common problems

Raspberry Pi Networking Cookbook

An epic collection of practical and engaging recipes for the Raspberry Pi!

Rick Golden

[PACKT] PUBLISHING

Raspberry Pi Networking Cookbook

An epic collection of practical and engaging recipes for the Raspberry Pi!

Rick Golden

[PACKT] PUBLISHING

BIRMINGHAM - MUMBAI

Raspberry Pi Networking Cookbook

First published: March 2013

Production Reference: 1270213

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84969-460-5

www.packtpub.com

Cover Image by Faiz Fattohi (faizfattohi@gmail.com)

Credits

Author
Rick Golden

Reviewers
Hector Cuesta-Arvizu
Shea Silverman

Acquisition Editor
Erol Staveley

Lead Technical Editor
Mayur Hule

Technical Editors
Sharvari Baet
Devdutt Kulkarni

Project Coordinator
Anurag Banerjee

Proofreader
Jonathan Todd

Indexer
Monica Ajmera Mehta

Production Coordinator
Shantanu Zagade

Cover Work
Shantanu Zagade

About the Author

Rick Golden sat in the computer lab at SUNY Fredonia and completed his first CAI tutorial for programming in APL. It was the summer of 1972; he was nine years old.

Most of the programming that he has done since then has been in Algol-based languages such as PL/I, FORTRAN, BASIC, Pascal, C, C++, C#, Objective C, and Java. He did occasionally write code in languages such as APL, FORTH, LISP, and Scheme; however, he could not find an employer that would actually pay him to develop solutions using those non-structured languages. In recent years he has had more success introducing organizations to scripting languages such as Python, Perl, TCL, Ruby, Groovy, and Node.js.

He also had the privilege to work in many different domains applying leading technologies through each cutting-edge wave of structured programming, architectural frameworks, and design patterns. He has championed distributed computing, scripting languages, SOA, browser applications, CMS, ESBs, web services, nosql and map-reduce, top-down structured approach, UML, use cases, XP - extreme programming, iterative development, and agile development. And, he is still moving forward.

Now, as he approaches his 40th year as a programmer, software architect, and product manager—a career that has spanned eighty percent of his life. He greatly enjoys guiding and coaching the next generation of programmers and software architects—awakening others to the same joy and passion for computing that he has had for the past 40 years.

I'd like to thank my family for giving me the space to complete this book. They have always been supportive and remain my biggest fans.

I'd also like to thank my colleagues Corny, David, Darren, and Pete who have always been available for advice and snippets of code when I needed them; Greg, John, and Steve who were long ago my interns but still remain sources of inspiration; and Ingo who is now and will remain always my muse.

And, most importantly, I'd like to thank my father, George H. Golden Jr., who sat me down in front of a teletype when I was eight years old and showed me how to play Hunt The Wumpus. Not only did my dad introduce me to computers and computer programming, he also introduced me to the Raspberry Pi. Without his encouragement, I could not have written this book.

About the Reviewers

Hector Cuesta-Arvizu provides consulting services for software engineering and data analysis with over 8 years of experience in a variety of industries including financial services, social networking, e-learning, and Human Resources. He is a Raspberry Pi enthusiast.

Hector holds a Bachelor's Degree in Informatics and a Master's Degree in Computer Science. His main research interests lie in Machine Learning, High Performance Computing, Simulation, and Visualization. He has published 12 Scientific Papers in International Conferences and Journals.

You can follow him on Twitter at `https://twitter.com/hmCuesta`.

Shea Silverman has been using computers since he was two years old. He has always been drawn to technology, video games, education, and the public sector. He is currently a member of the Orlando hackerspace FamiLAB, an alumni of the University of Central Florida, and is working towards his Masters in Nonprofit Management.

I would like to thank my family and friends for their ongoing support in my endeavors. I would also like to thank Liz, Eben, and the Raspberry Pi Foundation for the creation of the Raspberry Pi, and the wonderful community that has flourished since its release.

---

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.

Table of Contents

Preface

Back in 2006, Eben Upton and his colleagues at the University of Cambridge's Computer Laboratory noticed a disturbing trend—interviewees for degree course placement did not know enough about what a computer is or how it worked. So, he set out to design an inexpensive computer that would inspire kids to experiment with computers at home—a similar to the hobbyist computers, such as the Apple II, Amiga, and Commodore 64 computers of a generation before. On February 29, 2012, the first batch of 10,000 Raspberry Pis sold out within a few minutes, crashing the websites of the stores selling them. By the end of 2012 more than 500,000 Raspberry Pis have been sold and not just to school children.

The Raspberry Pi credit-card-sized single-board computer costs about $35 and has as much computing power as the early Xbox—more than enough power for playing games, running a home media center, a file server, a website, a small database, or a wireless access point. Its Broadcom **System on a chip** (**SoC**) architecture includes a powerful **graphics processing unit** (**GPU**), and the single-board design includes a network port, an HDMI connection, two USB ports, an SD card slot, and 512 MB of memory. There is more power and there are more features available on this small, inexpensive computer today than there were on the expensive desktop computers that ran the original Windows operating system.

This book contains recipes that take advantage of the power and features of the Raspberry Pi to create a number of practical solutions that can be realized without programming—solutions that anyone with minimal computer skills can apply in their home or office. This book is not about educating or inspiring children to learn computer programming. This book is for parents, hobbyists, and computer geeks who would like to learn more about the Raspberry Pi's "official" Raspbian Linux operating system and the advanced networking solutions that are available for the Raspberry Pi today.

What this book covers

Chapter 1, *Installation and Setup*, introduces the Raspberry Pi, explains how to download and install fresh images of popular Raspberry Pi distributions, and describes how to set up for the initial boot.

Chapter 2, *Administration*, contains a collection of recipes for the Raspberry Pi that cover the basic administration of the Raspberry Pi including how to access the Raspberry Pi remotely using Secure Shell.

Chapter 3, *Maintenance*, has recipes that are for the basic maintenance of the Raspberry Pi including installing and updating new software and accessing the built-in documentation.

Chapter 4, *File Sharing*, has recipes that are for sharing files with other computers on the same local network including automounting disks and installing a file server.

Chapter 5, *Advanced Networking*, has recipes that are for advanced networking solutions including setting up a webserver, a wiki, and a wireless access point.

What you need for this book

For the recipes in this book you will need the following:

- A Raspberry Pi
- A 5V power supply (the Raspberry Pi does not usually come with one)
- A keyboard
- A mouse
- A display (a TV or a monitor)
- A handful of SD cards
- USB devices (such as an external disk)
- A network connection

Some recipes require:

- An external USB disk
- A USB wireless network adapter

You may also want to purchase a case and a powered USB hub. They will help you protect your Raspberry Pi. The case protects the Raspberry Pi from the elements, and the powered USB hub protects it from the power drain that results from connecting too many devices.

The recipes in the first three chapters do not require any additional devices other than the Raspberry Pi and a network connection. The recipes in *Chapter 4*, *File Sharing*, show how to connect a USB disk to the Raspberry Pi. And in *Chapter 5, Advanced Networking*, the final recipe shows how to use a USB wireless network adapter to turn the Raspberry Pi into a wireless access point.

For most of the recipes in this book, you will just need the Raspberry Pi, a power supply, and a network connection. After completing the recipes in *Chapter 2, Administration*, the Raspberry Pi can be accessed remotely and does not require a display, keyboard, or mouse.

Who this book is for

This book is for those who would like to use the Raspberry Pi for more than just an educational tool or an experimenter's toy.

The book is also intended to turn the beginning Raspberry Pi user into an accomplished Linux administrator. Even an advanced Linux user will find the recipes in this book useful as a reference for creating advanced networking solutions with the Raspberry Pi.

The recipes in this book begin simply leading the reader through the installation and basic administration of the Raspberry Pi. As the book progresses, the solutions become more advanced, building on the knowledge gained from previous recipes. The final chapter contains a number of advanced networking solutions.

Although inexpensive, the Raspberry Pi has enough power for a number of practical solutions both at home and at the office. This book is for those who would like to use the Raspberry Pi in practical solutions, not just as an educational toy.

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: "This recipe shows how to update the Raspberry Pi using the `apt-get` command."

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

```
$ sudo apt-get install pianobar
```

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: "Clicking the **Next** button moves you to the next screen."

Warnings or important notes appear in a box like this.

Tips and tricks appear 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 via 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/submit-errata`, 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 on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

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.

1
Installation and Setup

In this chapter we will cover:

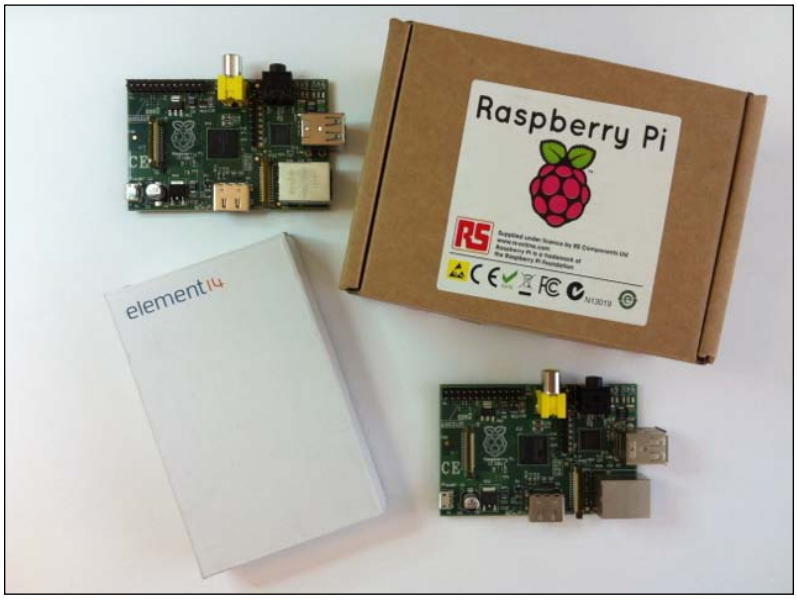
- Preparing for the initial boot
- Printing a case – the Punnet
- Setting up new SD cards
- Image writer for Windows cards (Win32DiskImager)
- Convert and copy for Linux (dd)
- Creating SD cards with BerryBoot
- Booting the "official" Raspbian Linux distribution
- Shutting down the Raspberry Pi (shutdown)

Introduction

This chapter introduces the Raspberry Pi and explains how to download and install fresh images of popular Raspberry Pi distributions and how to set them up for the initial boot.

Developed by Raspberry Pi Foundation in the U.K. for promoting the teaching of basic computer science in schools, the Raspberry Pi is a single-board computer about the size of a credit card. The Raspberry Pi is based on a Broadcom BCM2835 **System on a Chip** (**SoC**) that includes a 700 MHz ARM1176JZF-S processor. The Raspberry Pi is well designed for experimenting with computers and learning computer programming. Its eight **General-Purpose Input/Output** (**GPIO**), I2C bus, and SPI bus also make it an ideal choice for experimenting with computer hardware and peripheral devices.

The Raspberry Pi also has two USB ports, an onboard Ethernet network connector (RJ45), both HDMI and Composite RCA video outputs, and audio output via a 3.5 mm jack or over HDMI—the same type of high-speed connections that are found on most desktop computers and laptops. With these standard peripheral connections the Raspberry Pi has the potential to be much more than just an educational tool or an experimenter's toy.



The preceding image shows two Raspberry Pis with their original packaging.

The Raspberry Pi on the left-hand side comes from RS Components Ltd (`www.rs-online.com`). The one on the right-hand side comes from Premier Farnell/element 14 (`www.element14.com`). Both were shipped in electrostatic bags and contained only simple instructions on how to begin experimenting with the Raspberry Pi.

Although the Raspberry Pi was designed for experimenting, this book is about using the Raspberry Pi in practical networking solutions both at home and at the office. This chapter begins by listing the components you will need, in addition to the Raspberry Pi, for practical application of the solutions described in this book. The first recipe explains how to create a simple yet sturdy case for the Raspberry Pi, out of paper. The remaining recipes describe how to download, install, and configure a number of common Raspberry Pi-optimized operating systems.

Once you've completed this chapter, you will have created a simple case to protect your Raspberry Pi; you will have downloaded, installed, and configured an operating system for your Raspberry Pi; and you will have booted your Raspberry Pi for the first time. You will also understand how to create application cartridges for your Raspberry Pi.

Preparing for the initial boot

This recipe explains which components are needed in addition to the Raspberry Pi before it can be powered on for the first time; that is, the components needed for the initial boot.

The Raspberry Pi is shipped without a case and without a power supply. There is no keyboard. There is no monitor. Depending on how you intend to use the Raspberry Pi, you will need additional components. As a minimum, you will need a power supply, an SD card, and a network cable.

An HDMI cable (or composite video cable), a USB keyboard, and a USB mouse are needed if you'd like to use the Raspberry Pi as you would use a desktop computer. You may wish to attach additional peripherals depending on how you intend to use the Raspberry Pi. This recipe suggests a number of different hardware combinations.

After completing this recipe you will be ready for the initial boot of your Raspberry Pi.

The preceding image shows a Raspberry Pi without a case, an SD card, and a power supply.

Getting Ready

The following are the basic components required for this recipe:

- Raspberry Pi
- Class 4 SD card of 4 GB (or greater)
- 5V micro USB power supply

The Raspberry Pi draws its power from a 5V micro USB power supply and needs an SD card for its operating system. A single 4-GB SD card has more than enough room for hosting the operating system, as well as many useful applications. While no further components are required to boot the Raspberry Pi, the networking solutions in this book will require additional components.

The preceding image shows a Raspberry Pi with an SD card, a network, and power cables.

The essential networking required for this recipe is a network connection. For the simplest networking solutions, the only additional component that the Raspberry Pi needs is a network connection. Once the operating system on the SD card has been configured, remote logins to the Raspberry Pi are possible.

The media center required is a HDMI television or monitor. For the simplest media solutions, in addition to the basic components, the only additional component that the Raspberry Pi needs is an HDMI connection. Both audio and video can be streamed through the Raspberry Pi's HDMI connection. There is enough room on a 4 GB SD card to store a small collection of music and video files, in addition to the operating system.

The preceding image shows a Raspberry Pi in the Punnet with monitor, keyboard, and mouse.

The interactive whiteboard requires the following components:

- HDMI television or monitor
- Bluetooth adapter
- Bluetooth keyboard
- Bluetooth mouse

The Raspberry Pi has two USB ports, with enough power to support low-power devices such as a USB Bluetooth adapter, a simple keyboard, or a mouse.

The network hub requires the following components:

- Powered USB hub
- USB LAN adapter
- USB WLAN adapter
- USB hard drive
- USB printer

When using the Raspberry Pi as a firewall or wireless access point, an additional LAN or WLAN adapter is required. If the adapter is powered by the USB connection, an additional powered USB connector will be required for the adapter to operate reliably.

The teleconferencing center requires the following components:

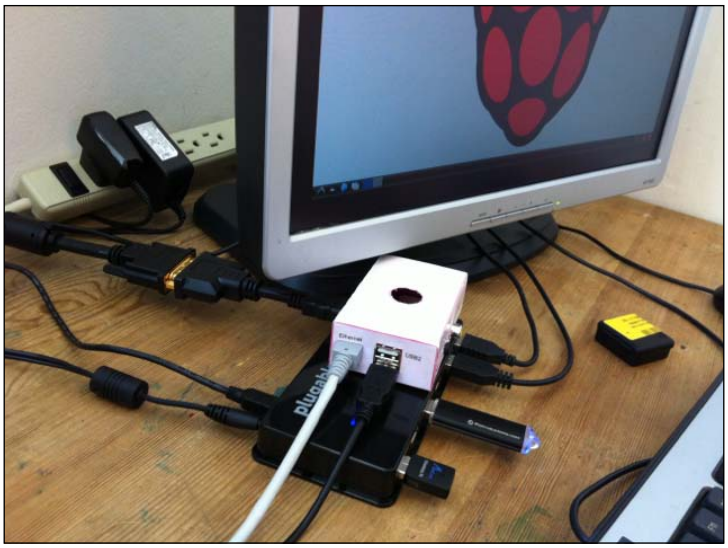
- Powered USB hub
- USB keyboard
- USB mouse
- USB camera

USB devices that require power over the USB connection, such as multimedia keyboards, gamer mice, cameras, printers, or external hard drives (including thumb drives), should be attached indirectly via a powered USB hub instead of directly attaching them to either of the Raspberry Pi's two USB ports. For greater reliability, the USB ports on the Raspberry Pi should typically be connected to powered USB hubs instead of directly connecting them to USB devices.

Gaming requires the following components:

- Powered USB hub
- USB game controllers

The Raspberry Pi is an excellent gaming platform whether for creating games, for playing single-player console games, or for playing multiplayer network games. Many of the older text-based games can be played on the Raspberry Pi with just a keyboard or via a remote login. However, USB game controllers can also be connected to the Raspberry Pi to further enrich the gameplay of multimedia action games.

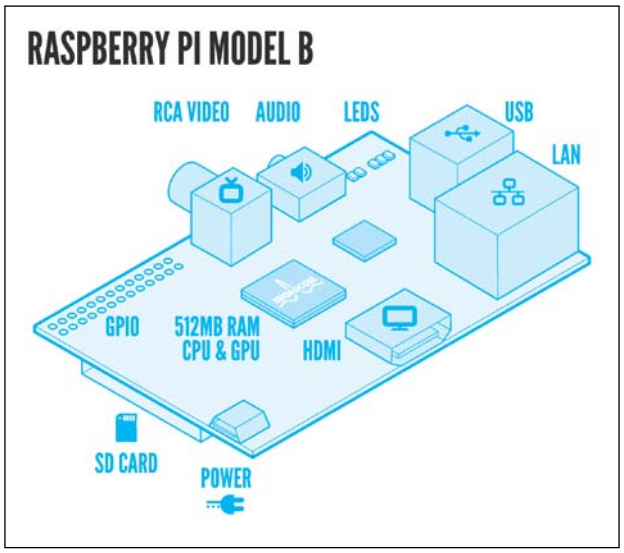


The preceding image shows a Raspberry Pi in the Punnet lying on a USB hub with devices attached.

The initial setup requires the following:

- Raspberry Pi
- Class 4 SD card of 4 GB (or greater)
- 5V micro USB power supply
- Powered USB hub
- Network cable
- HDMI or DVI monitor
- HDMI to DVI adapter (optional)
- Speakers
- Keyboard
- Mouse

A power supply, preformatted SD card, monitor, keyboard, and mouse are the bare minimum components needed for an initial setup. A DVI monitor can be attached to the Raspberry Pi using an HDMI-to-DVI adapter. Both the HDMI-to-DVI adapter and the speakers can be replaced with a single HDMI cable connected directly to an HDMI television. When connected with HDMI, the television will output audio as well as video. The yellow RCA connector also provides video output for older televisions.



The preceding image is a schematic diagram of the Raspberry Pi from Raspberry Pi Foundation (`http://www.raspberrypi.org`).

How to do it...

The following are the steps for booting the Raspberry Pi:

1. Build a case (optional).
2. Download the latest disk image.
3. Write the disk image to an SD card.
4. Insert the formatted SD card into the Raspberry Pi.
5. Attach a display to either the video connector or to the HDMI connector.
6. Attach a USB board and a USB mouse to the USB ports.
7. Attach a 5V micro USB power supply to the Raspberry Pi, and make sure it boots.
8. Finally, shut down the Raspberry Pi.

How it works...

The Raspberry Pi does not come with a case. While it is not necessary for experimenting, you'll probably want a case for protecting the Raspberry Pi. The next recipe in this chapter shows how to make a case from paper (see the *Printing a case – the Punnet* recipe given next).

Before you can boot the Raspberry Pi, you'll need an SD card with a bootable disk image on it. The "official" Raspbian Linux image for the Raspberry Pi is downloadable from `http://www.raspberrypi.org/downloads`.

Once the disk image has been downloaded, it needs to be written to an SD card (see the *Setting up new SD cards* recipe given later).

After the SD card has been prepared and inserted into the Raspberry Pi, the display, keyboard, and mouse can be connected to the Raspberry Pi; and it is ready for booting (see the *Booting the "official" Raspbian Linux distribution* recipe given later).

When it is time to turn the Raspberry Pi off, the operating system must first be shut down—the opposite of booting (see the *Shutting down the Raspberry Pi (shutdown)* recipe given later).

There's more...

The Raspberry Pi is a very low-cost, single-board computer ($35 for the current model). It is sold "bare bones" and requires a power supply, a preformatted SD card to hold its operating system, and a keyboard and a display before it can do anything useful. However, it does have a number of standard I/O connections that will enable it to connect to a large variety of devices.

The following is the list of interfaces:

- **Power (5V at 700mA)**: The Raspberry Pi has a micro USB power connector that should be connected directly to a power supply, neither to the USB port on a computer nor to a USB hub.
- **Preformatted SD card (class 4)**: The Raspberry Pi is designed to boot from a preformatted SD Card (4 GB or greater is recommended).
- **GPIO**: Analog and digital I/O connection for expansion and experimenting.
- **RCA video (composite video)**: The Raspberry Pi can be used with older televisions that have a composite video input.
- **Audio output (3.5 mm jack and stereo)**: The Raspberry Pi does not have an audio input connector; however, a USB mic or sound card can be added.
- **LEDs**: These are disk, power, and network traffic indicators. When these LEDs are flashing, the Raspberry Pi is actively processing.

- **USB 2.0 (two ports)**: There is limited power available on these ports. Devices connected to the Raspberry Pi via USB should have their own power supply or they should be connected indirectly via a powered USB hub.
- **Network (10/100 wired Ethernet RJ45)**: The onboard networking competes for bandwidth with the attached USB devices.
- **HDMI (1080p30)**: This may be used for both video and audio output. It cannot be used at the same time as the RCA video.

The following is the list of onboard components:

- System on chip
  - Broadcom BCM2835 media processor
  - CPU core – ARMv6 architecture; ARM11 core at 700MHz
  - GPU core – 24 GFLOPS of compute power
  - Memory – 512 MB SDRAM stacked on media processor
- LAN9512
  - 10/100 Mb Ethernet (Auto-MDIX)
  - 2x USB 2.0

The recommended accessories include a powered USB hub as it has the following advantages:

- It has its own power supply separate from the Raspberry Pi's power supply
- It has enough power to support the attached devices

However, the Raspberry Pi has some power supply problems. It is difficult to say how much power is actually needed by the Raspberry Pi, as it varies depending on how busy it is and which peripherals are connected. However, there have been problems reported that seem related to an inadequate supply of power. These problems are reduced or eliminated when the power supply for the Raspberry Pi produces at least 700mA at 5V and when USB devices are connected indirectly through a powered USB hub.

The following are the symptoms:

- Unreliable network connection
- Keyboard does not work after the GUI (X Window) is started
- Intermittent SD card errors

The following are the causes:

- Power supply is rated less than 700mA
- Complex keyboard or keyboard with built-in USB hub (for example, Apple Macintosh keyboards)
- A USB hard disk (or thumb drive) is attached directly to the Raspberry Pi instead of indirectly through a powered USB hub

The following are the solutions:

- Use a good quality regulated power supply of at least 700mA at 5V
- Only connect simple USB devices directly to the Raspberry Pi
- Connect USB devices to a powered USB hub, and only connect the hub directly to the Raspberry Pi

See also

- **Raspberry Pi**

  `http://en.wikipedia.org/wiki/Raspberry_pi`

  A detailed Wikipedia article about the Raspberry Pi.

- **DesignSpark – Raspberry Pi**

  `http://www.designspark.com/theme/raspberrypi.`

- **element14 Raspberry PI Group**

  `http://www.element14.com/community/groups/raspberry-pi`

  element14 is one of two distributors for the Raspberry Pi. The Raspberry Pi group on the element14 website has over 7,000 members, over 600 topics in the forums, technical documentations, and video tutorials.

- **The MagPi**

  `http://www.themagpi.com`

  The MagPi is a magazine for Raspberry Pi users. Monthly issues are available online.

- **The Raspberry Pi website**

  `http://www.raspberrypi.org`

  The official Raspberry Pi website contains history, news, and documentation for the Raspberry Pi as well as a quick start guide, a forum, a wiki, and a download area.

- **RPi Hub – eLinux.org**

  `http://elinux.org/R-Pi_Hub`

  The RPi Hub is the Embedded Linux community's wiki page for Raspberry Pi users. This wiki page has a buying guide, a beginners' guide, a list of verified peripherals, and a list of Raspberry Pi distributions larger than what is found on the official website. It is a wealth of well-organized, up-to-date information.

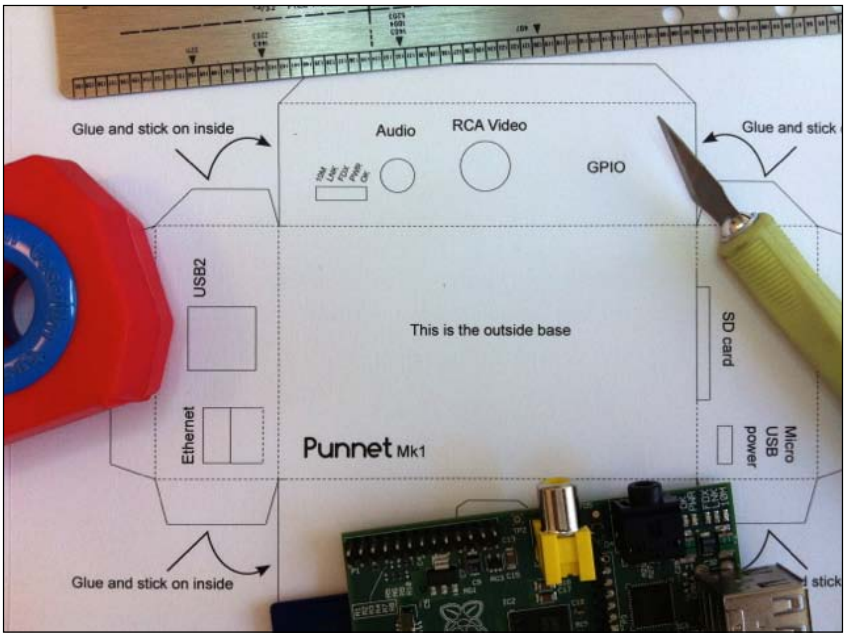
Printing a case – the Punnet

This recipe explains how to make a simple case out of paper.

The Raspberry Pi is a single-board computer without a case. Cases are available from a number of retailers (see the *See also* section of this recipe for a few suggestions). However, the Raspberry Pi is normally sold without a case.

For general experimentation and setup, the Raspberry Pi does not really need a case. It will function perfectly well sitting on top of the box that it came in, or on top of a powered USB hub. Although for regular use, as part of one of the solutions in this book, a case is recommended.

Once you finish with this recipe, you will have a simple protective case for your Raspberry Pi.



The preceding image shows the Punnet printed and ready to use.

Getting ready

The Punnet is a printable card case for the Raspberry Pi. It requires less than an hour to assemble; however, a couple of hours should be allowed for the glue to set firmly before use.

A printable PDF template of the Punnet can be found on the Raspberry Pi Foundation website (`http://www.raspberrypi.org/archives/1310`).

You will need the following components:

- A Raspberry Pi
- The Punnet PDF file (see the link given in the *See also* section)
- Heavy card stock paper
- White glue
- Cellophane tape
- A straightedge or ruler
- A hobby knife

How to do it...

The following are the steps for creating a Raspberry Pi case out of paper:

1. Print the PDF file on heavy card stock paper, or print the file on copy paper and then glue the copy paper to the heavier paper.
2. Carefully cut out the Punnet. Use the hobby knife with the straightedge as a guide for cutting straight lines.
3. Use the straightedge to score and crease the folds. This will make assembly easier.

4. Fold the Punnet around the Raspberry Pi and glue it, as shown in the previous screenshot. Apply the glue sparingly—too much glue causes the paper to warp.

The preceding image shows the Punnet in use. A printed Punnet was glued to pink card stock before folding. The card stock added strength and stability to the case.

How it works...

The Punnet is a simple paper box made from a single sheet of paper with cutouts marked that align with the Raspberry Pi's connectors. The 50-mm scale printed on the page can be used to validate the dimensions of the printout. Turn off any **Scale to fit** feature of the printer to ensure that the page is printed at 100 percent. Once the page is printed, the scale printed on the page can be measured to see if it is 50-mm long. If it is not 50-mm long, adjust the scale in the printer properties and try printing the page again.

The PDF page format is designed for A4 paper (used in Europe); however, it should print fine on 8½ x 11" paper (used in the USA) if the page is printed with actual size. The printed side of the page will become the outside of the Punnet.

There's more...

It is easier to assemble the Punnet around a Raspberry Pi than trying and putting a Raspberry Pi in a pre-assembled Punnet. First score the dotted lines so that it is easier to fold them, then set the Raspberry Pi on the Punnet. Fold up the tabs and spread a small amount of glue on the outside of each tab. Finally, fold up the sides around the Raspberry Pi and glue them together. Use cellophane tape to strengthen the corners.

A straightedge or ruler is a useful tool for cutting and folding on straight lines; and a hobby knife works better than scissors. White glue works well, if used sparingly.

Decorating the Punnet is easier before it is assembled, while it is still uncut.

Don't forget to cut a hole in the top to let cool air in. The Raspberry Pi can overheat if it is kept fully enclosed.

A number of commercial cases are available for the Raspberry Pi (see the links give next).

The preceding image shows three different Raspberry Pi cases.

See also

- **The Punnet**

  `http://www.raspberrypi.org/archives/1310`

  This is the original article featuring a home printable cardboard case for the Raspberry Pi.

- **Raspberry box**

  `http://www.amazon.com/shops/ATLOHWI71UDEX`

  This transparent acrylic box lets the inner beauty of the Raspberry Pi shine through. From Spain, the retailer also makes classical guitars.

- **Pi Holder**

  `http://www.piholder.com`

  The Pi Holder is a solid aluminium case with cooling pillars in the top half that attach directly to the three heat-emitting IC chips on the Raspberry Pi. This case keeps the Raspberry Pi cool and protected—a good choice for production use.

- **Other cases on the Raspberry Pi website**

  `http://www.raspberrypi.org/archives/tag/cases`

  This article features a number of very creative cases for the Raspberry Pi. Some of the cases on this site are very original.

- **Embedded Linux – RPi cases**

  `http://www.elinux.org/RPi_Cases`

  The Embedded Linux community maintains a long list of Raspberry Pi cases. The list features blueprints, photographs, and links to order forms.

Setting up new SD cards

The following recipes explain how to create bootable SD cards from downloaded disk images using `Win32DiskImager.exe`, `dd`, and BerryBoot.

The Raspberry Pi does not come with an operating system. Before the Raspberry Pi can boot, it needs an SD card with the operating system installed. Pre-installed SD cards are available for purchase; however, downloading and installing an operating system's image is not difficult.

How to do it...

The following are the steps on how to write an image to the SD card:

1. Download the "official" Raspbian Linux image from `http://www.raspberrypi.org/downloads`.
2. Write the image to an SD card.

How it works...

The "official" Raspbian Linux operating system image can be downloaded from the Raspberry Pi Foundation website (`http://www.raspberrypi.org/downloads`). Other operating system images can be found on the Embedded Linux community's wiki page (`http://elinux.org/RPi_Distributions`).

Once the operating system is downloaded, you'll need to write it to an SD card.

If you are writing the SD card from a Windows computer, use `Win32DiskImager.exe` (see the *Image Writer for Windows cards (Win32DiskImager)* recipe).

If you are using the Linux operating system or Mac OS to write the image to the SD card, use the `dd` command-line utility (see the *Convert and copy for Linux (dd)* recipe).

If you'd like to try a simpler way of installing the Raspberry Pi that works on any computer that can copy files to a formatted SD card, use BerryBoot (see the *Creating SD cards with BerryBoot* recipe).

See also

- **The Raspberry Pi website – downloads**

  `http://www.raspberrypi.org/downloads`

  This is the location of the "official" versions of Raspberry Pi optimized GNU Linux distributions. Currently, Raspberry Pi Foundation recommends four distributions of the Linux operating system, as follows:

  - Raspbian "Wheezy"
  - Soft-float Debian "Wheezy"
  - Arch Linux ARM
  - RISC OS

The Soft-float Debian distribution is a slower distribution that has not been optimized to use the Raspberry Pi's hardware-accelerated floating-point calculations. It has been made available for use with software that has not been optimized yet (such as Oracle's JVM). The other distributions are listed next with links to their original maintainers' websites.

- **Raspbian**

  `http://www.raspbian.org`

  Raspbian is a Linux operating system distribution based on Debian optimized for the Raspberry Pi and comes with more than 35,000 packages pre-compiled for the Raspberry Pi.

- **archlinux | ARM – Raspberry Pi**

  `http://archlinuxarm.org/platforms/armv6/raspberry-pi`

  Arch Linux ARM is a simple, lightweight distribution for shaping your own system. This distribution is built directly from source code and is optimized for the Raspberry Pi. It offers rolling releases of bleeding edge software; however, it is a minimalist base rather a complete desktop system.

- **RISC OS**

  `http://www.riscosopen.org/`

  The RISC OS was designed in Cambridge, originating from the same team that designed the ARM processor, and was originally released in 1987.

- **Embedded Linux – Rpi Distributions**

  `http://elinux.org/RPi_Distributions`

  The Embedded Linux community maintains an excellent wiki page on Raspberry Pi operating system distributions. The wiki page has a comparison table and links to downloadable image files. Many of these distributions are specialized for specific use – as a home theater, as firewall, as a development platform, or as a very inexpensive desktop PC.

- **GNU**

  `http://www.gnu.org`

  The homepage of the GNU operating system.

Image writer for Windows cards (Win32DiskImager)

This recipe explains how to install a Raspberry Pi operating system image on an SD card using the open source image writer for Windows – `Win32DiskImager.exe`.

Once you finish with this recipe, you will be able to write Raspberry Pi images to SD cards from a Windows computer.

Getting ready

The following are the ingredients:

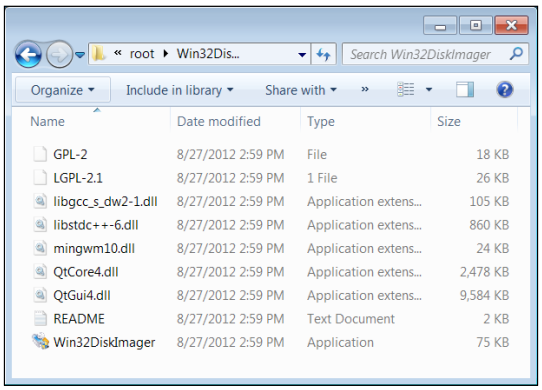
- A computer running Windows with an SD card writer
- A class 4 SD card of 4 GB (or greater)
- A Raspberry Pi operating system's image file
- A pre-compiled `Win32DiskImager` binary

The pre-compiled binary of `Win32DiskImager` is distributed as a ZIP file and can be downloaded from `https://launchpad.net/win32-image-writer`.

How to do it...

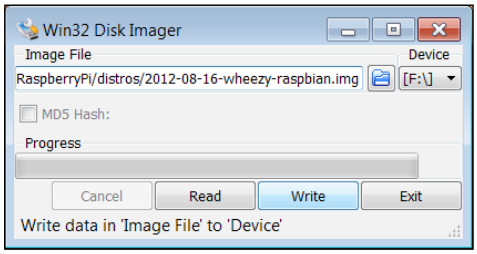
The following are the steps for writing a disk image to an SD card on a Windows computer:

1. Download the `Win32DiskImager` ZIP file (`https://launchpad.net/win32-image-writer`).
2. Expand the ZIP file to a folder on disk (for example, `C:\Win32DiskImager`).



The preceding screenshot shows the contents of the `Win32DiskImager` ZIP file.

3. Download a Raspberry Pi distribution disk image (`http://www.raspberrypi.org/downloads`).
4. Run `Win32DiskImager.exe` from the install folder.
5. Select the source image file and the target device, as shown in the following screenshot:



The preceding screenshot shows **Win32 Disk Imager** being used to write the recommended Raspbian GNU Linux distribution to an SD card.

6. Click on the **Write** button to copy the image to the SD card.

   Writing an image to a disk takes about 5 minutes for a 2-GB image file. Once the image is written to the SD card, the SD card may be ejected and used to boot the Raspberry Pi.

How it works...

First, the image writer for Windows (`Win32DiskImager`) is downloaded and installed.

`Win32DiskImager` is a standalone application. Its install files can be expanded to a folder located anywhere on the PC. Double-click on the **Win32DiskImager** executable file to start the application.

Then, a Raspberry Pi disk image is downloaded.

Finally, **Win32 Disk Imager** is used to write the Raspberry Pi disk image to the SD card.

There's more...

The utility, `Win32DiskImager`, was originally written to read and write disk images for a specific Linux distribution; since then, however, it has been generalized and is now a popular tool for many development projects such as the Raspberry Pi.

`Win32DiskImager` is also an excellent backup tool! After booting and configuring the Raspberry Pi, a backup can be made to preserve the image in case the SD card is damaged or lost.

To make a backup perform the following steps:

1. Run `Win32DiskImager.exe` from the install folder.
2. Select the SD card device as the source and the image file as the target.
3. Click on the **Read** button to read the SD card in the form of an image on disk.

A new backup should be created after each update to the Raspberry Pi's operating system, application software, or configuration.

See also

- **Image writer for Windows**

  `https://launchpad.net/win32-image-writer`

  This utility that was originally written to read and write disk images for a specific Linux distribution; however, since then it has been generalized and is now a popular tool for many development projects such as the Raspberry Pi.

Convert and copy for Linux (dd)

This recipe explains how to install an operating system's image on an SD card using the standard Linux utility, `dd`.

Most versions of Linux (and Mac OS) have the `dd` command installed. This powerful version of the copy command (`cp`) can be used to write blocks of data to devices such as an SD card.

Once you finish with this recipe, you will be able to write an SD card from a Linux (or Mac OS) computer.

Getting ready

The following are the ingredients:

- A computer running Linux with an SD card writer
- A class 4 SD card of 4 GB (or greater)
- A Raspberry Pi operating system's image file

The `dd` utility is normally installed by default with most Linux distributions. If it is not installed, use the appropriate Linux installation utility to install it.

All the commands in this example are executed as a privileged user (`root`).

How to do it...

The following are the steps for writing a disk image to an SD card on a Linux computer:

1. Download a Raspberry Pi distribution disk image (`http://www.raspberrypi.org/downloads`).
2. Excute the `df` command. Determine the name of the SD drive.
3. Execute the following command:

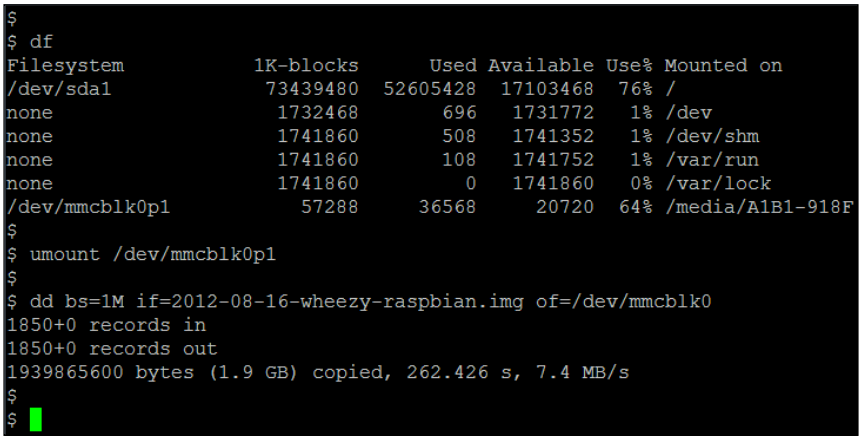
   ```
   umount /dev/mmcblk0p1
   ```

   Unmount the mounted disk partitions.

4. Execute the following command:

   ```
   dd bs=1M if=2012-08-16-wheezy-rasbian.img of=/dev/mmcblk0
   ```

   Use `dd` to copy the image to SD card, as shown in the following screenshot:



The preceding screenshot shows the `df` command being used to discover the name of the SD card drive, the `umount` command being used to unmount the SD card, and the `dd` command being used to write the recommended distribution to an SD card.

How it works...

First a Raspberry Pi disk image is downloaded.

Then, the name of the SD drive is discovered using the `df` command. The `df` command shows how much disk is free on each of the mounted disk drives. The SD card was just inserted, and its primary partition (`p1`) appears in this list as `/dev/mmcblk0p1`. The disk device is `/dev/mmcblk0`.

Finally, the `dd` command is used to write the Raspberry Pi disk image to the SD card. The following is the explanation for the command: `dd bs=1M if=2012-08-16-wheezy-rasbian.img of=/dev/mmcblk0`:

- Each disk block written is 1 MB (`bs=1M`)
- The input file (`if`) is `2012-08-16-wheezy-raspbian.img`
- The output file (`of`) is the SD card disk device (`/dev/mmcblk0`)

There's more...

The utility `dd` is one of the core GNU utilities found in most Linux distributions. It is a low-level utility that simply copies blocks of data from one file to another.

The previous screenshot shows an example of how the `df` command can be used to determine the name of the SD drive. The first partition of the SD disk, `/dev/mmcblk0p1`, is mounted at `/media/A1B1-918F`. Disk images cover a whole disk, not just one partition, so the correct name of the disk drive in the previous example is `/dev/mmcblk0` (notice that `p1` is missing).

Before the image is copied to the SD card in the previous example, the disk partition is unmounted (`umount /dev/mmcblk0p1`—notice there is no *n* in `umount`). It is a good practice to unmount all disk partitions before formatting or overwriting a disk.

When the image is copied with the `dd` command:

- `if=` specifies the input file (`2012-08-16-wheezy-raspbian.img`)
- `of=` specifies the output file (`/dev/mmcblk0`)
- `bs=` specifies the size of the blocks written to the disk

The `dd` utility can also be used as a backup tool. Just exchange the input file (`if=`) and output file (`of=`).

Use the following command to create a backup using the disk from the previous example:

```
dd bs=1M if=/dev/mmcblk0 of=backup-2012-08-16.img
```

See also

- **dd (Unix)**

  `http://en.wikipedia.org/wiki/Dd_(Unix)`

  This article from Wikipedia explains original application of the `dd` command.

- **dd – convert and copy a file**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=dd`

  The Debian man page for `dd`.

- **dd (gnu – coreutils)**
  `http://www.gnu.org/software/coreutils/manual/html_node/dd-invocation.html`

  The GNU operating system manual reference for `dd`.

Creating SD cards with BerryBoot

The following recipe explains how to install an operating system on an SD card using BerryBoot.

Unlike many Raspberry Pi distributions, BerryBoot is not a standalone disk image. It is a bare minimum Linux operating system running as a single application that automates the installation of the Raspberry Pi. It is by far the easiest way of selecting and installing an operating system distribution.

BerryBoot supports the installation of multiple Linux distributions on a single SD card and is by far the easiest way of installing an operating system for your Raspberry Pi. However, it does require an Internet connection during setup and installation.

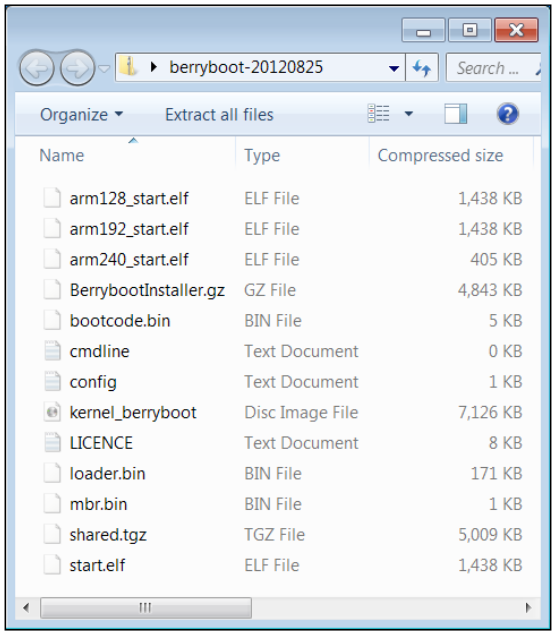
The installation takes place by booting the Raspberry Pi with an installer that can be easily copied to a FAT-formatted SD card using drag-and-drop from any PC. When the Raspberry Pi boots, the installer runs and uses the Raspberry Pi's network connection to download the latest distribution of selected operating systems.

Getting ready

You will need the following:

- An initial Raspberry Pi setup (see the *Preparing for the initial boot* recipe)
- The latest `berryboot.zip` file (see the link given in the *See also* section of this recipe)
- A class 4 SD card of 4 GB (or greater)
- A network connection

The Raspberry Pi needs to be connected to the Internet (via a home or office network) to complete this recipe. During installation, the network connection is used to download the selected distribution files. Without an Internet connection this recipe will not work.



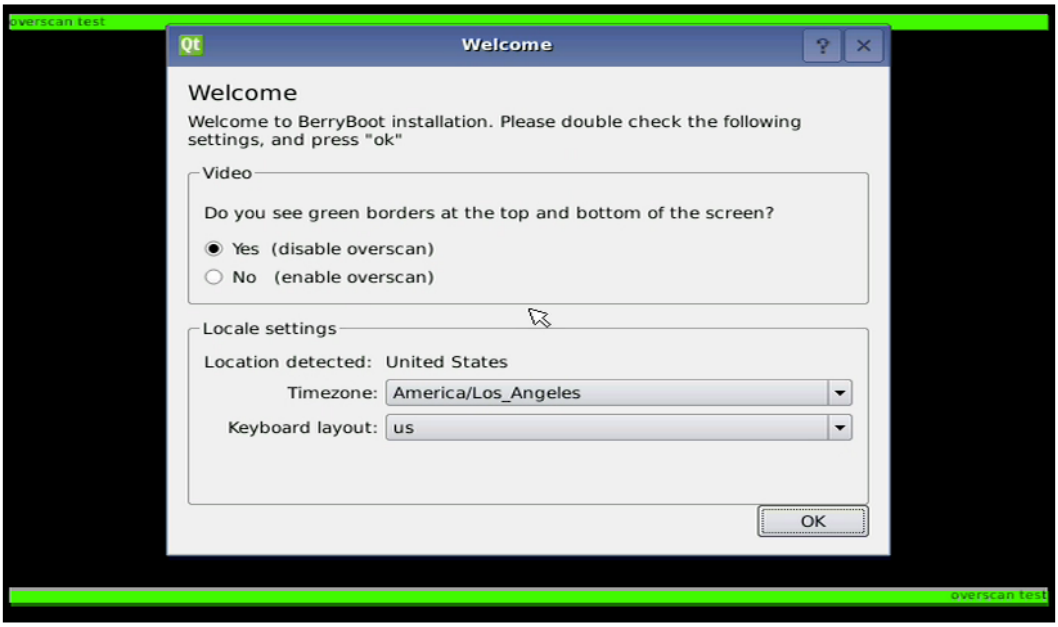
The preceding screenshot shows the contents of the `berryboot.zip` file.

How to do it...

The following are the steps for creating a boot disk with BerryBoot:

1. Format the SD card as a bootable FAT disk. Use the normal disk formatting tools that come with your PC's operating system.
2. Extract the contents of the `berryboot.zip` file to the newly formatted SD card. Again, the normal archival tools that come with your PC will do the job.
3. Connect your Raspberry Pi to a network with access to the Internet.

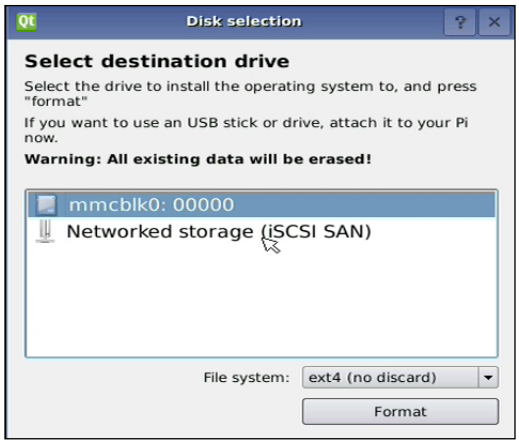
4. Insert the SD card in the Raspberry Pi and turn it on. This will start the BerryBoot installation process.



The preceding screenshot shows BerryBoot's **Welcome** screen.

5. After the network interfaces have been detected, the **Welcome** dialog box is displayed.
    i. Set the appropriate video scan option (the example shows a green overscan area indicating overscanning should be disabled).
    ii. The time zone and keyboard layout are also set on the **Welcome** dialog box.

The **Welcome** screen has video settings and locale settings. If green bars are displayed at the top and bottom of the screen (as in the preceding example), then the **Video** mode should be set to **Yes (disable overscan)**. This screen also contains locale settings for choosing a time zone and a keyboard layout. Click on the **OK** button to continue.



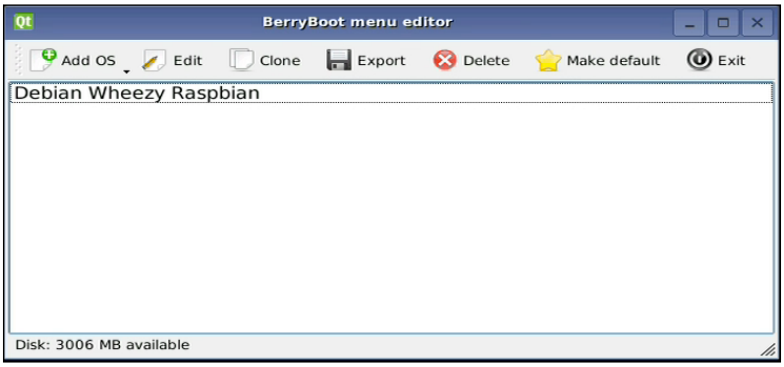
The preceding screenshot shows BerryBoot's **Disk selection** screen.

6. The **Disk selection** dialog box can be used to select an installation disk other than the SD card. The target disk may be formatted in one of the following three ways:
   - **B-tree file system** (**BTRFS**) for scalable storage
   - `ext4 – with discard flag set` for SDD disks
   - `ext4 – no discard` is the default

Even though the installation uses an external disk, the SD card is still required to boot.

The preceding screenshot shows BerryBoot's **Add OS** screen.

7. After the disk is formatted, BerryBoot downloads the list of available operating system images. The **Add OS** dialog box is used to select which operating system will be downloaded and installed next.



The preceding screenshot shows BerryBoot's **BerryBoot menu editor** screen.

8. The **BerryBoot menu editor** screen is used to add additional operating systems and manage those that have already been installed. All the installed images will appear in the **BerryBoot** boot menu.

   - The **Add OS** button is used to add another operating system to the boot menu (and repeat previous step)
   - The **Edit** button is used to change the name of the image displayed on the boot menu and to configure the memory split
   - The **Clone** button duplicates the current image in the boot menu
   - The **Export** button can be used to export the current disk image to an external disk
   - The **Delete** button deletes the current operating system from the boot menu
   - The **Make Default** button can be used to select which image will boot by default
   - The **Exit** button is used to reboot the Raspberry Pi

How it works...

First an SD card is formatted as a bootable FAT disk (using the utility of your choice). Then the BerryBoot installation files are copied to the disk. No special image writers are required! Just drag-and-drop all the unarchived files onto the SD card.

Once all the installation files are copied to the newly formatted SD card, the disk can be safely ejected and inserted into an unpowered Raspberry Pi. When the power cord is connected to the Raspberry Pi, the Raspberry Pi will boot automatically into the BerryBoot installation application.

After booting, the Raspberry Pi tries to detect all attached network cards. The installation application can detect many modern wireless USB network cards; however, only a limited number are actually configurable by the application. Once the network has been detected (and selected), the **Welcome** screen is displayed.

The BerryBoot install application will then detect any attached USB disk devices (or network storage). The installation application will permit the operating system to be installed on a disk other than the SD card. However, the Raspberry Pi will still need the BerryBoot SD card to boot. The Raspberry Pi will not boot from any other device. Choose `mmcblk0` to format the SD card. For most uses, the ext4 filesystem is the best choice. The **B-tree file system** (**BTRFS**) has advanced features that are not covered in this book. Choose `ext4 – with discard flag set` when formatting the SD card.

After the disk is formatted, the **Add OS** screen appears with a selection of Raspberry Linux distributions. There are distributions for using the Raspberry Pi as a media center (OpenELEC), as a classroom workstation (LTSP thin client BerryTerminal), and as a web server (BerryWebserver). There are also alternative Linux distributions (Puppy Linux and Sugar). And, of course, the "official" Raspbian Linux distribution is also included.

Once a Linux distribution has been chosen, it is downloaded and added to the boot menu. Choosing **Exit** from the **BerryBoot menu editor** screen reboots the Raspberry Pi. After reboot, the downloaded distributions are displayed on the boot menu and the user can select which distribution to boot.

Application cartridges

Each of the recipes in this book could be used to create an "application cartridge".

An **application cartridge** is like a game cartridge that is plugged into a game console, ready to play. The only difference is an application is stored on the cartridge instead of a game. So when the application cartridge is plugged into the Raspberry Pi, an application is started and is ready to use.

Using application cartridges, the Raspberry Pi can easily be repurposed just by switching the cartridge (the SD card). After shutting down and turning off the Raspberry Pi, the current SD card in the Raspberry Pi could be replaced with another SD card that has a different image installed on it – easily switching the Raspberry Pi from one purpose to another.

Write a multimedia home theatre distribution, for example OpenELEC or Raspbmc on one SD card to create a home theatre application cartridge. Write an IPFire image on another SD card to create a firewall application cartridge. A Berry Terminal image could be used to create an application cartridge for client access to a terminal server; or recipes from this book could be used to create application cartridges for a file server, a web server, or a wireless access point.

With a library of application cartridges, a Raspberry Pi can serve multiple purposes.

See also

- **BerryTerminal**

  `http://www.berryterminal.com`

  This Linux distribution turns the Raspberry Pi into a low-cost thin client allowing users to log in to a central **Linux Terminal Server Project** (**LTSP**) server. Thin clients can be used to run applications from Linux and Windows servers. Thin clients are reliable, cost-effective solutions for improving **total cost of ownership** (**TCO**) for organizations that only require simple desktop features (schools, call centers, factories).

- **IPFire**

  `http://en.wikipedia.org/wiki/IPFire`

  IPFire is a GNU Linux distribution that acts as a router and firewall. The Wikipedia article describes the distribution in more detail. You can download IPFire from `http://www.ipfire.org/`.

- **List of software based on XBMC**

  `http://en.wikipedia.org/wiki/List_of_software_based_on_XBMC`

  You can find a list of software distributions that are derived from the XBMC Media Center.

- **XMBC**

  `http://en.wikipedia.org/wiki/XBMC`

  You can find a Wikipedia article describing the free and open source media player application developed by the XBMC Foundation.

- **Installing OpenELEC on a Raspberry Pi**

  `http://wiki.openelec.tv/index.php?title=Installing_OpenELEC_on_Raspberry_Pi`

  You can find the installation instructions for putting OpenELEC on a Raspberry Pi.

- **IPFire on Raspberry Pi ready to first test**

  `http://planet.ipfire.org/post/ipfire-on-raspberry-pi-ready-to-first-test`

  A forum post describing how IPFire can be installed on a Raspberry Pi can be found.

- **OpenELEC**

  `http://www.openelec.tv`

  The Open Embedded Linux Entertainment Center Linux distribution can be found.

- **Raspbmc**

  `http://www.raspbmc.com/`

  A minimal XMBC Linux distribution developed specifically for the Raspberry Pi can be found.

Booting the "official" Raspbian Linux distribution

This recipe explains how to boot the official Raspbian distribution and run `raspi-config` to complete the installation of the Raspberry Pi.

The `raspi-config` command is run by default when you first boot the Raspberry Pi. In this recipe, the command is only used to update the tool itself. More detail on the use of `raspi-config` can be found in *Chapter 2*, *Administration*.

Once this recipe is complete, you will have booted the Raspberry Pi for the first time.

Getting ready

The following are the ingredients:

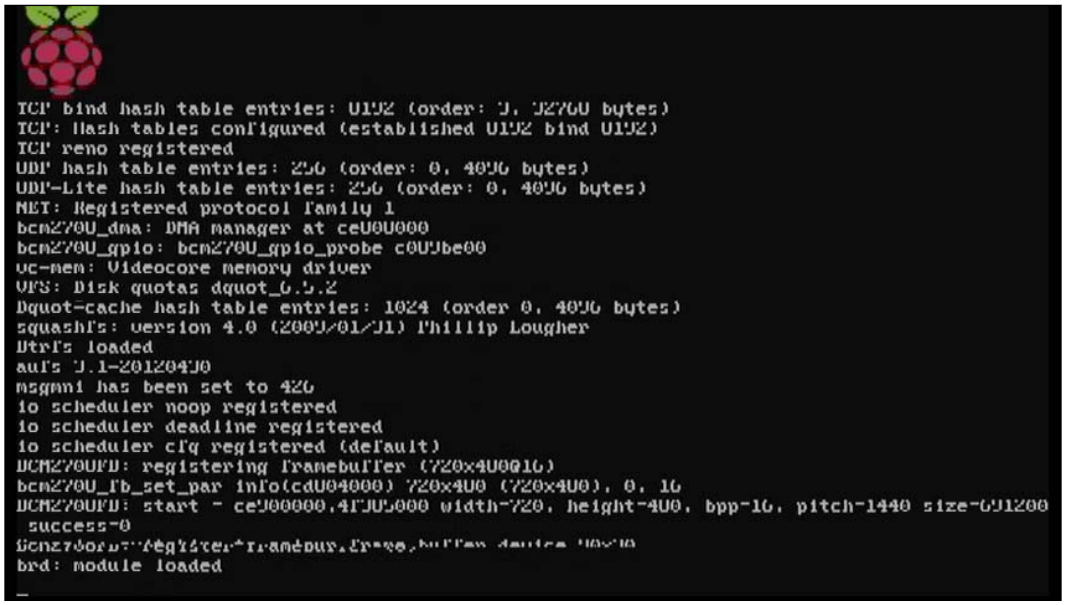
- An initial Raspberry Pi setup (see the *Preparing for the initial boot* recipe)
- An SD card formatted with the "official" Raspbian Linux image
- A network connection

The SD card should be formatted with a Raspbian image, a display and keyboard should be connected, and the Raspberry Pi should be ready to power on.

How to do it...

The following are the steps for booting the Raspberry Pi for the first time:

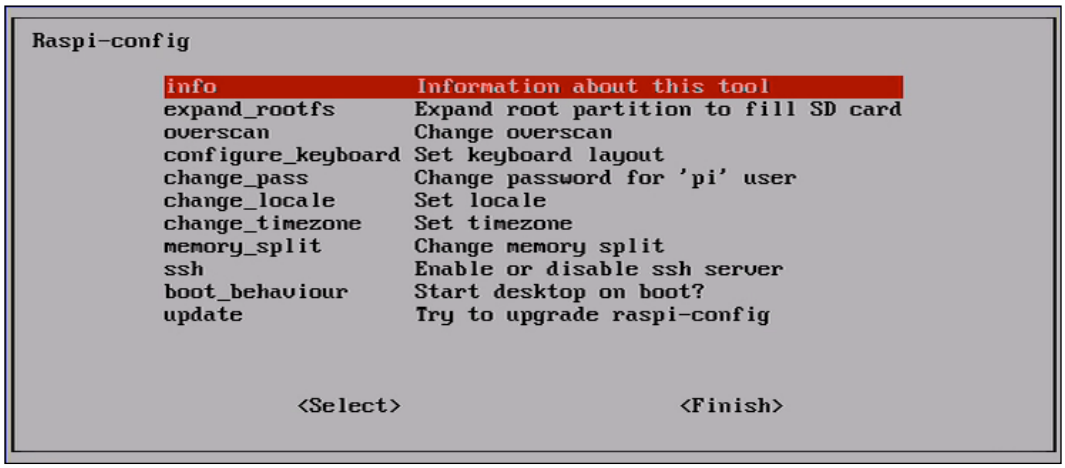
1. Insert the SD card into the Raspberry Pi and plug in the 5V power supply. The Raspberry Pi should start booting.



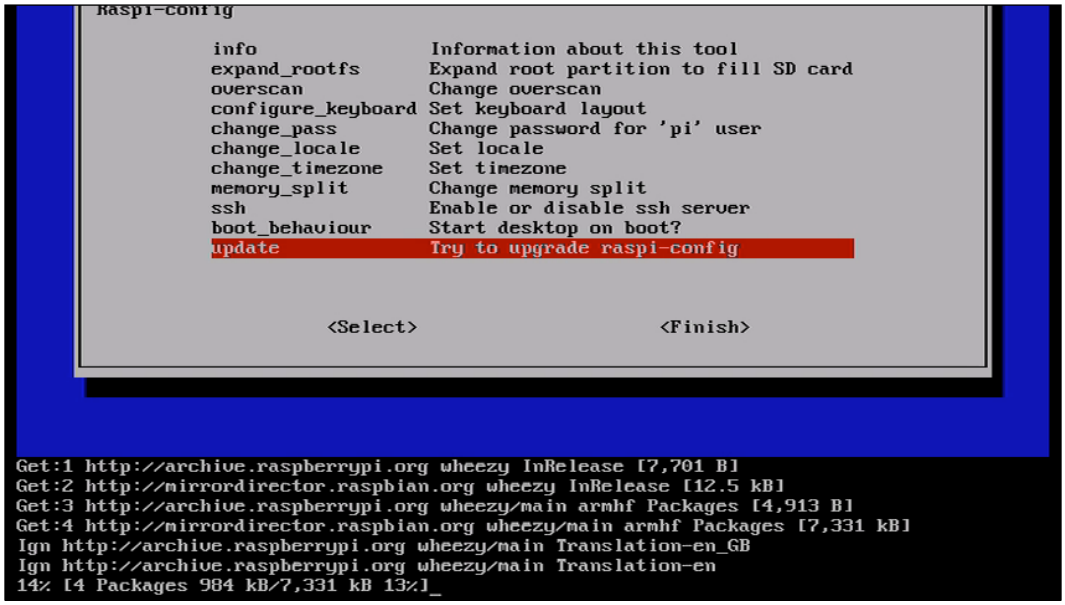
   The preceding screenshot shows the Raspberry Pi as it boots.

2. After a short initial boot, the **Raspi-config** main menu is displayed.
   - **info**: A short description of the tool
   - **expand_rootfs**: Resize the root partition to fill the SD card
   - **overscan**: Enable or disable overscan
   - **configure_keyboard**: Change the keyboard configuration
   - **change_locale**: Change the display language
   - **change_timezone**: Change the time zone
   - **memory_split**: Change how much memory is available
   - **ssh**: Enable or disable the SSH server
   - **boot_behavior**: Choose text mode or X windows GUI
   - **update**: Update the `raspi-config` utility

The following screenshot shows the **Raspi-config** main menu:



3. Choose **update** to download and install any updates to the configuration utility. Type in `sudo raspi-config` to return to the menu.



The preceding screenshot shows the updating process of the `raspi-config` utility.

4. Use the other menu items to configure the operating system.
5. Select **<Finish>** to complete the configuration and reboot the system.

How it works...

The official Raspbian Linux distribution comes with the `raspi-config` utility. It is run automatically on the first boot of the operating system. Afterwards, the system will boot without running the configuration utility.

When the **Raspi-config** main menu appears, the user can use the keyboard arrows, the *Tab* key, the Space bar, and the *Enter* key to navigate the menus.

The next chapter has recipes that use `raspi-config` to configure the Raspberry Pi. The example in this recipe selects the menu item update to fetch and install a current version of `raspi-config` from the Raspberry Pi package distribution center.

After the `raspi-config` command has been updated, the user is left at a command prompt and must restart `raspi-config`. The `raspi-config` command is privileged, so the `sudo` command is used as a prefix to temporarily grant privileges to the user.

After the user has configured the Raspberry Pi, selecting **<Finish>** from the main menu will cause the Raspberry Pi to reboot.

Once rebooted, the Raspberry Pi is ready for use!

There's more...

The `raspi-config` utility can be run at any time to reconfigure the Raspberry Pi. After logging in, enter the command `sudo raspi-config` to rerun the utility.

See also

- **sudo – execute a command as another user**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=sudo`

  The Debian man page for `sudo`.

- **RPi raspi-config**

  `http://elinux.org/RPi_raspi-config`

  This utility helps you configure the Raspberry Pi. It is a work in progress, expecting that the number of configuration items in this utility will increase over time.

Shutting down the Raspberry Pi (shutdown)

This recipe shuts down the Linux operating systems so that the Raspberry Pi can be powered off safely.

Before powering down the Raspberry Pi, it is important to first shut down the operating system so that all of the applications and services on the Raspberry Pi have completely finished writing to disk and are ready for the next boot.

External devices, such as hard disks, also need time to shut down and flush their buffers. The `shutdown` command also gives devices attached to the Raspberry Pi an opportunity to clean up and prepare for the next boot.

Getting ready

The following are the ingredients:

- An initial Raspberry Pi setup (see the *Preparing for the initial boot* recipe)
- An SD card formatted with the official Raspbian Linux image

The Raspberry Pi should already be powered on and booted before starting with this recipe.

How to do it...

The following are the steps for shutting down the Raspberry Pi:

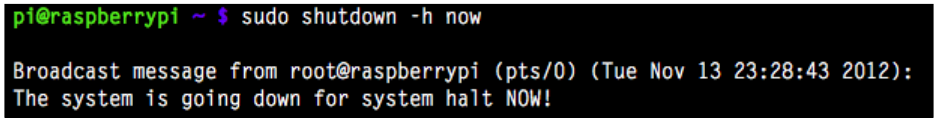
1. Log in to the Raspberry Pi with `pi` as the username (the default password is `raspberry`).

   The preceding screenshot shows the process of logging in to the Raspberry Pi with `pi` as the username.

2. Execute the following command:

   ```
   shutdown –h now
   ```

   Shut down and halt the operating system. This command is privileged. Use the prefix `sudo` to run the `shutdown` command as a privileged user.



   The preceding screenshot shows how to shut down the Raspberry Pi.

3. The Raspberry Pi will begin to shut down, displaying messages from applications, devices, and services as they clean up and prepare for the next boot.
4. Once the operating system has shut down, the Raspberry Pi will halt. Only a single red LED will remain lit on the Raspberry Pi as the Raspberry Pi is still powered on. The main display will be blank.
5. The power to the Raspberry Pi can now be shut off.

How it works...

Once the Raspberry Pi boots, it prompts for a username and a password. By default, the Raspbian Linux distribution has one user configured, the `pi` user. This user's default password is `raspberry`. The password for the `pi` user can be changed using the `raspi-config` command.

After login, the `shutdown` command is executed with the `–h` option telling the Raspberry Pi to halt the system (power it off) after the operating system has shut down. The `shutdown` command is privileged, so the `sudo` command is used as a prefix to temporarily grant privileges.

There's more...

The `shutdown` command can also be used to reboot the system. Use the `-r` option. Rebooting the system when logged in as the `pi` user can be done with the following command:

```
sudo shutdown -r now
```

A number of synonyms exist for the `shutdown` command including `halt`, `poweroff`, 'and `reboot`.

See also

- **halt, reboot, poweroff – stop the system**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=halt`

  The Debian man page for `halt`, `poweroff`, and `reboot`.

- **shutdown – bring the system down**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=shutdown`

  The Debian man page for `shutdown`.

2
Administration

In this chapter we will cover:

- Configuring remote access (raspi-config)
- Configuring memory usage (raspi-config)
- Remote access (SSH)
- Remote access (PuTTY)
- Changing the login password (passwd)

Introduction

This chapter contains a collection of recipes for the Raspberry Pi that cover the basic administration of the Raspberry Pi.

The chapter begins by configuring remote access to the Raspberry Pi. The second recipe shows how to change the video/processor memory split in the Raspberry Pi. Both of these recipes use the `raspi-config` tool provided with the "official" Raspbian Linux distribution that is available from Raspberry Pi Foundation.

The next two recipes show how the Raspberry Pi can be accessed remotely using an Open SSH client. One recipe is for Linux or Mac OS computers using the `ssh` command-line client. The other recipe is for Windows computers using the PuTTY client.

The final recipe is for changing the login password.

Once you complete this chapter, you will be able to access the Raspberry Pi command line remotely and complete a number of simple administration tasks.

Configuring remote access (raspi-config)

This recipe shows how to use the `raspi-config` utility during the initial boot of the "official" Raspbian Linux distribution to configure remote access.

During the normal installation of the "official" Raspbian Linux distribution, the Raspberry Pi configuration utility, `raspi-config`, is run automatically. This utility is used to configure many of the basic components of the Raspberry Pi including enabling the Secure Shell remote access server. This recipe shows how `raspi-config` is used to configure remote access.

The `raspi-config` utility was introduced in the *Booting the "official" Raspbian Linux distribution* recipe from *Chapter 1*, *Installation and Setup*. In that recipe, the utility was also used during installation. However, the utility can also be run from the command line after the system has been installed, as shown next in the *Configuring memory usage* recipe.

After you finish with this recipe, your Raspberry Pi will be accessible on your local network from another PC using the **Secure Shell** (**SSH**) protocol.

Getting ready

The following are the ingredients:

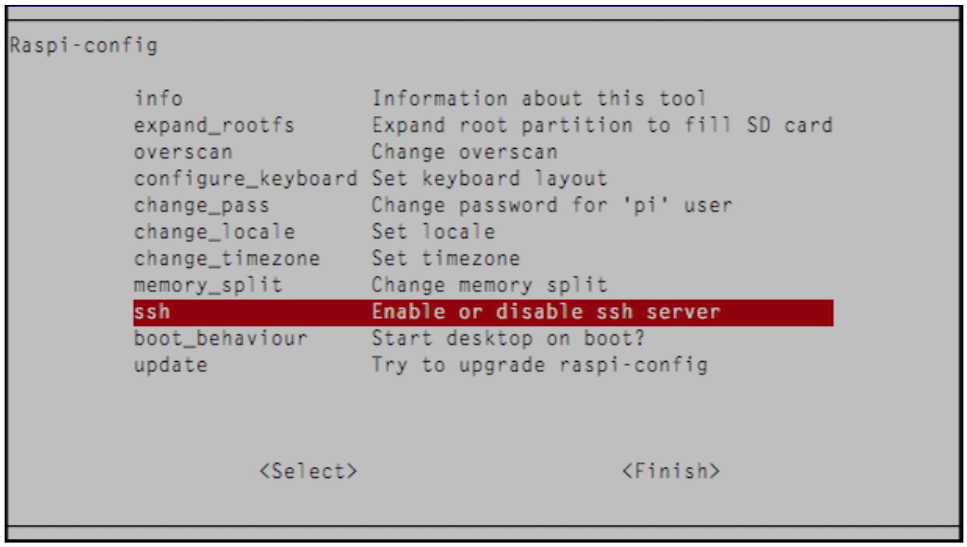
- An initial Raspberry Pi with a 5V power supply
- A connected display and keyboard
- A newly formatted SD card with the "official" Raspbian Linux image
- A network connection
- Another PC on the same network (for testing remote access)

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an LXTerminal.

How to do it...

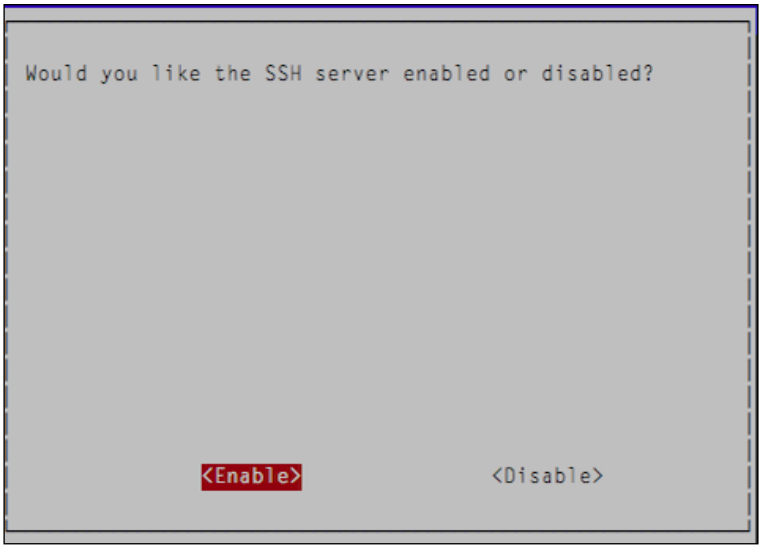
The following are the steps for configuring remote access to the Raspberry Pi:

1. Insert the SD card into the Raspberry Pi and plug in the 5V power supply. The Raspberry Pi should start booting.
2. After a short initial boot, the **Raspi-config** main menu is displayed.



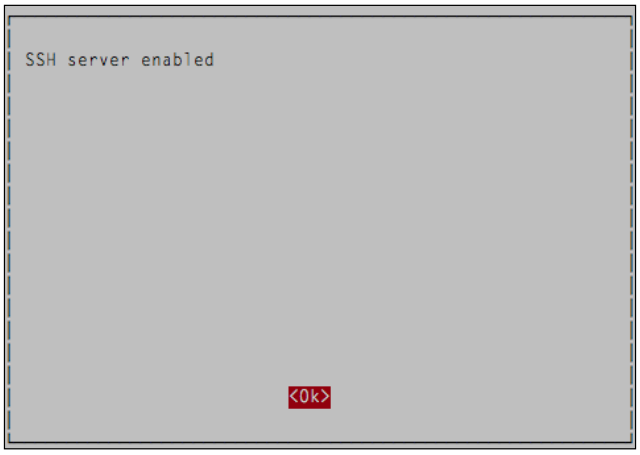
The preceding screenshot shows the **Raspi-config** main menu. The menu item for enabling remote access is selected.

3. Select **ssh** from the main menu to configure remote access.



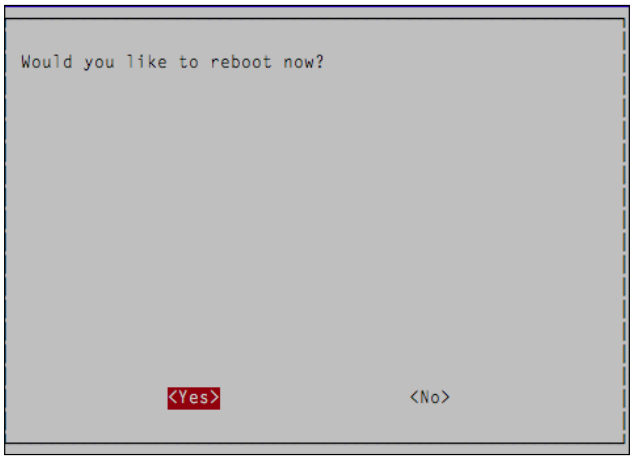
The preceding screenshot shows the `raspi-config` utility asking to enable remote access.

4. The following screen asks, **Would you like the SSH server enabled or disabled?**. Choose **<Enable>** to enable secure remote access to the Raspberry Pi.



The preceding screenshot shows that remote access to the Raspberry Pi has been enabled.

5. The **SSH server enabled** dialog appears. Press **<Ok>** to return to the main menu.
6. Choose **<Finish>** from the main menu, and a dialog appears asking, **Would you like to reboot now?**, as shown in the following screenshot:



The `raspi-config` utility asks to reboot.

7. Choose **<Yes>** to reboot the Raspberry Pi and enable remote access.

How it works...

One of the selections available from the **Raspi-config** main menu is for enabling the OpenSSH Secure Shell server. The OpenSSH server enables secure access to the Raspberry Pi over a network connection. Once enabled, the SSH server will be started automatically each time the Raspberry Pi boots.

To disable remote access, disable the OpenSSH server. The server can be disabled from the `raspi-config` utility using the same steps in this recipe, except that **<Disable>** should be selected when the utility asks, **Would you like the SSH server enabled or disabled?**.

The Secure Shell server runs on the Raspberry Pi. A client program such as `ssh` on Mac OS X or Linux, or PuTTY on Windows can be used to connect to the Secure Shell server (see the *See also* section of this recipe). The client program is a virtual terminal that, once connected, permits a user to interact remotely with the Raspberry Pi as though connected directly with a keyboard and a display. The virtual terminal cannot run X Windows directly, so you will need another tool to view the Raspberry Pi's GUI (see the *See also* section of this recipe).

There's more...

After the initial installation, the `raspi-config` configuration utility may be run again whenever there is a need to change the Raspberry Pi's configuration.

The utility can also be run remotely after logging in to the Raspberry Pi using `ssh` (see the *See also* section for an example).

See also

- **Secure Shell**

  `http://en.wikipedia.org/wiki/Secure_Shell`

  Secure Shell is a network protocol for secure data communication.

- **OpenSSH**

  `http://www.openssh.org/`

  An SSH1 and SSH2 implementation originally designed as part of the OpenBSD project. It is now included in most Linux distributions including the "official" Raspbian Linux.

Configuring memory usage (raspi-config)

This chapter features solutions that do not require a display. Many of these solutions would benefit from having the additional memory that the display would have been using. This recipe shows how to transfer the unused video memory in the Raspberry Pi to the CPU for using it in applications that do not require a display.

The `raspi-config` utility is used to change the Raspberry Pi's memory split. This utility is automatically run during installation of the "official" Raspbian Linux distribution. However, the utility can also be run from the command line. This recipe runs `raspi-config` from the command line.

You can use this recipe anytime when it is necessary to change the balance between memory dedicated for use by the Video Core and the memory available for the CPU. In this example, the CPU is given the maximum usage of the available memory.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection (optional)
- A client PC connected to the same network as the Raspberry Pi (optional)

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

How to do it...

The following are the steps for configuring the Raspberry Pi's memory usage:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following Raspberry Pi configuration command:

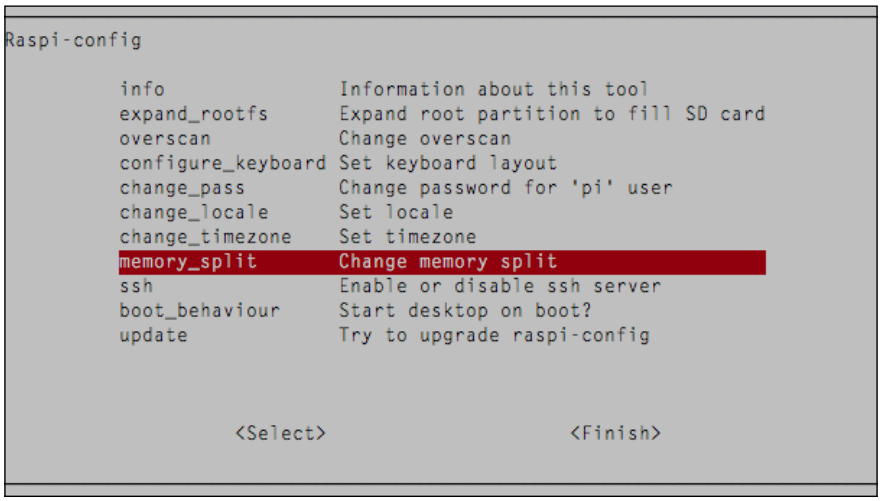
   `raspi-config`

   This command is privileged and needs to be run as a privileged user. Use the `sudo` command as a prefix to temporarily run the `raspi-config` command as a privileged user.

```
pi@raspberrypi ~ $ sudo raspi-config
```

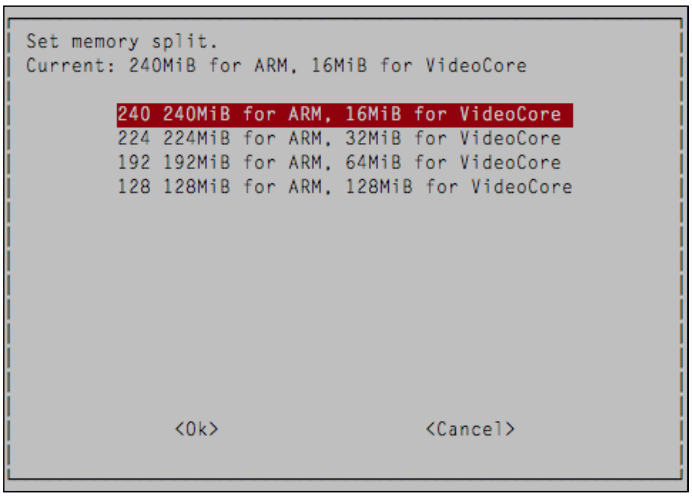
   The preceding screenshot shows how to execute the `raspi-config` command.

3. The `raspi-config` command will clear the screen and display its main menu.



The preceding screenshot shows the **Raspi-config** main menu. The **Change memory split** menu option is selected.

4. Select **memory_split** from the **Raspi-config** menu.



The preceding screenshot shows how to change the memory split using `raspi-config`.

5. Select **240** to give the Raspberry Pi's ARM CPU 240 MB of memory.

6. Select **<Ok>** to save the memory split value and continue.
7. Select **<Finish>** from the main menu and reboot the Raspberry Pi.

How it works...

The Raspberry Pi has 256 MB of memory that is shared between the ARM CPU and the Video Core. The memory can be split in four different ways, as follows:

- 240 MB for the CPU and 16 MB for the Video Core
- 224 MB for the CPU and 32 MB for the Video Core
- 192 MB for the CPU and 64 MB for the Video Core
- 128 MB for the CPU and 128 MB for the Video Core

For the Raspberry Pi solutions in this chapter, the Video Core only needs the minimum of 16 MB of memory. The maximum 240 MB should be assigned to the CPU. This will give the CPU the best performance.

For solutions that require a display, more Video Core memory should be configured. Multimedia solutions (for example, OMXPlayer and XBMC distributions) would benefit from having the maximum 128 MB of Video Core memory.

There's more...

In addition to changing the memory split, the Raspberry Pi can be overclocked to run at a faster processing speed. Overclocking, however, can void the Raspberry Pi's warranty.

The normal CPU frequency of the Raspberry Pi's ARM processor is 800 MHz. The CPU frequency of the ARM processor has successfully been overclocked to 1150 MHz. This is almost a 50 percent boost in power! Some Raspberry Pi users may wish to experiment with this boost of power even though it does void the warranty.

More information on overclocking the Raspberry Pi can be found on the Embedded Linux community wiki page (see the *See also* section of this recipe).

See also

- **Overclocking configuration**

  `http://elinux.org/RPi_config.txt#Overclocking_configuration`

Warning by the Embedded Linux community: Setting any of the parameters that overvolt your Raspberry Pi will set a permanent bit within the SOC and your warranty is void.

Remote access (SSH)

This recipe shows how to access the Raspberry Pi remotely using the `ssh` command.

After the Raspberry Pi has been configured to automatically start the Secure Shell server when it boots (see the previous recipe), it is possible to access the Raspberry Pi remotely using the Secure Shell client, `ssh`. The `ssh` command is built into the current version of the Mac OS X operating systems and is also available for most Linux distributions. For Microsoft Windows operating systems, another utility will be required (see this recipe).

This recipe starts by logging directly into the Raspberry Pi using an attached keyboard and display to discover the IP address of the Raspberry Pi. Once the IP address is known, the recipe shows how to log in to the Raspberry Pi from another computer using `ssh`.

After you finish with this recipe, you will no longer be required to log in directly to your Raspberry Pi with an attached keyboard. You will also not need a display. You will be able to remotely administer your Raspberry Pi from another PC on the same network.

Getting ready

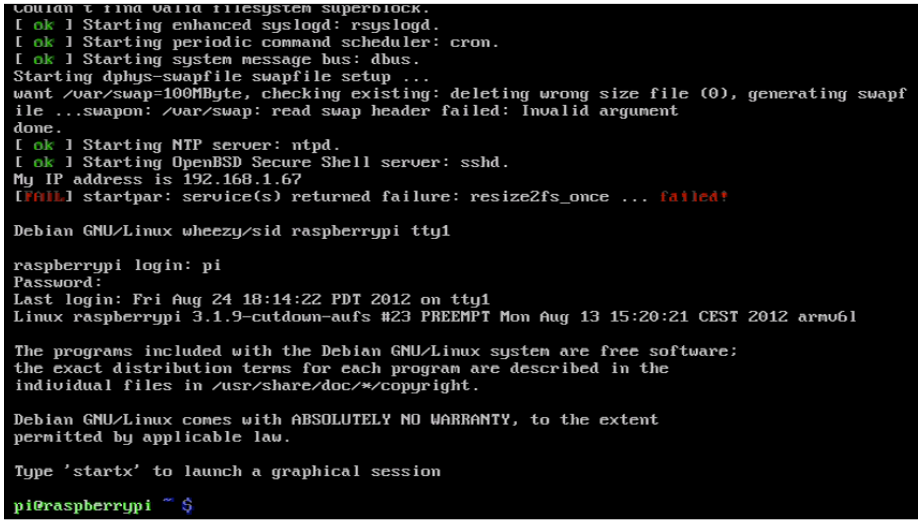
The following are the ingredients:

- An initial Raspberry Pi setup (see the preceding recipe)
- An installed and configured SD card (see the preceding recipe)
- A network connection
- A PC running Mac OS X, Linux, or OpenBSD

How to do it...

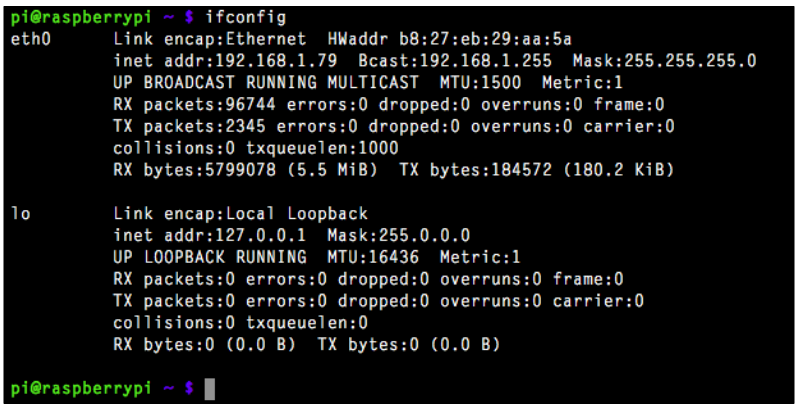
The following are the steps for remotely accessing the Raspberry Pi using the `ssh` command:

1. Insert the SD card into the Raspberry Pi and plug in the 5V power supply. The Raspberry Pi should start booting (you'll see the LEDs flashing).



   The preceding screenshot shows a successful login at the console after booting has been finished.

2. Once the Raspberry Pi has booted, log in using the keyboard and display.
3. Use the `ifconfig` command to discover the Raspberry Pi's IP address.



The preceding screenshot shows how the `ifconfig` utility is used to display the Raspberry Pi's IP address (`192.168.1.79`).

4. Use the discovered IP address (`192.168.1.79`) to log in to the Raspberry Pi from another PC using `ssh`. The format of the command is `ssh username@ipaddress`. For the "official" Raspbian Linux distribution, `pi` is the username, and `ipaddress` is the IP address of the Raspberry Pi (`pi@192.168.1.79`).

```
golden-imac:~ golden$ ssh pi@192.168.1.79
The authenticity of host '192.168.1.79 (192.168.1.79)' can't be established.
RSA key fingerprint is 39:49:70:f9:7e:92:38:bb:3a:4d:67:e0:0e:1a:dd:d5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.79' (RSA) to the list of known hosts.
pi@192.168.1.79's password:
Linux raspberrypi 3.1.9+ #272 PREEMPT Tue Aug 7 22:51:44 BST 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

Last login: Sun Sep  9 19:21:29 2012 from golden-imac
pi@raspberrypi ~ $
```

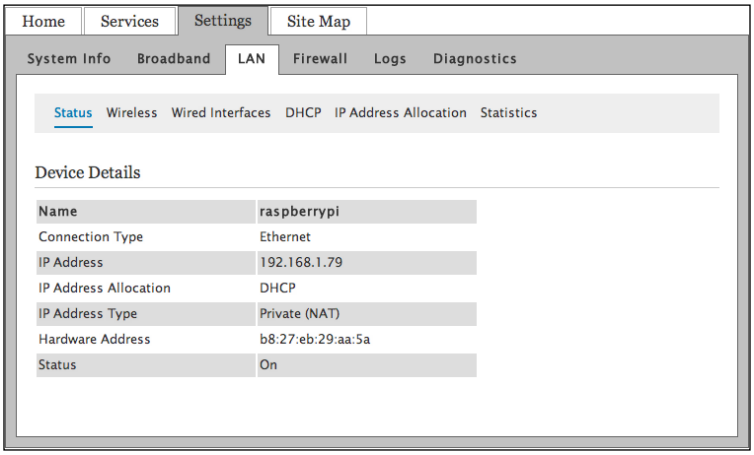
The preceding screenshot shows a remote login via `ssh` from `golden@golden-imac` to `pi@192.168.1.79`.

5. After entering the command, you are prompted for the password. The default password for the `pi` user is `raspberry`.
6. Answer `yes`, when asked, **Are you sure you want to continue connecting (yes/no)?**.
7. You are now connected to the Raspberry Pi remotely!
8. Type in `exit` to log out.

How it works...

A user on a client computer can use the Secure Shell command, `ssh`, to access the Raspberry Pi as though they have logged in directly using a keyboard and a display connected to the Raspberry Pi. Once the Secure Shell server is up and running, the Raspberry Pi can be accessed remotely and no longer needs a keyboard or a display.

Before remote login is possible, the network address of the Raspberry Pi must be known. In this recipe, the `ifconfig` command was used to discover the IP address directly from the Raspberry Pi via an attached keyboard and display. It is also possible to discover the IP address from the configuration interface of the local network gateway or DSL router.



The preceding screenshot shows how the IP address of a device named **raspberrypi** is displayed in a DSL router's configuration interface.

The IP address of a device (for example, the Raspberry Pi) is a mapping between the permanent hardware address of the device's network interface (**b8:27:eb:29:aa:5a**) and its current address in the topology of the local network (**192.168.1.79**). In the preceding example, the device named **raspberrypi** has been assigned network ID `79` in the `192.168.1` subnet of the local network.

Most home and small office networks attempt to assign network IDs semi-permanently to the same device (hardware address). So, once discovered, it is not likely that the Raspberry Pi's IP address will change.

There's more...

The Secure Shell utility, `ssh`, uses encryption to ensure that communication between the Raspberry Pi and other computers on the network remains secure. On first launch, the Secure Shell server creates a unique security key (like a fingerprint) that can be used to uniquely identify that particular Secure Shell server. The first time a Secure Shell client connects to a Secure Shell server at a specific IP address, the client prompts the user to verify that the server has the correct fingerprint and then stores the key for identifying the server during the next connection.

The Secure Shell client will prevent logins to a machine with the same IP address but a different security key. This helps to prevent a hacker from spoofing a device by stealing its IP address.

```
golden-imac:~ golden$ ssh pi@192.168.1.79
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
98:be:85:ba:3a:c7:76:13:9a:3c:a8:ed:7f:9f:6e:51.
Please contact your system administrator.
Add correct host key in /Users/golden/.ssh/known_hosts to get rid of this me
ssage.
Offending RSA key in /Users/golden/.ssh/known_hosts:1
RSA host key for 192.168.1.79 has changed and you have requested strict chec
king.
Host key verification failed.
golden-imac:~ golden$
```

The preceding screenshot shows that the Mac OS X `ssh` client suspects a hacker, but it's only a new install of the Raspberry Pi.

When the security key that the Secure Shell client receives from the server on the Raspberry Pi no longer matches with the one that the client has stored, access to the Raspberry Pi is denied.

When the Raspberry Pi is reinstalled, the Secure Shell server's security key changes because a new security key is generated each time the Secure Shell server is installed. The same safety check that prevents a hacker from completing a successful man-in-the-middle attack also prevents login to a newly reinstalled Raspberry Pi.

The old, invalid key of the Raspberry Pi's Secure Shell server stored on the client can be deleted using the `ssh-keygen` command. The IP address of the Raspberry Pi needs to be specified on the command line, as shown in the following example:

```
ssh-keygen -f "/Users/golden/.ssh/known_hosts" -R 192.168.1.79
```

Once the security key has been deleted, the Secure client will no longer block access to that IP address. Afterwards, login using `ssh` works the same way as during the initial login, and the new security key is stored on the client.

```
golden-imac:~ golden$ cd /Users/golden/.ssh
golden-imac:.ssh golden$ ssh-keygen -f "known_hosts" -R 192.168.1.79
known_hosts updated.
Original contents retained as known_hosts.old
golden-imac:.ssh golden$
```

The `ssh-keygen` command is used to remove (`-R`) the security key for IP address `192.168.1.79`.

See also

- **ifconfig – configure a network interface**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=ifconfig`

  The Debian man page for `ifconfig`.

- **ssh – OpenSSH SSH client (remote login program)**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=ssh`

  The Debian man page for `ssh`.

- **ssh-keygen – key generation, management, and conversion**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=ssh-keygen`

  The Debian man page for `ssh-keygen`.

Remote access (PuTTY)

This recipe shows how to create a secure connection to the Raspberry Pi using PuTTY, one of the most commonly used Windows Secure Shell clients.

The PuTTY virtual terminal program is not built into Windows. It must first be downloaded and installed.

This recipe assumes the Raspberry Pi's SSH server has been enabled (see the *Configuring remote access (raspi-config)* recipe), the IP address of the Raspberry Pi has been discovered (see the previous recipe) and you are now ready to access the Raspberry Pi remotely from a Windows PC using PuTTY.

Once you finish with this recipe, you will be able to remotely administer your Raspberry Pi from a Windows PC.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- A Windows PC connected to the same network as the Raspberry Pi
- PuTTY should be downloaded and installed on the Windows PC

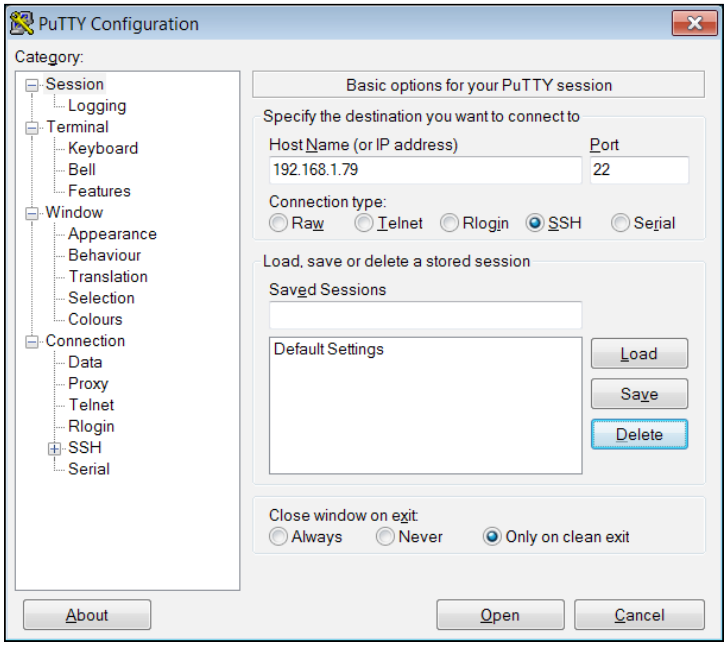
For this recipe, there is no need to have a keyboard or display connected directly to the Raspberry Pi.

PuTTY can be downloaded from
`http://www.chiark.greenend.org.uk/~sgtatham/putty/`.

How to do it...

The following are the steps for remotely accessing the Raspberry Pi using the PuTTY command:

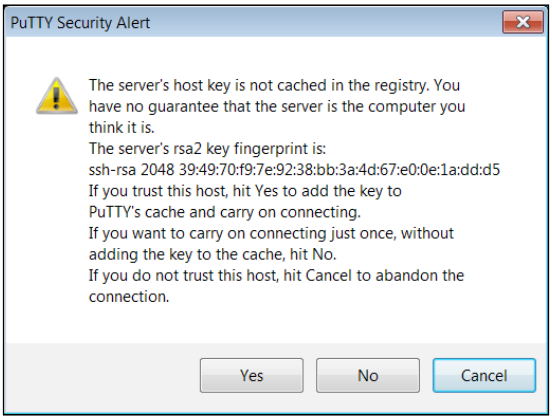
1. Insert the SD card into the Raspberry Pi and plug in the 5V power supply. The Raspberry Pi should start booting (you'll see LEDs flashing).
2. Once the Raspberry Pi has booted, PuTTY can be started on the Windows PC (double-click on the `PuTTY.exe` file).



In the preceding screenshot, the configuration window for PuTTY is being configured to connect to a Raspberry Pi at the IP address **192.168.1.79**.

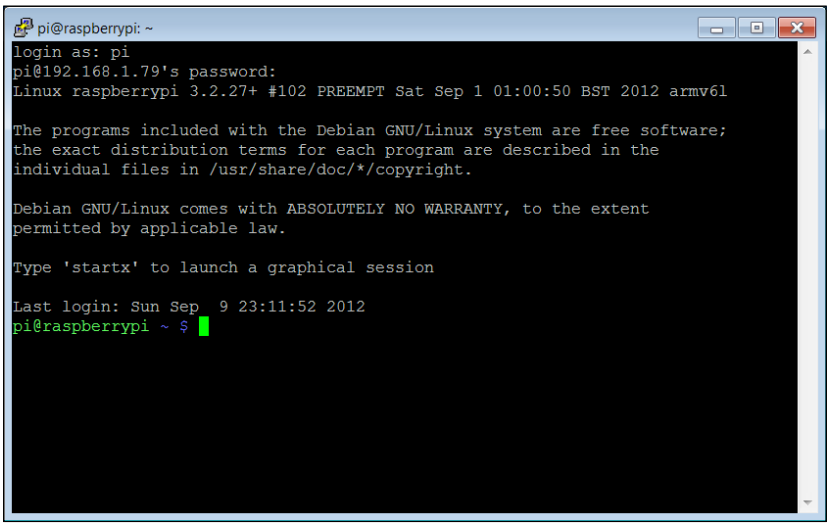
3. When PuTTY is started, the **PuTTY Configuration** screen is displayed.

4. Enter the Raspberry Pi's IP address in the **Host Name (or IP address)** field, and click on **Open** to connect to the Raspberry Pi.



In the preceding screenshot, PuTTY is displaying a warning message the first time it connects to the Raspberry Pi's SSH server.

5. The first time you connect to the Secure Shell server on the Raspberry Pi, PuTTY will display a warning message. This is because PuTTY does not yet have a stored security key for the Secure Shell server that is running on the Raspberry Pi.
6. Click on the **Yes** button to continue.

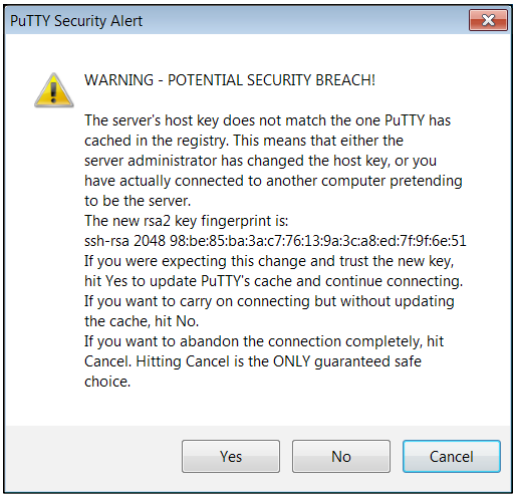


The preceding screenshot shows a successful login using PuTTY.

7. After you've accepted the security warning, you are asked for the login username and a password. Enter `pi` and the password that was configured during installation (`raspberry` is the default).
8. After you have successfully logged in to the Raspberry Pi remotely, you will no longer require a keyboard or a display to administer the system!
9. Type in `exit` to log out.

There's more...

When the Raspberry Pi is reinstalled, the security key of the Secure Shell server will be generated newly. PuTTY will not recognize the newly generated security key.



The preceding screenshot shows what happens when the Raspberry Pi has been reinstalled. PuTTY will think there has been a security breach!

When the **Putty Security Alert** dialog box appears after reinstalling the Raspberry Pi, click on **Yes** to accept and store the new security key.

If the Raspberry Pi has not been reinstalled, click on **Cancel** to disconnect and then make sure that there has not been a security breach!

See also

- **PuTTY**

  `http://www.chiark.greenend.org.uk/~sgtatham/putty/`

  PuTTY is a free SSH client for Windows. It is also a telnet client and an xterm emulator.

Changing the login password (passwd)

This recipe shows how to change the login password.

Once remote access to the Raspberry Pi has been enabled, anyone on the local network who knows the correct username and password will be able to log in remotely. To prevent unauthorized access to the Raspberry Pi, the default installation password should be changed immediately after the installation is complete.

The `raspi-config` utility can be used to change the login password; however, this recipe uses the `passwd` command. The `passwd` command is a standard GNU utility available with most Linux distributions and can generally be used, even if `raspi-config` is not available.

Changing the login password frequently will inhibit unauthorized access to the Raspberry Pi. You should use this recipe on a regular basis to protect access to your Raspberry Pi.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection (optional)

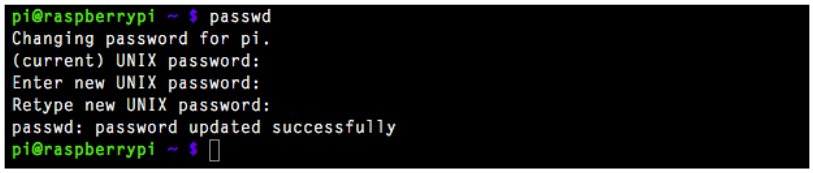
If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client (see the *Remote access (ssh)* recipe).

How to do it...

The following are the steps for changing the login password:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   passwd
   ```

   The preceding screenshot shows how the `passwd` command is used to change the login password.

3. Enter the current password – the password used while logging on. The passwords will not be displayed as they are entered.
4. Enter the new password.
5. Enter the new password for the second time.
6. The command responds with **passwd: password updated correctly**.
7. The login password has been successfully changed!

How it works...

The `passwd` command first prompts for the current login password to ensure that the user attempting to change the password is really authorized to do so. The password is not displayed as it is typed to protect the password from being overseen. If the current password is not entered correctly, the user will be asked to retype the password.

Once the user has been authorized, the command prompts for a password. Again, the new password is not displayed. The command prompts for the new password a second time to ensure that it was entered correctly. If the two new passwords differ, the user will be asked to type them again.

After the new password has been entered correctly, the user is notified that the password has been successfully changed. Changing the login password on a regular basis will help to prevent any unauthorized access to the Raspberry Pi.

See also

- **Passwd – change user password**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=passwd`

  The Debian man page for `passwd`.

3
Maintenance

In this chapter we will cover the following:

- Updating the operating system (`apt-get`)
- Searching for the software packages (`apt-cache`)
- Installing a package (`apt-get`)
- Package management (`aptitude`)
- Accessing the built-in documentation (`man`)
- Accessing the built-in documentation (`info`)

Introduction

The recipes in this chapter are for the basic maintenance of the Raspberry Pi. These recipes show how to update the operating system, install software, and access the built-in documentation.

After completing the recipes in this chapter you will be able to update the Raspberry Pi and read the built-in documentation.

Updating the operating system (apt-get)

This recipe shows how to update the Raspberry Pi using the `apt-get` command.

The Raspberry Pi and the operating systems that run on the Raspberry Pi, such as the "official" Raspbian Linux Distribution, are evolving at a rapid pace. Every week (if not every day) there are new updates and improvements that can be downloaded and installed.

The `apt-get` command is part of the Advanced Package Tool (`apt`) suite of command utilities. This command suite contains the primary package management tools for Debian-based Linux distributions such as the "official" Raspbian Linux distribution. The Advanced Package Tool not only installs software improvements but also security fixes. Used on a regular basis, the tool not only keeps the Raspberry Pi up to date, it also keeps it secure.

You should use this recipe on a regular basis (at least monthly) to ensure that your Raspberry Pi is running with the latest software and security updates.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- A client PC connected to the same network as the Raspberry Pi (optional)

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

The Raspberry Pi will need access to the Internet to reach the update server(s) from which it will pull the software updates and security fixes.

How to do it...

The steps for updating the Raspberry Pi's operating system are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   apt-get update
   ```

   This command is privileged and needs to be run as a privileged user. Use `sudo` as a prefix to run the command as a privileged user.

```
pi@raspberrypi ~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org wheezy InRelease [7,701 B]
Get:2 http://mirrordirector.raspbian.org wheezy InRelease [12.5 kB]
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Get:3 http://mirrordirector.raspbian.org wheezy/main armhf Packages [7,351 k
B]
Ign http://archive.raspberrypi.org wheezy/main Translation-en_US
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Get:4 http://mirrordirector.raspbian.org wheezy/contrib armhf Packages [23.3
 kB]
Get:5 http://mirrordirector.raspbian.org wheezy/non-free armhf Packages [46.
4 kB]
Get:6 http://mirrordirector.raspbian.org wheezy/rpi armhf Packages [14 B]
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
Fetched 7,441 kB in 1min 11s (105 kB/s)
Reading package lists... Done
pi@raspberrypi ~ $
```

The preceding screenshot is an example of using the `apt-get update` command to fetch the new software distribution catalogs.

3. Execute the command:

   `apt-get dist-upgrade`

   This command is privileged and needs to be run as a privileged user.

```
pi@raspberrypi ~ $ sudo apt-get dist-upgrade
```

The preceding screenshot shows the `apt-get dist-upgrade` command used to upgrade the Raspberry Pi.

4. This command calculates and then displays the changes that will be applied to the system during the upgrade:

```
Calculating upgrade... Done
The following packages will be REMOVED:
  linux-sound-base
The following NEW packages will be installed:
  ncurses-term
The following packages will be upgraded:
  alsa-base aptitude aptitude-common base-files console-setup
  console-setup-linux debconf debconf-i18n debconf-utils desktop-base dpkg
  dpkg-dev gcc-4.7-base gconf-service gconf2 gconf2-common ifupdown
  initscripts iso-codes keyboard-configuration kmod libbsd0 libc-bin
  libc-dev-bin libc6 libc6-dev libdpkg-perl libgail-3-0 libgail18 libgcc1
  libgconf-2-4 libgdu0 libgfortran3 libgl1-mesa-glx libglapi-mesa libgomp1
  libgpg-error0 libgtk-3-0 libgtk-3-bin libgtk-3-common libgtk2.0-0
  libgtk2.0-bin libgtk2.0-common libgudev-1.0-0 libicu-dev libicu48
  libkmod2 liblapack3 liblapack3gf libluajit-5.1-common libpulse0
  libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0
  libsasl2-2 libsasl2-modules libsmbclient libssh2-1 libstdc++6
  libthai-data libthai0 libudev-dev libudev0 libwbclient0 locales lua5.1
  luajit module-init-tools multiarch-support omxplayer openssh-client
  openssh-server raspberrypi-bootloader samba-common ssh sysv-rc sysvinit
  sysvinit-utils tasksel tasksel-data udev xkb-data
83 upgraded, 1 newly installed, 1 to remove and 0 not upgraded.
Need to get 143 MB of archives.
After this operation, 51.2 kB disk space will be freed.
Do you want to continue [Y/n]?
```

In the preceding screenshot, the `apt-get dist-upgrade` command has calculated which packages need upgrading.

5. The user is prompted, **Do you want to continue [Y/n]?**. Press the *return* key to accept the default of **Y** to continue.

```
Get:83 http://archive.raspberrypi.org/debian/ wheezy/main libraspberrypi0 ar
mhf 1.20120831-1 [333 kB]
Get:84 http://archive.raspberrypi.org/debian/ wheezy/main raspberrypi-bootlo
ader armhf 1.20120831-1 [36.4 MB]
89% [84 raspberrypi-bootloader 21.3 MB/36.4 MB 58%]      69.2 kB/s 3min 39s
```

The preceding screenshot shows the `apt-get dist-upgrade` command downloading packages.

6. Once accepted, the command first downloads the new packages. The bytes remaining, the average transfer speed, and the time remaining for each package are displayed as each package is downloaded.

This step could take quite a while depending on the number of updates and the speed of the Internet connection.

```
Unpacking replacement libraspberrypi-bin ...
Preparing to replace libraspberrypi0 1.20120810-1 (using .../libraspberrypi0
_1.20120831-1_armhf.deb) ...
Unpacking replacement libraspberrypi0 ...
Preparing to replace raspberrypi-bootloader 1.20120810-1 (using .../raspberr
ypi-bootloader_1.20120831-1_armhf.deb) ...
```

The preceding screenshot shows the `apt-get dist-upgrade` command downloading and installing packages.

7. Once the command completes, the Raspberry Pi will be fully updated and all of the current security fixes will also have been applied.

```
Setting up libraspberrypi0 (1.20120831-1) ...
ln: failed to create symbolic link `/lib/ld-linux.so.3': File exists
Setting up libraspberrypi-dev (1.20120831-1) ...
Setting up libraspberrypi-doc (1.20120831-1) ...
Setting up libraspberrypi-bin (1.20120831-1) ...
Setting up ssh (1:6.0p1-3) ...
Setting up tasksel-data (3.12) ...
Setting up tasksel (3.12) ...
pi@raspberrypi ~ $
```

In the preceding screenshot, the `apt-get dist-upgrade` command has completed downloading and installing the new and updated packages.

8. Use the following command to reboot the Raspberry Pi:

   `shutdown -r`

   This command is privileged, so use `sudo` as a prefix.

```
pi@raspberrypi ~ $ sudo shutdown -r now

Broadcast message from root@raspberrypi (pts/0) (Sat Sep 15 20:32:09 2012):
The system is going down for reboot NOW!
pi@raspberrypi ~ $ Connection to 192.168.1.79 closed by remote host.
Connection to 192.168.1.79 closed.
golden-imac:.ssh golden$
```

In the preceding screenshot, the `shutdown -r now` command was used to reboot the Raspberry Pi from a remote PC.

9. Once the Raspberry Pi has rebooted, the upgrade is complete!

How it works...

The "official" Raspbian Linux distribution, like most operating system distributions for the Raspberry Pi, is organized as a collection of software packages. Each software package contains one or more applications, with their configuration files and support libraries. Each package is also labeled with its current version and its dependency on other software packages. The **Advanced Package Tool** (`apt`) and its supporting utilities (such as `apt-get`) are used to manage the software packages of the "official" Raspbian Linux distribution.

The `apt-get update` command is used to fetch the current software catalogs from the software distribution sites that are configured in the `/etc/apt/sources.list` file and those configured in the files located in the `/etc/apt/sources.list.d` directory.

```
pi@raspberrypi ~ $ cat /etc/apt/sources.list
deb http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-fre
e rpi
pi@raspberrypi ~ $ ls /etc/apt/sources.list.d/
raspi.list
pi@raspberrypi ~ $ cat /etc/apt/sources.list.d/raspi.list
deb http://archive.raspberrypi.org/debian/ wheezy main
pi@raspberrypi ~ $
```

In the preceding screenshot, the `ls` and `cat` commands are used to display the contents of the `apt` command's configuration files.

The `sources.list` file located in the `/etc/apt` directory points to the base operating system distribution and the `raspi.list` file located in the `/etc/apt/sources.list.d` directory points to the location of additional packages. These are the only two `apt` configuration files in the "official" Raspbian Linux distribution. These configuration files each follow the same format.

Each line of the configuration files describes the root of one software distribution. The most preferred software distribution is listed first. The entire `sources.list` file is preferred over the files in the `sources.list.d` directory.

The first item in each line defines the distribution format. For Debian-based distributions, such as the "official" Raspbian Linux distribution, the definition is either `deb` or `deb-src`. The `deb-src` catalogs include source code as well as compiled binaries.

The second item in each line is the location of the root of the software distribution. The `sources.list` file points to the base `raspbian` operating system distribution located on the `mirrordirector.raspbian.org` website (`http://mirrordirector.raspbian.org/raspbian/`). The `raspi.list` file points to the additional `debian` packages provided by the Raspberry Pi foundation located on the `archive.raspberrypi.org` website (`http://archive.raspberrypi.org/debian/`).

The third item in each line is the name of the version to use. Both files specify `wheezy` as the name of the distribution version.

The remainder of the line is a list of distribution components. The configuration of the base distribution (`/etc/apt/source.list`) points to the "official" Raspbian Linux distribution and has a number of components defined. The secondary configuration file, `raspi.list`, located in the `/etc/apt/sources.list.d` directory, points to a small distribution of Raspberry Pi specific packages that has only one component, `main`.

The components of the "official" Raspbian Linux distribution are as follows:

- `main` – the base distribution supported by the Raspbian team
- `contrib` – components contributed to Raspbian, but supported outside of the Raspbian team
- `non-free` – components that are not open source or have some rights restrictions, and are also supported outside of the Raspbian team
- `rpi` – Raspberry Pi specific packages supported by the Raspberry Pi community

The single component defined in `raspi.list` is as follows:

- `main` – the packages supported by the Raspberry Pi Foundation

These files are located under the `/etc/apt` directory and will require a privileged user to edit them. The `/etc` directory is the root of the file hierarchy where configuration files are stored on the "official" Raspbian Linux distribution. This file hierarchy is protected and requires root permission to change the files.

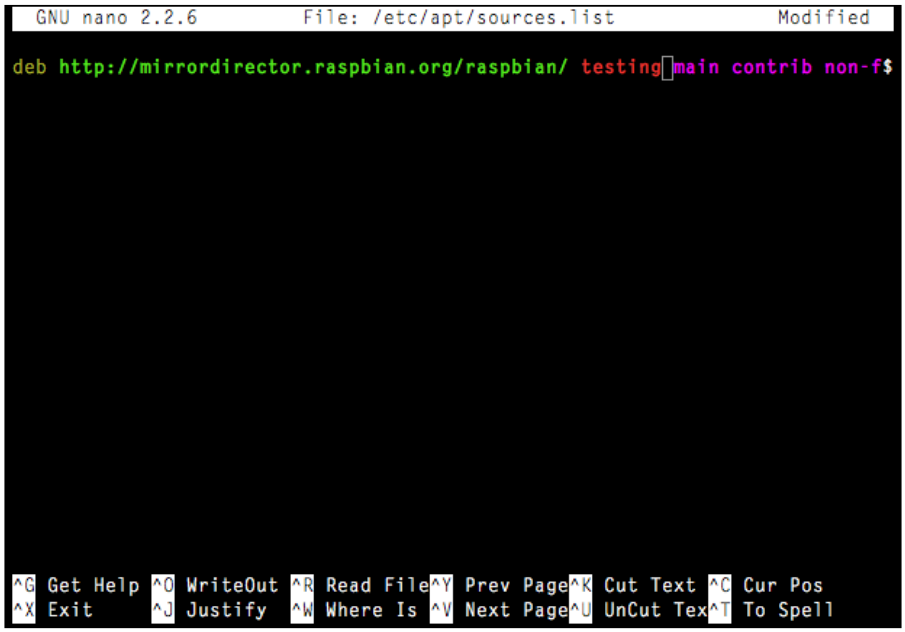
The configuration files under `/etc` may be edited by using the `sudo` command together with a text-editing command, such as `nano`.

```
pi@raspberrypi ~ $ sudo nano /etc/apt/sources.list
```

In the preceding screenshot, the `nano` command is used to edit the Advanced Package Tool's configuration file.

There's more...

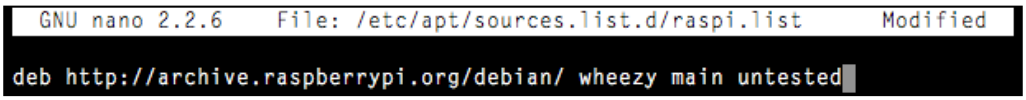
In addition to the normal named distributions, such as `wheezy`, the "official" Raspbian Linux distribution also has a package distribution named `testing`. The `testing` distribution is, as the name implies, primarily for testing. However, this package distribution does contain the most cutting-edge versions of Raspberry Pi software available from the "official" Raspbian Linux distribution.



The preceding screenshot shows the `nano` command used to select the most cutting-edge software distribution: **testing**.

If you would like to work on the cutting edge, and are willing to update often (daily), then `testing` might be an appropriate distribution for you. If you would prefer a more stable distribution, or do not wish to update often, then the current default distribution would be better.

The Raspberry Pi foundation also distributes cutting-edge software packages in the `untested` components of the `wheezy` distribution. The packages in untested are very cutting edge and should be used with caution. Their developers have tested the packages; however, the general public has not yet tested them.



The preceding screenshot shows how `nano` is used to modify the `raspi.list` configuration file to include the untested software packages.

After changing the `apt` configuration, be sure to rerun both of the `apt-get update` and `apt-get dist-upgrade` commands so that the Raspberry Pi is updated with the new configuration.

These cutting-edge packages are being released to the general public for their help in testing and debugging them. Use at your own risk. If you do use these packages and find a bug, an error, or other software deficiency, or you would like to suggest an improvement, please post a positive, constructive report of the error or suggestion on one of the Raspberry Pi Foundation's forums. User support in debugging and improving software applications is one of the strengths of open source software distributions such as the "official" Raspbian Linux distribution.

See also

- **apt – Advance Package Tool**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=apt`

  The Debian man page for `apt`.

- **apt-get – APT package handling**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=apt-get`

  The Debian man page for `apt-get`.

- **sources.list – Package resource list for APT**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=sources.list`

  The Debian man page for `sources.list`.

- **nano – Nano's ANOther editor, an enhanced free Pico clone**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=nano`

  The Debian man page for `nano`.

- **Raspberry Pi Foundation User Forums**

  `http://www.raspberrypi.org/phpBB3`

  The Raspberry Pi Foundation has a number of active forums on a variety of topics.

- **Raspbian Bug Reports**

  `http://www.raspbian.org/RaspbianBugs`

  This is where to post any bugs you might discover in the Raspbian operating system.

Searching for the software packages (apt-cache)

This recipe shows how to search for the software packages using `apt-cache`.

The "official" Raspbian Linux distribution contains a large number of prebuilt software packages, ready to be downloaded and installed. The distributed software packages cover a wide range of uses including utilities for administering and securing the Raspberry Pi. Using the `apt-cache` command, this large repository can be searched for keywords.

Once you've completed this recipe, you will be able locate software packages by keyword using the `apt-cache search` command.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

The Raspberry Pi will need access to the Internet to reach the update server(s) from which it will pull the software updates.

How to do it...

The steps for searching the Raspbian Linux software packages are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   `apt-cache search fortune –names-only`

   Searching the cache does not require privileges.

```
pi@raspberrypi ~ $ apt-cache search fortune --names-only
fortune-mod - provides fortune cookies on demand
fortune-zh - Chinese Data files for fortune
fortunes - Data files containing fortune cookies
fortunes-bg - Bulgarian data files for fortune
fortunes-bofh-excuses - BOFH excuses for fortune
fortunes-br - Data files with fortune cookies in Portuguese
```

The preceding screenshot shows how the `apt-cache` command is used to search for packages with **fortune** in their names.

3. The command displays a list of packages with "fortune" in the package name.

How it works...

In this recipe, the configured Raspberry Pi distributions are searched for packages that have the keyword "fortune" in their package name.

When `apt-cache` command is run without the option `-names-only`, the command summary is also searched.

Of the packages found, the package **fortune-mod** is a command-line utility that provides fortune cookies on demand. The packages that begin with **fortunes-** are the collections of fortunes that the command-line utility selects from when displaying a fortune cookie on demand.

Instructions on how to install the `fortune` package are in the next recipe.

See also

- **apt-cache – APT cache manipulator**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=apt-cache`

  The Debian man page for apt-cache.

- **fortune – sample lines from a file**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=fortune`

Installing a package (apt-get)

This recipe shows how the `apt-get install` command is used to install the `fortune-mod` package.

The `apt-get` command is not only used to `update` the package cache and `dist-upgrade` the packages installed on the Raspberry Pi. The `apt-get` command is also used to `install` new packages.

After completing this recipe, you will be able to install software packages using the `apt-get install` command.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

The Raspberry Pi will need access to the Internet to reach the update server(s) from which it will pull the new software packages.

How to do it...

The steps for installing a software package are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   apt-get install fortune-mod
   ```

   This command is privileged and needs to be run as a privileged user. Use `sudo` as a prefix to run the command as a privileged user.

```
pi@raspberrypi ~ $ sudo apt-get install fortune-mod
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  fortunes-min librecode0
Suggested packages:
  fortunes
The following NEW packages will be installed:
  fortune-mod fortunes-min librecode0
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 832 kB of archives.
After this operation, 1,664 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

In the preceding screenshot, the `apt-get install` command is used to install the `fortunes-mod` package.

3. The `apt-get install` command calculates the package dependencies and the additional space that will be used and then the user is prompted, **Do you want to continue [Y/n]?**.
4. Press the *return* key to accept the default of **Y** to continue.

```
Get:1 http://mirrordirector.raspbian.org/raspbian/ testing/main librecode0 a
rmhf 3.6-20 [708 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian/ testing/main fortune-mod
armhf 1:1.99.1-4 [50.9 kB]
Get:3 http://mirrordirector.raspbian.org/raspbian/ testing/main fortunes-min
 all 1:1.99.1-4 [73.0 kB]
Fetched 832 kB in 2s (337 kB/s)
Selecting previously unselected package librecode0:armhf.
(Reading database ... 57709 files and directories currently installed.)
Unpacking librecode0:armhf (from .../librecode0_3.6-20_armhf.deb) ...
Selecting previously unselected package fortune-mod.
Unpacking fortune-mod (from .../fortune-mod_1%3a1.99.1-4_armhf.deb) ...
Selecting previously unselected package fortunes-min.
Unpacking fortunes-min (from .../fortunes-min_1%3a1.99.1-4_all.deb) ...
Processing triggers for man-db ...
Setting up librecode0:armhf (3.6-20) ...
Setting up fortune-mod (1:1.99.1-4) ...
Setting up fortunes-min (1:1.99.1-4) ...
pi@raspberrypi ~ $
```

The preceding screenshot shows the `apt-get install` command after it has finished installing the `fortunes-mod` package.

5. When told to continue, the command downloads the selected package and its dependencies, unpacks them, and then sets them up.

   Installation of the `fortune-mod` package is now complete.

```
pi@raspberrypi ~ $ fortune
To be or not to be.
                -- Shakespeare
To do is to be.
                -- Nietzsche
To be is to do.
                -- Sartre
Do be do be do.
                -- Sinatra
pi@raspberrypi ~ $
```

The preceding screenshot shows the newly installed `fortune` command in action.

How it works...

The `apt-get install` command uses the same cached package information as the `apt-cache search` command. Both commands rely on `apt-get update` to download the updated package information. It is a good idea to both update the package cache and to run `apt-get dist-upgrade` to update the currently installed packages, before installing any new packages.

By default, `apt-get install` will not only download and install the selected package(s); the command will also automatically download and install any dependent packages. If more packages need to be installed than the one(s) selected by the user, a list of packages is displayed together with their total disk space and the user is asked if the installation should continue.

When the installation continues, the new package and all dependent packages are first downloaded and then set up (installed). Unless documented elsewhere, commands installed with `apt-get install` are available for use immediately after the command is successfully completed.

Package management (aptitude)

This recipe uses the `aptitude` frontend to find and install the `pianobar` application.

There are a number of frontends to the Advance Package Tool (`apt`) that provide a more feature-rich user interface. Behind the scenes, these frontends still call the `apt-get` command and the `apt-cache` command; however, they combine the functionality of all of the `apt` tools into a single user interface.

After completing this recipe, you will be able to use the `aptitude` application to find and install software packages.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

The Raspberry Pi will need access to the Internet to reach the update server(s) from which it will pull new software packages.

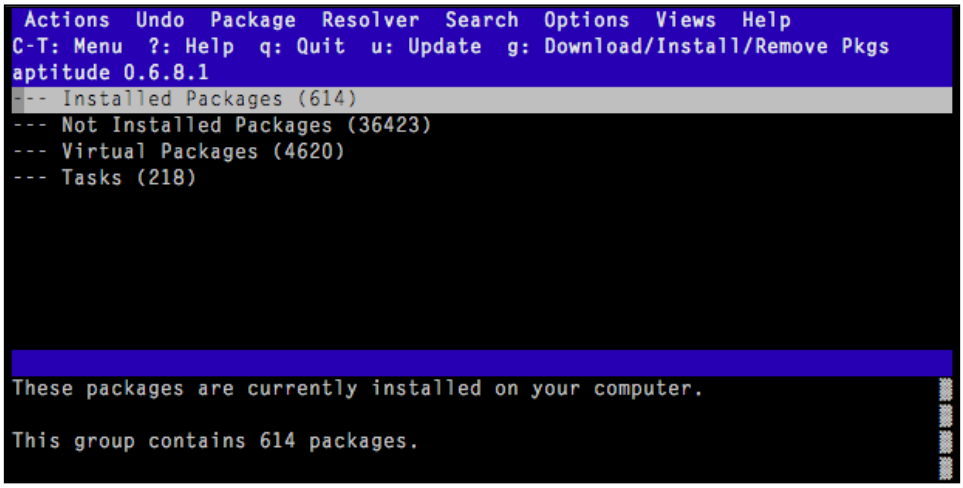
How to do it...

The steps for managing software packages with `aptitude` are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

```
aptitude
```

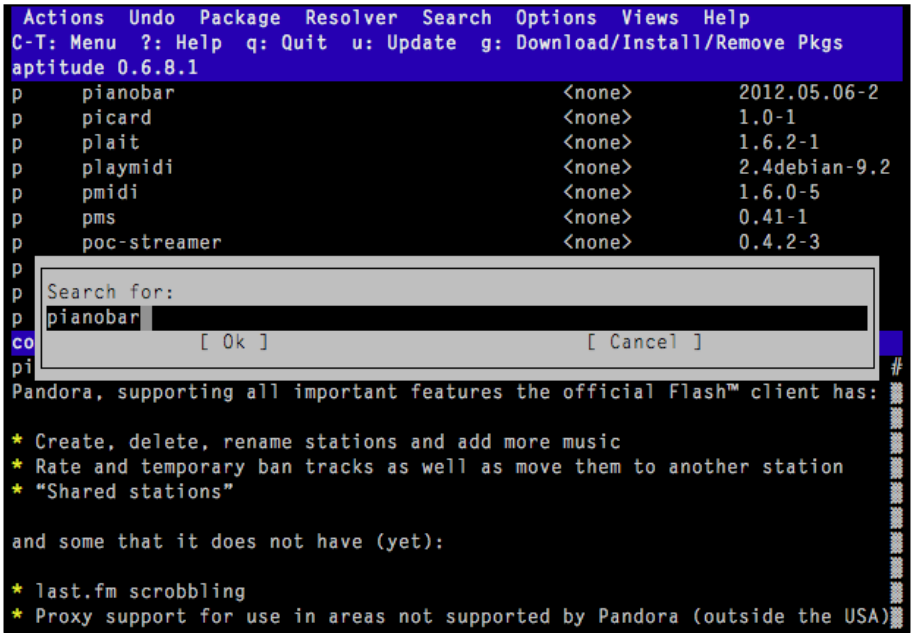
This command is privileged and needs to be run as a privileged user. Use `sudo` to run the command as a privileged user.



The preceding screenshot shows the main screen of the `aptitude` command that is a frontend for the Advance Package Tool (`apt`).

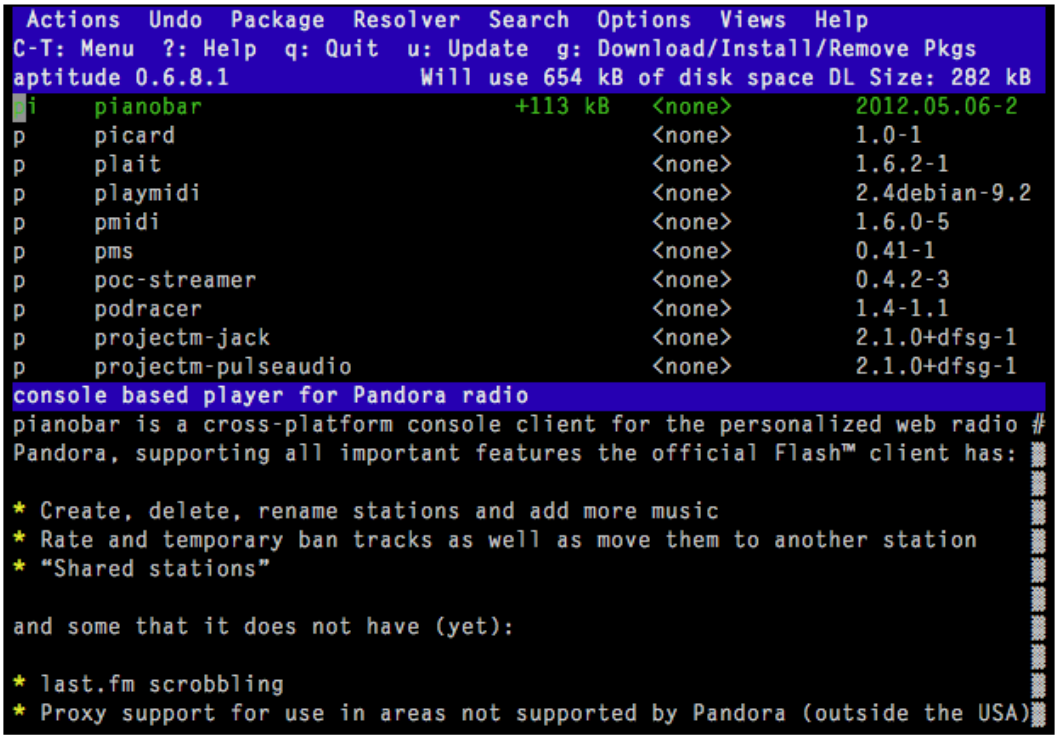
3. Type `/pianobar` to search for the `pianobar` package.

   Press *return* to go to the first package that is found.



The preceding screenshot shows the `aptitude` search dialog—searching for the package: **pianobar**

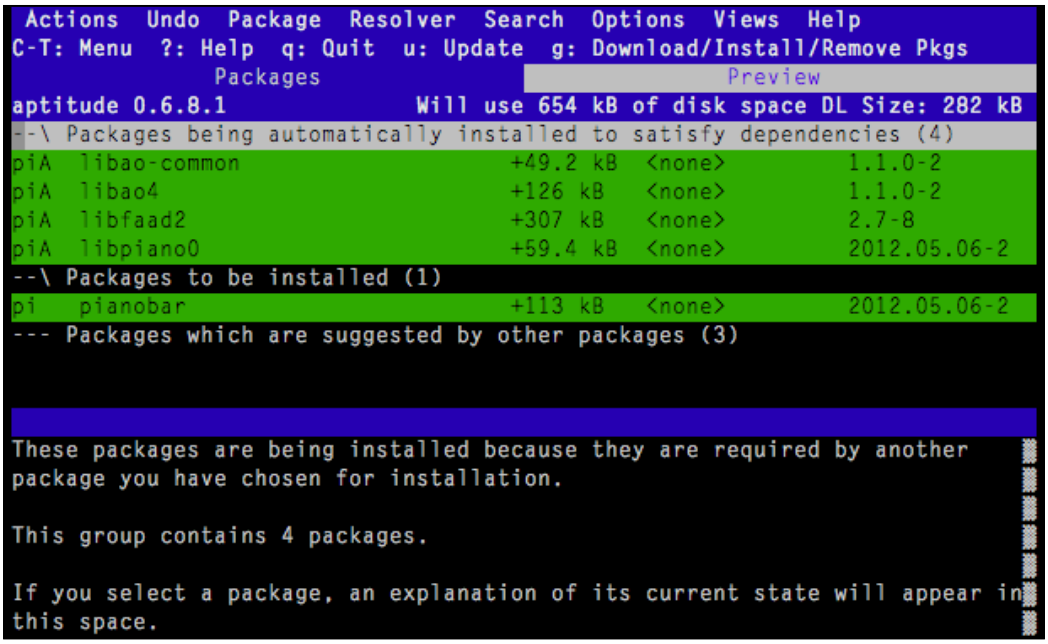
4. Press **+** to select the package for installation.



The preceding screenshot shows that the `pianobar` package is selected and ready for installation.

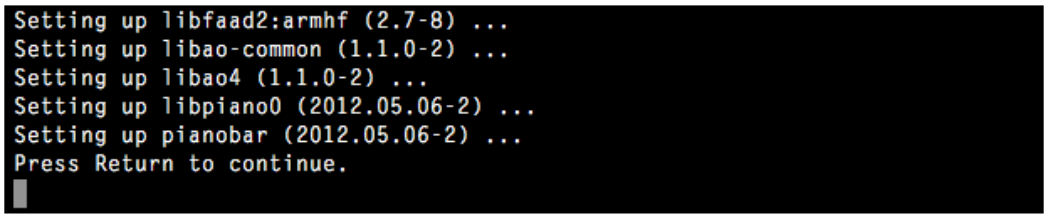
5. Press **g** to preview the installation of the selected package(s).

   The list of dependent packages is displayed.



   The preceding screenshot shows how the `aptitude` command calculates the dependencies of the `pianobar` package.

6. Press **g** a second time to complete the installation.
7. The `aptitude` command clears the screen and then runs the `apt-get install` command. Press *return* after the installation is complete to return to `aptitude`.



   The preceding screenshot shows the `aptitude` command after it has finished running the `apt-get install`. Press *return* to continue.

8. After returning to `aptitude`, press `q` and then `y` to quit.

How it works...

The `aptitude` application combines the functionality of the entire suite of Advanced Package Tools into one user interface. From `aptitude`, the user can perform the following tasks:

- Update the package cache (press *u* lowercase-u)
- Select updated packages (press *U* uppercase-U)
- Search packages (press */* slash)
- Select packages (press + plus to select, press – minus to unselect)
- Install selected packages (press *g* lowercase-g)

The direction keys are used to navigate the packages, and *q* is used to quit.

Full instructions can be found under the **Help** menu or by pressing *?*. Press *Ctrl* + *T* to activate the menus.

Although not always necessary, rebooting after an install is always a good idea.

There's more...

The `aptitude` application is one of many frontends for managing packages.

See also

- **aptitude – high-level interface to the package manager**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=aptitude`

  The Debian man page for `aptitude`.

- **pianobar – personalized online radio client**

  `http://6xq.net/projects/pianobar/`

  The `pianobar` application is an open source, console-based client for Pandora.

Reading the built-in documentation (man)

This recipe shows how to read the man page of the `fortune` command using the `man` command.

The "official" Raspbian Linux distribution comes with a large amount of documentation built into the system. There are three primary sources of built-in documentation: man pages, Info documents, and the `/usr/share/docs` directory.

After completing this recipe, you will able to read the man page documentation built in for the Raspberry Pi.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card

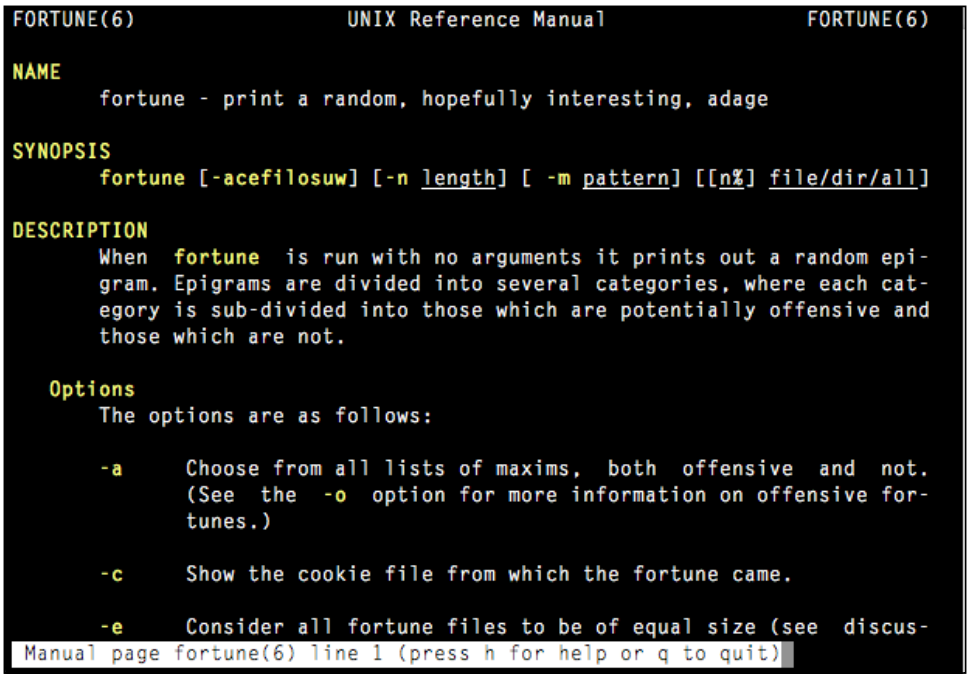
If the Raspberry Pi is connected to a network, and its Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

How to do it...

The steps for reading the built-in documentation using `man` are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

```
man fortune
```



The preceding screenshot shows the man page for the `fortune` command.

3. Press *Space* to page through the manual.

   Press *h* for a list of the key commands used for reading and searching.

   Press *q* to quit.

How it works...

Most command-line utilities have a man page. There are also man pages for configuration files and software libraries. All the man pages are accessible using the `man` command.

For more information, use the `man` command to read its own man page as follows:

```
man man
```

There's more...

It is also possible to search the library of man pages using the `apropos` command (or the `-k` option of the `man` command). The `apropos` command searches the man pages database for keywords and returns a list of man pages.

```
pi@raspberrypi ~ $ man -k music
pianobar (1)         - console pandora.com music player
pi@raspberrypi ~ $
```

In the preceding screenshot, the apropos command (`man -k`) found the man page for pianobar when searching for the keyword `music`.

See also

- **apropos - search the manual page names and descriptions**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=apropos`

  The Debian man page for `apropos`.

- **man – an interface to the online reference manuals**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=man`

  The Debian man page for `man`.

Reading the built-in documentation (info)

This recipe shows how to read the `info` file documentation of the `nano` command.

Another source of documentation installed with the "official" Raspbian Linux distribution is the `info` command. The `info` command is used to display documentation in the info file format.

After completing this recipe, you will be able to read the documentation built in to the Raspbian Linux distribution that has been stored in the `info` file format.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card

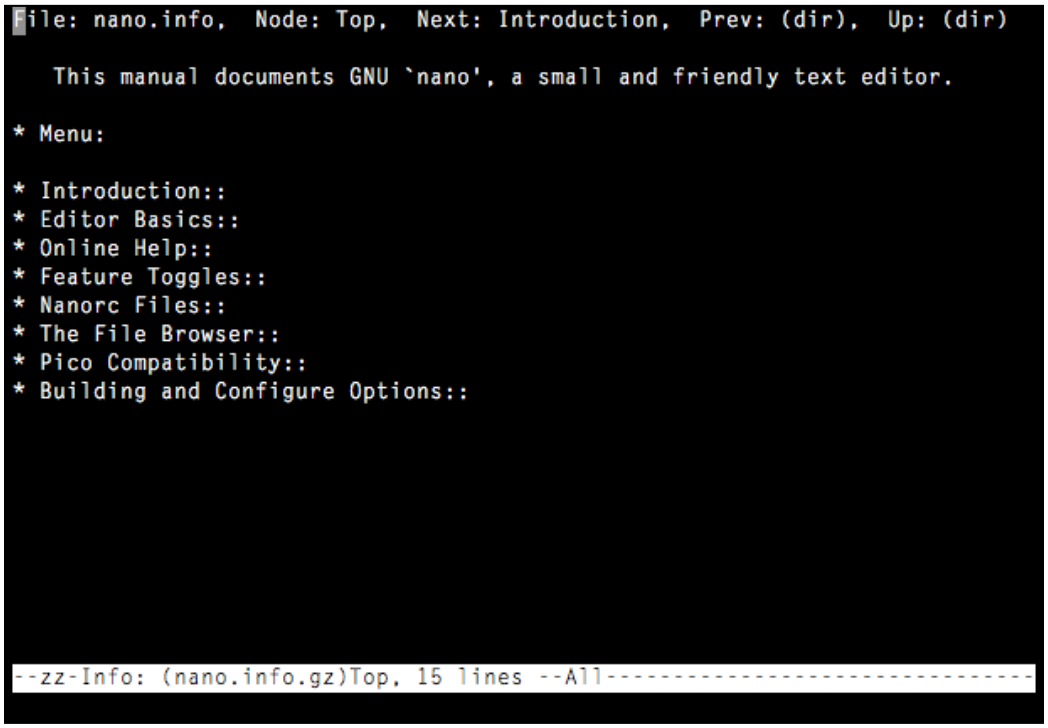
If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

How to do it...

The steps for reading the built-in documentation using `info` are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the command:

```
info nano
```



The preceding screenshot shows the top of the `info` file for the nano text editor.

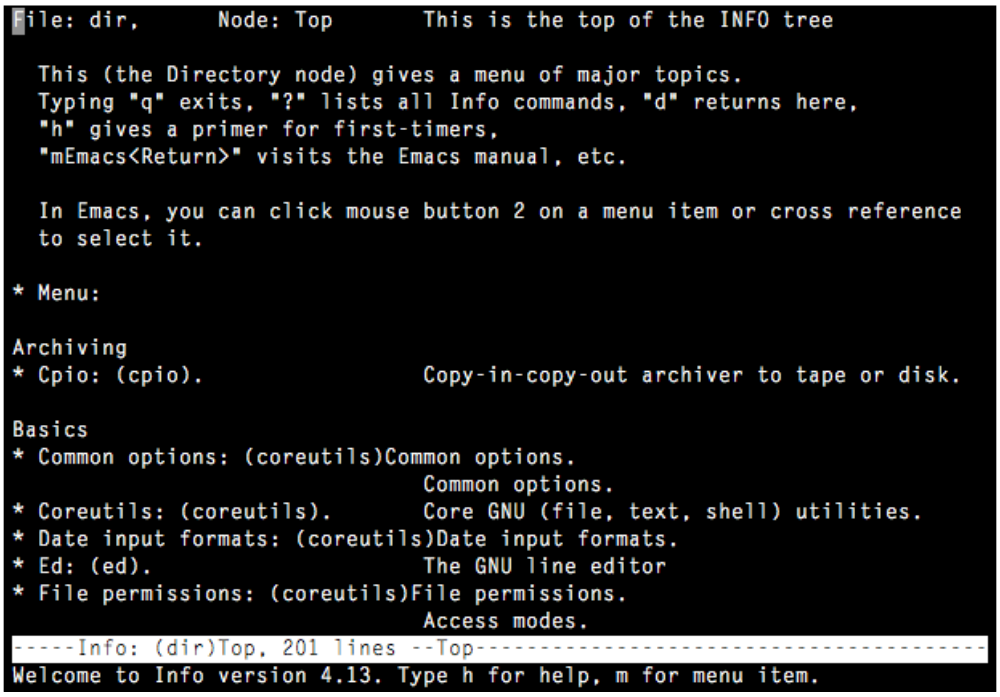
3. Press *Space* to page through the file.

   This is the easiest way to read the file from top to bottom.

4. Use the direction keys to select a menu item and then press *return* to jump to that location in the file.
5. Press *h* to learn more about the `info` command.
6. Press *q* to quit.

How it works...

The `info` command reads specially formatted Info files and displays them in a text browser. Info files are well structured with menu hierarchies for navigating documentation by concepts. The root of all Info files is displayed when the `info` command is entered without parameters: `info`



The preceding screenshot shows the root node of the `info` file documentation.

The `Coreutils` section of the Info documentation covers the most common command-line utilities used in the "official" Raspbian Linux distribution. In the Info files, there is also documentation on file permissions, access modes, and other concepts useful for managing the Raspberry Pi.

The Info file documentation goes beyond just a summary and list of options to also explain the concepts behind the commands and their intended use. Ironically, the Info file for the `info` command is not included with the "official" Raspbian Linux distribution. Instead, use the command `man info` to read more about the `info` command.

There's more...

The Info pages can also be searched using the `-k` option of the `info` command. Try searching (`-k`) both `man` and `info` for keywords such as permission, access, or download when you cannot remember the name of a command.

```
pi@raspberrypi ~ $ info -k access
"(coreutils)Access permission tests" -- access permission tests
"(coreutils)chmod invocation" -- access permissions, changing
"(coreutils)dd invocation" -- access time
"(coreutils)touch invocation" -- access time, changing
"(coreutils)Sorting the output" -- access time, printing or sorting files by
"(coreutils)du invocation" -- access time, show the most recent
"(coreutils)chmod invocation" -- changing access permissions
"(coreutils)chmod invocation" -- permissions, changing access
"(coreutils)chmod invocation" -- recursively changing access permissions
pi@raspberrypi ~ $
```

The preceding screenshot shows the result of using `info -k` to search for documentation on access permissions.

See also

- **info – read Info documents**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=info`

  The Debian man page for `info`.

4
File Sharing

In this chapter we will cover:

- Mounting USB drives (pmount)
- Sharing folders from other computers (mount.cifs)
- Automounting USB disks at boot (/etc/fstab)
- Automounting a shared folder at boot
- Creating a file server (Samba)
- Sharing an attached USB disk via Samba
- Accessing another computer's files (smbclient)

Introduction

The Raspberry Pi recipes in this chapter are for sharing files with other computers on the same local network.

The chapter begins with recipes for mounting USB disks attached to the Raspberry Pi. The next recipes show how to exchange files with other computers using the SMB (CIFS) protocol. The chapter ends with a recipe for setting up the Raspberry Pi as a file server.

After completing the recipes in this chapter, you will be able to exchange files between the Raspberry Pi and other computers using USB disks and the SMB (CIFS) protocol.

Mounting USB drives (pmount)

This recipe installs and applies `pmount`, a command that mounts USB disks attached directly to the Raspberry Pi in the same manner as the desktop file manager.

The Raspberry Pi desktop (GUI) file manager will ask the desktop user if a USB disk should be mounted as it is connected to the Raspberry Pi. However, this recipe does not rely on the Raspberry Pi desktop or the file manager. Instead, the command-line utility `pmount` is installed and then used to mount USB.

Once you finish with this recipe, you will be able to use both large USB storage devices and small USB flash drives to exchange files with other computers without relying on the Raspberry Pi desktop file manager.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A powered USB hub (recommended)
- At least one USB disk drive

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be completed remotely using a Secure Shell client (see *Chapter 2*, *Administration*).

The examples in this recipe use two USB drives – a 32-GB flash drive and a 500-GB disk.

How to do it...

The following are the steps to mount USB disks:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   apt-get install pmount
   ```

   This command needs to be run as a privileged user (use `sudo`).

The package management application `aptitude` could be used as an alternative to the command utility `apt-get` for downloading and installing `pmount`.

```
pi@raspberrypi ~ $ sudo apt-get install pmount
```

The preceding screenshot shows the `apt-get` command being used to install the `pmount` software package.

3. The `apt-get install` command downloads and installs `pmount`.
4. Connect one or more USB disks to the Raspberry Pi. The example includes two disks – a 32-GB flash drive and a 500-GB disk.
5. Execute the following command:

   `fdisk -l`

   This command also needs to be run as a privileged user (use `sudo`).

```
pi@raspberrypi $ sudo fdisk -l

Disk /dev/mmcblk0: 4025 MB, 4025483264 bytes
4 heads, 16 sectors/track, 122848 cylinders, total 7862272 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000dbfc6

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880     7862271     3869696   83  Linux

Disk /dev/sda: 31.5 GB, 31457280000 bytes
64 heads, 32 sectors/track, 30000 cylinders, total 61440000 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x6f20736b

This doesn't look like a partition table
Probably you selected the wrong device.

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   ?   778135908  1919645538   570754815+  72  Unknown
/dev/sda2   ?   168689522  2104717761   968014120   65  Novell Netware 386
/dev/sda3   ?  1869881465  3805909656   968014096   79  Unknown
/dev/sda4   ?  2885681152  2885736650       27749+   d  Unknown

Partition table entries are not in disk order

Disk /dev/sdb: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x002317d5

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048   976773167   488385560    7  HPFS/NTFS/exFAT
pi@raspberrypi $
```

In the preceding screenshot, the `fdisk` command is used to display information about the disk drives attached to the Raspberry Pi.

6. The `fdisk` command displays information about the disks that are attached to the Raspberry Pi. The example has three disks, as follows:
   - The 4-GB SD card (the boot disk), which has two partitions – `/dev/mmcblk0p1` and `/dev/mmcblk0p2`
   - The 32-GB flash drive is located at `/dev/sda` and has no partitions
   - The 500-GB disk has its primary partition located at `/dev/sdb1`
7. Execute the following command:

   `pmount /dev/sda`

   Mount the 32-GB flash drive. This is not a privileged command.
8. Execute the following command:

   `pmount /dev/sdb1`

   Mount the 500-GB disk drive.

```
pi@raspberrypi ~ $ pmount /dev/sda
pi@raspberrypi ~ $ pmount /dev/sdb1
pi@raspberrypi ~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          3.7G  1.5G  2.1G  42% /
/dev/root       3.7G  1.5G  2.1G  42% /
devtmpfs        117M     0  117M   0% /dev
tmpfs            24M  236K   23M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            47M     0   47M   0% /run/shm
/dev/mmcblk0p1   56M   34M   23M  60% /boot
/dev/sda         30G   26G  4.1G  87% /media/sda
/dev/sdb1       459G  260G  176G  60% /media/sdb1
pi@raspberrypi ~ $ ls -l /media
total 20
drwx------ 10 pi pi 16384 Dec 31  1969 sda
drwxr-xr-x 20 pi pi  4096 Jun 28 19:46 sdb1
pi@raspberrypi ~ $
```

In the preceding screenshot, the `pmount` command is used to mount two disks.

9. Execute the following command:

   `df -h`

   Display the mounted disks and their mount points.

10. Execute the following command:

    ```
    ls –l /media
    ```

    List the contents of the directory where disks are automatically mounted.

11. The 32-GB flash drive (`/dev/sda`) is mounted at `/media/sda`. The only partition of the 500-GB drive is mounted at `/media/sdb1`.

How it works...

After `pmount` is installed and the USB disks have been attached, the `fdisk –l` command is used to list the partition tables of the disks attached to the Raspberry Pi.

In the example, the 32-GB flash drive does not have a partition table, so the `fdisk` utility erroneously displays four partitions. However, the utility does warn the user with the message, **This doesn't look like a partition table**.

The 500-GB disk does have a partition table. The `fdisk` utility correctly displays partition information for the disk's only partition, `/dev/sdb1`.

For each disk drive discovered during boot, a device file is created for the disk and stored in the `/dev` directory. The names of each of these device files are assigned sequentially as the disks are discovered.

The assigned names of USB drives all begin with the characters `sd`. The first disk drive is assigned the device file `/dev/sda`, the second `/dev/sdb`, the third `/dev/sdc`, and so on.

The assigned names of the SD card drives begin with `mmcblk`, and a digit is added for each device discovered starting at zero (0). The only SD card drive in the Raspberry Pi (the boot disk) has its device file located at `/dev/mmcblk0`.

For each disk partition discovered in a disk's partition table, there is also a device file created. Each of these disk partition device files has the same name as the disk's device file with a digit attached corresponding to its location in the partition table.

In the example, the 4-GB SD card is assigned the device file `/dev/mmcblk0`. It has two partitions – `/dev/mmcblk0p1` and `/dev/mmcblk0p2`. If the SD card had yet another partition, the third partition's device file would be `/dev/mmcblk0p3`.

When a disk has no partition table, use the `pmount` command with the device file assigned to the disk drive. In the example, the 32-GB flash drive has no partition, so the command `pmount /dev/sda` is used to mount the disk.

When a disk has a partition table, each partition needs to be mounted separately. Use the `pmount` command with the partition's device file and not the disk device file. The 500-GB disk in the example has one partition that is mounted with the command `pmount /dev/sdb1`.

When the disks are mounted, a mount point is created for the disk in the `/media` directory. By default the name of the mount point is the same as the name of the device file used to mount the disk (or disk partition). The 32-GB flash drive was assigned the device file `/dev/sda`, so by default its mount point is `/media/sda`.

To create a mount point filename under the `/media` directory other than the default device filename, pass the desired filename as a second parameter, as shown in the following command:

```
pmount /dev/sdb1 mydisk
```

```
pi@raspberrypi ~ $ pmount /dev/sdb1 mydisk
pi@raspberrypi ~ $ ls -l /media
total 20
drwxr-xr-x 20 pi pi  4096 Jun 28 19:46 mydisk
drwx------ 10 pi pi 16384 Dec 31  1969 sda
pi@raspberrypi ~ $
```

The preceding screenshot shows that the primary partition of the 500-GB disk is mounted at `/media/mydisk`.

Use the `pumount` command to unmount the disks (there is no *n* in the command name). After the disks are unmounted, their mount points are deleted from the `/media` directory and from the `/etc/fstab` file.

```
pi@raspberrypi ~ $ pumount /dev/sda
pi@raspberrypi ~ $ pumount /dev/sdb1
pi@raspberrypi ~ $ ls /media
pi@raspberrypi ~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          3.7G  1.5G  2.1G  42% /
/dev/root       3.7G  1.5G  2.1G  42% /
devtmpfs        117M     0  117M   0% /dev
tmpfs            24M  236K   23M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            47M     0   47M   0% /run/shm
/dev/mmcblk0p1   56M   34M   23M  60% /boot
pi@raspberrypi ~ $
```

The preceding screenshot shows how the `pumount` command is used to unmount both disks. The `ls` and `df` commands are used to show that the disks are no longer mounted.

`pmount` is not a privileged command. Its purpose is to enable non-privileged users to mount removable disks. The `mount` command is privileged. During the boot process, the `mount` command is used to automatically mount disks, for example the SD card from which the Raspberry Pi boots. The next recipe shows how to configure the Raspberry Pi to automatically mount attached drives when it boots.

The file manager included with Raspberry Pi desktop, PCManFM, will also mount disks in the `/media` directory. However, the PCManFM file manager uses the disk's label instead of the disk's name as the name of the mount point. An example recipe is found in the next chapter.

There's more...

The typical class 6 SD card has a transfer rate of 6 MB per second, whereas the transfer rate of a typical 7,200-RPM external hard drive is more than 1 GB/sec. The typical external USB disk is almost 20 times faster than an SD card!

Even though SD cards can be purchased in increasingly larger sizes, they are not the best choice for high-speed performance. The recipes in this chapter will perform better if the files that they are using are stored on an external drive.

External disks and flash drives that have no power supply of their own draw power from the USB connection. The power needed by these disk drives will be taken from the Raspberry Pi, if they are directly connected to the Raspberry Pi.

The amount of power needed by external USB devices could easily exceed the amount of power that is available from the Raspberry Pi. If too much power is drawn from the Raspberry Pi, other USB devices including the network interface (which internally uses the USB bus) may cease to function properly or they may cease to function at all.

For best performance and reliability, it is recommended that all external devices, including external disk drives, should be connected to the Raspberry Pi via a powered USB hub instead of being connected to the Raspberry Pi directly.

See also

- **Hard disk drive**

  `http://en.wikipedia.org/wiki/Hard_disk_drive`

  A Wikipedia article about disk drives.

- **USB flash drive**

  `http://en.wikipedia.org/wiki/USB_flash_drive`

  A Wikipedia article that provides detailed information about USB drives.

- **Secure digital (SD) cards**

  `http://en.wikipedia.org/wiki/Secure_Digital`

  A Wikipedia article about the **Secure Digital** (**SD**) card format.

- **mount (Unix)**

  `http://en.wikipedia.org/wiki/Mount_(Unix)`

  A Wikipedia article about the `mount` command.

- **df – report file system disk space usage**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=df`

  The Debian man page for `df`.

- **fdisk – partition table manipulator for Linux**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=fdisk`

  The Debian man page for `fdisk`.

- **ls – list directory contents**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=ls`

  The Debian man page for `ls`.

- **pmount – mount arbitrary hot pluggable devices as normal user**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=pmount`

  The Debian man page for `pmount`.

- **pumount – unmount arbitrary hot pluggable devices**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=pumount`

  The Debian man page for `pumount`.

- **udev – Linux dynamic device management**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=udev`

  The Debian man page for `udev`.

Sharing folders from other computers (mount.cifs)

This recipe shows how to mount folders that have been shared on another computer.

Recent Linux kernels, including the "official" Raspbian Linux distribution for the Raspberry Pi, have built-in support for mounting shared folders using the SMB (CIFS) protocol. This is the file-sharing protocol used commonly by Windows computers.

This is the simplest recipe for mounting a shared folder from the command line.

Once you've completed this recipe, you will be able to share files with other computers on the same network using the SMB (CIFS) protocol.

SMB and CIFS are synonyms for the same file-sharing protocol.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- Another PC connected to the same network as the Raspberry Pi that has one or more open file shares

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be remotely completed using a Secure Shell client (see *Chapter 2*, *Administration*).

The examples in this recipe will connect the Raspberry Pi to a local network computer named `GOLDEN-XP` that is running Windows XP and is sharing one folder, `XFER`. The shared folder is configured for guest access. Configuration for other computers using the SMB (CIFS) protocol will be similar.

How to do it...

The following are the steps for sharing folders from other computers:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

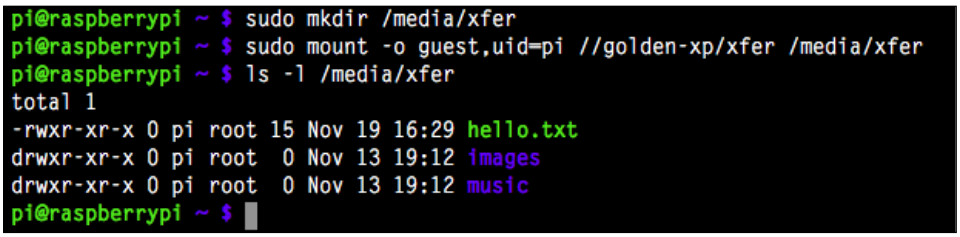
   ```
   mkdir /media/xfer
   ```

   Create a mount point for the shared folder. The `/media` directory is protected (use `sudo`).

3. Execute the following command:

   ```
   sudo mount -o guest,uid=pi //golden-xp/xfer /media/xfer
   ```

   Mount the shared folder on the newly created mount point. The `mount` command is protected (use `sudo`).



   In the preceding screenshot, a shared folder named `xfer` is mounted from the computer `golden-xp` at the mount point `/media/xfer`.

4. Execute the following command:

   ```
   ls –l /media/xfer
   ```

   List the files in the shared folder.

How it works...

First the `mkdir` command is used to create a mount point for the shared folder in the `/media` directory. The mount point is an empty directory and serves as a placeholder in the filesystem. The `/media` directory is protected, so the `sudo` command needs to be used as a command prefix to temporarily grant system privileges to the user `pi`.

Then, the shared folder `xfer` from the computer `golden-xp` is mounted at the newly created mount point. The `mount` command is privileged, so the `sudo` command is used as a command prefix. The options to the `mount` command `(-o guest,uid=pi)` tell the command that the share is to be accessed without a username or password (`guest`) and that the mounted files will belong to the user `pi` (`uid=pi`). The source computer (`golden-xp`) and the share (`xfer`) are specified followed by the target mount point (`/media/xfer`).

Finally, the files in the mount point are listed using the command `ls -l` to verify that they are indeed identical to the files in the Windows share (note – the reported file sizes may be slightly different between the two computers).

There's more...

The example uses a Windows share that does not require a username or a password. If a Windows share is password protected, then the options on the `mount` command need to be expanded to include `username=USER,password=PASS`. `USER` and `PASS` should be replaced with the respective username and password of a user who has access to the Windows share.

```
pi@raspberrypi ~ $ sudo mount -o uid=pi,user=golden,password=pass //golden-x
p/C$ /media/xfer
pi@raspberrypi ~ $ ls -l /media/xfer
total 1573161
-rwxr-xr-x 0 pi root          0 Nov  1 02:34 AUTOEXEC.BAT
-rwxr-xr-x 0 pi root        211 Nov  1 01:18 boot.ini
drwxr-xr-x 0 pi root          0 Nov 14 11:15 Config.Msi
-rwxr-xr-x 0 pi root          0 Nov  1 02:34 CONFIG.SYS
drwxr-xr-x 0 pi root          0 Nov  1 01:41 Documents and Settings
drwxr-xr-x 0 pi root          0 Nov  4 02:42 ea19e4d36fb3ce40c1ac12
-r-xr-xr-x 0 pi root          0 Nov  1 02:34 IO.SYS
-r-xr-xr-x 0 pi root          0 Nov  1 02:34 MSDOS.SYS
-r-xr-xr-x 0 pi root      47564 Feb 28  2006 NTDETECT.COM
-r-xr-xr-x 0 pi root     250048 Nov  1 04:47 ntldr
-rwxr-xr-x 0 pi root 1610612736 Nov 25 16:29 pagefile.sys
dr-xr-xr-x 0 pi root          0 Nov 14 00:28 Program Files
drwxr-xr-x 0 pi root          0 Nov  1 01:52 RECYCLER
drwxr-xr-x 0 pi root          0 Nov  4 21:56 System Volume Information
drwxr-xr-x 0 pi root          0 Nov 14 22:50 WINDOWS
drwxr-xr-x 0 pi root          0 Nov 25 16:53 xfer
pi@raspberrypi ~ $
```

The preceding screenshot shows how to mount the `C:` drive of a Windows computer (`//golden-xp/C$`) using a username (`golden`) and password (`pass`). Once mounted, the files will belong to the user `pi` (`uid=pi`).

Use the `umount` command to unmount the disks (there is no *n* in the command name). After the disks are unmounted, the mount points remain under the `/media` directory; however, they are once again empty. The `umount` command is privileged (use `sudo`).

```
pi@raspberrypi ~ $ sudo umount /media/xfer
pi@raspberrypi ~ $ ls -l /media/xfer
total 0
pi@raspberrypi ~ $
```

The preceding screenshot shows how to unmount (`umount`) the Windows share from the mount point at `/media/xfer`.

See also

- **Filesystem**

  `http://en.wikipedia.org/wiki/Filesystem`

  A Wikipedia article on filesystems.

- **Server Message Block (SMB)**

  `http://en.wikipedia.org/wiki/Server_Message_Block`

  A Wikipedia article on the **Server Message Block** (**SMB**) protocol.

- **Uniform Naming Convention (UNC)**

  `http://en.wikipedia.org/wiki/Uniform_Naming_Convention`

  A Wikipedia article on the **Uniform Naming Convention** (**UNC**) protocol.

- **mount – mount a filesystem**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=mount`

  The Debian man page for `mount`.

- **Mount.cifs – mount using the CIFS filesystem**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=mount.cifs`

  The Debian man page for `mount.cifs`.

- **umount – unmounts file systems**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=umount`

  The Debian man page for `umount`.

Automounting USB disks at boot (/etc/fstab)

This recipe shows how to configure the Raspberry Pi so that it automatically mounts disk drives during the boot process.

The goal of this recipe is to mount the example disks at boot time with the same configuration that is used by the `pmount` command.

The first few steps of this recipe use the `pmount` and `mount` commands to discover the correct filesystem table configuration parameters for two example disk drives. Once the configuration parameters are known, the filesystem table configuration file (`fstab`) is updated so that the example disks are mounted as the Raspberry Pi boots.

This recipe does not provide details on the meaning of each configuration parameter. Instead, the recipe reuses the same configuration that is used by the `pmount` command. More detail on the configuration parameters can be found in the man pages for the `mount` command.

After completing this recipe, the Raspberry Pi will mount USB disk drives at boot time with a configuration that is consistent with the `pmount` utility.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A powered USB hub (recommended)
- At least one USB disk drive

For continuous use with the Raspberry Pi, USB disks should have their own power supply, or be attached indirectly to the Raspberry Pi using a powered USB hub to ensure that there is enough power for the Raspberry Pi and its internal devices to operate properly.

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be completed remotely using a Secure Shell client (see *Chapter 2*, *Administration*).

The examples in this recipe use two USB drives – a 32-GB flash drive and a 500-GB disk.

The Raspberry Pi should already have the `pmount` command installed (see the previous recipe).

Be careful! A broken `/etc/fstab` may prevent the system from booting. So, make sure that you've tested the configuration before rebooting.

How to do it...

The following are the steps to automount USB disks at boot:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   pmount /dev/sda thumbdrive
   ```

   Mount the 32-GB flash drive (see recipe above).

3. Execute the following command:

   ```
   pmount /dev/sdb1 bigdisk
   ```

   Mount the 500-GB USB drive.

4. Execute the `mount` command:

```
pi@raspberrypi ~ $ pmount /dev/sda thumbdrive
pi@raspberrypi ~ $ pmount /dev/sdb1 bigdisk
pi@raspberrypi ~ $ mount
/dev/root on / type ext4 (rw,noatime,user_xattr,barrier=1,data=ordered)
devtmpfs on /dev type devtmpfs (rw,relatime,size=118872k,nr_inodes=29718,mod
e=755)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=23788k,mode=755)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=47560k)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620)
/dev/mmcblk0p1 on /boot type vfat (rw,relatime,fmask=0022,dmask=0022,codepag
e=cp437,iocharset=ascii,shortname=mixed,errors=remount-ro)
/dev/sda on /media/thumbdrive type vfat (rw,nosuid,nodev,noexec,relatime,uid
=1000,gid=1000,fmask=0177,dmask=0077,codepage=cp437,iocharset=iso8859-1,shor
tname=mixed,quiet,utf8,errors=remount-ro)
/dev/sdb1 on /media/bigdisk type ext4 (rw,nosuid,nodev,noexec,relatime,error
s=remount-ro,user_xattr,acl,barrier=1,data=ordered)
pi@raspberrypi ~ $
```

   In the preceding screenshot, the `mount` command displays the contents of the filesystem table.

5. The `mount` command displays the filesystem table including the mount parameters needed for each disk (or disk partition).
6. Execute the following command:

   ```
   cp /etc/fstab /etc/fstab.orig
   ```

   Save a copy of the original filesystem table configuration file. Files can only be created in (or copied into) the directory `/etc` by a privileged user (use `sudo`).

7. Execute the following command:

   ```
   nano /etc/fstab
   ```

   Edit the filesystem configuration table `fstab`. The `fstab` file may only be modified by a privileged user (use `sudo`).

8. Use the `nano` editor to add the configuration parameters for each disk to the bottom of the filesystem table configuration file. The parameters for each disk can be copied from the output of the `mount` command.

In the preceding screenshot, the `nano` editor is used to change the disk configuration (`/etc/fstab`).

9. For the example 32-GB flash drive the configuration parameters are as follows:
   - The device file (`/dev/sda`)
   - The mountpoint (`/media/thumbdrive`)
   - The mount options (`rw,nosuid,nodev,noexec,relatime,uid=1000,gid=1000,fmask=0177,dmask=0077,codepage=437,iocharset=iso8859-1,shortname=mixed,quiet,utf8,errors=remount-ro`)

     Notice the slight difference in syntax when entering the `codepage` option. The output of the `mount` command shows the `codepage` value `cp437`. However, the value of the `codepage` option entered in `fstab` should be `437`.
   - Dump option (`0`)
   - Boot phase (`2`)
10. For the example 500-GB USB disk the configuration parameters are as follows:
    - The device file (`/dev/sdb1`)
    - The mountpoint (`/media/bigdisk`)
    - The mount options (`rw,nosuid,nodev,noexec,relatime,errors=remount-ro,user_xattr,acl,barrier=1,data=ordered`)
    - Dump option (`0`)
    - Boot phase (`2`)
11. After entering the configuration parameters for each disk, save the file and exit the editor.
12. Execute the following command:

    `umount /media/thumbdrive`

    Unmount the 32-GB thumb drive. The `mount` command is privileged (use `sudo`).
13. Execute the following command:

    `umount /media/bigdisk`

    Unmount the 500-GB USB disk. Remember to use `sudo`.
14. Execute the following command:

    `mount -a`

    Mount all configured disks. Only a privileged user can use the `mount -a` command (use `sudo`).

15. Execute the following command:

    ```
    ls -l /media
    ```

    List the mount points of the example disks.

```
pi@raspberrypi ~ $ sudo umount /media/thumbdrive
pi@raspberrypi ~ $ sudo umount /media/bigdisk
pi@raspberrypi ~ $ sudo mount -a
pi@raspberrypi ~ $ ls -l /media
total 20
drwxr-xr-x 20 pi pi  4096 Jun 28 19:46 bigdisk
drwx------ 10 pi pi 16384 Dec 31  1969 thumbdrive
pi@raspberrypi ~ $
```

    The preceding screenshot shows how to test the filesystem table configuration.

16. Execute the `reboot` command. Reboot the system. This is a privileged command (use `sudo`).
17. The disks are now mounted automatically at boot!

How it works...

The disks are first mounted with the `pmount` command (see the previous recipe for installation instructions), and then their filesystem table configuration parameters are displayed with the `mount` command.

The `nano` editor is used to add the configuration parameters for each disk to the bottom of the filesystem table configuration file (`/etc/fstab`).

Each line of the configuration file is used to configure where one disk (or disk partition) is mounted. Each line has the following six fields separated by blanks (or tabs):

- The first field is the device file. The device file is assigned automatically to each disk and can be discovered using the `fdisk -l` command (see the previous recipe). The 32-GB thumb drive has no partitions; its mountable device file is `/dev/sda`. The 500-GB disk has one partition with the mountable device file `/dev/sdb1`.
- The second field is the mount point. Mount points can be located anywhere in the filesystem; however, current convention is to locate them in the `/media` directory. The mount point for the 32-GB flash drive is `/media/thumbdrive`. The mount point for the 500-GB disk is `/media/bigdisk`.
- The third field is the type of filesystem on the device. The 32-GB flash drive has the `vfat` filesystem, and the 500-GB drive has an `ext4` filesystem.

- The fourth field is a comma-separated list of configuration parameters. They should be copied from the parenthetical list of parameters in the output of the `mount` command. There should be no spaces in this field and no parenthesis.

One configuration parameter displayed by the `mount` command is inconsistent with the format of the `/etc/fstab` file. The `codepage` parameter, when entered in the `/etc/fstab` file, should be configured with the value `437` instead of the displayed value of `cp437`.

- The fifth field is a dump flag. It is usually zero (`0`), indicating that the filesystem does not need to be dumped. Both disks have `0` in this field.
- The sixth field selects the boot phase in which the disk will be mounted. The disk mounted as the root filesystem (`/`) is mounted in phase one; all other disks should be mounted in phase two. The example disks will not be mounted as the root filesystem, so they both have `2` in this field.

Once the filesystem table's configuration file (`fstab`) has been updated, the configuration is tested before rebooting the system.

First the disks are unmounted. Now that the disks are configured in `/etc/fstab`, the privileged command `umount` must be used to unmount them (notice that there is no *n* in the command name).

Once the disks are unmounted, the 'mount all disks' command, `mount -a`, is used to mount the example disks. If there is no error, the system can be rebooted.

```
pi@raspberrypi ~ $ sudo mount -a
mount: wrong fs type, bad option, bad superblock on /dev/sda,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so

pi@raspberrypi ~ $
```

The preceding screenshot shows that the `mount -a` command has detected an error in the filesystem table configuration file.

If the `mount` command does display an error, edit the `/etc/fstab` file again and look for typos and extra spaces. Correct any errors and try again.

If there are still errors, replace the configuration file with its original version using the following command:

```
sudo cp -f /etc/fstab.orig /etc/fstab
```

Remember that a broken `/etc/fstab` may prevent the system from booting. So, make sure that you've tested the configuration before rebooting.

There's more...

A reference to the current filesystem table is kept in the `/proc` filesystem and can be displayed with the following command:

```
cat /proc/mounts
```

See also

- **procfs**

  `http://en.wikipedia.org/wiki/Procfs`

  A Wikipedia article on the `/proc` filesystem.

- **cat – concatenate files and print on the standard output**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=cat`

  The Debian man page for `cat`.

- **cp – copy files and directories**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=cp`

  The Debian man page for `cp`.

- **fstab – static information about the filesystems**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=fstab`

  The Debian man page for `fstab`.

Automounting a shared folder at boot

The goal of this recipe is to mount a shared folder from another computer at boot time.

The previous recipe showed how to configure `/etc/fstab` for mounting USB disks. This recipe shows how a similar configuration can be used to automount a Windows share at boot time (or any other set of files shared using the SMB (CIFS) protocol).

At home or at the office, it is common for a local network to have some form of **Networked Attached Storage** (**NAS**) available to network users as shared folders using the SMB (CIFS) protocol. This recipe shows how the Raspberry Pi can be configured to automatically mount a shared folder at boot time.

After completing this recipe, a shared folder from another computer will be attached to the root filesystem every time the Raspberry Pi boots.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- A client PC connected to the same network as the Raspberry Pi

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be remotely completed using a Secure Shell client (see *Chapter 2*, *Administration*).

The examples in this recipe will connect the Raspberry Pi to a local network computer named `GOLDEN-XP`. The computer is running Windows XP and is sharing one folder, `XFER`. The shared folder is configured for guest access. Configuration for other computers using the SMB (CIFS) protocol will be similar.

How to do it...

The following are the steps for automounting shared folders at boot:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   cp /etc/fstab /etc/fstab.orig
   ```

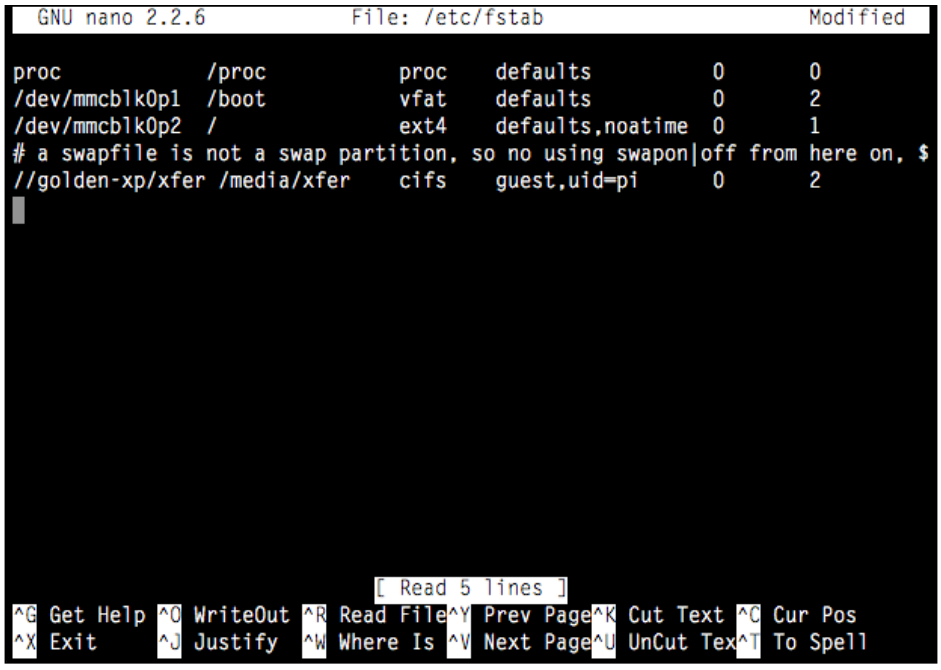
   Save a copy of the original filesystem table configuration file. Files can only be created in (or copied into) the directory `/etc` by a privileged user (use `sudo`).

3. Execute the following command:

   ```
   nano /etc/fstab
   ```

   Edit the filesystem configuration table `fstab`. The `fstab` file may only be modified by a privileged user (use `sudo`).

4. Use the `nano` editor to add the configuration parameters for the Windows shared folder (`//golden-xp/xfer`) to the bottom of the filesystem table configuration file.



   In the preceding screenshot, the `nano` editor is used to change the disk configuration in `/etc/fstab`.

5. For the example Windows shared folder the configuration is as follows:
   - The network location (`//golden-xp/xfer`)
   - The mount point (`/media/xfer`)
   - The mount options (`guest,uid=pi`)
   - Dump option (`0`)
   - Boot phase (`2`)
6. After entering the configuration parameters for each disk, save the file and exit the editor.
7. Execute the following command:

   ```
   mount -a
   ```

   Mount all mount points in `/etc/fstab` (that are not already mounted). The `mount` command is privileged (use `sudo`).
8. Execute the following command:

   ```
   ls /media/xfer
   ```

   List all the files in the mounted Windows shared folder.
9. Execute the following command:

   ```
   touch /media/xfer/foo
   ```

   Create an empty file in the Windows shared folder.
10. Execute the following command:

    ```
    ls /media/xfer
    ```

    List files in the Windows shared folder, including the newly created file.
11. Execute the following command:

    ```
    rm /media/xfer/foo
    ```

    Remove the newly created file.

12. Execute the following command:

```
ls /media/xfer
```

List the files again. The newly created file has been deleted.

```
pi@raspberrypi ~ $ sudo nano /etc/fstab
pi@raspberrypi ~ $ sudo mount -a
pi@raspberrypi ~ $ ls /media/xfer/
hello.txt  images  music
pi@raspberrypi ~ $ touch /media/xfer/foo
pi@raspberrypi ~ $ ls /media/xfer/
foo  hello.txt  images  music
pi@raspberrypi ~ $ rm /media/xfer/foo
pi@raspberrypi ~ $ ls /media/xfer/
hello.txt  images  music
pi@raspberrypi ~ $
```

In the preceding screenshot, the filesystem table configuration is tested and the Windows shared folder is validated.

13. Execute the `reboot` command. Reboot the system. This is a privileged command (use `sudo`).
14. The Windows share `//golden-xp/xfer` is now mounted at boot!

How it works...

The goal of this recipe is to mount the Windows shared folder `//golden-xp/xfer` at boot time.

The `nano` editor is used to add the configuration parameters for the Windows shared folder to the bottom of the filesystem table configuration file (`/etc/fstab`).

The configuration for a Windows shared folder (that anyone can access) is as follows:

- The first field is the network location – `//golden-xp/xfer`. The UNC format used on Windows computers is `\\golden-xp\xfer`. The use of `/` instead of `\` is consistent with other GNU Linux paths.
- The second field is the mount point, `/media/xfer`.
- The third field is the type of filesystem, `cifs`. This is the protocol used by Windows computers to share folders.
- The fourth field is a comma-separated list of options: `guest,uid=pi`. Additional options include `user=` and `password=` for specifying the username and password when a shared folder is protected.
- The fifth field is a dump flag. It is zero (`0`), indicating that the shared folder does not need to be dumped.

- The sixth field selects the boot phase in which the disk will be mounted. The disk mounted as the root filesystem (`/`) is mounted in phase one; all other disks should be mounted in phase two.

Once the filesystem table's configuration file (`fstab`) has been updated, the configuration is tested before rebooting the system. The 'mount all disks' command, `mount -a`, is used to mount the example disks. Afterwards, a foo file is created and then deleted to test write access to the shared folder.

If there is no error, the system can be rebooted.

If the `mount` command displays an error, edit the `/etc/fstab` file again and look for typos and extra spaces. Correct any errors and try again.

If there are still errors, replace the configuration file with its original version using the following command:

```
sudo cp -f /etc/fstab.orig /etc/fstab
```

See also

- **Network Attached Storage**

  `http://en.wikipedia.org/wiki/Network_Attached_Storage`

  A Wikipedia on **Network Attached Storage** (**NAS**).

- **rm – remove files or directories**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=cp`

  The Debian man page for `rm`.

- **touch – change file timestamps**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=touch`

  The Debian man page for `touch`.

Creating a file server (Samba)

This recipe shows how the Raspberry Pi can be configured to become a file server on the local network.

The Raspberry Pi with attached file storage functions well as a file server. Such a file server could be used as a central location for sharing files and documents, for storing backups of other computers, and for storing large media files such as photo, music, and video files.

This recipe installs and configures `samba` and `samba-common-bin`. The Samba software distribution package, `samba`, contains a server for the SMB (CIFS) protocol used by Windows computers for setting up 'shared drives' or 'shared folders'. The `samba-common-bin` package contains a small collection of utilities for managing access to shared files.

The recipe includes setting a file-sharing password for the user `pi` and providing read/write access to the files in `pi` user's home directory. However, it does not set up a new file share or show how to share a USB disk. The next recipe shows how to do that.

After completing this recipe, other computers can exchange files with the default user `pi`.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- A client PC connected to the same network as the Raspberry Pi

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be remotely completed using a Secure Shell client (see *Chapter 2*, *Administration*).

How to do it...

The following are the steps for creating a file server:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   apt-get install samba samba-common-bin
   ```

   Download and install `samba` with the other packages that it depends on. This command needs to be run as a privileged user (use `sudo`).

The package management application aptitude could be used as an alternative to the `apt-get` command utility for downloading and installing `samba` and `samba-common-bin`.

```
pi@raspberrypi ~ $ sudo apt-get install samba samba-common-bin
```

In the preceding screenshot, the `apt-get` command is used to install the `samba` and `samba-common-bin` software distribution packages.

3. Execute the following command:

   `nano /etc/samba/smb.conf`

   Edit the `samba` configuration file. The `smb.conf` file is protected and needs to be accessed as a privileged user (use `sudo`).

```
pi@raspberrypi ~ $ sudo nano /etc/samba/smb.conf
```

The preceding screenshot starts the `nano` editor to edit the `/etc/samba/smb.conf` file.

4. Change the `security = user` line. Uncomment the line (remove the hash, `#`, from the beginning of the line).

```
# "security = user" is always a good idea. This will require a Unix account
# in this server for every user accessing the server. See
# /usr/share/doc/samba-doc/htmldocs/Samba3-HOWTO/ServerType.html
# in the samba-doc package for details.
   security = user
```

The preceding screenshot of the Samba configuration file shows how to change samba security to use the Raspberry Pi's user accounts.

5. Change the `read only = yes` line to be `read only = no`, as shown in the following screenshot:

```
# By default, the home directories are exported read-only. Change the
# next parameter to 'no' if you want to be able to write to them.
   read only = no
```

The preceding screenshot shows how to change the Samba configuration file to permit adding new files to user shares (`read only = no`).

6. Save and exit the `nano` editor.
7. Execute the following command:

   `/etc/init.d/samba reload`

   Tell the Samba server to reload its configuration file. This command is privileged (use `sudo`).

```
pi@raspberrypi ~ $ sudo /etc/init.d/samba reload
[ ok ] Reloading /etc/samba/smb.conf: smbd only.
pi@raspberrypi ~ $
```

   In the preceding screenshot, the Samba server's configuration file is reloaded with the `/etc/init.d/samba` command.

8. Execute the following command:

   `smbpasswd -a pi`

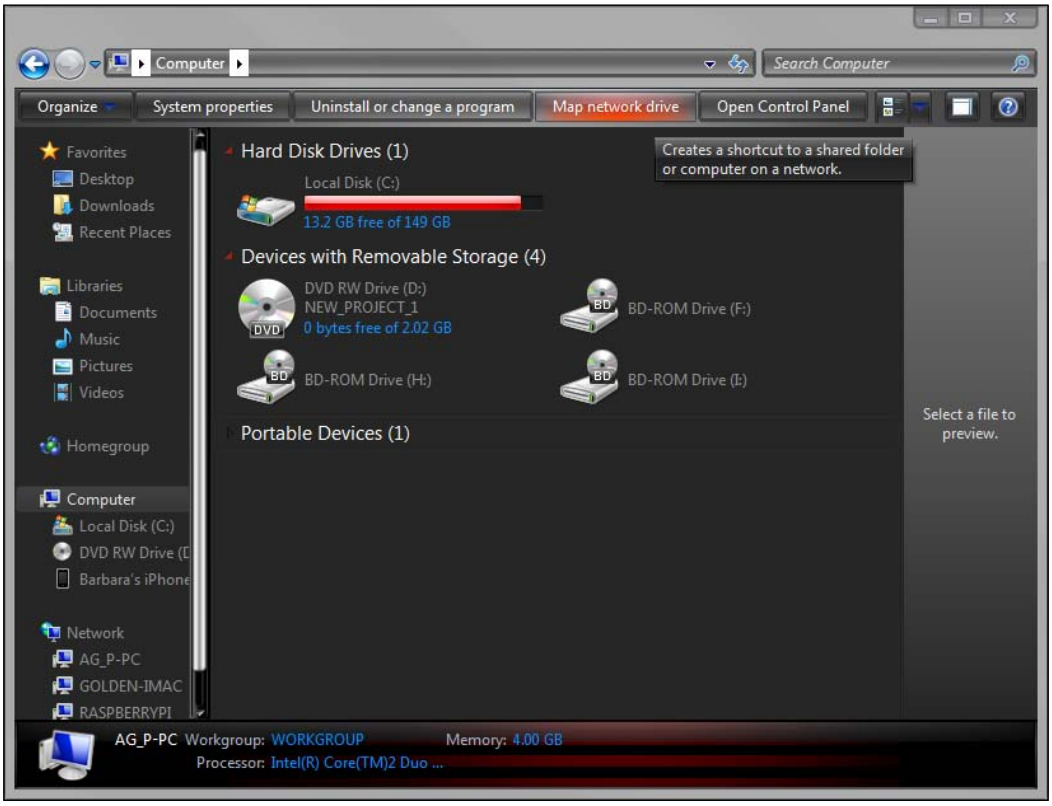
   This command needs to be run as a privileged user (use `sudo`). Enter the password (twice) that will be used for SMB (CIFS) file sharing.

```
pi@raspberrypi ~ $ sudo smbpasswd -a pi
New SMB password:
Retype new SMB password:
Added user pi.
pi@raspberrypi ~ $
```

   The preceding screenshot shows how to add an SMB password for the `pi` user.

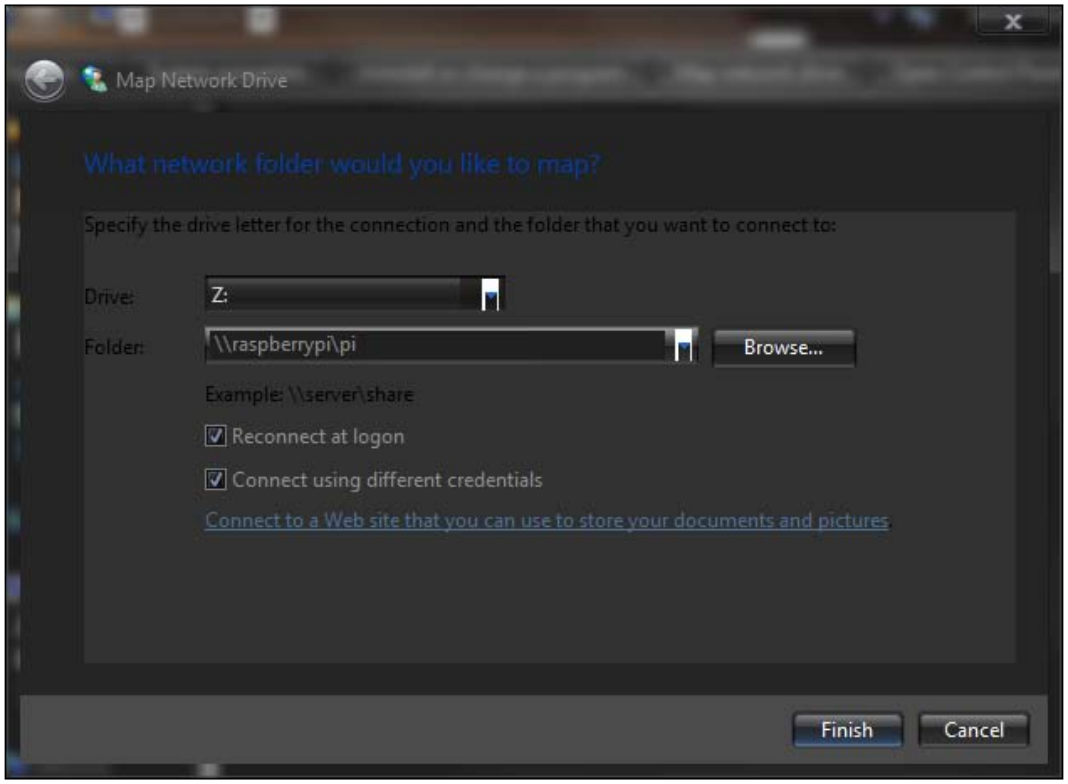
9. The Raspberry Pi is now accessible as a Windows share!

10. From a Windows computer, use **Map network drive** to mount the Raspberry Pi as a network disk, as follows:



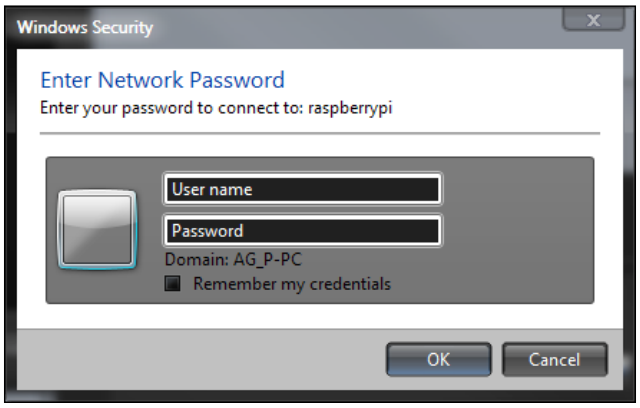
The preceding screenshot starts mapping a network drive to the Raspberry Pi on Windows 7.

11. Enter the UNC address `\\raspberrypi\pi` as the network folder. Choose an appropriate drive letter. The example uses the `Z:` drive. Select **Connect using different credentials** and click on **Finish**, as shown in the following screenshot:



The preceding screenshot finishes mapping a network drive to the Raspberry Pi.

12. Log in using the newly configured SMB (CIFS) password (from step 7).



    In the screenshot, a dialog box is displayed for logging in to the Raspberry Pi with the SMB (CIFS) username and password.

13. The Raspberry Pi is now accessible as a Windows share!

    Only the home directory of the `pi` user is accessible at this point. The next recipe configures a USB disk for using it as a shared drive.

How it works...

This recipe begins by installing two software distribution packages – `samba` and `samba-common-bin`. This recipe uses the `apt-get install` command; however, the `aptitude` package management application could also be used to install software packages.

The Samba package contains an implementation of the **Server Message Block** (**SMB**) protocol (also known as the **Common Internet File System**, **CIFS**). The SMB protocol is used by Microsoft Windows computers for sharing files and printers.

The `samba-common-bin` package contains the `smbpasswd` command. This command is used to set up user passwords exclusively for using them with the SMB protocol.

After the packages are installed, the Samba configuration file `/etc/samba/smb.conf` is updated. The file is updated to turn on user security and to enable writing files to user home directories.

After the configuration file is updated, the Samba server is told to reload its configuration. At this point, the Raspberry Pi should be visible to other machines on the local network that are using the SMB protocol. However, the passwords for the authorized users have not yet been configured.

The `smbpasswd` command is used to add (`-a`) the `pi` user to the list of users authorized to share files with the Raspberry Pi using the SMB protocol. The passwords for file sharing are managed separately from the passwords used to log in to the Raspberry Pi either directly or remotely. The `smbpasswd` command is used to set the password for Samba file sharing.

After the password has been added for the `pi` user, the Raspberry Pi should be accessible from any machine on the local network that is configured for the SMB protocol.

The last steps of the recipe configure access to the Raspberry Pi from a Windows 7 PC using a mapped network drive. The UNC name for the file share, `\\raspberrypi\pi`, could also be used to access the share directly from **Windows Explorer**.

There's more...

This is a very simple configuration for sharing files. It enables file sharing for users with a login to the Raspberry Pi. However, it only permits the files in the user home directories to be shared. The next recipe describes how to add a new a file share.

In addition to the SMB protocol server, `smbd`, the Samba software distribution package also contains a NetBIOS name server, `nmbd`. The NetBIOS name server provides naming services to computers using the SMB protocol. The `nmbd` server broadcasts the configured name of the Raspberry Pi, `raspberrypi`, to other computers on the local network.

In addition to file sharing, a Samba server could also be used as a **Primary Domain Controller** (**PDC**) – a central network server that is used to provide logins and security for all computers on a LAN. More information on using the Samba package as a PDC can be found on the links given next.

See also

- **Samba (software)**

  `http://en.wikipedia.org/wiki/Samba_(software)`

  A Wikipedia article on the Samba software suite.

- **nmbd - NetBIOS over IP naming service**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=nmbd`

  The Debian man page for `nmbd`.

- **samba – a Windows SMB/CIFS file server for UNIX**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=samba`

  The Debian man page for `samba`.

- **smb.conf - the configuration file for the Samba suite**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=smb.conf`

  The Debian man page for `smb.conf`.

- **smbd - server to provide SMB/CIFS services to clients**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=smbd`

  The Debian man page for `smbd`.

- **smbpasswd - change a user's SMB password**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=smbpasswd`

  The Debian man page for `smbpasswd`.

- **System initialization**

  `http://www.debian.org/doc/manuals/debian-reference/ch04.en.html`

  The Debian Reference Manual article on system initialization.

- **Samba.org**

  `http://www.samba.org`

  The Samba software website.

Sharing an attached USB disk via Samba

This recipe extends the default Samba configuration to include sharing a USB disk.

The previous recipe showed how to install Samba and set up file sharing. However, the default Samba configuration only shares files that are in the home directory of the `pi` user. This recipe extends the Samba configuration file to include a file share definition that points to a USB hard disk.

After completing this recipe, the Raspberry Pi can be used as a file server for any computer that shares its files via the SMB (CIFS) protocol (for example, Windows, Mac, and Linux computers).

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection (optional)

- A client PC connected to the same network as the Raspberry Pi (optional)
- A powered USB hub (recommended)
- At least one USB disk drive (the example uses a 500-GB USB disk)

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be remotely completed using a Secure Shell client (see *Chapter 2*, *Administration*).

The examples in this recipe use a USB disk mounted at `/media/bigdisk`.

The Raspberry Pi should already have Samba installed (see the previous recipe).

How to do it...

The following are the steps for sharing an attached USB disk:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

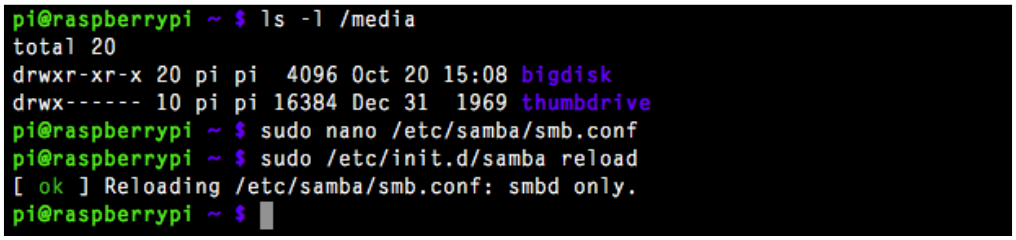
   ```
   ls -l /media
   ```

   List the mounted disk drives.

3. Execute the following command:

   ```
   nano /etc/samba/smb.conf
   ```

   Edit the Samba configuration file. The `smb.conf` file is protected and needs to be accessed as a privileged user (use `sudo`).



   The preceding screenshot lists the attached disks with `ls` and then uses the `nano` editor to edit the `/etc/samba/smb.conf` file. Finally, the configuration file is reloaded (`samba reload`).

4. Add a new share configuration to the bottom of the file, as follows:

```
[bigdisk]
        comment = A really big disk!
        path = /media/bigdisk
        valid users = pi
        admin users = pi
        read only = No
```

5. Save and exit the `nano` editor.

```
############ Shares ############

[bigdisk]
        comment = A really big disk!
        path = /media/bigdisk
        valid users = pi
        admin users = pi
        read only = No
```

In the preceding screenshot, a new `bigdisk` share is configured at the bottom of the `/etc/samba/smb.conf` file.

6. Execute the following command:

```
/etc/init.d/samba reload
```

Tell the Samba server to reload its configuration file. This command is privileged (use `sudo`).

7. The file share `\\raspberrypi\bigdisk` is now accessible from the local network!

Connect as the `pi` user.

How it works...

First the recipe lists the disks that have been mounted in the `/media` directory. This is the directory used by the file manager included with the Raspberry Pi desktop, PCManFM, when automounting USB disks. It is also the directory used by the `pmount` command and the *Automounting USB disks at boot (/etc/fstab)* recipe for automounting USB disks at boot time.

Then, the `nano` editor is used to modify the Samba configuration file, `smb.conf`. The configuration for a new share is added to the bottom of the file. The following is its explanation:

- `[bigdisk]` starts a new section ('[',']') in the config file and sets `bigdisk` as the name of the share
- `Comment = A really big disk!` defines a comment to display when users browse the share from another computer
- `Path = /media/bigdisk` defines the location of the files on the Raspberry Pi
- `valid users = pi` declares that only the `pi` user can access the files
- `admin users = pi` gives the `pi` user administrative access to the files
- `read only = no` allows files to be written to and deleted from the share

After the configuration is saved and the `nano` editor has been exited, the Samba server is told to reload its configuration with the command `/etc/init.d/samba reload`.

Once the configuration file is reloaded, the Samba server is ready to share files from the `/media/bigdata` directory over the file share `bigdata` at the UNC address `\\raspberrypi\bigdata`.

There's more...

The files in the new share are protected and require that users on other computers connect to the share using the Samba password of the `pi` user.

The Samba password of a user is maintained separately from the login password. Use the `smbpasswd` command to manage Samba passwords.

Accessing another computer's files (smbclient)

The goal of this recipe is to transfer files between the Raspberry Pi and other computers using the `smbclient` command.

The previous recipes showed how to mount Widows shared folders either temporarily (by using the `mount` or `pmount` command) or permanently (by configuring `/etc/fstab`). This recipe shows how to list the available network shared folders and how to copy individual files from a computer with shared folders to the Raspberry Pi using the `smbclient` command.

After completing this recipe, you will be able to list the available file shares on the local network and transfer files directly between another computer and the Raspberry Pi using the SMB (CIFS) protocol.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- A client PC connected to the same network as the Raspberry Pi

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running and it has a network connection, this recipe can be remotely completed using a Secure Shell client (see *Chapter 2*, *Administration*).

The examples in this recipe will connect the Raspberry Pi to a local network computer named `GOLDEN-XP`. This computer is running Windows XP and has one shared folder, `XFER`. Configuration for other computers using the SMB (CIFS) protocol will be similar.

How to do it...

The following are the steps to access another computer's files:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   apt-get install smbclient
   ```

   Download and install the `smbclient` software package. This command needs to be run as a privileged user (use `sudo`).

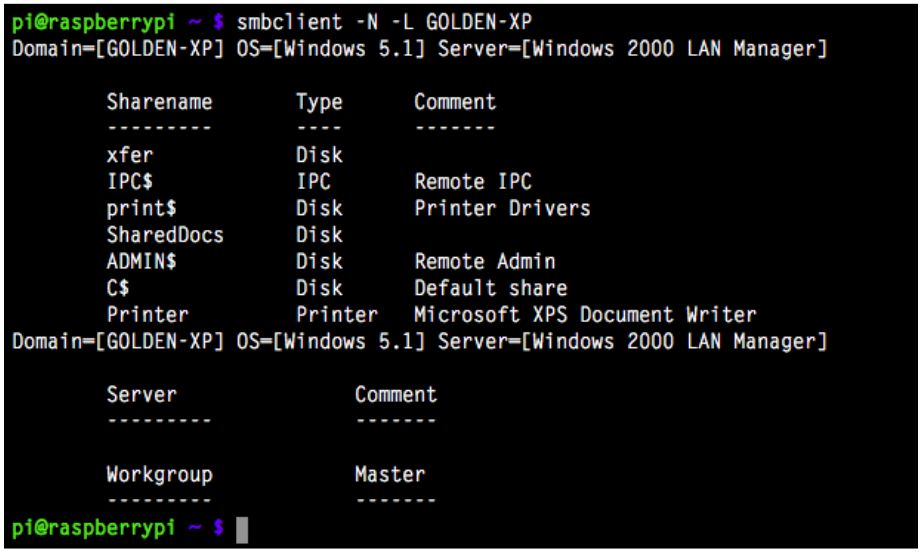
   ```
   pi@raspberrypi ~ $ sudo apt-get install smbclient
   ```

   In the preceding screenshot, the `apt-get` command is used to install the `smbclient` software distribution packages.

3. Execute the following command:

```
smbclient -N -L GOLDEN-XP
```

List the SMB services on the local network computer `GOLDEN-XP`.

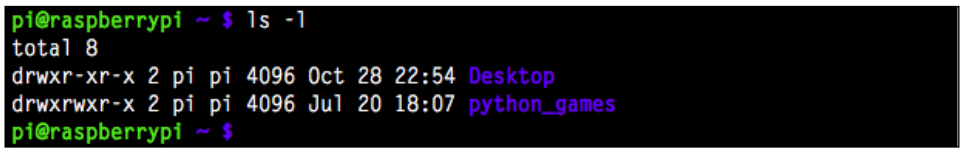


The preceding screenshot shows how to use the `smbclient` command to list the services (`-L`) on the local network computer `GOLDEN-XP`. No password is required (`-N`).

4. A list of SMB services from the computer `GOLDEN-XP` is displayed.
5. Execute the following command:

```
ls -l
```

List the files in the current directory.

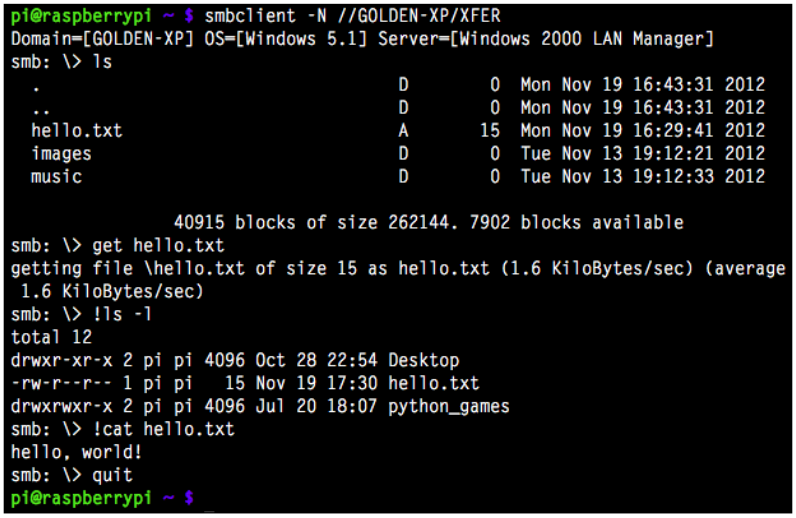


The preceding screenshot shows how to list the files in the current directory.

6. Execute the following command:

   `smbclient //GOLDEN-XP/XFER`

   Begin an SMB conversation with the remote computer `GOLDEN-XP`.



   The preceding screenshot uses `smbclient` to converse with `GOLDEN-XP` and transfer the `hello.txt` file to the Raspberry Pi.

7. The prompt `smb: \>` is displayed.

   The `smbclient` command is ready to converse with `GOLDEN-XP`.

8. Execute the `ls` command.

   The files and folders in the `XFER` share of `GOLDEN-XP` are displayed.

9. Execute the following command:

   `get hello.txt`

   The `hello.txt` file is transferred from `GOLDEN-XP` to the Raspberry Pi.

10. Execute the following command:

    `!ls -l`

    The contents of the current directory on the Raspberry Pi are displayed.

11. Execute the following command:

    `!cat hello.txt`

    The contents of the local file `hello.txt` are displayed.

12. Execute the `quit` command.

    The SMB protocol conversation with `GOLDEN-XP` is complete.

How it works...

First the software distribution package `smbclient` is installed.

This recipe uses the `apt-get install` command; however, the aptitude package management application could also be used to install software packages (for an example, see *Chapter 2*, *Administration*).

The `smbclient` package contains the command-line application `smbclient`. This application is used to communicate with other computers using the SMB protocol.

After the software distribution package is installed, the `smbclient` command is used to list the available services on a remote computer named `GOLDEN-XP`. The `-N` option tells the command that no password is required for the remote computer, and the `-L` option tells the command to just list the available shares and printers.

Before `smbclient` is used to start a conversation with `GOLDEN-XP`, the current contents of the local directory are displayed using the `ls -l` command. The `-l` option tells the `ls` command to use the long listing format.

The command `smbclient -N //GOLDEN-XP/XFER` is used to start a conversation between the Raspberry Pi and the remote computer `GOLDEN-XP` using the file share named `XFER`. The command prompt changes to `smb: \>`, telling the user that the conversation has begun.

The user then enters the `ls` command, which lists the files on the remote computer. The `dir` command could be used instead and will return the same result as the `ls` command.

When the user enters the command `get hello.txt` the `hello.txt` file is transferred from the remote computer to the Raspberry Pi. The `smbclient` command reports that the file transferred has a size of 15 bytes and was transferred at a rate of 1.6 KB per second.

After the file has been transferred the `!ls -l` command is used to list the contents of the local directory on the Raspberry Pi. Notice "`!`" at the beginning of the command. This special character tells `smbclient` to execute the command on the Raspberry Pi, instead of the remote computer. So, the list of files displayed is of those on the Raspberry Pi. This is the same command that was executed in step 5, only now the list of files includes the `hello.txt` file.

The `!cat hello.txt` command is also executed on the Raspberry Pi (notice "`!`"). The `cat` command is being used to display the contents of the `hello.txt` file, which is the familiar greeting – `hello, world!`.

Finally, the `quit` command is used to tell `smbclient` that the conversation with the remote computer `GOLDEN-XP` is now over. The command prompt returns to `pi@raspberrypi ~ $`, showing that the conversation is complete.

There's more...

The `smbclient` command has its own rich set of commands for communicating with remote computers via the SMB protocol. The `help` command can be used to list the commands available from the `smb: \>` prompt.

```
smb: \> help
?              allinfo        altname        archive        blocksize
cancel         case_sensitive cd             chmod          chown
close          del            dir            du             echo
exit           get            getfacl        geteas         hardlink
help           history        iosize         lcd            link
lock           lowercase      ls             l              mask
md             mget           mkdir          more           mput
newer          open           posix          posix_encrypt  posix_open
posix_mkdir    posix_rmdir    posix_unlink   print          prompt
put            pwd            q              queue          quit
readlink       rd             recurse        reget          rename
reput          rm             rmdir          showacls       setea
setmode        stat           symlink        tar            tarmode
translate      unlock         volume         vuid           wdel
logon          listconnect    showconnect    ..             !
smb: \> help get
HELP get:
       <remote name> [local name] get a file

smb: \>
```

The preceding screenshot shows how `help` can be used to display a list of available `smbclient` commands.

The `cd` command in `smbclient` can be used to change directories on the remote computer in the same way `cd` is used on the Raspberry Pi. Notice that the current directory path is reflected in the `smbclient` command prompt.

```
smb: \> ls
  .                                   D        0  Mon Nov 19 16:43:31 2012
  ..                                  D        0  Mon Nov 19 16:43:31 2012
  hello.txt                           A       15  Mon Nov 19 16:29:41 2012
  images                              D        0  Tue Nov 13 19:12:21 2012
  music                               D        0  Tue Nov 13 19:12:33 2012

                40915 blocks of size 262144. 7902 blocks available
smb: \> cd music
smb: \music\> ls
  .                                   D        0  Tue Nov 13 19:12:33 2012
  ..                                  D        0  Tue Nov 13 19:12:33 2012
  Seeed                               D        0  Mon Nov 19 16:46:57 2012

                40915 blocks of size 262144. 7902 blocks available
smb: \music\> cd Seeed
smb: \music\Seeed\> ls
  .                                   D        0  Mon Nov 19 16:46:57 2012
  ..                                  D        0  Mon Nov 19 16:46:57 2012
  03-seeed-aufstehn.mp3               A  6505317  Tue Nov 13 19:11:33 2012
  13-seeed-light_the_sun.mp3          A  4821141  Tue Nov 13 19:11:29 2012

                40915 blocks of size 262144. 7902 blocks available
smb: \music\Seeed\>
```

The preceding screenshot shows how the `smbclient cd` command is used to navigate to a music folder on a remote computer.

If the location of a file on the remote computer is already known, it can be fetched directly with a single `smbclient` command.

```
pi@raspberrypi ~ $ smbclient -N //golden-xp/xfer -c 'get \music\Seeed\03-see
ed-aufstehn.mp3'
Domain=[GOLDEN-XP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
getting file \music\Seeed\03-seeed-aufstehn.mp3 of size 6505317 as \music\Se
eed\03-seeed-aufstehn.mp3 (10414.5 KiloBytes/sec) (average 10414.5 KiloBytes
/sec)
pi@raspberrypi ~ $
```

The preceding screenshot shows how to fetch a file from a remote computer with a single `smbclient` command.

Notice that the SMB (CIFS) protocol uses \ instead of / as a path separator. The path separator is one of the major differences between the Windows operating system and GNU Linux-based operating systems such as the Raspbian Linux distribution used by the Raspberry Pi.

See also

- **cat – concatenate files and print on the standard output**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=cat`

  The Debian man page for `cat`.

- **cd – change the working directory**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=cd`

  The Debian man page for `cd`.

- **smbclient – client to access SMB/CIFS resources on servers**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=smbclient`

  The Debian man page for `smbclient`.

5
Advanced Networking

In this chapter we will cover the following:

- Creating a firewall with `ufw`
- Connecting to the desktop remotely (`xrdp`)
- Installing a web server (`apache`, `lighttpd`, `nginx`)
- Installing a wiki (MediaWiki)
- Creating a wireless access point with `hostapd`

Introduction

The recipes in this chapter are for advanced networking solutions.

The recipes in this chapter include setting up a firewall, accessing the Raspberry Pi desktop remotely, installing a web server and installing a wiki server. The final recipe in this chapter sets up the Raspberry Pi as a wireless access point.

Once you've completed the recipes in this chapter, your Raspberry Pi can be used for a number of advanced networking solutions.

Creating a firewall with ufw

This recipe uses a simple, yet powerful command-line tool (`ufw`) to configure a firewall.

This recipe can be used to completely block access to the Raspberry Pi. It can also be used to configure access through the firewall to specific applications (such as a web server or file server).

After completing this recipe, you will be able to protect the Raspberry Pi with a firewall.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

How to do it...

The steps to create a firewall using `ufw` are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the command: `apt-get install ufw`.

   This command needs to be run as a privileged user (use `sudo`).

The package management application `aptitude` can be used as an alternative to the `apt-get` command utility for downloading and installing software distribution packages.

```
pi@raspberrypi ~ $ sudo apt-get install ufw
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ufw
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 166 kB of archives.
After this operation, 708 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main ufw all 0.31.
1-2 [166 kB]
Fetched 166 kB in 1s (115 kB/s)
Preconfiguring packages ...
Selecting previously unselected package ufw.
(Reading database ... 57780 files and directories currently installed.)
Unpacking ufw (from .../archives/ufw_0.31.1-2_all.deb) ...
Processing triggers for man-db ...
Setting up ufw (0.31.1-2) ...

Creating config file /etc/ufw/before.rules with new version

Creating config file /etc/ufw/before6.rules with new version

Creating config file /etc/ufw/after.rules with new version

Creating config file /etc/ufw/after6.rules with new version
pi@raspberrypi ~ $
```

The preceding screenshot shows how the `apt-get install` command is used to install the `ufw` package with a default set of firewall rules.

3. The `apt-get install` command downloads and installs `ufw`.

   The installation of `ufw` includes a default set of firewall rules.

4. Execute the following command:

   `ufw allow ssh`

   This command needs to be run as a privileged user (use `sudo`).

```
pi@raspberrypi ~ $ sudo ufw allow ssh
Rule updated
pi@raspberrypi ~ $
```

In the preceding screenshot, the `ufw allow ssh` command is used to permit remote access to the Raspberry Pi via the SSH protocol.

5. The `ufw allow` command updates the firewall rules (`Rule updated`).
6. Execute the following command:

   `ufw enable.`

   This command also needs to be run as a privileged user (use `sudo`).

```
pi@raspberrypi ~ $ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
pi@raspberrypi ~ $
```

In the preceding screenshot, the `ufw enable` command is used to activate the firewall immediately and also when the Raspberry Pi boots.

7. The `ufw enable` command enables the defined firewall rules.

   When the firewall rules are enabled via a remote connection, the user is warned and prompted to continue. Answer `y` to continue.

8. The firewall rules are enabled immediately and also on startup!

How it works...

The Uncomplicated Firewall (`ufw`) is downloaded and installed using `apt-get install`. Then, a firewall rule is defined to permit remote access to the Raspberry Pi using `ssh` (`ufw allow ssh`). Finally, the firewall is turned on (`ufw enable`).

There's more...

The Uncomplicated Firewall (`ufw`) is used to define firewall rules. The `ufw` command is not actually a firewall. Instead, the command is used to configure firewall rules. The rule definitions are stored in the `/etc/ufw` directory; however, those definition files should not normally be modified by hand. That is the purpose of the `ufw` command.

In this recipe, after the command is installed, a firewall rule, "allow SSH access" (`ufw allow ssh`) is added to the rule definitions before the rules are enabled. Without this rule, the next attempt to connect remotely via `ssh` would fail.

```
golden-imac:~ golden$ ssh pi@192.168.1.79
ssh: connect to host 192.168.1.79 port 22: Operation timed out
golden-imac:~ golden$
```

In the preceding image, the `ssh` command fails because there is no firewall rule permitting access to the Raspberry Pi.

The firewall rules defined using `ufw` will not prevent a user from logging on using a keyboard and display connected directly to the Raspberry Pi. So, if remote access is no longer possible, log in directly to the Raspberry Pi and then use the `ufw disable` command to disable the firewall rules.

```
pi@raspberrypi ~ $ sudo ufw disable
Firewall stopped and disabled on system startup
pi@raspberrypi ~ $
```

The preceding screenshot shows how to disable the firewall rules with the `ufw` command.

If the Raspberry Pi has been set up as a file server (see *Chapter 3*, *Maintenance*), execute the `ufw allow cifs` command to allow the SMB (CIFS) protocol through the firewall.

```
pi@raspberrypi ~ $ sudo ufw allow cifs
Rule added
pi@raspberrypi ~ $
```

The preceding screenshot shows how to use `ufw` to allow SMB (CIFS) file sharing through the Raspberry Pi's firewall.

Similarly, if the Raspberry Pi has been set up as a web server, execute the `ufw allow http` command to allow the HTTP protocol through the firewall.

The Uncomplicated Firewall has predefined rules for a number of applications, in addition to `ssh`. The applications currently recognized by `ufw` can be displayed with the `ufw app list` command.

Remote connections for these applications can be enabled using the `ufw accept $APPNAME` command and disabled using the `ufw deny $APPNAME` command. Where `$APPNAME` is replaced by the name of a recognized application (for example, `ssh`).

By default, the `ufw` rules deny any remote access to the Raspberry Pi. However, also by default, the firewall rules are not enabled.

Use the `ufw reset` command to return the firewall rules to their defaults: no remote access at all and firewall disabled. After resetting the default rules, the firewall can be re-enabled with the `ufw enable` command.

See also

- **Uncomplicated Firewall (ufw)**

  `http://en.wikipedia.org/wiki/Uncomplicated_Firewall`

  A Wikipedia article about the Uncomplicated Firewall.

- **ufw – program for managing a netfilter firewall**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=ufw`

  The Debian man page for `ufw`.

- **iptables – administration tool for IPv4 packet filtering and NAT**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=iptables`

  The Debian man page for `iptables`.

Connecting to the desktop remotely (xrdp)

This recipe enables remote access to the Raspberry Pi desktop using `xrdp`.

Using `xrdp`, teachers, support engineers, and hobbyists can access the Raspberry Pi display from another computer. Teachers could use the remote desktop access to help students complete an assignment. Remote support engineers could help a Raspberry Pi user debug, install, or configure software. And hobbyists can use `xrdp` to access the Raspberry Pi attached to the TV from another room.

After completing this recipe, you will be able to use `xrdp` to remotely access the Raspberry Pi desktop.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection

Running this application requires the desktop GUI. Use `raspi-config` to configure the Raspberry Pi to automatically boot with the desktop GUI (see *Chapter 2*, *Administration*).

The installation portion of this recipe does not require the desktop and can be run from the text-based console (or from within an LXTerminal).

If the Raspberry Pi's Secure Shell server is running, the installation portion of this recipe can also be completed remotely using a Secure Shell client (see the `ssh` recipe in *Chapter 2*, *Administration*).

How to do it...

The steps for connecting to the Raspberry Pi desktop remotely are as follows:

1. Log in to the Raspberry Pi either directly or remotely.

   If you are using the Raspberry Pi desktop, open an LXTerminal window.

2. Execute the command:

   `apt-get install xrdp.`

   This command needs to be run as a privileged user (use `sudo`).

The package management application `aptitude` can be used as an alternative to the `apt-get` command utility for downloading and installing software distribution packages.

```
pi@raspberrypi ~ $ sudo apt-get install xrdp
```

The preceding screenshot shows how the `apt-get install` command is used to install the `xrdp` software distribution package.

3. The `apt-get install` command downloads and installs `xrdp`.

   Installation of the software package includes starting the `xrdp` services.

```
Setting up xrdp (0.5.0-2) ...
[....] Generating xrdp RSA keys......
Generating 512 bit rsa key...

ssl_gen_key_xrdp1 ok

saving to /etc/xrdp/rsakeys.ini

done (done).
[ ok ] Starting Remote Desktop Protocol server : xrdp sesman.
Processing triggers for menu ...
pi@raspberrypi ~ $
```

   The preceding screenshot shows that after the install of the `xrdp` software package, the xrdp session manager (`sesman`) is started.

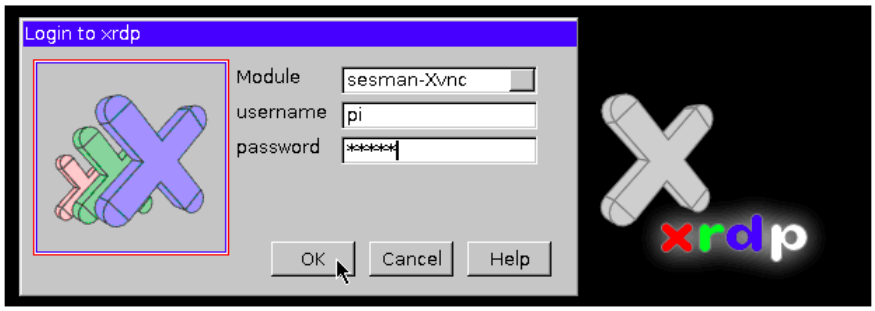
4. Use a **Remote Desktop Protocol** (**RDP**) client (such as Microsoft's Remote Desktop Connection) to connect to the Raspberry Pi using its IP address (**192.168.2.2**).

   The preceding screenshot shows how to connect to the Raspberry Pi using Microsoft's Remote Desktop Connection for Mac.

5. Once connected, the `xrdp` session manager (`sesman`) displays a login screen.
6. Log in to the `xrdp` session.

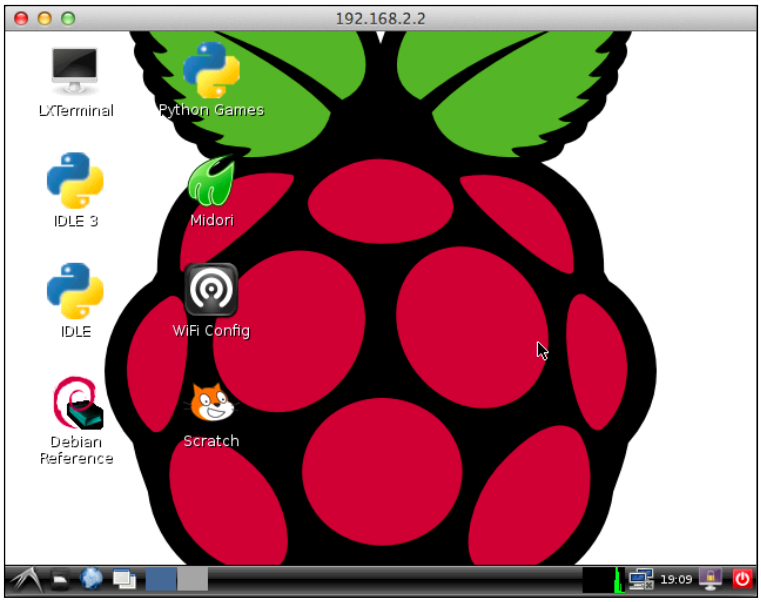
   Choose `sesman-Xvnc` as the Module.

7. Use the same username (`pi`) and password (`raspberry`) that is used to log in via the console (or to the desktop).



The preceding screenshot shows how to start a remote desktop session with the Raspberry Pi using `xrdp`.

8. After a successful login, the Raspberry Pi desktop is displayed.



The preceding screenshot shows a remote connection to the Raspberry Pi desktop via an RDP connection (client resolution was set to 640x480).

How it works...

The `xrdb` software package is downloaded and installed using `apt-get install`.

The installation includes starting the `xrdb` session manager (`sesman`).

Once `sesman` has started, a remote session to the Raspberry Pi can be created from another computer that has an RDP client installed.

The example in the recipe uses Microsoft's Remote Desktop Connection for Mac. The Microsoft RDP client prompts for the remote computer's name or IP address. The IP address of the Raspberry Pi is entered to create the remote connection.

Before the Raspberry Pi desktop is displayed, an RDP session is created by logging in to the Raspberry Pi with the same username and password (`pi` and `raspberry`) that is used to login directly to the Raspberry Pi via the console or the desktop.

The remote desktop connection displays the Raspberry Pi desktop in much the same way as the desktop would be seen if the Raspberry Pi were connected directly to a display. There are some exceptions.

The most noticeable limitation that a remote display has is not being able to display streaming video. Most video streaming applications stream directly to the video frame buffer, the part of memory that is shared with the display. So, the video is streamed directly to the display and bypasses the desktop. Because the video bypasses the desktop, it is also not sent to the remote desktop.

There's more...

The **Remote Desktop Protocol** (**RDP**) is a client server protocol developed by Microsoft. The `xrdp` client is used to display the graphical user interface sent by an RDP server. The Microsoft RDP client (`mstsc.exe`) is included in every Windows version since XP.

The `xrdp` software package also includes the TightVNC server (`tightvncsever`), which communicates via the **Virtual Network Computing** (**VNC**) protocol—another graphical desktop sharing protocol. A number of open source and proprietary clients exist for both the RDP and VNC remote desktop protocols, including clients for mobile devices and tablet PCs.

See also

- **Remote Desktop Protocol**

  `http://en.wikipedia.org/wiki/Remote_Desktop_Protocol`

  A Wikipedia article about the **Remote Desktop Protocol** (**RDP**).

- **Virtual Network Computing (VNC)**

  `http://en.wikipedia.org/wiki/Vnc`

  A Wikipedia article about **Virtual Network Computing**.

- **Comparison of remote desktop software**

  `http://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software`

  A Wikipedia article that compares remote desktop software.

- **xrdp – a Remote Desktop Protocol (RDP) server**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=xrdp`

  The Debian man page for `xrdp`.

- **sesman – a xrdp(8) session manager**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=sesman`

  The Debian man page for `sesman`.

- **TightVNC software**

  `http://tightvnc.com/`

  The TightVNC software website.

- **xrdp – An open source Remote Desktop Protocol (RDP) server**

  `http://www.xrdp.org/`

  The `xrdp` website.

Installing a web server (Apache, lighttpd, Nginx)

This recipe installs the Apache HTTP web server. Installation differences for the lighttpd and Nginx web servers are found at the end of this recipe.

The Apache web server is one of the most commonly used web servers on GNU Linux platforms, such as the "official" Raspbian Linux distribution used by the Raspberry Pi. Apache is a mature, extensible web server that can be used to host a variety of applications as well as static web pages.

After completing this recipe, the Raspberry Pi will be able to serve static and dynamic web pages from the SD card boot disk and from an external disk.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A network connection
- A client PC connected to the same network as the Raspberry Pi (for testing)

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

The examples in this recipe also use an external hard disk mounted at `/media/bigdisk` (see the some of the previous recipes for more information).

Bear in mind that the IP address assigned to the Raspberry Pi will depend on your local area network configuration. The `ifconfig` command (see the *Remote Access (ssh)* recipe in *Chapter 2*, *Administration*) can be used to discover the IP address of the Raspberry Pi.

How to do it...

The steps for installing the Apache web server are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   ```
   apt-get install apache2
   ```

   Download and install the Apache HTTP web server.

   This command needs to be run as a privileged user (use `sudo`).

The package management applications `aptitude` or `synaptic` could be used as alternatives to the command utility `apt-get` for downloading and installing `apache2`.

```
pi@raspberrypi ~ $ sudo apt-get install apache2
```

The preceding screenshot shows the `apt-get` being used to install the Apache web server's software distribution package.

3. (optional) Execute the following command:

   ```
   ufw allow http
   ```

   Allow the HTTP (web server) protocol through the firewall.

   Use only if the `ufw` firewall utility has been installed.

   This command needs to be run as a privileged user (use `sudo`).

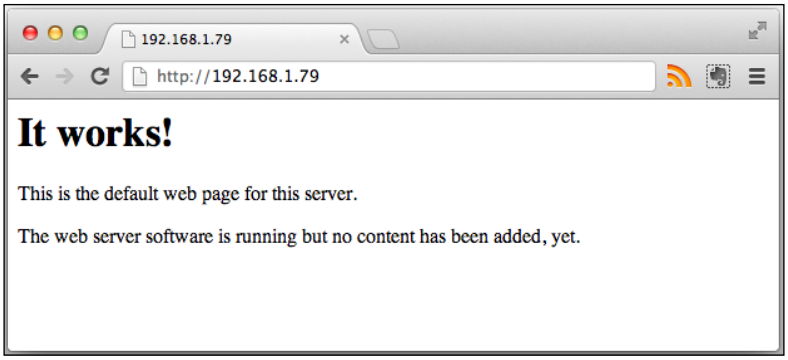
```
pi@raspberrypi ~ $ sudo ufw allow http
Rule added
pi@raspberrypi ~ $
```

   The preceding screenshot shows how to use the `ufw` command to allow the HTTP (web server) protocol to pass through the Raspberry Pi's firewall.

4. Use a web browser on another PC to connect to the Raspberry Pi.

   Use the IP address of the Raspberry Pi to address the web server (`http://192.168.1.79`).

   The Raspberry Pi is now accessible as a web server!



   In the preceding screenshot, a web browser on another PC is used to access the default web page of the Apache web server running on the Raspberry Pi.

5. Execute the following command:

   ```
   ls -l /var/www
   ```

   List the contents of the web server's root directory.

   There is one file: `index.html`.

6. Execute the following command:

   ```
   ls -l /media/bigdisk/MyWebsite/
   ```

   List the contents of a website stored on an external disk.

   The disk has already been mounted.

7. Execute the following command:

   ```
   ln -s /media/bigdisk/MyWebsite /var/www/MyWebsite
   ```

   Create a symbolic link to an external disk from the web server's root.

8. Execute the following command:

   ```
   ls -l /var/www/
   ```

   The link to the external disk is visible in the web server's root directory.

9. Execute the following command:

   ```
   ls -l /var/www/MyWebsite/
   ```

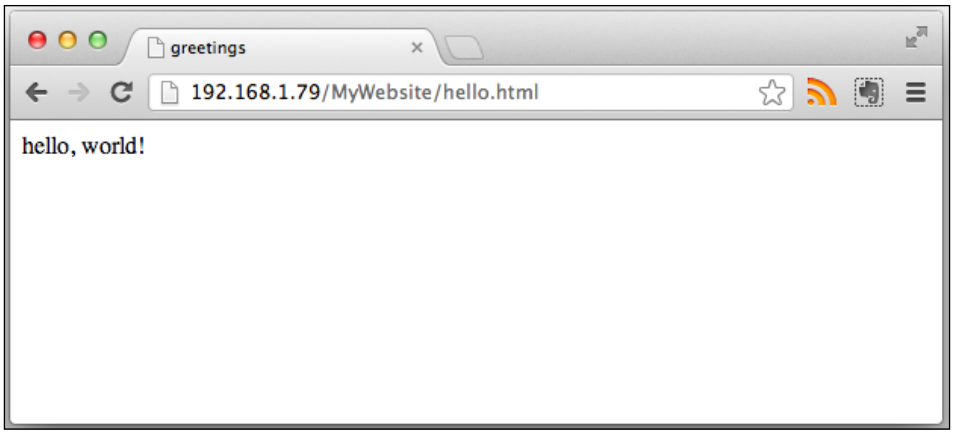
   The linked external drive is accessible from the web server's root.

```
pi@raspberrypi ~ $ ls -l /var/www/
total 4
-rw-r--r-- 1 root root 177 Dec  9 15:16 index.html
pi@raspberrypi ~ $ ls -l /media/bigdisk/MyWebsite/
total 4
-rw-r--r-- 1 pi root 92 Dec  9 16:15 hello.html
pi@raspberrypi ~ $ sudo ln -s /media/bigdisk/MyWebsite /var/www/MyWebsite
pi@raspberrypi ~ $ ls -l /var/www/
total 4
-rw-r--r-- 1 root root 177 Dec  9 15:16 index.html
lrwxrwxrwx 1 root root  24 Dec  9 16:23 MyWebsite -> /media/bigdisk/MyWebsite
pi@raspberrypi ~ $ ls -l /var/www/MyWebsite/
total 4
-rw-r--r-- 1 pi root 92 Dec  9 16:15 hello.html
pi@raspberrypi ~ $
```

In the preceding screenshot, the `ln -s` command is used to create a link from the web server root to a website stored on an external disk.

10. Use a web browser on another PC to display the website (`http://192.168.1.79/MyWebsite/hello.html`).

    The website on the external disk is now accessible!



    The preceding screenshot shows how a web browser on another PC is used to access a web page stored on an external disk attached to the Raspberry Pi.

11. Execute the following command:

    `sudo nano /usr/lib/cgi-bin/timestamp`

    Use the `nano` editor to create a dynamic web page.

    The `/usr/lib/cgi-bin` directory is protected (use `sudo`).

```
pi@raspberrypi ~ $ sudo nano /usr/lib/cgi-bin/timestamp
```

    The preceding screenshot shows the `nano` command used to create a dynamic web page.

12. Add the following lines to the dynamic web page:

```
#!/bin/sh
echo "Content-Type: text/plain"
echo ""
echo $(date)
```

```
#!/bin/bash

echo "Content-Type: text/plain"
echo ""
echo "$(date)"
```

The preceding screenshot shows the contents of the dynamic web page.

13. Execute the following command:

```
chmod a+rx /usr/lib/cgi-bin/timestamp
```

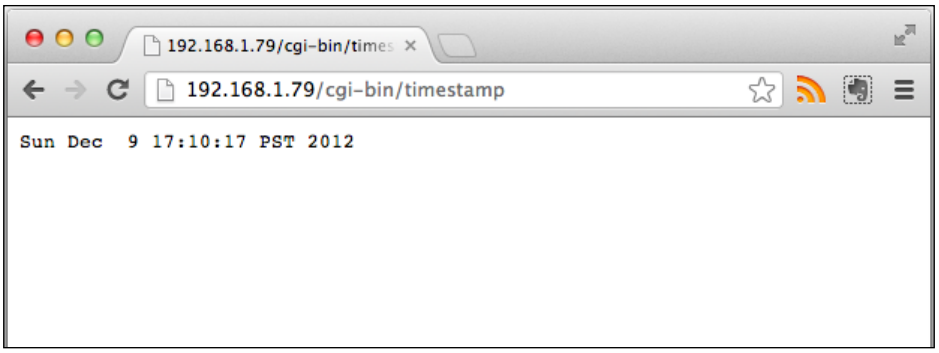
Make the dynamic web page executable.

```
pi@raspberrypi ~ $ sudo chmod a+rx /usr/lib/cgi-bin/timestamp
```

The preceding screenshot shows how to change the mode of a file so that it is executable by everyone.

14. Use a web browser on another PC to display the dynamic web page (`http://192.168.1.79/cgi-bin/timestamp`).

The dynamic web page displays the current time with each page refresh!



The preceding screenshot shows the dynamic web page in action—with every page refresh the timestamp is updated to the current time.

15. Raspberry Pi is now a web server displaying static and dynamic pages.
16. Execute the following command:

```
ls -l /var/log/apache2/
```

Display the contents of the web server's logfiles directory.

```
pi@raspberrypi ~ $ ls -l /var/log/apache2/
total 20
-rw-r----- 1 root adm  10689 Dec  9 17:57 access.log
-rw-r----- 1 root adm   4468 Dec 10 02:17 error.log
-rw-r--r-- 1 root root     0 Dec  9 15:16 other_vhosts_access.log
pi@raspberrypi ~ $
```

The preceding screenshot shows how the `ls` command is used to display the contents of the Apache web server's logfiles directory.

How it works...

After logging in to the Raspberry Pi, this recipe begins by downloading and installing the Apache HTTP server software distribution metapackage. The `apache2` software distribution includes the Apache HTTP server and supporting utilities. The configuration files for the server are located in the `/etc/apache` directory. However, there is no need to change the configuration as the default configuration is sufficient for serving static web pages from the `/var/www` directory and CGI-based dynamic web pages from the `/usr/lib/cgi-bin` directory.

The Apache HTTP server is started as soon as it is installed. However, if the Raspberry Pi has been protected with a firewall, the firewall will need to be configured to allow HTTP connections. The `ufw allow http` command configures the Raspberry Pi to allow connections from other computers using the HTTP protocol.

After the firewall has been configured, a web browser is used to connect to the Raspberry Pi from another computer. The Raspberry Pi will most likely need to be addressed using its IP address (`192.168.1.79`) instead of its computer name (`raspberrypi`). The `ifconfig` command (see *Chapter 2*, *Administration*) can be used to discover the IP address of the Raspberry Pi.

The IP address of the Raspberry Pi is dependent on the configuration of your local area network. The IP address of your Raspberry Pi is most likely different from the example used in this book. See the *Remote Access (ssh)* recipe in *Chapter 2*, *Administration*, which shows you how to determine the IP address of your Raspberry Pi.

The directory `/var/www` is the default location of static web pages, as well as static content such as images, fonts, video, and sounds. Any file located under this directory can be displayed in a web browser using the URL of the Raspberry Pi plus the name of the file (or path to the file). Initially, there is only one file in this directory, `index.html`. In the example it has the URL `http://192.168.1.79/index.html`. The `/var/www` directory is protected; use the `sudo` command to add, modify, or delete files in this directory.

The easiest way to add files from an external disk to the web server is to first mount the external disk in the `/media` directory and then create a symbolic link to the external disk in the root directory of the web server (`/var/www`). A symbolic link does not copy files, it just points to where files are located.

In the example, a website directory on the external disk (`/media/bigdisk/MyWebsite`) is linked into the web server's root directory at `/var/www/MyWebsite`. The example website has only one file, which is physically located on the external disk at `/media/bigdisk/MyWebsite/hello.html`. After the symbolic link is made, that file is also located symbolically at `/var/www/MyWebsite/hello.html` and can be reached from a web browser at the URL `http://192.168.1.79/MyWebsite/hello.html`.

The recipe continues by creating a simple dynamic web page that displays the current time. In the default Apache HTTP web server configuration, dynamic web pages use the **Common Gate Interface** (**CGI**) and are located in the `/usr/lib/cgi-bin` directory.

The `nano` editor is used to create the file `timestamp` in the `cgi-bin` directory. The directory is protected, so the `sudo` command is used as a prefix to temporarily grant the user the privilege of creating and editing the file.

The `timestamp` dynamic web page uses the BASH scripting language (`#!/bin/bash`). The web page first outputs a header describing the content of the web page as plain text (`echo "Content-Type: text/plain"`) followed by a blank line (`echo ""`) to separate the HTTP response header from the content of the web page. The last line of the web page displays the current time by executing the GNU `date` utility (`echo "$(date)"`).

In order for the script to be called dynamically, the execution bit of the file's mode needs to be set with the `chmod` command (`chmod a+rx`). The file is located in a protected directory, so the `sudo` command is used again to temporarily grant privileges.

Refreshing the web page (`http://192.168.1.79/cgi-bin/timestamp`) in the browser on another computer shows that the page is dynamic as each refresh of the browser updates the web page to the current time.

Finally, the contents of the web server's logfile directory (`/var/log/apache2/`) are displayed. Each request sent to the Apache HTTP web server is logged in the `access.log` file—one line per request. Error messages are appended to the end of the `error.log` file. If a dynamic web page does not display, or does not display properly, look at the end of the error log for a message describing the cause. All of the logfiles in this directory should be deleted (or archived) on a regular basis to free up disk space.

There's more...

The Apache HTTP web server is not the only web server that runs on the Raspberry Pi; however, it is perhaps the most famous and full-featured open source web server available today. Since 2009, more than 100 million websites have been hosted by Apache HTTP web servers. Although robust and complete, the Apache HTTP web server can also be resource heavy, consuming more memory and compute power than other servers.

lighttpd

The lighttpd web server is lighter on resources than Apache and has the potential to scale to a higher number of requests per second. Originally designed as a proof-of-concept to handle 10,000 parallel connections on one server, lighttpd has since become a popular web server. A very small bootable image containing lighttpd, PHP, and SQLite is distributed via BerryBoot (see *Chapter 1*, *Installation and Setup*).

The lighttpd web server can also be installed with the `apt-get install lighttpd` command. The command is privileged, so use `sudo` as a command prefix.

```
pi@raspberrypi ~ $ apt-cache search lighttpd | grep -v ^lib
collectd-core - statistics collection and monitoring daemon (core system)
lighttpd - fast webserver with minimal memory footprint
lighttpd-doc - documentation for lighttpd
lighttpd-mod-cml - cache meta language module for lighttpd
lighttpd-mod-magnet - control the request handling module for lighttpd
lighttpd-mod-mysql-vhost - MySQL-based virtual host configuration for lightt
pd
lighttpd-mod-trigger-b4-dl - anti-deep-linking module for lighttpd
lighttpd-mod-webdav - WebDAV module for lighttpd
pi@raspberrypi ~ $ sudo apt-get install lighttpd
```

In the preceding screenshot, the commands `apt-cache` and `grep` are used to display a list of `lighttpd` software packages; and the `apt-get` command is used to install `lighttpd`.

Although lighttpd does use `/var/www` as the root directory of the web server, it does not run CGI dynamic web pages by default. The `lighty-enable-mod cgi` command is used to enable the lighttpd CGI module (the command is privileged, so use `sudo`). The lighttpd CGI module expects the `cgi-bin` directory to be located in the root directory of the web server instead of in the `/usr/lib` directory. The `ln -s` command can be used to create a symbolic link from the web server's root directory to the default location for dynamic web pages (`/usr/lib/cgi-bin`).

```
pi@raspberrypi ~ $ sudo lighty-enable-mod cgi
Enabling cgi: ok
Run /etc/init.d/lighttpd force-reload to enable changes
pi@raspberrypi ~ $ sudo /etc/init.d/lighttpd force-reload
[ ok ] Reloading web server configuration: lighttpd.
pi@raspberrypi ~ $ sudo ln -s /usr/lib/cgi-bin /var/www/cgi-bin
pi@raspberrypi ~ $
```

The preceding screenshot shows how to use the `lighty-enable-mod` command to enable CGI dynamic web pages and how to use the `ln` command to create a symbolic link to the `cgi-bin` directory.

The lighttpd web server's configuration files are stored in the `/etc/lighttpd` directory, and its logfiles are written to the `/var/log/lighttpd` directory.

Nginx

Another popular modern web server is Nginx. Using an event-driven approach to handling requests, Nginx can provide more predictable performance under high loads than the Apache web server. Nginx has a number of options for deploying web applications including modular support for the popular web framework Ruby on Rails.

Nginx can be installed with the `apt-get install nginx` command. The command is privileged, so use `sudo` as a command prefix.

```
pi@raspberrypi ~ $ apt-cache search nginx
collectd-core - statistics collection and monitoring daemon (core system)
fcgiwrap - simple server to run CGI applications over FastCGI
gitweb - fast, scalable, distributed revision control system (web interface)
nginx - small, powerful, scalable web/proxy server
nginx-common - small, powerful, scalable web/proxy server - common files
nginx-doc - small, powerful, scalable web/proxy server - documentation
nginx-extras - nginx web/proxy server (extended version)
nginx-extras-dbg - nginx web/proxy server (extended version) - debugging sym
bols
nginx-full - nginx web/proxy server (standard version)
nginx-full-dbg - nginx web/proxy server (standard version) - debugging symbo
ls
nginx-light - nginx web/proxy server (basic version)
nginx-light-dbg - nginx web/proxy server (basic version) - debugging symbols
nginx-naxsi - nginx web/proxy server (version with naxsi)
nginx-naxsi-dbg - nginx web/proxy server (version with naxsi) - debugging sy
mbols
nginx-naxsi-ui - nginx web/proxy server - naxsi configuration front-end
ruby-passenger - Rails and Rack support for Apache2 and Nginx
stud - scalable TLS unwrapping daemon
pi@raspberrypi ~ $ sudo apt-get install nginx
```

The preceding screenshot shows how to use the `apt-cache` command to display a list of Nginx software packages and how to use the `apt-get` command to install the `nginx` package.

The Nginx web server does not use the same root directory as the Apache HTTP and lighttpd web servers; instead, static web pages are stored in the `/usr/share/ngix/www` directory. The Nginx web server also does not support CGI; however, the Nginx web server can produce dynamic web pages using FastCGI or Ruby Passenger.

The Nginx web server's configuration files are stored in the `/etc/nginx` directory, and its logfiles are written to the `/var/log/nginx` directory.

In addition to Apache, lighttpd, and Nginx, the Raspberry Pi can also run a number of lesser-known web servers including AOLserver, Node.js, and Tntnet.

```
pi@raspberrypi ~ $ apt-cache search httpd | grep -i server | grep -v '^lib'
aolserver4-core - AOL web server version 4 - core libraries
aolserver4-daemon - AOL web server version 4 - program files
apache2-mpm-event - Apache HTTP Server - event driven model
apache2-mpm-prefork - Apache HTTP Server - traditional non-threaded model
apache2-mpm-worker - Apache HTTP Server - high speed threaded model
boa - Lightweight and high performance web server
bozohttpd - Bozotic HTTP server
ebhttpd - specialized HTTP server to access CD-ROM books
lighttpd - fast webserver with minimal memory footprint
mathopd - Very small, yet very fast HTTP server
micro-httpd - really small HTTP server
mini-httpd - a small HTTP server
monkey - fast, efficient, small and easy to configure web server
nginx-extras - nginx web/proxy server (extended version)
nginx-full - nginx web/proxy server (standard version)
nginx-light - nginx web/proxy server (basic version)
nginx-naxsi - nginx web/proxy server (version with naxsi)
nginx-naxsi-ui - nginx web/proxy server - naxsi configuration front-end
ocsigen - web server and programming framework in OCaml
ocsigenserver - web server of the Ocsigen project
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
tntnet - modular, multithreaded web application server for C++
webfs - lightweight HTTP server for static content
yaws - High performance HTTP 1.1 webserver written in Erlang
pi@raspberrypi ~ $
```

In the preceding screenshot, `apt-cache` and `grep` are used to display a list of available HTTP web servers.

See also

- **bash – GNU Bourne-Again SHell**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=bash`

  The Debian man page for `bash`.

- **chmod – change file mode bits**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=chmod`

  The Debian man page for `chmod`.

- **date – print or set the system date and time**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=date`

  The Debian man page for `date`.

- **grep – print lines matching a pattern**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=grep`

  The Debian man page for `grep`.

- **ln -s – make links between files**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=ln`

  The Debian man page for `ln`.

- **AOLserver**

  `http://www.aolserver.com`

  The AOLserver website.

- Apache HTTP Server Project

  `http://httpd.apache.org`

  The Apache web server website.

- **Bash – the GNU Bourne-Again SHell**

  `http://tiswww.case.edu/php/chet/bash/bashtop.html`

  The Bash maintainer's web page.

- **lighttpd – fly light**

  `http://www.lighttpd.net/`

  The lighttpd website.

- **Nginx**

  `http://nginx.org/en`

  The Nginx website.

- **Node.js**

  `http://nodejs.org`

  The `Node.js` website.

- **PHP**

  http://php.net

  The PHP website.

- **SQLite**

  `http://www.sqlite.org/`

  The SQLite website.

- **tntnet**

  `http://www.tntnet.org`

  The tntnet website.

Installing a wiki (MediaWiki)

This recipe installs the same wiki used by Wikipedia.

Wikis are useful collaborative environments for teams that share a continuously evolving set of documentation. Each team member can contribute to the creation and editing of pages within the wiki. The resulting pages of documentation stored in the wiki are the result of a group effort and reflect the team's combined knowledge.

This recipe is used to install the MediaWiki. This is the same wiki software used by Wikipedia. The Wikipedia website is hosted by a cluster of high-powered servers so that it can continuously serve millions of users. This recipe uses a single Raspberry Pi that has more than enough power to manage a wiki for a team or small office of collaborators.

MediaWiki is a web application. Users of a web application generally access the application via a web browser. The web application is hosted on a web server that the web browser accesses via a link or URL. Within the web server is a scripting language that defines the layout of the application web pages and the implementation of the application services. Most web applications use a database to store user data together with the application's configuration. A MediaWiki installation is configured by default to use Apache2 as the web server and MySQL as the database. MediaWiki is written in the PHP5 scripting language.

After completing this recipe, you will have a wiki that is ready for collaboration.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD
- A network connection
- Another PC on the same network (for test and configuration)

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running, the installation can be completed remotely using a Secure Shell client.

Once installed, the MediaWiki is accessed and configured from a web browser. Configuration can be done from another PC on the same network as the Raspberry Pi.

For better performance, the video memory of the Raspberry Pi should be set as low as possible (see *Chapter 2*, *Administration*) so that there is more memory available for MediaWiki.

How to do it...

The steps for installing the MediaWiki wiki server are:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command: `apt-get install mediawiki`.

   This command needs to be run as a privileged user (use `sudo`).

The package management application `aptitude` can be used as an alternative to the command utility `apt-get` for downloading and installing software distribution packages.

```
pi@raspberrypi ~ $ sudo apt-get install mediawiki
```

The preceding screenshot shows the `apt-get` install command that is used to install the MediaWiki software package (`mediawiki`).

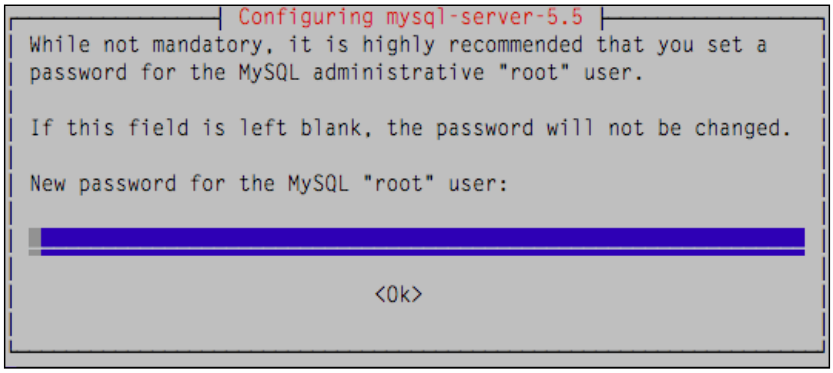
3. The `apt-get install` command downloads and installs MediaWiki.

   The installation of `mediawiki` includes `apache2`, `mysql`, and `php5`.

   The complete installation time will be at least 15 - 30 minutes.

4. During install, the `mysql` package will prompt for a new root password that should be used when managing the database.

   Enter a root password and repeat it on the next screen.



   In the preceding screenshot, the installation of `mysql` prompts for a root password.

5. After accepting the root password for the `mysql` database (twice), the installation of `mediawiki` completes.

6. Execute the following command:

   `nano /etc/mediawiki/apache.conf`

   Edit the MediaWiki website configuration file.

   The configuration file is protected (use `sudo`).

```
pi@raspberrypi ~ $ sudo nano /etc/mediawiki/apache.conf
```

   The preceding screenshot shows the `nano` command for editing the Apache web server configuration.

7. Uncomment the `Alias` for the MediaWiki website by removing the # from the beginning of the line. Save and close the file.

```
  GNU nano 2.2.6       File: /etc/mediawiki/apache.conf          Modified

# Uncomment this to add an alias.
# This does not work properly with virtual hosts..
Alias /mediawiki /var/lib/mediawiki
<Directory /var/lib/mediawiki/>
```

   In the preceding screenshot, the `nano` editor is used to uncomment the website alias for the MediaWiki by removing the # from the beginning of the line.

8. Execute the following command:

   `apachectl restart.`

   Restart the Apache web server.

   The command is privileged (use `sudo`).

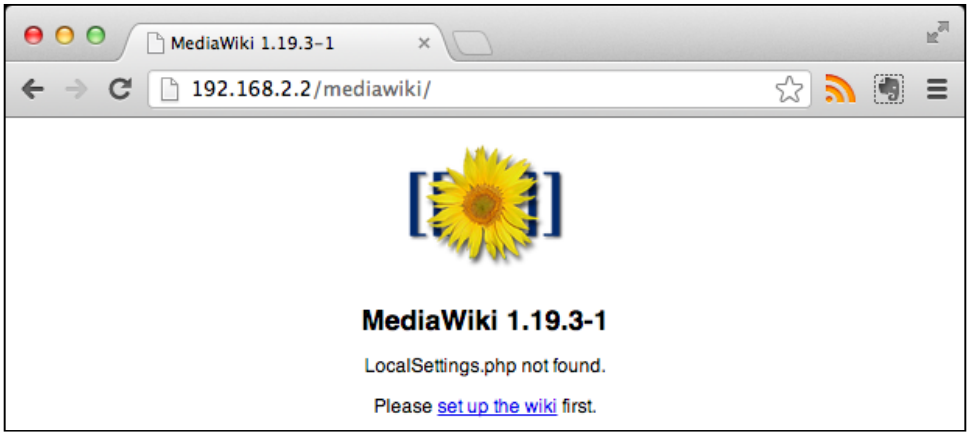
```
pi@raspberrypi ~ $ sudo apachectl restart
apache2: Could not reliably determine the server's fully qualified domain na
me, using 127.0.1.1 for ServerName
pi@raspberrypi ~ $
```

   The preceding screenshot shows how to restart the Apache web server.

9. Once the Apache web server has restarted, the configuration of MediaWiki continues in the web browser.

10. Open a web browser (possibly on another PC) and browse to the URL of the MediaWiki: `http://192.168.2.2/mediawiki/`.

The IP address (`192.168.2.2`) of your Raspberry Pi will be different. The `ipconfig` command can be used to display the IP address of the Raspberry Pi—see the *Remote Access (ssh)* recipe in *Chapter 2, Administration*, for an example of using the `ipconfig` command.



The preceding screenshot shows the web page that says MediaWiki is ready for the initial configuration.

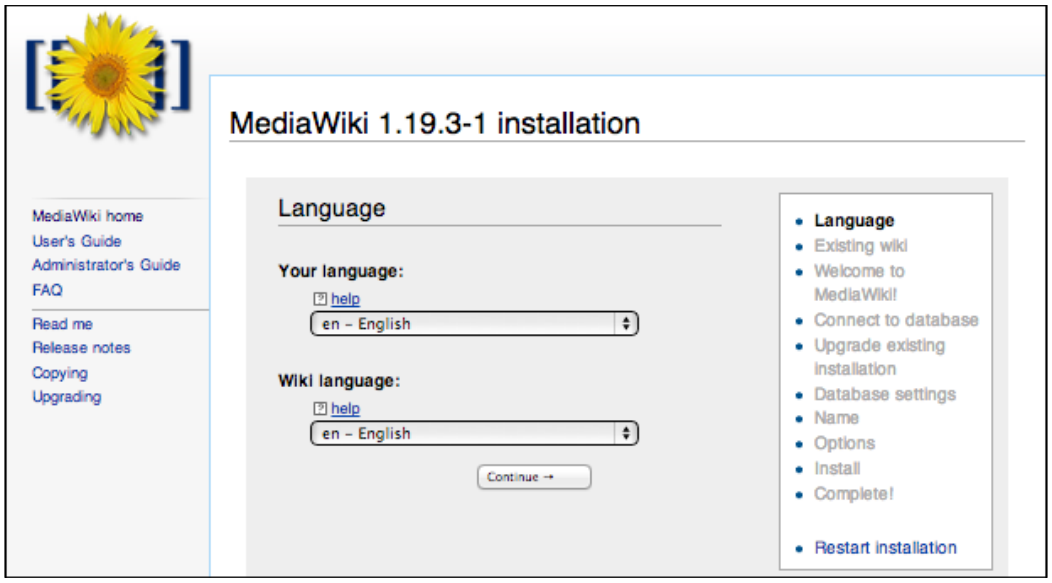
11. The MediaWiki initial configuration page is displayed in the browser.

    MediaWiki is ready to be configured.

    Click the **set up the wiki** link to continue.

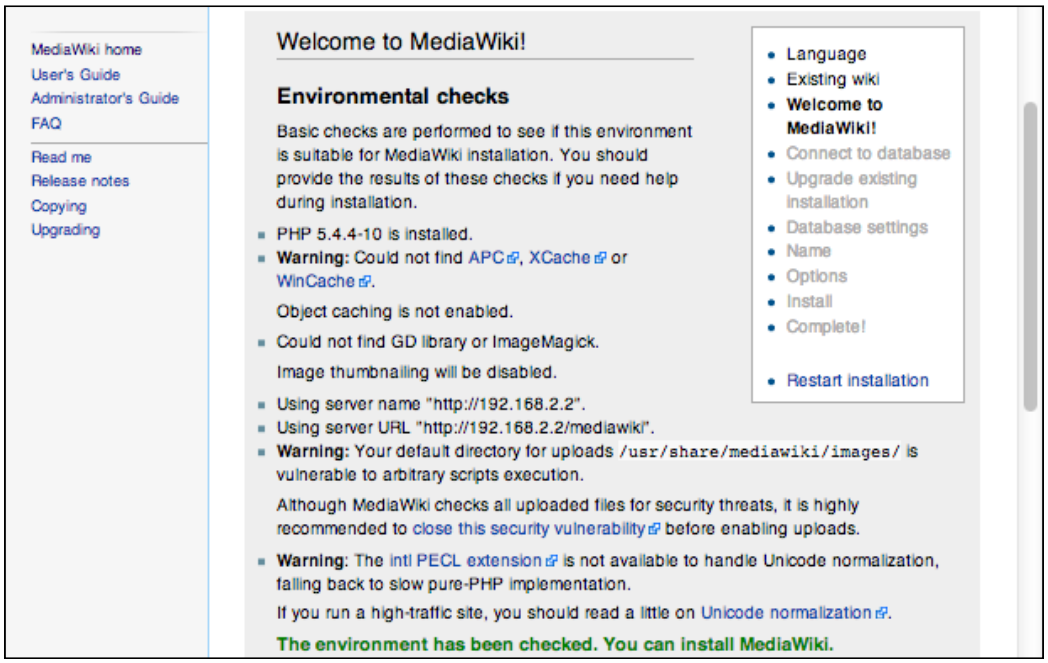
12. Choose the language to be used by MediaWiki.

    Click **Continue ->** for the next configuration page.



The web page in the preceding screenshot shows how to configure the languages used by the MediaWiki.

13. Once the language has been set, MediaWiki runs a number of tests to determine if (and how) it can be set up on the Raspberry Pi.



The preceding screenshot shows a web page verifying that the Raspberry Pi is ready for MediaWiki.

14. The configuration tests have completed successfully. **You can install MediaWiki** (if you do not see this green message, click **Restart installation**).

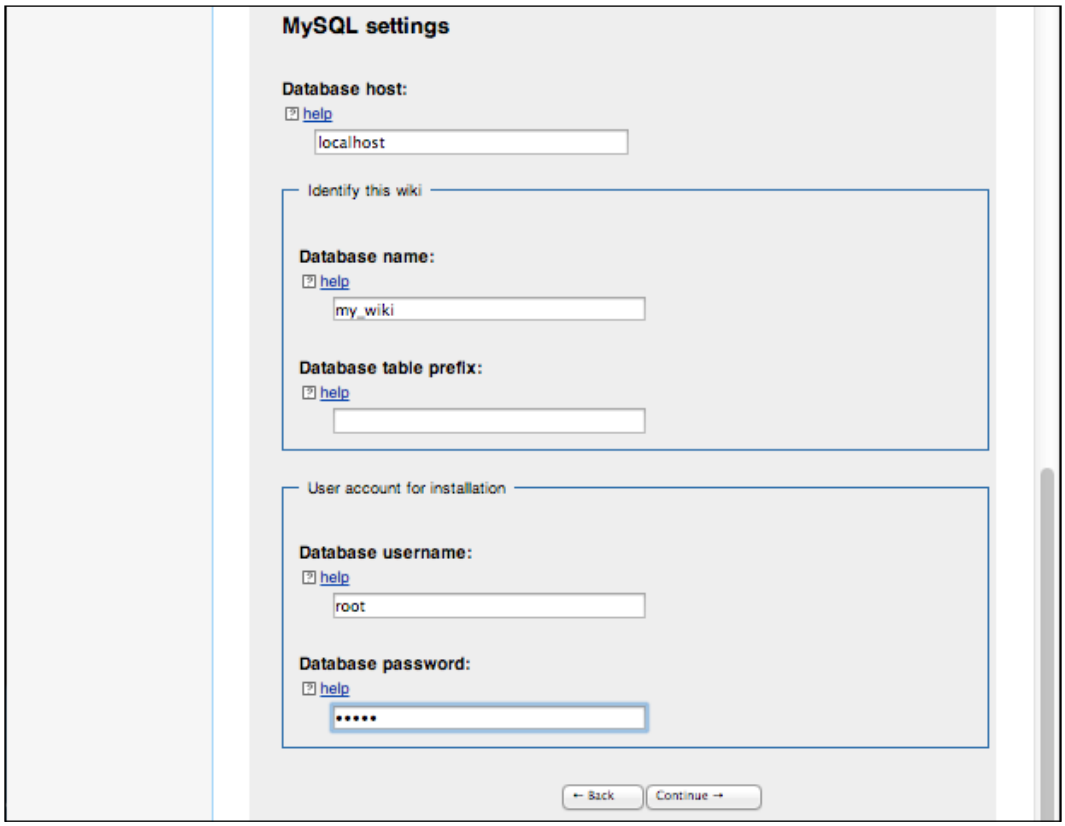
15. Scroll to the bottom of the page.

    Click **Continue ->** to accept the license and continue the installation.

    The preceding screenshot shows how to accept the MediaWiki license agreement.

16. Once the license agreement has been accepted, configuration continues with the MySQL settings.

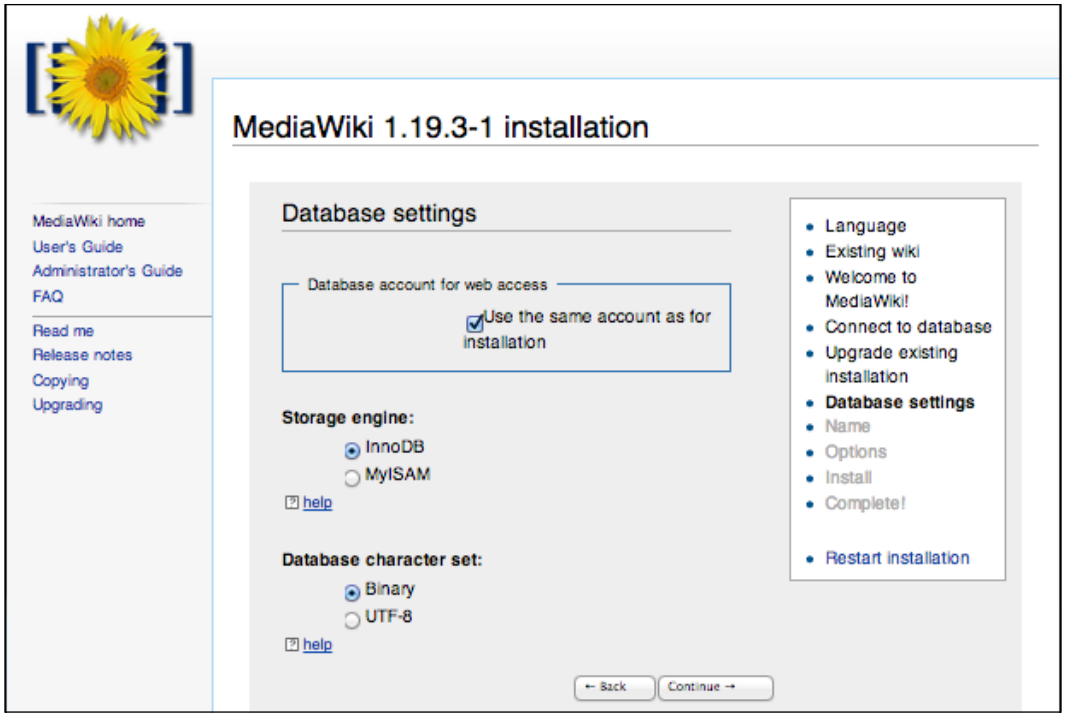
17. Enter the password for the MySQL root user from Step 4.



The preceding screenshot shows how to enter the MySQL database settings.

18. After the database name and root password for MySQL have been entered, configuration continues with some additional database setting for web access.

19. Accept the defaults by clicking **Continue ->**.



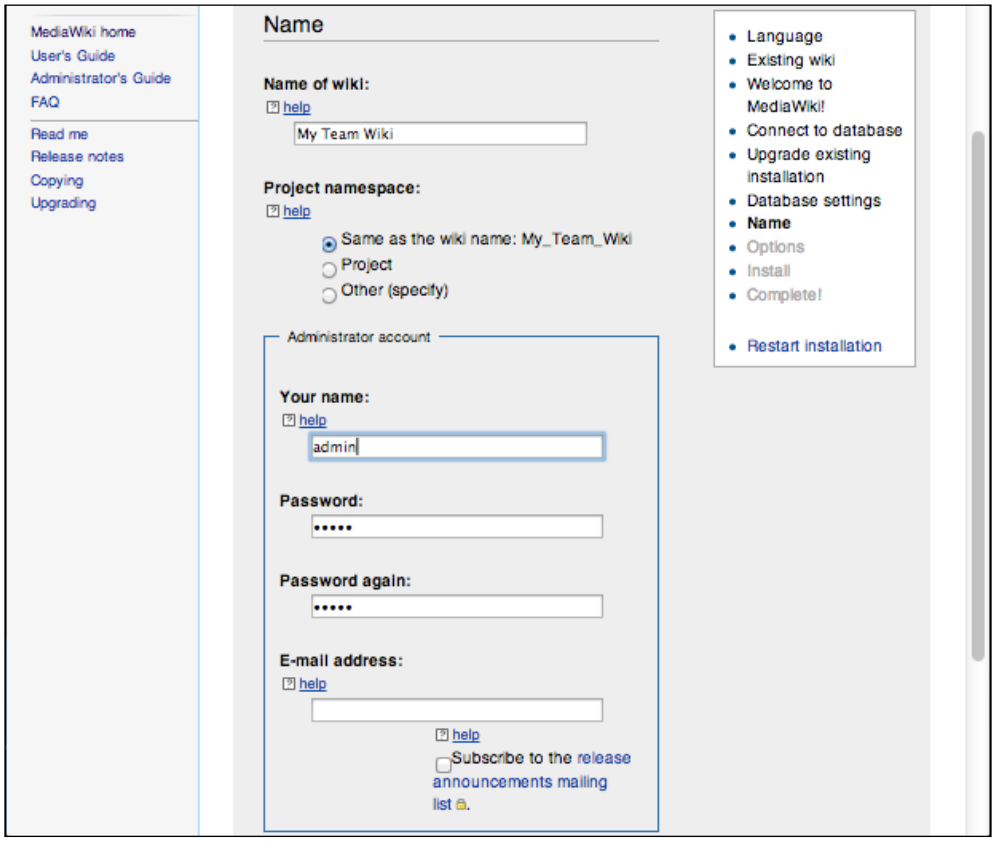
The preceding screenshot shows how to accept the default database settings.

20. After the default database settings have been accepted, configuration continues by entering the name of the wiki and the username and password for the Administrator Account.

21. Enter a **Name** for the wiki.

    Enter a username and password for the MediaWiki **Administrator Account**.

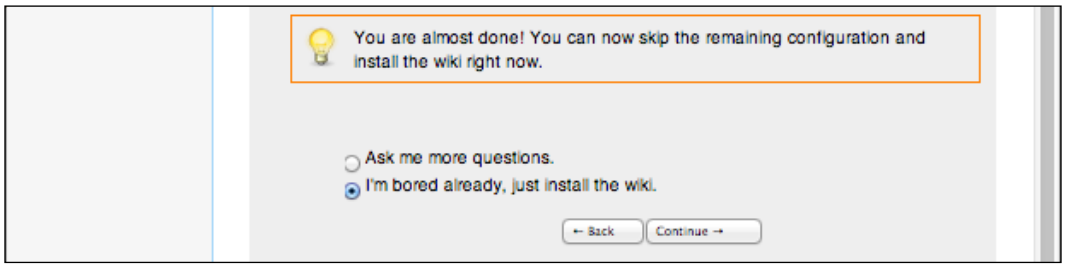
    Click **Continue ->.**



The preceding screenshot shows the web page used to enter the **Name of wiki** and the username and password for the **Administrator Account**.

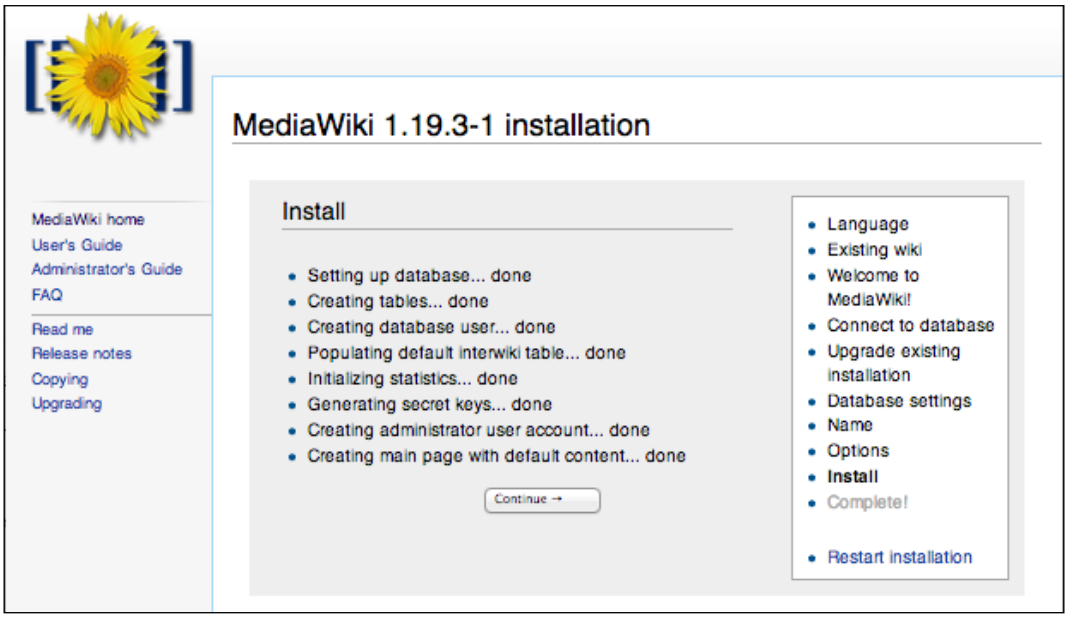
22. After the wiki name and the MediaWiki Administrator's username and password have been entered, scroll to the bottom of the page.

23. Choose **I'm bored already, just install the wiki** and click **Continue ->**.



The preceding screenshot shows how to continue the MediaWiki installation without configuring optional settings.

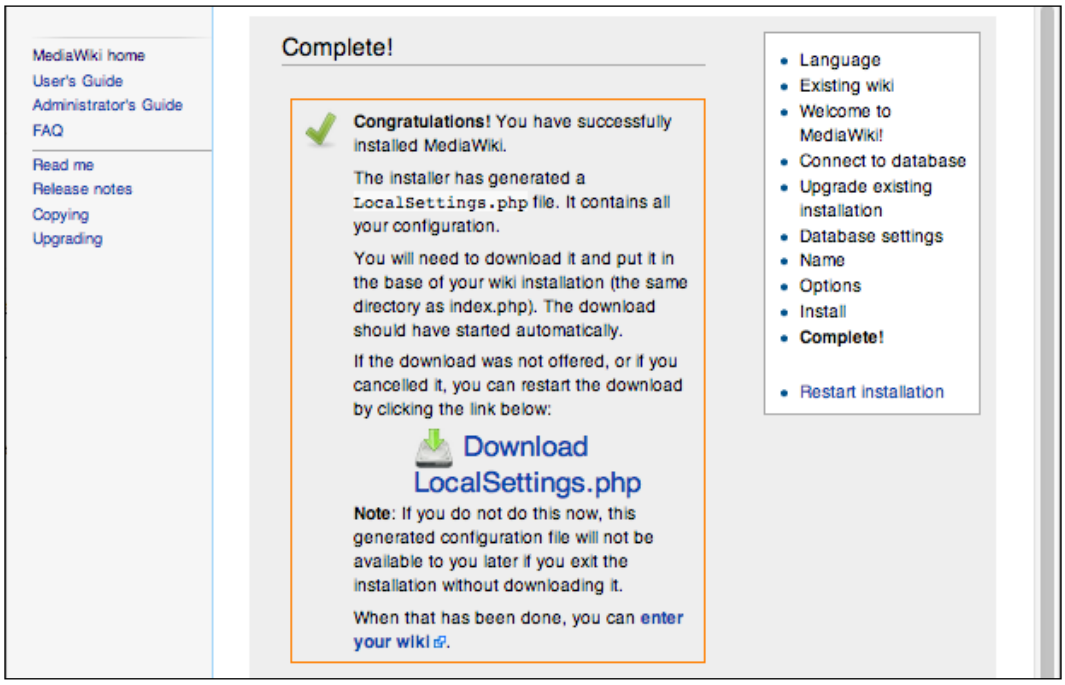
24. MediaWiki uses the supplied configuration parameters to set up the database displaying a web page listing the completed installation steps.
25. Click **Continue ->**.



The preceding screenshot shows the web page that displays the list of completed installation steps.

26. After the displayed installation list has been accepted a congratulation page is displayed saying that the installation is complete. However, the `LocalSettings.php` file still needs to be downloaded and installed.

27. Download the `LocalSettings.php` file.



The preceding screenshot shows the web page that is used to download the `LocalSettings.php` file.

28. Once the `LocalSettings.php` file has been downloaded, it needs to be moved to the root of the MediaWiki installation on the Raspberry Pi.

If the browser used for configuration was on another PC, the `LocalSettings.php` file will need to be copied to the Raspberry Pi.

If the other PC is running Linux or MacOS, execute the `scp LocalSettings.php pi@192.168.2.2` command to copy the downloaded `LocalSettings.php` file to the Raspberry Pi.

```
golden@golden-imac:Downloads $ scp LocalSettings.php pi@192.168.2.2:
pi@192.168.2.2's password:
LocalSettings.php                             100% 4512     4.4KB/s   00:00
golden@golden-imac:Downloads $
```

In the preceding screenshot, `scp` is used to copy the `LocalSettings.php` file to the Raspberry Pi.

29. On the Raspberry Pi, execute the following command:

```
mv LocalSettings.php /var/lib/mediawiki
```

Move the `LocalSettings.php` file to the root of the MediaWiki (`/var/lib/mediawiki`).

The MediaWiki root is protected (use `sudo`).

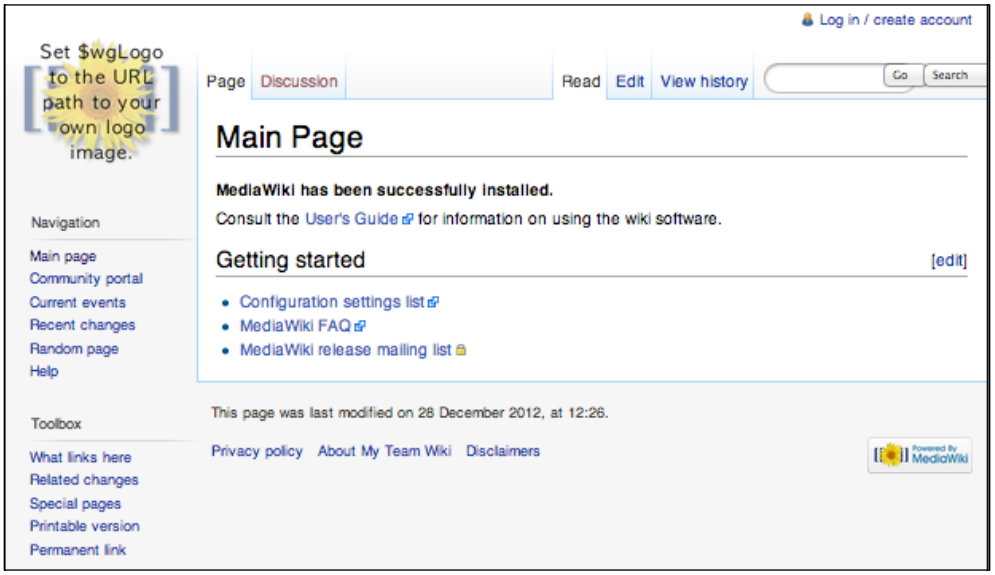
```
pi@raspberrypi ~ $ sudo mv LocalSettings.php /var/lib/mediawiki
pi@raspberrypi ~ $
```

The preceding screenshot shows how to move the `LocalSettings.php` file to the MediaWiki root (`/var/lib/mediawiki`).

30. After the `LocalSettings.php` file has been moved to the root of the MediaWiki installation, the MediaWiki is ready to use!
31. Browse to the MediaWiki URL on the Raspberry Pi:

```
http://192.168.2.2/mediawiki/
```

32. The MediaWiki is set up and running; however, it still needs to be customized for your team (by adding a custom team logo, fonts, and styles). Follow the `User's Guide` link for more information on customizing your installation and detailed instructions on how to use the MediaWiki.



The preceding screenshot shows that MediaWiki is up and running. Click **User's Guide** for detailed instructions on using the wiki.

How it works...

At the beginning of this recipe, the MediaWiki software distribution package (`mediawiki`) is downloaded and installed using `apt-get install`. The MediaWiki installation includes the installation of Apache2, MySQL, and PHP5. Each of these packages can also be installed separately to provide a web server (Apache2), a database (MySQL), or a scripting language (PHP5).

The MediaWiki installation begins by installing Apache2, MySQL, and PHP5. Neither the Apache2 installation nor the PHP5 installation requires any user input to complete successfully. However, the MySQL installation requires a password for the root user of the database.

The MySQL administrative "root" password is for the setup and configuration of the MySQL database management server. It is not the same password used by the user pi. The security of the Raspberry Pi is improved if the password for the user pi and the password for the database are different.

After the MySQL "root" password has been entered (twice), the installation of Apache2, MySQL, PHP5, and the MediaWiki application files continues and completes. All the MediaWiki application files are installed and the Apache2 web server is installed and running. However, the Apache2 web server has not been configured to host the MediaWiki.

The nano editor is used to edit the `apache.conf` configuration file and assign the website that the MediaWiki will use. The configuration file is located in the MediaWiki configuration directory, `/etc/mediawiki/`, and is also linked to the web server configuration directory, `/etc/apache2/conf.d/`.

The only line that needs to be changed in the configuration file is the line that begins `#Alias`. The `#` at the beginning of the line turns the line into a comment. Comment lines are ignored when the Apache web server reads the configuration file. Removing the `#` from the beginning of the line will activate the MediaWiki the next time the web server is started.

The Alias statement has two parameters: the website's URL prefix (`/mediawiki`) and the location of the website's application files (`/var/lib/mediawiki`). The URL prefix becomes part of the wiki's URL (`http://192.168.2.2/mediawiki/`). Changing the URL prefix would also change the URL used to access the wiki. Changing the prefix to `/TeamWiki` would change the wiki's URL to `http://192.168.2.2/TeamWiki/`.

After the web server configuration file has been edited and the alias line has been uncommented, the Apache web server is restarted with the `apachectl restart` command. This privileged command is used to restart the Apache web server.

Restarting the web server forces it to reload its configuration files, including MediaWiki's web server configuration file. Now that MediaWiki's alias has been uncommented, the MediaWiki website is available; however, the wiki's database still needs to be set up.

Now that the MediaWiki website is running, a web browser (possible on another PC) can be used to complete the wiki's database setup. The example URL is `http://192.168.2.2/mediawiki/`. The IP address of your Raspberry Pi will most likely be different. *Chapter 2*, *Administration*, has a recipe for displaying a Raspberry Pi's IP address.

The first page displayed by the MediaWiki says the `LocalSettings.php` file cannot be found. The link labeled `setup the wiki` leads to the web pages that are used to complete the setup of the wiki. Click on **setup the wiki**.

The first setup page is used to define the language used by the wiki administrator (**Your language**) and the language used by other wiki users (**Wiki language**).

The next setup page shows the results of a system configuration examination and a license that must be accepted to complete the setup.

Then, the database settings are entered. The password entered earlier for the MySQL administrative "root" users should be entered here.

Some additional database settings appear on the next screen. Accept the defaults.

Finally, there is a setup page that asks for the wiki name and a username and password for the MediaWiki administrator. This new password is only used for configuring the wiki and is neither the MySQL administrative "root" password nor the password of the user pi.

At the bottom of the page is an opportunity to enter more configuration data. The example in the recipe chose **I'm bored already, just install the wiki** and clicked **Continue ->** to complete the database installation.

After a status page is displayed, the last page of the setup triggers the download of the `LocalSettings.php` file, which is used to complete the installation of the wiki. If the file was downloaded on another PC, the scp (if available) could be used to copy the file to the Raspberry Pi as shown in the example.

The `LocalSettings.php` file (written in the PHP5 scripting language) is a summary of the configuration data that was entered into the web browser. This file needs to be moved to the MediaWiki installation directory (`/var/lib/mediawiki/`) to complete the setup of the wiki.

After the `LocalSettings.php` file has been moved to the MediaWiki installation directory, the MediaWiki **Main Page** is displayed at the configured URL (`http://192.168.2.2/mediawiki/`). Some customization could still be done to personalize the wiki (for example, change the wiki logo); however, the general setup is complete and the wiki can now be used for collaboration.

More information on using and customizing MediaWiki can be found by following the **User's Guide** link on the **Main Page** of the wiki.

See also

- **MediaWiki**

  `http://en.wikipedia.org/wiki/Mediawiki`

  A Wikipedia about the MediaWiki wiki server.

- **apachectl – Apache HTTP server control interface**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=apachectl`

  The Debian man page for `apachectl`.

- **mv – move (rename) files**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=mv`

  The Debian man page for `mv`.

- **scp – secure copy (remote file copy program)**

  `http://manpages.debian.net/cgi-bin/man.cgi?query=scp`

  The Debian man page for `scp`.

Creating a wireless access point with hostapd

This recipe sets up the Raspberry Pi as a wireless access point—as a hub that other wireless devices can use to connect to the local network—or to services that are running on the Raspberry Pi.

Out of the box, the "official" Raspbian Linux distribution supports connecting the Raspberry Pi to an existing wireless network using a wireless USB adapter. This recipe goes beyond using the Raspberry Pi as a network client and instead configures the Raspberry Pi to be a network hub for other wireless devices. If it is also connected to a local network via a TCP cable, the Raspberry Pi can also act as a wireless network router enabling other devices to connect to the local network through the Raspberry Pi.

Not all Wireless USB Adapters support Access Point (AP) mode. The recipe begins with a test to ensure that the wireless adapter is supported.

After completing this recipe, you will be able to create a wireless network with the Raspberry Pi acting as a network hub.

Getting ready

The following are the ingredients:

- A Raspberry Pi with a 5V power supply
- An installed and configured "official" Raspbian Linux SD card
- A supported wireless USB adapter
- A powered USB hub
- A network connection

This recipe does not require the desktop GUI and could either be run from the text-based console or from within an `LXTerminal`.

If the Raspberry Pi's Secure Shell server is running, this recipe can be completed remotely using a Secure Shell client.

There are a limited number of wireless USB adapters that can work with the Raspberry Pi and can also be configured as wireless access points. The links at the end of this chapter can be used to find current wireless USB adapters that can be used together with the Raspberry Pi to create a wireless access point.

Some wireless USB adapters consume more power than the Raspberry Pi can support consistently on a continual basis. Connecting the wireless USB adapter to the Raspberry Pi indirectly via a USB hub will lead to better performance and reduce the likelihood that other USB devices (such as the onboard network card!) will be starved for power.

How to do it...

The steps for configuring the Raspberry Pi as a wireless access point are as follows:

1. Log in to the Raspberry Pi either directly or remotely.
2. Execute the following command:

   `apt-get install hostapd udhcpd.`

   This command needs to be run as a privileged user (use `sudo`).

The package management application `aptitude` can be used as an alternative to the `apt-get` command utility for downloading and installing software distribution packages.

```
pi@raspberrypi ~ $ sudo apt-get install hostapd udhcpd
```

The preceding screenshot shows how the `apt-get install` command is used to install the `hostapd` and `udhcpd` software distribution packages.

3. The `apt-get install` command downloads and installs the software packages `hostapd` and `udhcpd`.

   Check your wireless device using the following steps:

4. Execute the following command: `iw list |less`

```
pi@raspberrypi ~ $ iw list |less
```

   The preceding screenshot shows how to use the `less` command to page through the list of wireless interface capabilities.

5. The `iw list` command displays a list of wireless capabilities.

```
        Supported interface modes:
                 * IBSS
                 * managed
                 * AP
                 * AP/VLAN
                 * WDS
                 * monitor
                 * mesh point
        software interface modes (can always be added):
                 * AP/VLAN
                 * monitor
```

   The preceding screenshot shows how to use the `iw list` command to verify that the Access Point (**AP**) interface mode is supported.

6. Execute the following command: `cat /etc/resolv.conf`

```
pi@raspberrypi ~ $ cat /etc/resolv.conf
nameserver 192.168.2.1
pi@raspberrypi ~ $
```

   The preceding screenshot shows how to use the `cat` command to display the Raspberry Pi's `nameserver` address that is stored in `/etc/resolve.conf`.

7. The DNS `nameserver` used by the Raspberry Pi is displayed (`192.168.2.1`).

   To configure the DHCP server (`udhcpd`), follow these steps:

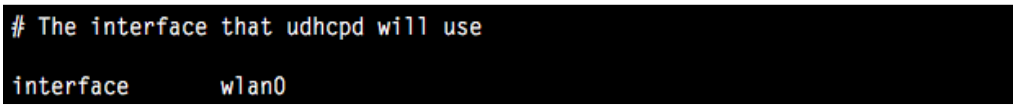
8. Execute the command:

   `nano /etc/udhcpd.conf`

   Files in the `/etc` directory are protected (use `sudo`).

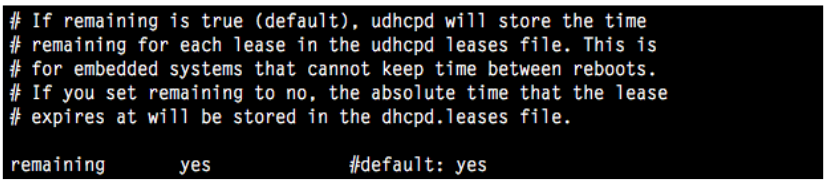
   The preceding screenshot is the `nano` command used to edit the configuration file for `udhcpd`.

9. The `nano` editor opens the `udhcpd` configuration file (`/etc/udhcpd.conf`).
10. Change the line beginning with `interface` to read `interface wlan0`



   The preceding screenshot shows how to change the `udhcpd` network interface to be `wlan0`.

11. Uncomment the line beginning with `#remaining` by removing the `#`.



   The preceding screenshot show how to uncomment the remaining time configuration in udhcpd.

12. Change the section beginning with `#Examples` to read:

```
#Examples
opt     dns     192.168.2.1
option  subnet  255.255.255.0
opt     router  192.168.0.1
option  domain  local
option  lease   864000
```

Replace the example DNS address (`192.168.2.1`) with the `nameserver` address displayed by your Raspberry Pi in step 4.

```
#Examples
opt     dns     192.168.2.1
option  subnet  255.255.255.0
opt     router  192.168.0.1
option  domain  local
option  lease   864000          # 10 days of seconds
```

The preceding screenshot shows the options in the uhdcpd configuration file.

13. Save the file and exit the editor.
14. Execute the following command:

    `nano /etc/default/udhcpd`

    Files in the `/etc` directory are protected (use `sudo`).

```
pi@raspberrypi ~ $ nano /etc/default/udhcpd
```

The preceding screenshot shows the `nano` command for editing the service defaults file of the `udhcpd` service.

15. The `nano` editor opens the service defaults for the `udhcpd` service (`/etc/default/udhcpd`).
16. Comment the line `DHCP_ENABLED="no"` by placing a `#` at the beginning of the line.

```
# Comment the following line to enable
#DHCPD_ENABLED="no"
```

The preceding screenshot shows how to comment (`#`) the `DHCPD_ENABLED="no"` line so that `udhcpd` is enabled.

17. Save the file and exit the editor.

To configure the Wireless Access Point server (hostapd), follow these steps:

1. Execute the command:

   `nano /etc/hostapd/hostapd.conf`

   Files under the `/etc` directory are protected (use `sudo`).

```
pi@raspberrypi ~ $ sudo nano /etc/hostapd/hostapd.conf
```

The preceding screenshot show the `nano` command for creating a configuration file for `hostapd`.

2. The `nano` editor creates a configuration file for `hostapd` (`/etc/hostapd/hostapd.conf`).
3. Edit the file so that it contains the following:

```
interface=wlan0
driver=nl80211
ssid=Raspi_AP
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Pr0t3ct3d
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

For the sake of security, remember to change ssid and wpa_passphrase to something else for your own network! We'll continue assuming you use the above.

```
interface=wlan0
driver=nl80211
ssid=RasPi_AP
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Pr0t3ct3d
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

The preceding screenshot shows the contents of the configuration for `hostapd`.

4. Save the file and exit the editor.
5. Execute the following command:

   `nano /etc/default/hostapd`

   Files in the `/etc` directory are protected (use `sudo`).

```
pi@raspberrypi ~ $ sudo nano /etc/default/hostapd
```

   The preceding screenshot shows the `nano` command for editing the service defaults file for the `hostapd` service.

6. The `nano` editor opens the service defaults for the `hostapd` service (`/etc/default/hostapd`).
7. Replace the line beginning with `#DAEMON_CONF` with the line:

   `DAEMON_CONF="/etc/hostapd/hostapd.conf"`

```
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configura$
# file and hostapd will be started during system boot. An example configura$
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

   The preceding screenshot shows how to uncomment and configure the `DAEMON_CONF` variable that enables the `hostapd` daemon.

8. Save the file and exit the editor.

   To Set up IP forwarding follow these steps:

9. Execute the following command:

   `nano /etc/sysctl.conf`

   Files in the `/etc` directory are protected (use **`sudo`**).

```
pi@raspberrypi ~ $ sudo nano /etc/sysctl.conf
```

   The preceding screenshot shows the `nano` command for editing the kernel parameters.

10. The `nano` editor opens the kernel parameters file (`/etc/sysctl.conf`).

11. Uncomment the line beginning with `net.ipv4.ip_forward` by removing the `#` from the beginning of the line. The line should read:

    `net.ipv4.ip_forward=1`

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

    The preceding screenshot shows how to uncomment the kernel parameters for IP packet forwarding.

12. Save the file and exit the editor.
13. Execute the following commands:

    ```
    iptables –t nat –A POSTROUTING –o eth0 –j MASQUERADE
    iptables –A FORWARD –i eth0 –o wlan0 –m state –state
    RELATED,ESTABLISHED –j ACCEPT
    iptables –A FORWARD –i wlan0 –o eth0 –j ACCEPT
    ifconfig wlan0 192.168.0.1
    ```

    `sh –c "iptables-save > /etc/iptables.ipv4.nat"`

    Each of these commands is privileged (use `sudo`).

```
pi@raspberrypi ~ $ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
pi@raspberrypi ~ $ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --stat
e RELATED,ESTABLISHED -j ACCEPT
pi@raspberrypi ~ $ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
pi@raspberrypi ~ $
pi@raspberrypi ~ $ sudo ifconfig wlan0 192.168.0.1
pi@raspberrypi ~ $
pi@raspberrypi ~ $ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
pi@raspberrypi ~ $
```

    The preceding screenshot shows the commands for enabling IP forwarding and setting a static IP address for the wireless network interface (`wlan0`).

14. The wireless network interface is assigned a static IP address (`192.168.0.1`) and the rules for forwarding network traffic between the wireless network (`wlan0`) and the wired network (`eth0`) are configured and saved.

To configure the boot parameters, follow these steps:

1. Execute the following command:

   `nano /etc/network/interfaces`

   Files under the `/etc` directory are protected (use `sudo`).

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
```

   The preceding screenshot shows the `nano` command for editing the network interface configuration file.

2. The `nano` editor opens the network interface configuration file (`/etc/network/interfaces`).
3. Change the file so that it only contains the following lines:

```
auto lo
iface lo    inet loopback
iface eth0  inet dhcp
iface wlan0 inet static
  address 192.168.0.1
  netmask 255.255.255.0
up iptables-restore < /etc/iptables.ipv4.nat
```

```
auto lo

iface lo   inet loopback

iface eth0 inet dhcp

iface wlan0 inet static
  address 192.168.0.1
  netmask 255.255.255.0

up iptables-restore < /etc/iptables.ipv4.nat
```

   The preceding screenshot shows the network interface configuration.

4. Save the file and exit the editor.

   To Start the Wireless Access Point follow these steps:

5. Execute the following commands:

```
update-rc.d hostapd enable
update-rc.d udhcpd enable
```

   The `update-rc.d` command is privileged (use `sudo`).

```
pi@raspberrypi ~ $ sudo update-rc.d hostapd enable
update-rc.d: using dependency based boot sequencing
pi@raspberrypi ~ $ sudo update-rc.d udhcpd enable
update-rc.d: using dependency based boot sequencing
pi@raspberrypi ~ $
```

The preceding screenshot shows how to use `update-rc.d` to configure the Raspberry Pi to start `hostpad` and `udhcp` during boot.

6. The Raspberry Pi is configured to start `hostapd` and `udhcp` during boot.
7. Execute the following commands:

```
service hostapd start
service udhcpd start
```

The `service` command is privileged (use `sudo`).

```
pi@raspberrypi ~ $ sudo service hostapd start
pi@raspberrypi ~ $ sudo service udhcpd start
Starting very small Busybox based DHCP server: Starting /usr/sbin/udhcpd...
udhcpd.
pi@raspberrypi ~ $
```

The preceding screenshot shows how to start the `hostpad` and `udhcpd` services.

8. The `hostapd` and `udhcpd` services are started.
9. The Raspberry Pi is now a secure wireless access point with the SSID `Raspi_AP` and the password `Pr0t3ct3d`! This is unless you changed the configuration file earlier on.

How it works...

The recipe starts by downloading and installing two servers: a Wireless Access Point server (`hostapd`) and a DHCP server (`udhcpd`).

Before configuration begins, the wireless USB adapter is tested for compatibility by listing the available features of the adapter.

Check the wireless USB adapter

Not all wireless USB adapters support Access Point (AP) mode. The `iw list` command is used to list the wireless capabilities of any attached wireless devices. If AP mode is supported, it will be listed in the section `Supported interface modes`.

The output of the `iw list` command is quite long. The `less` command is used to page through the output of the `iw list` command. A pipe (`|`) is used to connect the output of the `iw list` command to the input of the `less` command. The `less` command is used to control the output of other commands; each press of *space* displays one page of output. Press *q* to quit the `less` command.

If the wireless adapter is not compatible with `hostapd`, the `iw list` command will display the message "`nl80211 not found`.

Once the wireless adapter has been tested and shown to support AP mode, configuration of the Raspberry Pi continues.

Configure the DHCP server (udhcpd)

The Dynamic Host Configuration Protocol (DHCP) server (udhcpd) assigns client computers network configuration information, such as the address of a DNS name server, an IP address, and a default route (or gateway). When this recipe is complete, the Raspberry Pi will configure its wireless clients using DCHP.

Three changes are made to the DHCP configuration file (`/etc/udhcpd.conf`):

- The wireless interface is selected (`wlan0`)
- The time remaining flag is set to support embedded devices (like the Raspberry Pi)
- The default network parameters are defined for wireless clients

The default network parameters are:

- `dns` – the DNS name server to use. Defined in `/etc/resolv.conf`. Displayed in step 4.
- `subnet` – how many IP addresses are part of the same network subnet. The value `255.255.255.0` is a net mask that matches computers with the same numbers in the first three bytes of the IP address. In the example, computers with IP addresses that begin `192.168.0` will be part of the same subnet.
- `domain` – the name of the network (`local`).
- `lease` – how long a network address will be assigned to a specific computer (`864000` seconds or 10 days).

This recipe relies on the default configuration to set the range of IP values (from `192.168.0.20` to `192.168.0.254`).

If the Raspberry Pi will always be attached to a wired network that has a network timeserver, the remaining time flag does not need to be set.

After udhcpd has been configured, its boot script parameter file (`/etc/default/udhcpd`) also needs to be changed. The parameter `DHCP_ENABLED="no"` needs to be commented out (by placing a `#` at the beginning of the line) so that the DHCP server (`udhcpd`) is enabled.

Configure the Wireless Access Point server (hostpad)

The Wireless Access Point server (`hostapd`) manages the wireless connection between other wireless devices and the Raspberry Pi. This includes establishing a secure connection using an encryption protocol like Wi-Fi Protect Access (WPA) and setting the Service Set ID (SSID) and the pre-shared key (PSK).

The created `hostapd` configuration parameter file includes the following:

- `interface` – the wireless interface (`wlan0`)
- `driver` – the wireless device driver (`nl80211`)
- `ssid` – the network ID (`Raspi_AP`)
- `hw_mode` – the hardware mode (`g`)
- `channel` – the radio frequency channel (`6`)
- `macaddr_acl` – access control list (`0`)
- `auth_algs` – the authorization algorithm to use (`1` – open auth)
- `ignore_broadcast_ssid` – enable broadcasting the network ID (`0` – don't ignore broadcasting)
- `wpa` – which version of WPA (`2`)
- `wpa_passphrase` – the passphrase or pre-shared key (`Pr0t3ct3d`)
- `wpa_key_mgmt` – which key manages algorithm (`WPA-PSK`)
- `wpa_pairwise` – WPA v1 data encryption (`TKIP`)
- `rsn_pairwise` – WPA v2 data encryption (`CCMP`)

After `hostapd` has been configured, its boot script parameter file (`/etc/default/hostapd`) also needs to be changed. The parameter `DAEMON_CONF` needs to be set to the location of the `hostapd` configuration file (`/etc/hostapd/hostapd.conf`) so that the Wireless Access Point server is enabled.

Set up IP Forwarding

IP Forwarding is used to pass (forward) network traffic between network interfaces. In this recipe, IP Forwarding is used to pass network traffic between the wireless network and the wired network. Using IP Forwarding, the Raspberry Pi connects the clients of the wireless network to the wired network.

The first configuration step is to enable IP Forwarding in the Raspberry Pi's Linux kernel. The kernel parameters file (`/etc/sysctl.conf`) has an IP Forwarding entry (`net.ipv4.ip_forward=1`) that is by default commented out. Uncommenting this entry, by removing the `#` at the beginning of the line, enables IP Forwarding in the Linux kernel.

After IP Forwarding is enabled, the `iptables` command is used to define the Netfilter rules that determine which network packets are allowed to cross the Linux kernel's internal firewall. The Linux kernel's firewall organizes its rules into tables that define how network packets pass through the kernel. The `iptables` command is used to manage the rules stored in these tables.

The first Linux kernel firewall rule defined in this step is appended to the post processing rules (`-A POSTROUTING`) of the network address translation table (`-t nat`). This rule masquerades network packets (`MASQUERADE`) as they are passed to the wired network (`-o eth0`). The IP addresses of wireless network clients are translated into the IP address of the Raspberry Pi's wired network connection as they are passed to the wired network. This is how the network packets from multiple wireless network clients are translated so they can pass through the Raspberry Pi's single wired network connection.

The second rule is appended to the packet forwarding rules (`-A FORWARD`) of the filter table (the default table). This rule allows (`-j ACCEPT`) network packets to be forwarded (`-A FORWARDED`) from the wired network (`-i eth0`) to the wireless network (`-o wlan0`) when they are related to an established connection (`-m state –state RELATED,ESTABLISHED`).

The last `iptables` rule in this step is appended to the forwarding rules of the filter table (`-A FORWARD`). This rule allows packets to pass from the wireless network (`-i wlan0`) to the wired network (`-o eth0`).

The next command, `ifconfig wlan0`, sets the IP address of the Raspberry Pi's wireless network connection to `192.168.0.1`.

Finally, the `iptables-save` command is used to save a copy of these rules in a configuration file (`/etc/iptables.ipv4.nat`) that can be used during boot. This command is run in subshell (`sh –c`) so that the output redirect (`>`) in the command inherits the privileges of the `sudo` command prefix.

After this step is complete, the IP Forwarding rules have been defined and saved in a configuration file. The IP Forwarding rules are also active.

Configure the boot parameters

The network interfaces definitions used during boot are stored in a configuration file (`/etc/network/interfaces`). The file defines the network address, network mask, and the default route for each network interface.

The configuration file used in this recipe defines three network interfaces:

- `lo` – the loopback network
- `eth0` – the wired network
- `wlan0` – the wireless network

The loopback interface (`lo`) is loaded automatically (`auto`).

The wired interface (`eth0`) is configured dynamically using the DHCP protocol.

The wireless interface (`wlan0`) has a static definition (`static`)—it is in this file. The wireless interface's IP `address` is defined to be `192.168.0.1`. The interface's defined network mask (`255.255.255.0`) is big enough to support 256 unique addresses on the same subnet.

After the network interfaces are brought up (`up`), the IP Forwarding definitions (`/etc/iptables.ipv4.nat`) are restored (`iptables-restore`) that were saved earlier in this recipe (using `iptables-save`).

Once the network interface definitions have been saved, the network can be started.

Start the Wireless Access Point

Now that the configuration files have been updated the boot scripts for the wireless access point daemon (`hostapd`) and the dynamic host configuration protocol daemon (`udhcp`) can be enabled (`update-rc.d enable`).

The Raspberry Pi will now become a wireless access point every time it boots!

Finally, the two daemons are started (`service start`).

The Wireless Access Point is ready to use! Wireless devices can now connect to the Raspberry Pi using your chosen SSID and passphrase (SSID: `Raspi_AP` / Passphrase: `Pr0t3ct3d by default`.

There's more...

This recipe works well when combined with other recipes in this book.

Together with the file-sharing recipes in *Chapter 4*, *File Sharing*, the Raspberry Pi could become a file server for both wired and wireless devices connected to the local network - for exchanging document and media files; or for backup and storage.

When combined with other advanced networking recipes in this chapter, the Raspberry Pi could become a network firewall, protecting wireless access to a wired network; a teaching or support tool with remote access to desktop devices; a communication tool that serves web pages or a collaboration tool that hosts wiki pages.

Within the Raspberry Pi and open source GNU Linux community there are numerous other tools and applications that could be combined with this recipe to turn the Raspberry Pi into a dynamic network hub for wireless devices.

See also

- **hostapd**

  `http://en.wikipedia.org/wiki/Hostapd`

  A Wikipedia article about hostapd.

- **hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator**

  `http://w1.fi/hostapd/`

  The hostapd website.

- **Debian Linux Kernel Handbook**

  `http://kernel-handbook.alioth.debian.org/`

  The Debian Linux Kernel Handbook.

- **Linux wireless**

  `http://linuxwireless.org/`

  The Linux wireless website.

- **RPI-Wireless-Hotspot**

  `http://elinux.org/RPI-Wireless-Hotspot`

  An article about wireless hotspots on the Embedded Linux website.

- **USB Wifi Adapters**

  `http://elinux.org/RPi_VerifiedPeripherals#USB_Wi-Fi_Adapters`

  An article about WiFi adapters on the Embedded Linux Wiki.

- **Raspberry Pi WiFi adapter testing**

  `http://www.element14.com/community/docs/DOC-44703/l/raspberry-pi-wifi-adapter-testing`

  An article about tested wireless adapters on the Element14 Community website.

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Thank you for buying
Raspberry Pi Networking Cookbook

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.

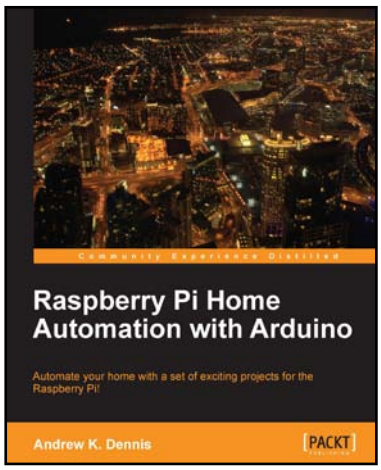
Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

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.

[PACKT]
PUBLISHING



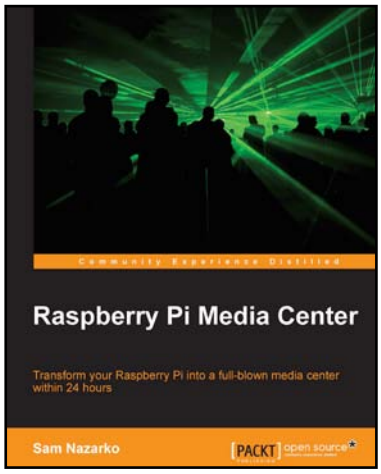
Raspberry Pi Home Automation with Arduino

ISBN: 978-1-84969-586-2 Paperback: 176 pages

Automate your home with a set of exciting projects for the Raspberry Pi!

1. Learn how to dynamically adjust your living environment with detailed step-by-step examples
2. Discover how you can utilize the combined power of the Raspberry Pi and Arduino for your own projects
3. Revolutionize the way you interact with your home on a daily basis

---



Raspberry Pi Media Center

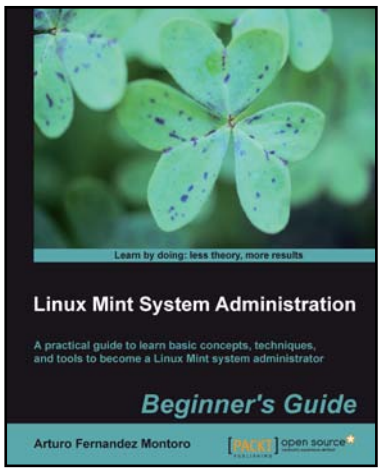
ISBN: 978-1-78216-302-2 Paperback: 108 pages

Transform your Raspberry Pi into a full-blown media center within 24 hours

1. Discover how you can stream video, music, and photos straight to your TV
2. Play existing content from your computer or USB drive
3. Watch and record TV via satellite, cable, or terrestrial
4. Build your very own library that automatically includes detailed information and cover material

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

[PACKT] PUBLISHING



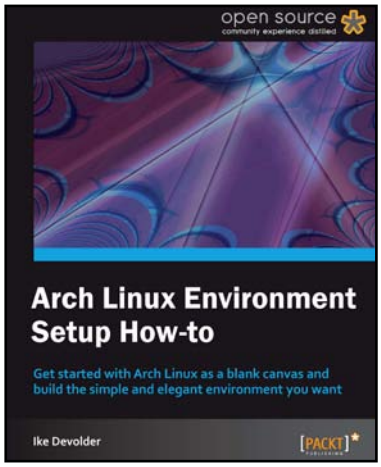
Linux Mint System Administration Beginner's Guide

ISBN: 978-1-84951-960-1 Paperback: 146 pages

A practical guide to learn basic concepts, techniques, and tools to become a Linux Mint system administrator

1. Discover Linux Mint and learn how to install it
2. Learn basic shell commands and how to deal with user accounts
3. Find out how to carry out system administrator tasks such as monitoring, backups, and network configuration

---



Arch Linux Environment Setup How-to

ISBN: 978-1-84951-972-4 Paperback: 68 pages

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

1. Learn something new in an Instant! A short, fast, focused guide delivering immediate results.
2. Install and configure Arch Linux to set up your optimum environment for building applications
3. Boot and manage services, add and remove packages

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