Chapter 3: Troubleshooting

Sometimes, things don’t go entirely smoothly. The more complex the device, the more complex the problems that can occur—and the Pi is an extremely complex device indeed.

Thankfully, many of the most common problems are straightforward to diagnose and fix. In this chapter, we’ll look at some of the most common reasons for the Pi to misbehave and how to fix them.

Keyboard and Mouse Diagnostics

Perhaps the most common problem that users experience with the Raspberry Pi is when the keyboard repeats certain characters. For example, if the command `startx` appears onscreen as `sttttttttttartxxxxxxxxxxxx`, it will, understandably, fail to work when the Enter key is pressed.

There are typically two reasons why a USB keyboard fails to operate correctly when connected to the Raspberry Pi: it’s drawing too much power, or its internal chipset is conflicting with the USB circuitry on the Pi.

Check the documentation for your keyboard, or the label on its underside, to see if it has a power rating given in milliamps (mA). This is how much power the keyboard attempts to draw from the USB port when it’s in use.

The Pi’s USB ports have a component called a polyfuse connected to them, which protects the Pi in the event that a device attempts to draw too much power. When this polyfuse is tripped, it causes the USB port to shut off, at around 150 mA. If your keyboard draws anywhere around that much power, it may operate strangely—or not at all. This can be a problem for keyboards that have built-in LED lighting, which require far more power to operate than a standard keyboard.

If you find that your USB keyboard may be drawing too much power, try connecting it to a powered USB hub instead of directly to the Pi. This will allow the keyboard to draw its power from the hub’s power supply unit, instead of from the Pi itself. Alternatively, swap the keyboard out for a model with lower power demands. The repeating-letter problem may also be traced to an inadequate power supply for the Pi itself, which is addressed in the next section, “Power Diagnostics”.

The issue of compatibility, sadly, is harder to diagnose. While the overwhelming majority of keyboards work just fine with the Pi, a small number exhibit strange symptoms. These range from intermittent response, the repeating-letter syndrome or even crashes that prevent the Pi from operating. Sometimes, these issues don’t appear until other USB devices are connected to the Pi. If your keyboard was working fine until another USB device, in particular a USB wireless adapter, was connected, you may have an issue of incompatibility.

If possible, try swapping the keyboard out for another model. If the new keyboard works, your old one may be incompatible with the Pi. For a list of known-incompatible keyboards, visit the eLinux wiki:
http://elinux.org/RPi_VerifiedPeripherals#Problem_USB_Keyboards

The same advice on checking compatibility in advance applies to problems with the mouse: the majority of USB mice and trackballs work fine, but some exhibit incompatibility with the Pi’s own USB circuitry. This usually results in symptoms like a jerky or unresponsive mouse pointer, but it can sometimes lead to the Pi failing to load or crashing at random intervals. If you’re looking to buy a new mouse, an up-to-date list of models known to work with the Pi is available at the eLinux wiki site:
http://elinux.org/RPi_VerifiedPeripherals#Working_USB_Mouse_Devices

Power Diagnostics

Many problems with the Raspberry Pi can be traced to an inadequate power supply. The Model A requires a 5 V supply capable of providing a 500 mA current, while the Model B’s extra components bump up the current requirement to 700 mA. Not all USB power adapters are designed to offer this much power, even if their labelling claims otherwise.

TIP

The formal USB standard states that devices should draw no more than 500 mA, with even that level of power only available to the device following a process called negotiation. Because the Pi doesn’t negotiate for power, it’s unlikely that it will work if you connect it to the USB ports on a desktop or laptop computer.

If you’re having intermittent problems with your Pi—particularly if it works until you connect something to a USB port or start a processor-intensive operation like playing video—the chances are that the power supply in use is inadequate. The Pi provides a relatively easy way to check if this is the case in the form of two voltage test points.

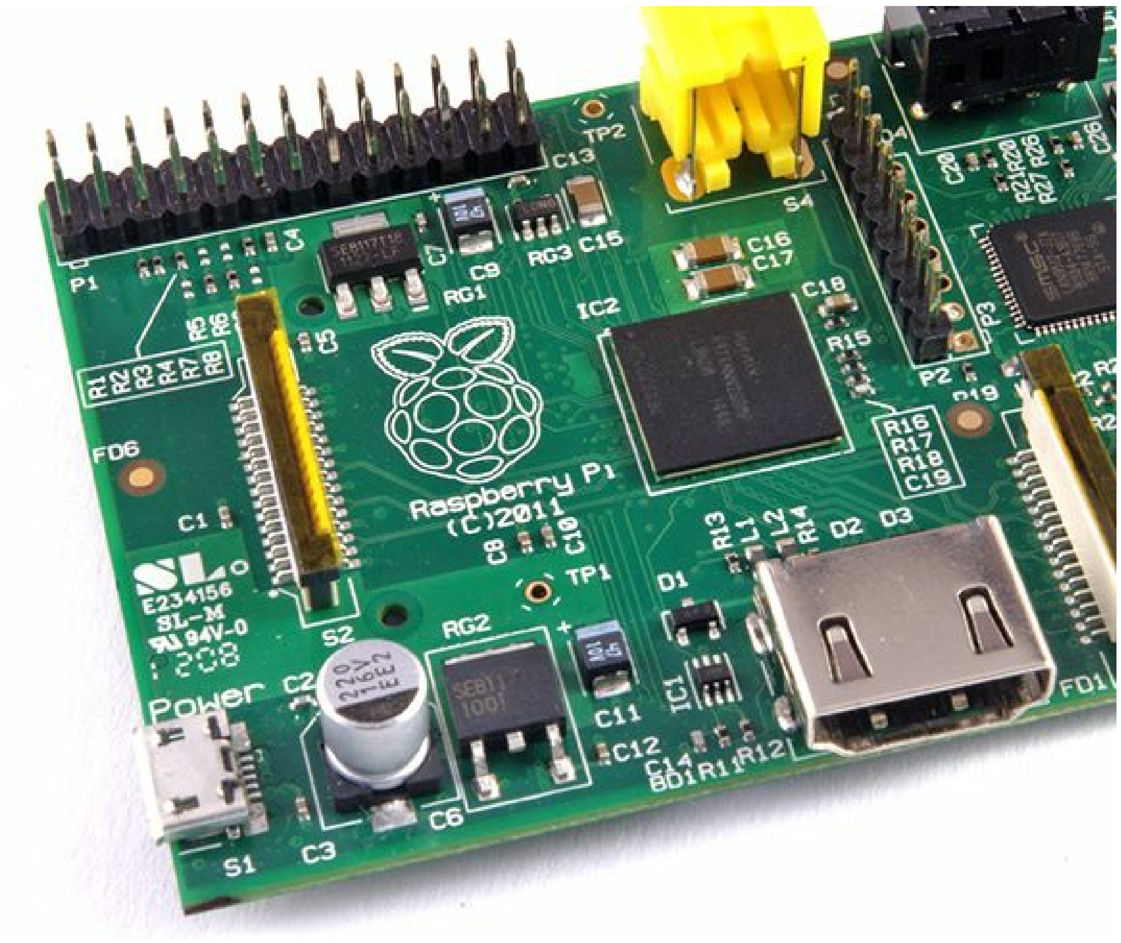
To use the voltage test points, you'll need a voltmeter or multimeter with direct current (DC) voltage measuring capabilities. If your meter has multiple inputs for different voltages, use an appropriate setting.

WARNING

Avoid touching the test probes to anything not labelled as a test point. It's possible to bridge the 5 V supply that comes in to the Pi to the internal 3.3 V supply, creating a short circuit which can damage the device. Be especially careful around exposed header pins.

The two test points are small, copper-clad holes known as vias, which are connected to the Pi's 5 V and ground circuits. Put the positive (red) meter probe on TP1, located to the left of the board just above a small black component called a regulator labelled RG2. Connect the black (negative) meter probe to TP2, located between the copper GPIO pins and the yellow-and-silver RCA phono connector at the top-left of the board (see Figure 3-1).

Figure 3-1: The two voltage test points, labelled TP1 and TP2



The reading on the voltmeter should be somewhere between 4.8 V and 5 V. If it's lower than 4.8 V, this indicates that the Pi is not being provided with enough power. Try swapping the USB adapter for a different model, and check that the label says it can supply 700 mA or more. A model rated at 1A is recommended, but beware of cheap models—they sometimes have inaccurate labelling, and fail to supply the promised current. Genuine branded mobile phone chargers rarely have this problem, but cheap unbranded devices—often sold as compatible adapters—should be avoided

If your voltmeter reads a negative number, don't worry: this just means you've got the positive and negative probes in the wrong place. Either swap them around or just ignore the negative sign when noting your reading.

Display Diagnostics

Although the Pi is designed to work with almost any HDMI, DVI or composite video display device, it simply may not work as expected when you plug it in. For example, you may find that your picture is shifted to the side or not fully displayed, or is only visible as a postage-stamp-sized cut-out in the middle of the screen or in black-and-white—or even missing entirely.

First, check the type of device to which the Pi is connected. This is especially important when you're using the composite RCA connection to plug the Pi into a TV. Different countries use different standards for TV video, meaning that a Pi configured for one country may not work in another. This is the usual explanation for a Pi showing black-and-white video. You'll learn how to adjust this setting in Chapter 6, "Configuring the Raspberry Pi".

When you use the HDMI output, the display type is usually automatically detected. If you're using an HDMI to DVI adapter to plug the Pi into a computer monitor, however, this occasionally goes awry. Common symptoms include snow-like static, missing picture portions or no display at all. To fix this, note the resolution and refresh rate of your connected display, and then jump to Chapter 6 to find out how to set these manually.

Another issue is a too-large or too-small image, either missing portions at the edge of the screen or sitting in the middle of a large black border. This is caused by a setting known as overscan, which is used when the Pi is connected to TVs to avoid printing to portions of the display which may be hidden under a bezel. As with other display-related settings, you will learn how to adjust—or even completely disable—overscan in Chapter 6.

Boot Diagnostics

The most common cause for a Pi to fail to boot is a problem with the SD card. Unlike a desktop or laptop computer, the Pi relies on files stored on the SD card for everything. If Pi can't talk to the card, it won't display anything on the screen or show any signs of life at all.

If your Pi's power light glows when you connect the micro-USB power supply, but nothing else happens and the OK light remains dark, you have an SD card problem. First, ensure that the card works when you connect it to a PC, and that it shows the partitions and files expected of a well-flashed card. (For more details, see Chapter 2, "Linux System Administration", particularly the section titled "File System Layout" in that chapter.)

If the card works on a PC but not in the Pi, it may be a compatibility problem. Some SD cards—especially high-speed cards marked as Class 10 on their labelling—don't operate correctly when connected to the Pi's onboard SD card reader. A list of cards known to cause compatibility problems with the Pi can be found on the eLinux wiki: `http://elinux.org/RPi_VerifiedPeripherals#Problem_SD_Cards`

Sadly, if you have one of the cards on the list, you may need to replace it with a different card in order for the Pi to work. As the Pi's software base is developed, however, work is being carried out to ensure that a wider range of cards operate correctly with the Pi. Before giving up on a high-speed card completely, check to see if an updated version of your chosen Linux distribution is available. (See Chapter 1, "Meet the Raspberry Pi", for more information about distributions.)

Network Diagnostics

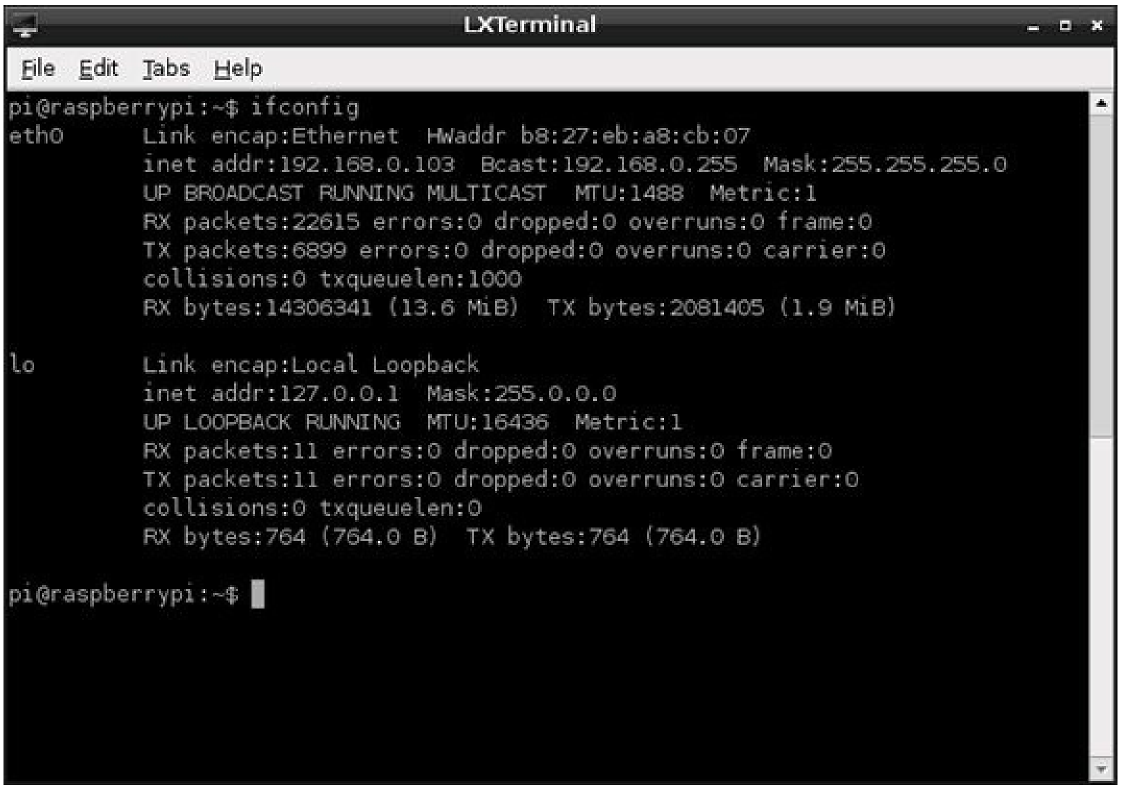
The most useful tool for diagnosing network problems is `ifconfig`. If you're using a wireless network connection, jump to Chapter 4, "Network Configuration", for information on a similar tool for those devices. Otherwise, read on.

Designed to provide information on connected network ports, `ifconfig` is a powerful tool for controlling and configuring the Pi's network ports. For its most basic usage, simply type the tool's name in the terminal:

```
ifconfig
```

Called in this manner, `ifconfig` provides information on all the network ports it can find (see Figure 3-2). For the standard Raspberry Pi Model B, there are two ports: the physical Ethernet port on the right side of the board, and a virtual loopback interface that allows programs on the Pi to talk to each other.

Figure 3-2: The output of `ifconfig` on a Raspberry Pi Model B



```
pi@raspberrypi:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:a8:cb:07
          inet addr:192.168.0.103  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1488  Metric:1
          RX packets:22615 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6899 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14306341 (13.6 MiB)  TX bytes:2081405 (1.9 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:764 (764.0 B)  TX bytes:764 (764.0 B)

pi@raspberrypi:~$
```

The output of `ifconfig` is split into the following sections:

- Link encap—The type of encapsulation used by the network, which on the Model B will either read `Ethernet` for the physical network port or `Local Loopback` for the virtual loopback adaptor.
- Hwaddr—The Media Access Control (MAC) address of the network interface, written in hexadecimal. This is unique for every device on the network, and each Pi has its own MAC address, which is set at the factory.
- inet addr—The internet protocol (IP) address of the network interface. This is how you find the Pi on the network if you're using it to run a network-accessible service, such as a web server or file server.
- Bcast—The broadcast address for the network to which the Pi is connected. Any traffic sent to this address will be received by every device on the network.
- Mask—The network mask, which controls the maximum size of the network to which the Pi is connected. For most home users, this will read 255.255.255.0.
- MTU—The maximum transmission unit size, which is how big a single packet of data can be before the system needs to split it into multiple packets.
- RX—This section provides feedback on the received network traffic, including the number of errors and dropped packets recorded. If you start to see errors appearing in this section, there's something wrong with the network.
- TX—This provides the same information as the RX section, but for transmitted packets. Again, any errors recorded here indicate a problem with the network.
- collisions—If two systems on the network try to talk at the same time, you get a collision which requires them to retransmit their packets. Small numbers of collisions aren't a problem, but a large number here indicates a network issue.
- txqueuelen—The length of the transmission queue, which will usually be set to 1000 and rarely needs changing.
- RX bytes, TX bytes—A summary of the amount of traffic the network interface has passed.

If you're having problems with the network on the Pi, you should first try to disable and then re-enable the network interface. The easiest way to do this is with two tools called `ifup` and `ifdown`.

If the network is up, but not working correctly—for example, if `ifconfig` doesn't list anything in the `inet addr` section—start by disabling the network port. From the terminal, type the following command:

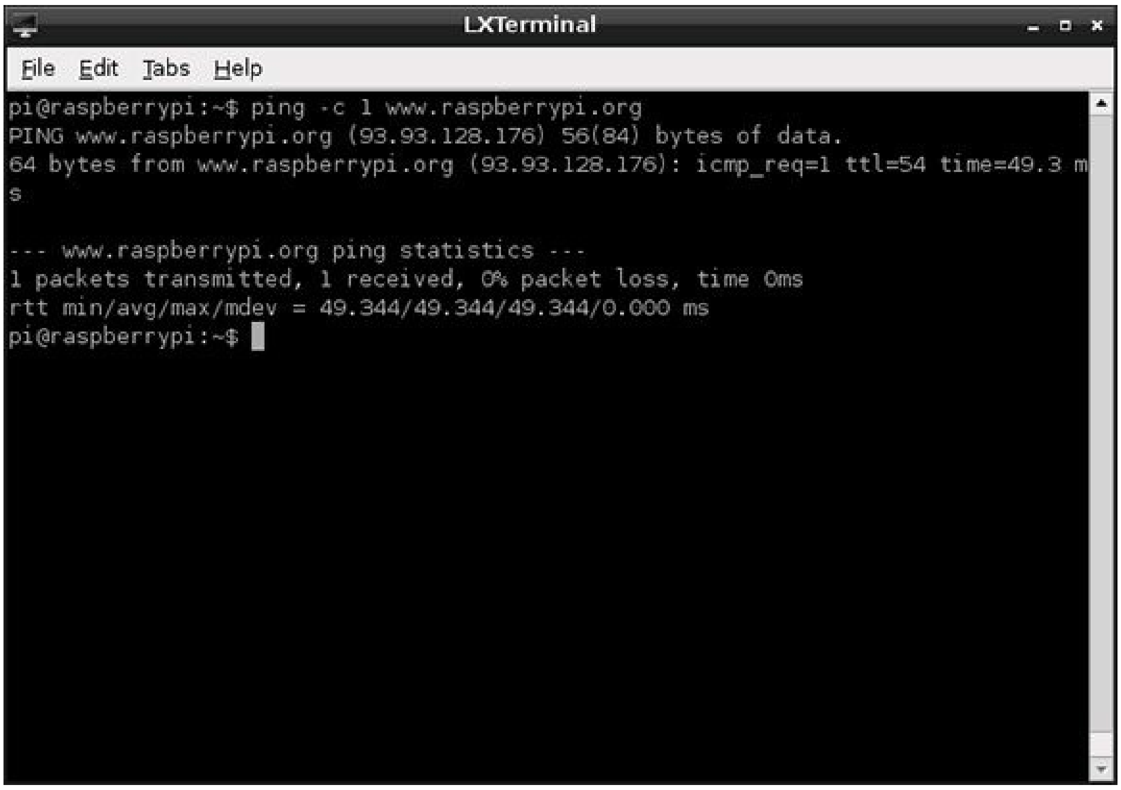
```
sudo ifdown eth0
```

Once the network is disabled, make sure that the cable is inserted tightly at both ends, and that whatever network device the Pi is connected to (hub, switch or router) is powered on and working. Then bring the interface back up again with the following command:

```
sudo ifup eth0
```

You can test the networking by using the `ping` command, which sends data to a remote computer and waits for a response. If everything's working, you should see the same response as shown in Figure 3-3. If not, you may need to manually configure your network settings, which you'll learn how to do in Chapter 4, "Network Configuration".

Figure 3-3: The result of a successful test of the network, using the ping command



```
pi@raspberrypi:~$ ping -c 1 www.raspberrypi.org
PING www.raspberrypi.org (93.93.128.176) 56(84) bytes of data.
64 bytes from www.raspberrypi.org (93.93.128.176): icmp_req=1 ttl=54 time=49.3 m
s

--- www.raspberrypi.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 49.344/49.344/49.344/0.000 ms
pi@raspberrypi:~$
```

The Emergency Kernel

The Linux kernel is the heart of the operating system that drives the Pi. It's responsible for everything from making sure that you can access your files to allowing programs to talk to other programs.

When switched on, your Pi will load the normal, default kernel. There's also a second kernel included in most distributions, which sits unused. This is the emergency kernel, and as the name suggests, it is typically used only when the normal kernel isn't working.

It's highly unlikely that you'll ever need to boot a Pi using the emergency kernel, but it's worth learning how to do so just in case. This is especially important if you're upgrading your kernel or are using a new and potentially poorly tested distribution. Sometimes, newly-released software can have bugs which aren't spotted before its release. When encountering strange errors after upgrading, the emergency kernel can be used to narrow down the problem to the new kernel version.

The Linux kernel is a single file located in the `/boot` directory called `kernel.img`. When the Pi is first switched on and begins to load the operating system, it looks for this file, and if the file is missing, the Pi won't work. The emergency kernel is a second file, again in the `/boot` directory, called `kernel_emergency.img`.

The emergency kernel is, in most cases, almost identical to the standard kernel. When changes are made to the standard kernel, to boost performance or add new features for example, the emergency kernel is left unaltered. This way, if the changes to the standard kernel cause stability problems, a user can simply tell the Pi to load the emergency kernel instead.

There are two ways to boot into the emergency kernel, and both require the use of a PC and an SD card reader if the Pi can't boot. Otherwise, the following can be carried out on the Pi itself.

The easiest way to boot the emergency kernel is to rename the existing `kernel.img` file to `kernel.img.bak`, and then rename the `kernel_emergency.img` file to `kernel.img`. When the Pi loads, it will now load the emergency kernel by default. To go back to the standard kernel, simply reverse the process: rename `kernel.img` to `kernel_emergency.img` and `kernel.img.bak` to `kernel.img`.

An alternative method to load the emergency kernel is to edit the `cmdline.xt` file (located in the `/boot` directory) by adding the following entry at the end of the existing command line:

```
kernel=kernel_emergency.img
```

This tells the Pi that it should load the kernel named `kernel_emergency.img` instead of the usual `kernel.img`. Reversing the process is as simple as opening `cmdline.txt` again and removing the entry.

You'll learn more about `cmdline.txt` and how it affects the operation of the Raspberry Pi in Chapter 6, "Configuring the Raspberry Pi".

Chapter 4: Network Configuration

For most users, configuring the Pi's network is as easy as plugging a cable into the Model B's Ethernet port—or a USB Ethernet adapter in the case of the Model A. For others, however, the network requires manual configuration.

If you know that your network doesn't have a Dynamic Host Configuration Protocol (DHCP) server—a system that tells the Pi and other devices on the network how they should connect—or if you want to use a USB wireless adapter with the Pi, read on.

Wired Networking

If the network still doesn't work, you may need to configure it manually. Normally, the network in a home, school or office has a DHCP server that tells the Pi and other devices on the network how they should connect. Some networks, however, don't have a DHCP server and need to be set up manually.

The list of network interfaces, along with information about how they should be configured, is stored in a file called `interfaces` located in the folder `/etc/network`. This is a file only the root user can edit, because removing a network interface from this list will cause it to stop working.

From the terminal, you can edit this file using a variety of different text editors. For simplicity, the `nano` text editor should be used for this process. Open the file for editing with the following command:

```
sudo nano /etc/network/interfaces
```

`Nano` is a powerful yet lightweight text editor, with a simple user interface (see Figure 4-1). You can move your cursor around the document with the arrow keys, save by holding down the CTRL key and pressing O, and quit by holding down the CTRL key and pressing X.

The line you need to edit for manual configuration starts with `iface eth0 inet`. Delete `dhcp` from the end of this line and replace it with `static`, press Enter to start a new line, and then fill in the remaining details in the following format with a tab at the start of each line:

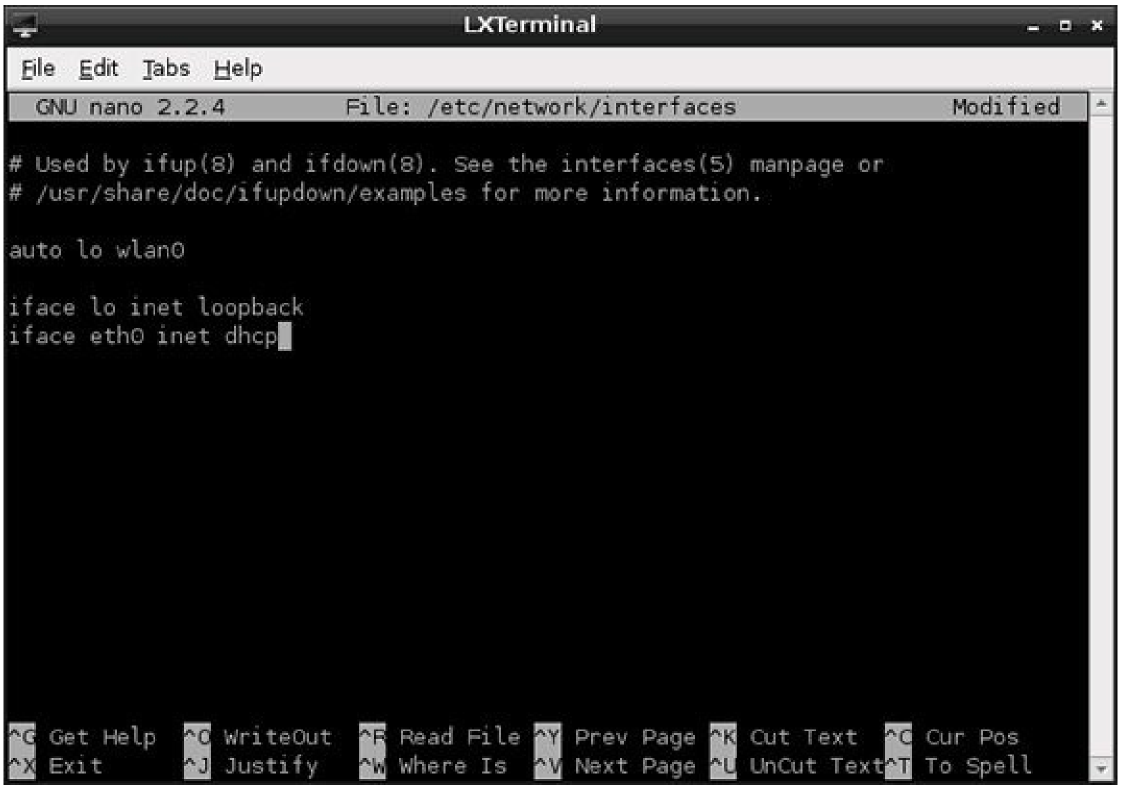
```
[Tab] address xxx.xxx.xxx.xxx
[Tab] netmask xxx.xxx.xxx.xxx
[Tab] gateway xxx.xxx.xxx.xxx
```

Make sure that you press the Tab key at the start of each line, and don't actually type [Tab]. The x characters in the configuration lines represent network addresses you'll need to enter. For `address`, you should enter the static IP address that you want to assign to the Pi. For `netmask`, you should enter the network mask—which controls the size of the connected network—in what is known as dotted-quad format. If you're using a home network, this is typically 255.255.255.0. For `gateway`, you should enter the IP address of your router or cable modem.

As an example, the settings for a common home network would look like this:

```
iface eth0 inet static
[Tab] address 192.168.0.10
[Tab] netmask 255.255.255.0
[Tab] gateway 192.168.0.254
```

Figure 4-1: Editing /etc/network/interfaces with nano



LXTerminal
File Edit Tabs Help

```
  GNU nano 2.2.4          File: /etc/network/interfaces                  Modified

# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.

auto lo wlan0

iface lo inet loopback
iface eth0 inet dhcp
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

When you've finished editing the file, press CTRL + O to save it, and then press CTRL + X to leave `nano` and return to the terminal. To use your new network settings, restart the networking service by typing the following:

```
sudo /etc/init.d/networking restart
```

If you need to return to automatic settings via DHCP, you need to edit the `interfaces` file again and delete the `address`, `netmask` and `gateway` settings. Replace `static` with `dhcp` at the end of the `iface` line, and then restart the networking service again.

Setting a manual IP address isn't quite enough to get your Pi connected to the outside world. Computers on modern networks have both a numerical address identifier known as an IP address and a hostname or domain name. It's this latter, friendly name which means you can simply type `www.raspberrypi.org` into your browser, instead of trying to remember `93.93.128.176`.

A system called a Domain Name Service (DNS) server is responsible for looking up the friendly names you supply and converting them into the numbers required to access the system. It operates much like an automated telephone directory. Before you'll be able to access Internet-connected systems via their domain names, you'll need to tell the Pi which DNS servers to use.

The list of DNS servers, known as nameservers in Linux parlance, is stored in `/etc/resolv.conf`. When the system gets its details through DHCP, this file is automatically filled in. When you set an address manually, you need to provide the addresses of the nameservers on your network. Normally, this would be the address of your router as found in the `gateway` line from the `interfaces` file (described earlier in this chapter).

To set the nameservers, open the file with `nano` by typing the following command at the terminal:

```
sudo nano /etc/resolv.conf
```

Add each nameserver on a separate line, prefaced with `nameserver` and a space. As an example, the `resolv.conf` configuration for a network which uses Google's publicly-accessible nameservers to resolve domain names would appear like this:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

You'll notice that the nameserver addresses need to be supplied as IP addresses, rather than domain names. If you provided domain names instead, the Pi would enter an infinite loop of trying to find a nameserver to ask how it can find the nameservers.

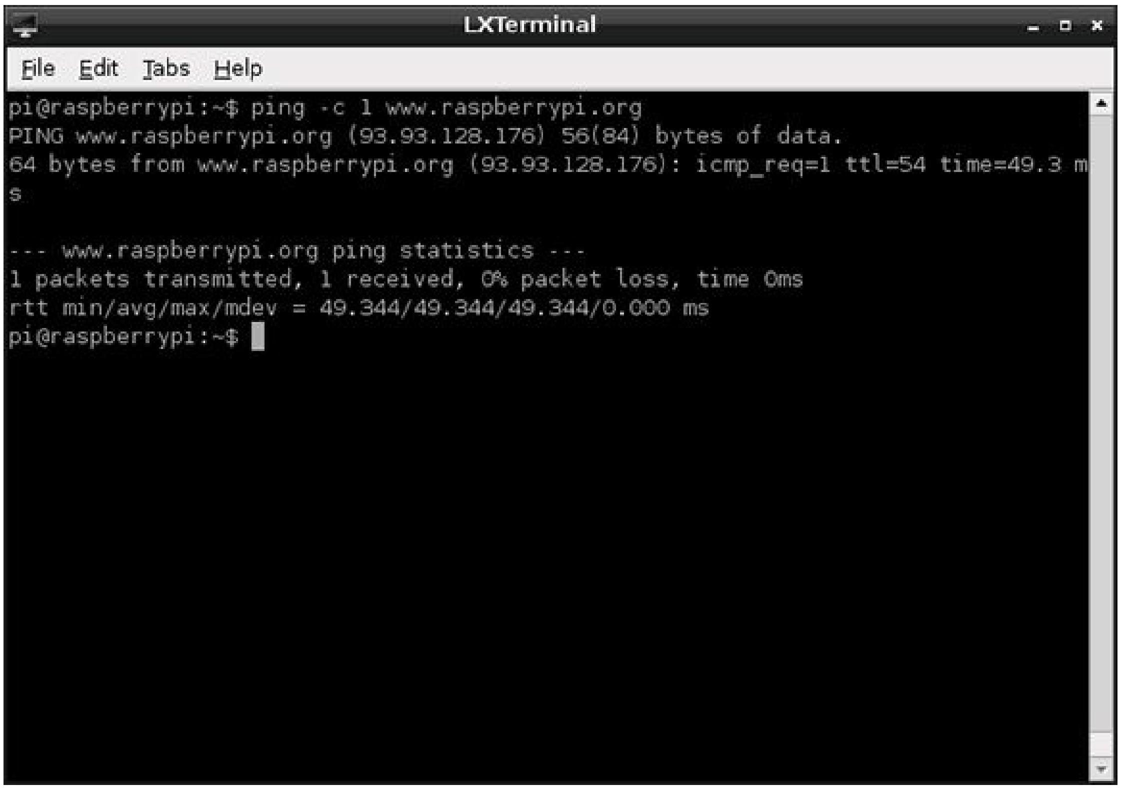
Save the file by pressing CTRL + O, and then quit `nano` by pressing CTRL + X. Restart the networking interface by typing the following:

```
sudo /etc/init.d/networking restart
```

Then test the settings by either opening a web browser or using the following `ping` command (see Figure 4-2):

```
ping -c 1 www.raspberrypi.org
```

Figure 4-2: A successful test of networking on the Raspberry Pi Model B



```
pi@raspberrypi:~$ ping -c 1 www.raspberrypi.org
PING www.raspberrypi.org (93.93.128.176) 56(84) bytes of data.
64 bytes from www.raspberrypi.org (93.93.128.176): icmp_req=1 ttl=54 time=49.3 m
s

--- www.raspberrypi.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 49.344/49.344/49.344/0.000 ms
pi@raspberrypi:~$ 
```

Wireless Networking

Although no current models of the Raspberry Pi include Wi-Fi networking hardware onboard, it's possible to add wireless connectivity with a simple USB Wi-Fi adapter. However, you will need to configure the adapter before you can use it to get your Pi online.

TIP

USB Wi-Fi adapters are very power-hungry. If you connect one directly to the Pi's USB port, the chances are it simply won't work. Instead, connect a powered USB hub to the Pi, and then insert the Wi-Fi adapter into that.

Before you start to set up the wireless interface, you'll need to know the Service Set Identifier (SSID)—also known as the network name—of the wireless router to which you want to connect, along with the type of encryption in use and the password required. You'll also need to know what type of wireless network it is. A USB adapter designed for 802.11a Wi-Fi may not connect to an 802.11g network, and vice versa.

In order for the USB wireless adapter to be addressed by the system, a software bundle known as a firmware is required. While some distributions include a selection of the most common Wi-Fi firmware installed by default, others do not. At present, to save space, most distributions designed for the Raspberry Pi need the firmware files for a wireless card installing manually.

This, unfortunately, can lead to a Catch-22 situation: in order to download the firmware files, the Pi must be connected to the Internet. If you can spare a wired port on your router or gateway for a few minutes, that's not a problem. However, if wireless is your only way of getting online, you'll need to manually download the firmware installation package on a different computer, and then transfer it across to the Pi by either copying it to the Pi's SD card or connecting an external storage device such as a USB flash drive.

To find the correct firmware file to download, you'll need to know what type of wireless adapter you have. Although various companies sell branded USB wireless adapters, the number of companies that actually manufacture the components is a lot smaller. Several different manufacturers may use the same type of chip inside their USB wireless adapters, making them all compatible with the same firmware. As a result, the labelling on a device or its packaging is not enough to know which firmware you should install. Instead, you'll need to connect the device to the Pi and check the kernel ring buffer for error messages. If you've already connected the wireless adapter as instructed in Chapter 1, "Meet the Raspberry Pi", you can continue. If not, connect the adapter now.

The kernel ring buffer is a special portion of memory used by the Linux kernel to store its human-readable output. It's an important part of the Linux operating system: the text flashes by too quickly to read while the Pi boots, so it's critical that users are able to view the messages at a later date to read errors and diagnose problems.

With the adapter connected but no wireless firmware packages installed, the kernel will print a series of error messages to the ring buffer. To read these messages, you can use the `dmesg` command to print the contents of the buffer to the screen. At the terminal, or at the console if you haven't loaded the desktop environment, simply type the following command to view the buffer:

```
dmesg
```

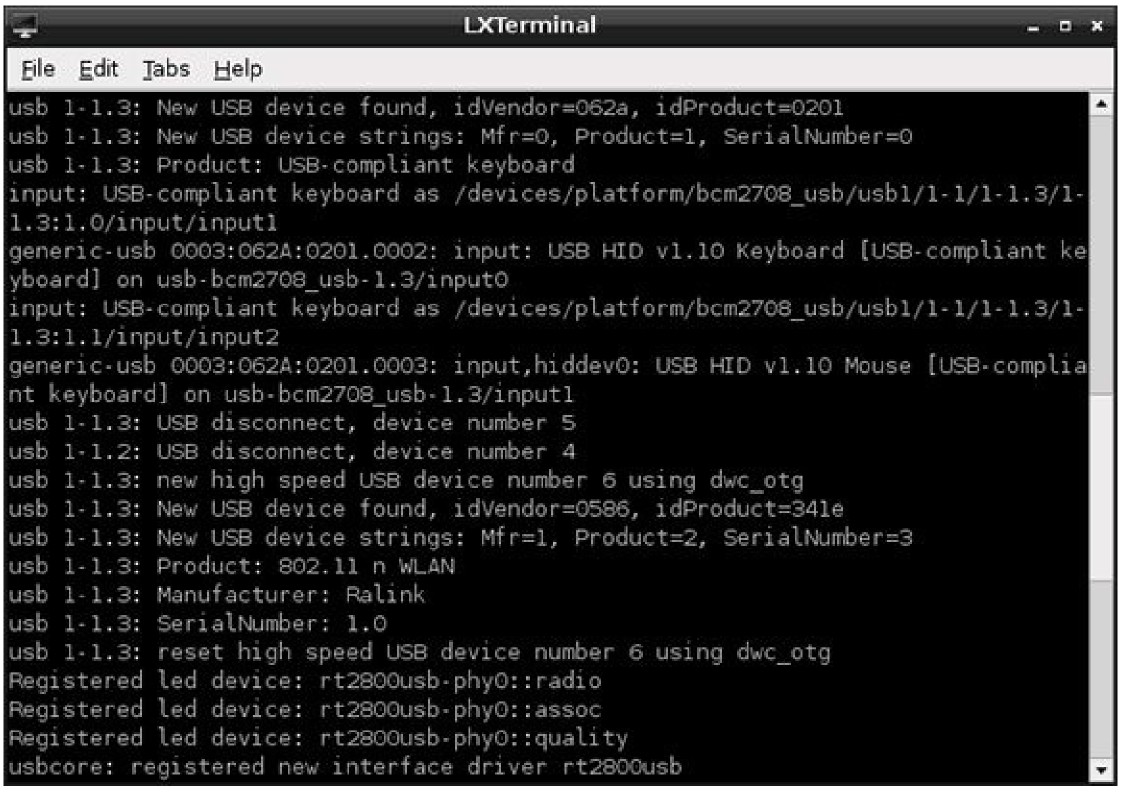
This will print out the entire kernel ring buffer, which will contain all messages output by the kernel since the Pi was switched on. If the Pi has been running a while, that can be a lot of text. To locate error message particular to the wireless adapter, it can help to send the output of `dmesg` through a tool called `grep`. Using `grep`, you can search through the buffer for text relating to missing firmware. By piping the output of `dmesg` through `grep` with a search term, things become significantly clearer. Type the following at the terminal:

```
dmesg | grep ^usb
```

The `|` symbol is known as a pipe, and tells Linux to send the output of one program—which would normally go to a file or the screen—to the input of another. Multiple programs can be chained this way. In this example, `grep` is being told to search through the output of `dmesg`—the screens full of text from the earlier command—for any use of the term `usb` at the start of the line (denoted by the `^` character).

The exact output of that search will depend on the manufacturer of your USB wireless adapter. In Figure 4-3, the output is shown with a Zyxel NWD2015 Wireless USB Adapter connected to the Pi.

Figure 4-3: Searching the kernel ring buffer for `usb` with a Zyxel wireless adapter connected



```
usb 1-1.3: New USB device found, idVendor=062a, idProduct=0201
usb 1-1.3: New USB device strings: Mfr=0, Product=1, SerialNumber=0
usb 1-1.3: Product: USB-compliant keyboard
input: USB-compliant keyboard as /devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-
1.3:1.0/input/input1
generic-usb 0003:062A:0201.0002: input: USB HID v1.10 Keyboard [USB-compliant ke
yboard] on usb-bcm2708_usb-1.3/input0
input: USB-compliant keyboard as /devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-
1.3:1.1/input/input2
generic-usb 0003:062A:0201.0003: input,hiddev0: USB HID v1.10 Mouse [USB-complia
nt keyboard] on usb-bcm2708_usb-1.3/input1
usb 1-1.3: USB disconnect, device number 5
usb 1-1.2: USB disconnect, device number 4
usb 1-1.3: new high speed USB device number 6 using dwc_otg
usb 1-1.3: New USB device found, idVendor=0586, idProduct=341e
usb 1-1.3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1.3: Product: 802.11 n WLAN
usb 1-1.3: Manufacturer: Ralink
usb 1-1.3: SerialNumber: 1.0
usb 1-1.3: reset high speed USB device number 6 using dwc_otg
Registered led device: rt2800usb-phy0::radio
Registered led device: rt2800usb-phy0::assoc
Registered led device: rt2800usb-phy0::quality
usbcore: registered new interface driver rt2800usb
```

The important part of this output is the line that reads `Manufacturer`. In the case of the example Zyxel NWD2105, this reads `Ralink`, which is the company that makes the actual chip found inside Zyxel USB wireless adapter. It's this company's firmware that must be installed for the wireless adapter to work.

TIP

If you couldn't find anything using usb as a search term, you can try the same command using the search term firmware, wlan or wireless. If you still can't see anything useful, type lsusb for a list of all USB devices connected to the system.

Using the manufacturer name from `dmesg`, search for the firmware files using the `apt-cache` search tool introduced earlier in this chapter. For the example Zyxel NWD2015 adapter, the `apt-cache` command would be

```
apt-cache search ralink
```

If `apt-cache` fails to find the firmware, you may need to make a guess based on the firmware packages in the following list. Don't worry if you install the wrong one—any firmware can be quickly uninstalled using `apt-get remove`, and having multiple firmware packages does no harm. The following wireless firmware packages are available in the recommended Debian distribution on the Raspberry Pi:

- atmel-firmware—For devices based on the Atmel AT76C50X chipset
- firmware-atheros—For devices based on Atheros chipsets
- firmware-brcm80211—For devices based on Broadcom chipsets
- firmware-intelwimax—For devices based on Intel's WiMAX chipsets
- firmware-ipw2x00—For Intel Pro Wireless adapters (including 2100, 2200 and 2915)
- firmware-iwlwifi—For other Intel wireless adapters (including 3945, 4965 and the 5000 series)
- firmware-ralink—For devices based on Ralink chipsets
- firmware-realtek—For devices based on Realtek chipsets
- zd1211-firmware—For devices based on the ZyDAS 1211 chipset

The firmware for the example Zyxel wireless adapter is provided by the `firmware-ralink` package in this list. This package can be installed using `apt-get`, but only while the Pi is connected to the Internet through its wired Ethernet port or a USB Ethernet adapter. When connected, install the firmware by typing the following:

```
sudo apt-get install firmwarepackage
```

Replace firmwarepackage in this command with the name of the package that you found by using `apt-cache`. For the example Zyxel NWD2105, the full command would be `sudo apt-get install firmware-ralink.`

Installing Wireless Firmware Offline

If you can't connect the Pi to the Internet using any method other than a wireless connection, you'll need to download the firmware on a different computer. In a web browser, load a search engine and type the name of the firmware package followed by the name of the distribution you're using and its version.

If you're using the recommend Debian distribution, the firmware for the Ralink RT2x00 chipset from the example can be found by searching for `firmware-ralink debian wheezy`. The search will lead you to a package file to download. In the case of Debian, this is a `.deb` file. For Fedora Remix, the same firmware is provided as a `.rpm` file.

Download this file, and then copy it to the Pi's SD card in the `/home/pi` directory, or onto a USB flash drive or other external storage device. Load the Pi, and then when it comes time to install the firmware, replace the package name with the name of the file you downloaded. For the example Zyxel NWD2105 card, the command would be the following:

```
sudo apt-get install firmware-ralink_0.35_all.deb
```

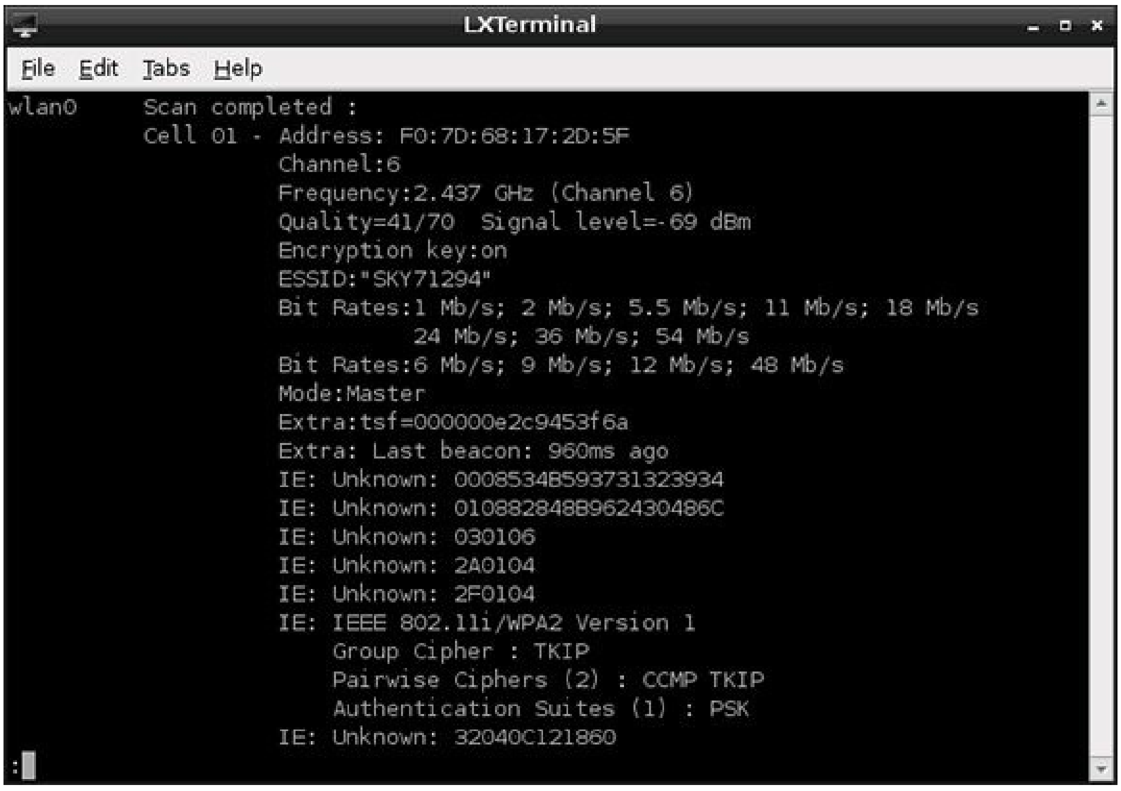
With the firmware installed, disconnect the USB wireless adapter and reconnect it to the Pi. This will restart the kernel's search for the firmware files, which it will now be able to find. These files will remain in place, and load automatically when the USB wireless adapter is connected. You will only have to perform the installation process once.

With the firmware installed, setting the wireless connection up should be straightforward. First, check that the USB wireless adapter is working as it should by using the `iwlist` command to scan for nearby wireless access points. This list will probably be larger than a single screen, so pipe the command's output through `less` to pause after each screenful, like this:

```
sudo iwlist scan | less
```

This command will return a list of all the wireless networks reachable from the Pi and their details (see Figure 4-4). If you receive an error message at this point—in particular, one that claims the network or interface is down—check that you have installed the correct firmware, and that the USB wireless adapter is connected to a powered USB hub.

Figure 4-4: Scanning for wireless networks with iwlist



```
wlan0     Scan completed :
          Cell 01 - Address: F0:7D:68:17:2D:5F
                    Channel:6
                    Frequency:2.437 GHz (Channel 6)
                    Quality=41/70  Signal level=-69 dBm
                    Encryption key:on
                    ESSID:"SKY71294"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                              24 Mb/s; 36 Mb/s; 54 Mb/s
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
                    Mode:Master
                    Extra:tsf=000000e2c9453f6a
                    Extra: Last beacon: 960ms ago
                    IE: Unknown: 0008534B593731323934
                    IE: Unknown: 010882848B962430486C
                    IE: Unknown: 030106
                    IE: Unknown: 2A0104
                    IE: Unknown: 2F0104
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (2) : CCMP TKIP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 32040C121860
:
```

You can check the current status of the network using the `iwconfig` command. Like `ifconfig`, the `iwconfig` command allows you to check the status of a network interface and issue configuration commands. Unlike `ifconfig`, however, `iwconfig` is specifically designed for wireless networks and includes specific features for this. Type the command name at the terminal as follows:

```
iwconfig
```

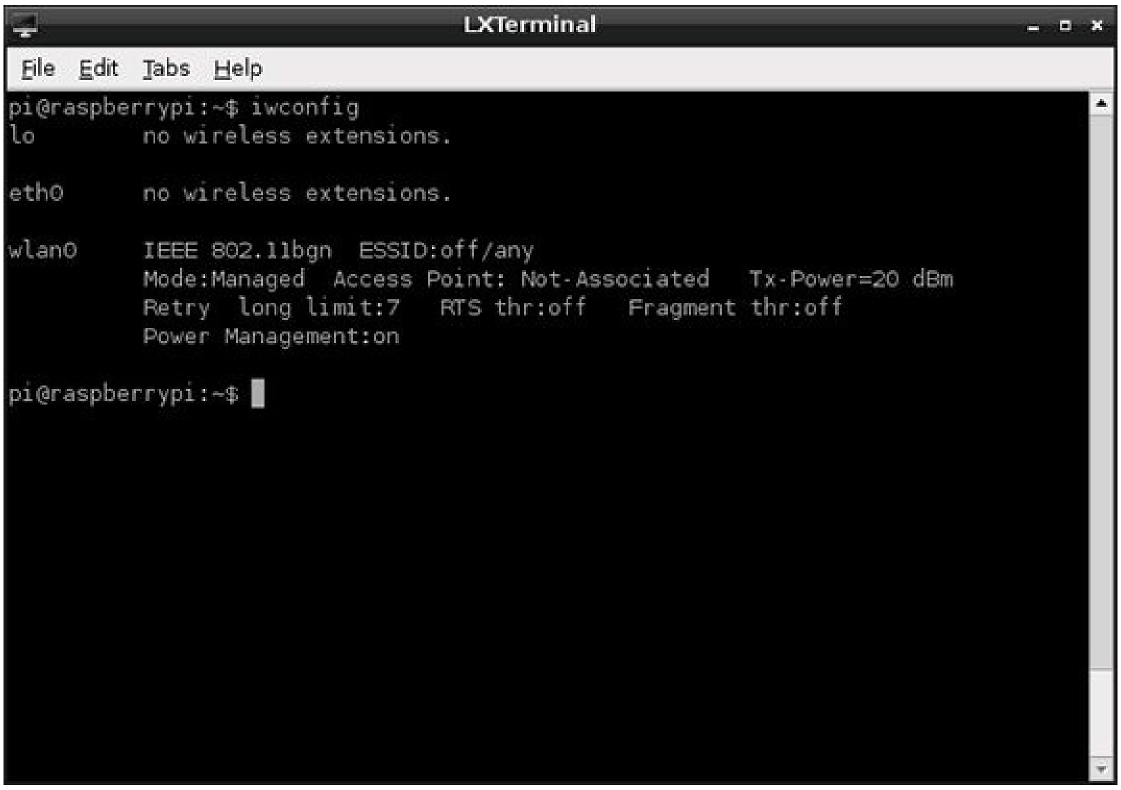
The output of `iwconfig`, as shown in Figure 4-5, is split into the following sections:

- Interface Name—Each device has its own interface name, as with wired networks. If the interface is a wireless connection, additional details will be shown. The default name for a Pi's wireless connection is `wlan0`.
- Standard—The IEEE 802.11 wireless standards have a variety of different types, distinguished by a letter suffix. This section lists the standards supported by the USB wireless adapter. For the example adapter, this reads `IEEE 802.11bgn` for the network types it can address.
- ESSID—The SSID of the network to which the adapter is connected. If the adapter is not currently connected to a network, this will read `off/any`.
- Mode—The mode that the adapter is currently operating in, which will be one of the following:
  - Managed—A standard wireless network, with clients connecting to access points. This is the mode used for almost all home and business networks.
  - Ad-Hoc—A device-to-device wireless network, with no access points.
  - Monitor—A special mode in which the card listens out for all traffic whether or not it is the addressee. This mode is typically used in network troubleshooting for capturing wireless network traffic.
  - Repeater—A special mode that forces a wireless card to forward traffic on to other network clients, to boost signal strength.
  - Secondary—A subset of the Repeater mode, which forces the wireless card to act as a backup repeater.
- Access Point—The address of the access point to which the wireless adapter is currently connected. If the adapter isn't connected to a wireless access point, this will read `Not-Associated`.
- Tx-Power—The transmission power of the wireless adapter. The number displayed here indicates the strength of the signal that the adapter is sending: the higher the number, the stronger the signal.
- Retry—The current setting for the wireless adapter's transmission retry, used on congested networks. This does not normally need changing, and some cards won't allow it to be changed.
- RTS—The adapter's current setting for Ready To Send and Clear To Send (RTS/CTS) handshaking, used on busy networks to prevent collisions. This is normally set by the access point on connection.
- Fragment—The maximum fragment size, used on busy networks to split packets up into multiple fragments. This is normally

set by the access point on connection.

• Power Management—The current status of the adapter's power management functionality, which reduces the device's power demands when the wireless network is idle. This has little effect on the Pi, but is typically enabled for battery-powered devices like a laptop.

Figure 4-5: The output of iwconfig when not connected to a wireless network



```
pi@raspberrypi:~$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

pi@raspberrypi:~$
```

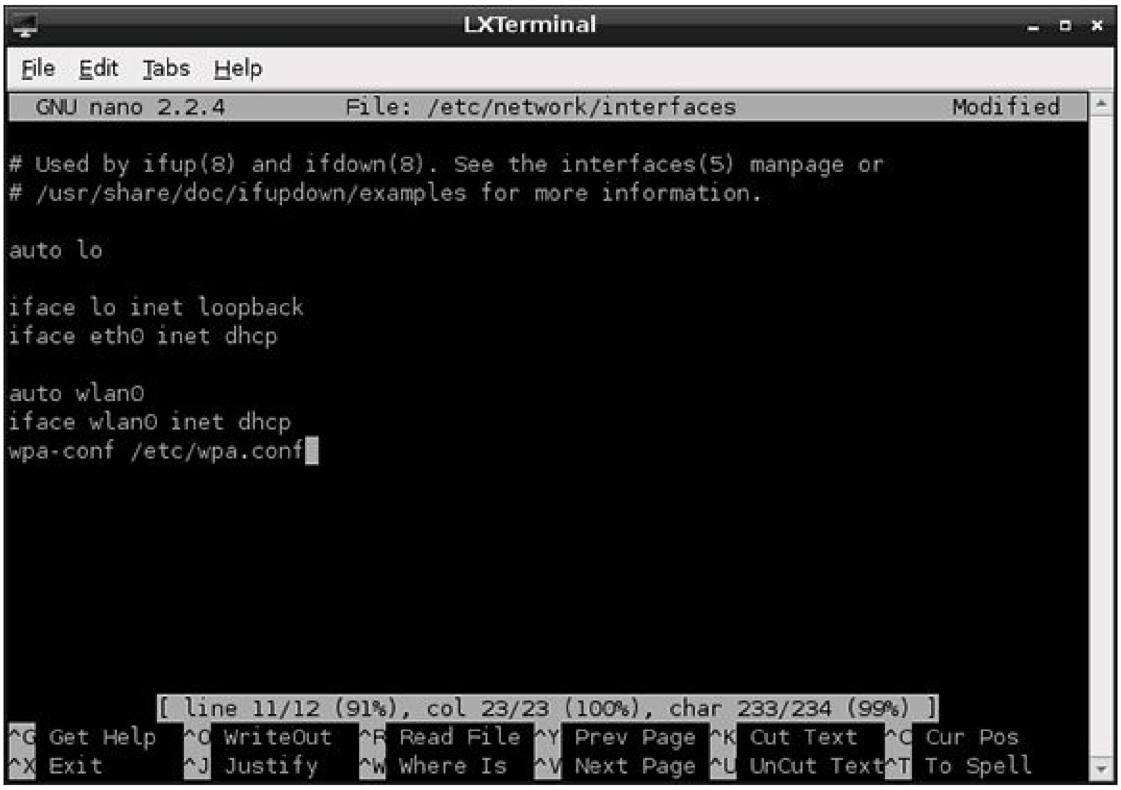
To connect the Pi to a wireless network, you will need to add some lines into the `/etc/network/interfaces` file. (For full details on how this file is laid out, see the "Wired Networking" section earlier in this chapter.) First, open the file in the `nano` text editor:

```
sudo nano /etc/network/interfaces
```

At the bottom of the file, create a new entry for the USB wireless adapter that reads as follows (see Figure 4-6):

```
auto wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa.conf
```

Figure 4-6: Editing the interfaces file for wireless network access



```
  GNU nano 2.2.4          File: /etc/network/interfaces               Modified

# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.

auto lo

iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa.conf

                  [ line 11/12 (91%), col 23/23 (100%), char 233/234 (99%) ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Once the entry is in place, save the file by pressing CTRL + O and then quit `nano` with CTRL + X.

TIP

The device ID of wlan0 is correct if this is the first wireless device you've set up on your Pi. If it isn't, the number at the end will be different. Type iwconfig to see a current list of wireless devices, and change the lines in the preceding code example accordingly.

The last line of the `interfaces` file makes reference to a configuration file, `wpa.conf`, which does not yet exist. This file is used by a tool known as `wpasupplicant`, designed to provide Linux with an easy way to connect to networks secured with Wireless Protected Access (WPA) encryption.

Using `wpasupplicant`, you can connect the Pi to almost any wireless network—regardless of whether it's protected by WPA or its newer replacement WPA2—in both Advanced Encryption Standard (AES) and Temporal Key Integrity Protocol (TKIP) modes. Despite its name, `wpasupplicant` also allows connection to wireless networks using the older Wired Equivalent Privacy (WEP) encryption standard.

The `wpasupplicant` program stores its configuration in a file called `wpa.conf`, located in the `/etc` directory. To begin configuring the Pi for wireless access, first open a new blank file for editing by typing the following:

```
sudo nano /etc/wpa.conf
```

Enter the following two lines, which again, are the same for any wireless network type. Replace Your_SSID with the SSID for the wireless network to which you want to connect, and then finish the file with the lines that match your network's encryption type.

```
network={
[Tab] ssid="Your_SSID"
```

At this point in the configuration file, the details required differ depending on the type of wireless network you are configuring. The following subsections provide instructions for completing the configuration for unencrypted, WEP and WPA networks.

No Encryption

If your wireless network has no encryption in place, finish the `wpa.conf` file as follows:

```
[Tab] key_mgmt=NONE
}
```

Save the file with CTRL + O, and then exit `nano` with CTRL + X.

WEP Encryption

If your wireless network uses WEP encryption, finish the `wpa.conf` file as follows:

```
[Tab] key_mgmt=NONE
[Tab] wep_key0="Your_WEP_Key"
}
```

Replace Your_WEP_Key with the ASCII key for your wireless network's WEP encryption. Save the file with CTRL + O, and then exit `nano` with CTRL + X.

TIP

WEP encryption is extremely insecure. Readily-available software can break the encryption on a WEP-protected network in just a few minutes, allowing a third party to use your network. If you're still running WEP, consider switching to WPA or WPA2 for better security.

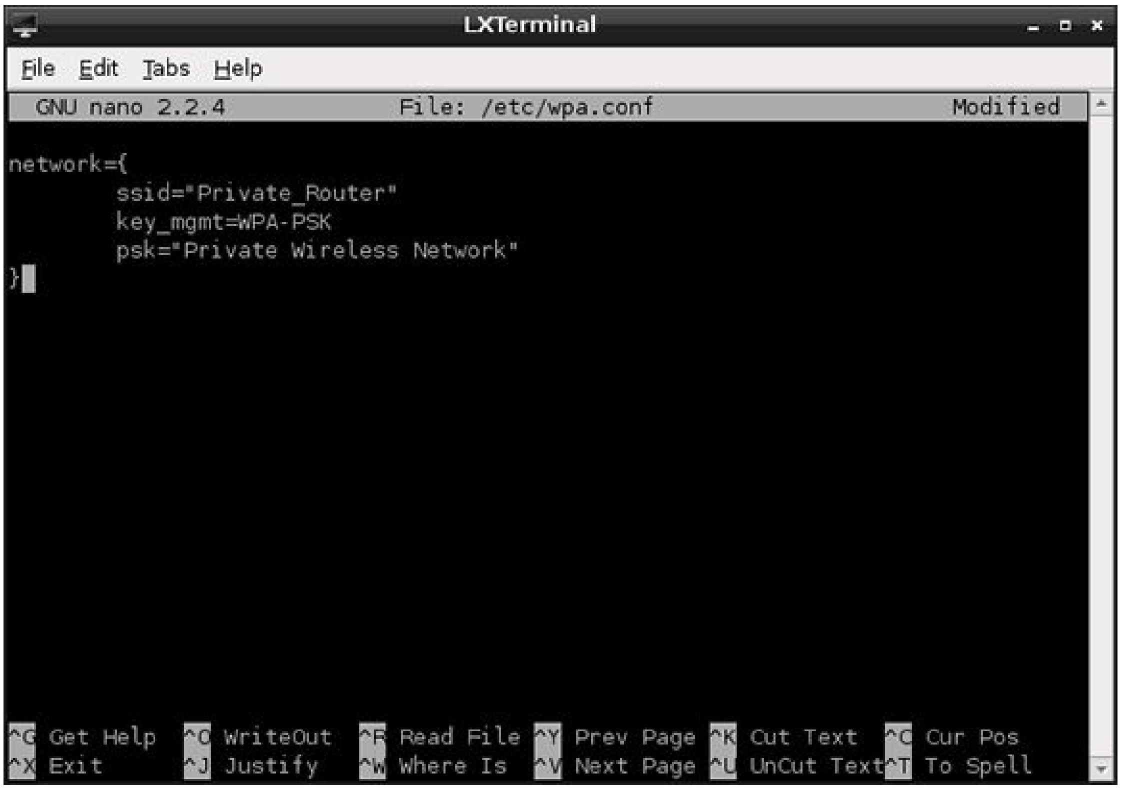
WPA/WPA2 Encryption

If your wireless network uses WPA or WPA2 encryption, finish the `wpa.conf` file as follows:

```
[Tab] key_mgmt=WPA-PSK
[Tab] psk="Your_WPA_Key"
}
```

Replace Your_WPA_Key with the pass phrase for your wireless network's encryption. Figure 4-7 shows an example configuration for a wireless network with the SSID "`Private_Router`" and the WPA pass phrase "`Private Wireless Network`". Save the file with CTRL + O, and then exit `nano` with CTRL + X.

Figure 4-7: Editing the `wpa.conf` file for a WPA-protected network



```
  GNU nano 2.2.4              File: /etc/wpa.conf                    Modified

network={
        ssid="Private_Router"
        key_mgmt=WPA-PSK
        psk="Private Wireless Network"
}

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Connecting to the Wireless Network

The Pi's wireless networking is now configured, and will begin the next time the Pi is restarted. To start the wireless network without rebooting, type the following:

```
sudo ifup wlan0
```

To make sure that the network is operational, unplug the Pi's Ethernet cable (if attached) and type the following:

```
ping -c 1 www.raspberrypi.org
```

TIP

If you start having problems with your Pi following the installation of a USB wireless adapter, it could be due to a conflict with other USB devices. Some adapter models are known to cause problems with certain USB keyboards. For an up-to-date list of adapters that are known to be good, as well as those that are known to cause conflicts, visit http://www.element14.com/community/docs/DOC-44703/l/raspberry-pi-wifi-adapter-testing or the eLinux wiki at http://elinux.org/RPi_VerifiedPeripherals#Working_USB_Wifi_Adapters.

Chapter 5: Partition Management

Having the Raspberry Pi's operating system provided as an image of somebody else's SD card is convenient, but a little inflexible. Most distribution images available for download assume a 2 GB or 4 GB SD card, meaning that people with 8 GB or larger cards find much of their space wasted.

Creating a New Partition

One way to make the most of a large SD on the Raspberry Pi is to create a new partition in the empty space at the end of the card. This partition can be used to store any large files that you want your Pi to be able to access without having to use an external storage device.

All the tasks required for this can be carried out directly on the Pi, without having to remove the SD card and connect it to a PC. All you need is an SD card flashed with one of the Raspberry Pi Linux images (as described in Chapter 1, "Meet the Raspberry Pi") and some free space.

WARNING

When using tools that can modify the partitions on a disk, it's important not to delete any existing partitions. In particular, be careful not to delete the /boot or root file system partitions, or the Pi will be unable to load.

Follow these steps to create a new partition on an SD card:

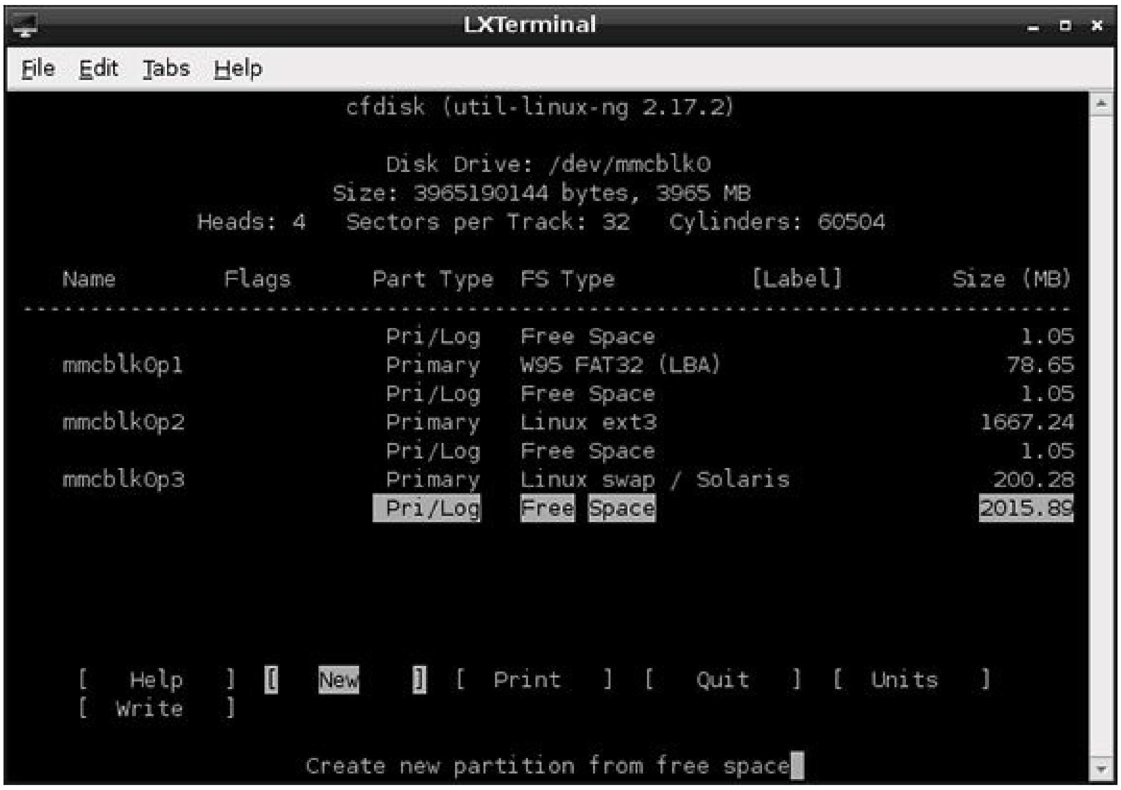
1. Open a terminal window, and type `sudo fdisk -l` to list the storage devices connected to the Pi and their current partitions. The SD card will appear as `/dev/mmcblk0` with a series of partitions numbered `p0`, `p1` and so on.

2. Load the `cfdisk` menu-based partition management tool by typing `sudo cfdisk /dev/mmcblk0`. This will use the SD card as the target device. If you're trying to create a partition on a USB Mass Storage device, replace `mmcblk0` with the device identifier for the storage device (for example, `sda`).

3. Using the cursor keys, move the partition selection highlight—which prints the currently selected partition in inverse video, typically black on white—down the partition list to the section labelled Free Space at the bottom.

4. Again using the cursor keys, move the option selection highlight, located at the bottom of the screen and displayed in inverse video, right to the New option and press Enter (see Figure 5-1).

Figure 5-1: Creating a new partition using cfdisk



```
                         cfdisk (util-linux-ng 2.17.2)

                            Disk Drive: /dev/mmcblk0
                        Size: 3965190144 bytes, 3965 MB
              Heads: 4   Sectors per Track: 32   Cylinders: 60504

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                            Pri/Log   Free Space                           1.05
    mmcblk0p1               Primary   W95 FAT32 (LBA)                     78.65
                            Pri/Log   Free Space                           1.05
    mmcblk0p2               Primary   Linux ext3                        1667.24
                            Pri/Log   Free Space                           1.05
    mmcblk0p3               Primary   Linux swap / Solaris               200.28
                            Pri/Log   Free Space                        2015.89




     [   Help   ]  [   New    ]  [  Print   ]  [   Quit   ]  [  Units   ]
     [  Write   ]

                       Create new partition from free space
```

5. You will be prompted to create a Primary or Logical partition. The default is to create a Primary partition, so just press

Enter to accept this.

6. The next prompt will ask you how big the partition should be, in megabytes (MB). The default is to create a partition that fills the entire available free space on the device, so again, just press Enter.

7. Next, the new details—known as a partition table—need to be written to the disk. Use the cursor keys to move the option selection highlight to Write and press Enter.

8. You will be prompted to make sure that the changes are correct. Double-check that you've created the partition as instructed, and then type `yes` and press Enter.

9. Exit `cfdisk` by moving the option selection highlight at the bottom of the screen to Quit with the cursor keys and then pressing Enter.

Although the partition table has now been updated with the freshly created partition, it won't be visible to the operating system until it has been reloaded. The easiest way to achieve that is to restart the Pi by typing the following:

```
sudo reboot
```

When the Pi has restarted and you've logged back in, you can use `fdisk` to verify that the new partition is ready for use with the following command:

```
sudo fdisk -l
```

Before you can store files on the new partition, however, you need to put a file system in place by formatting the drive using the `mkfs` tool as follows.

1. Type `fdisk -l` to list the storage devices connected to the Pi and their current partitions. Note the new partition, which will appear as `/dev/mmcblk0p`N where N is the partition number. If you're doing this on the recommended Debian distribution, the partition will be `/dev/mmcblk0p3`.

2. Create a new EXT4 file system in the partition by typing `sudo mkfs.ext4 /dev/mmcblk0p`N, replacing N with the new partition's number. Make sure that you've picked the right partition: the `mkfs` (make file system) command will wipe any data on the partition it is told to format.

3. Before the new file system can be used, it must be mounted. Create a mount point (an empty directory) by typing `sudo mkdir /storage` at the terminal.

4. Use the `mount` command to make the new partition accessible on the mount point you just created by typing `sudo mount /dev/mmcblk0p`N `/storage`, where N is the new partition number.

WARNING

When using mkfs to create a new, blank file system on a partition, always double-check the partition details before continuing. If you give mkfs the wrong partition, it will erase any files you have stored there. If it's the /boot or root file system that has been wiped, the Pi will no longer load until you flash the SD card again.

While this gives you a storage device that the root superuser can access, the pi standard user and any user accounts you've created yourself don't currently have permission to store files there. That can be changed with a trio of commands: `chown`, `chgrp` and `chmod`.

The first command, `chown`, is short for change ownership and allows files created by one user to be passed across to another; `chgrp` changes the group to which a file belongs so all the members of that group can access it; and `chmod` modifies the permissions on a file or directory.

To allow all users to access your new partition, change the group membership from `root` to `users` with the `chgrp` command, using the `-R` (recursive) flag to affect the directory's entire contents as follows:

```
sudo chgrp -R users /storage
```

You'll also need to allow all members of the group to write to the directory. To do this, you use the `chmod` command with the option `g+w`, which tells `chmod` to allow write access from the group:

```
sudo chmod -R g+w /storage
```

The new partition is now ready for use, but there's still one more task to carry out. At present, the partition needs to be manually mounted (using the `mount` command combined with the `sudo` command for running as the root user) each time the Pi reboots. To save time, you can tell the Pi to automatically mount the new partition instead by editing the `fstab` file.

Short for file system table, the `fstab` file—located in the `/etc` directory—tells Linux what file systems should be mounted on which mount points. This table may look complicated at first glance, but its layout follows a logical tabular pattern.

From left to right, the columns tell Linux the location of the device to be mounted, the directory where the device should be accessible (the mount point), the file system type, any options required, and finally, two numbers that control whether the file system should be dumped in the event of a system problem and whether it should be checked by the `fsck` (file system check) tool.

To make the system mount the new partition automatically, first open the `fstab` file in `nano`:

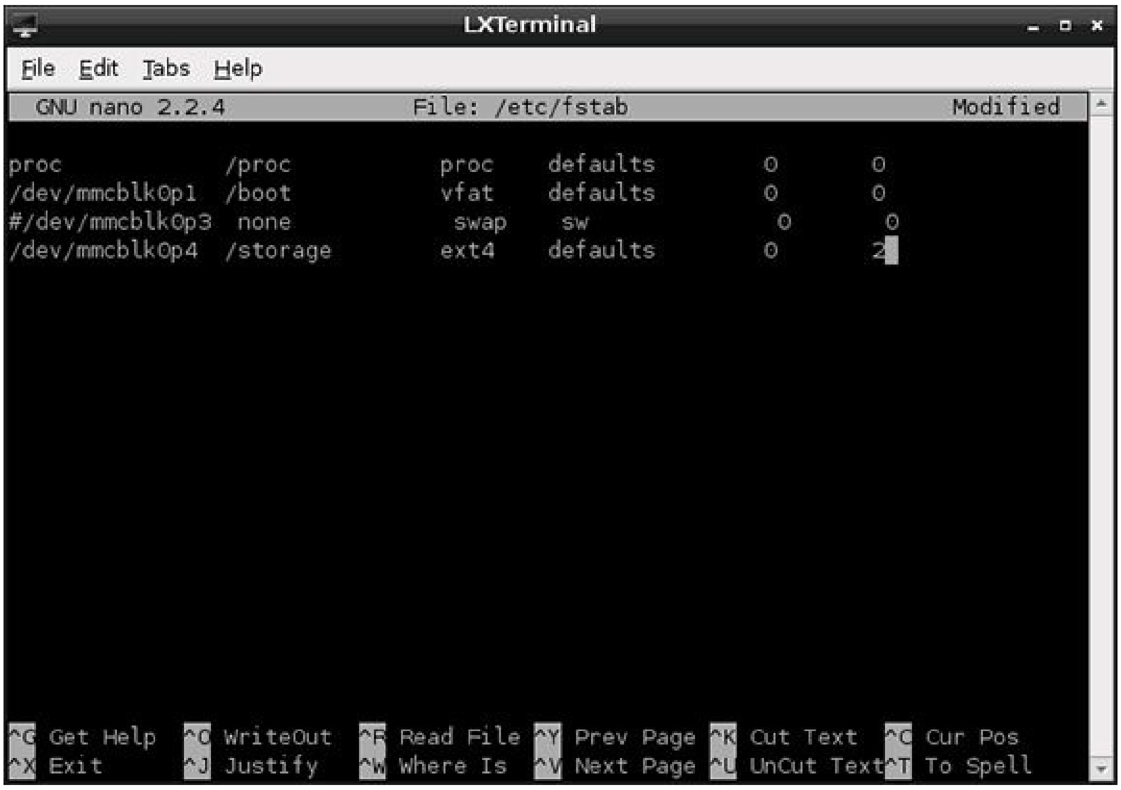
```
sudo nano /etc/fstab
```

Add a new line at the bottom of the file, defining the various options required by the new partition, with tabs between each field:

```
/dev/mmcblk0pN [Tab]/storage[Tab] ext4[Tab] defaults[Tab] 0[Tab] 2
```

Remember to change N for the partition number of the new partition (see Figure 5-2). If you're using `fstab` to mount external storage devices, use the device name `/dev/sd`XN where X is the device letter and N is the partition number. Save the file with CTRL + W, and then exit `nano` with CTRL + X. When you're back at the terminal, reboot your system and check if `/storage` is mounted automatically by typing `mount`. If not, double-check your new `fstab` entry. Remember that you need to press the Tab key each time you come to the end of a field.

Figure 5-2: Editing fstab to automatically mount the new partition



```
  GNU nano 2.2.4              File: /etc/fstab                    Modified

proc            /proc           proc    defaults        0       0
/dev/mmcblk0p1  /boot           vfat    defaults        0       0
#/dev/mmcblk0p3  none            swap    sw               0       0
/dev/mmcblk0p4  /storage        ext4    defaults        0       2







^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Resizing Existing Partitions

Creating a new partition is one way to make use of a larger SD card, but it's not the most flexible. A better method is to resize existing partitions to make use of the free space. To do this reliably, you'll need to unplug the Pi, remove the SD card and insert it into a desktop or laptop through a card reader.

Automatic Resizing

The Debian Linux distribution for the Raspberry Pi comes with a tool called `raspi-config`, which loads when the system is booted for the first time. It can also be loaded manually at any time by typing `sudo raspi-config` at the console or in a terminal window. This tool provides an interface for many common configuration tasks, including the ability to resize the root file system to make full use of the available space on an SD card automatically.

WARNING

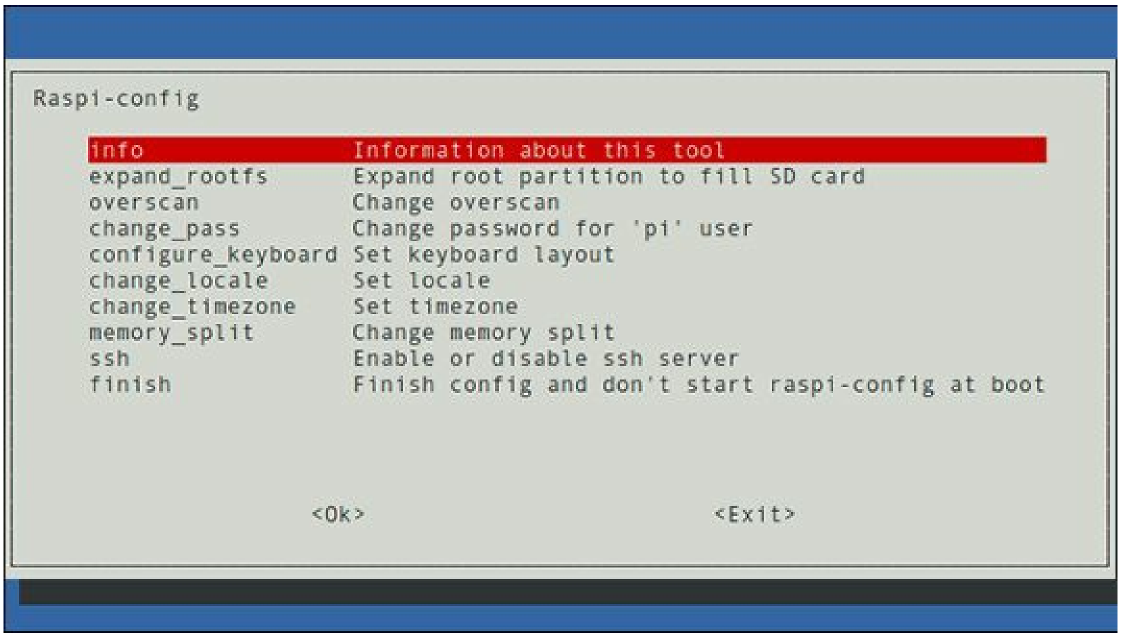
Using the raspi-config tool to resize the root file system can, in rare cases, result in data corruption. If you have data stored on the Pi which you can't afford to lose, back it up first or consider following the more reliable manual resizing instructions found later in this chapter.

To resize the root file system using the `raspi-config` tool, follow these instructions:

1. If this is the first time you have loaded Debian on the Raspberry Pi, `raspi-config` will load automatically. If it does not, type `sudo raspi-config` at the console or terminal to load the tool manually.

2. In the `raspi-config` menu (see Figure 5-3), press the down arrow on the keyboard to highlight the `expand_rootfs` option and then press Enter.

Figure 5-3: The raspi-config tool's menu screen



```
Raspi-config

    info               Information about this tool
    expand_rootfs      Expand root partition to fill SD card
    overscan           Change overscan
    change_pass        Change password for 'pi' user
    configure_keyboard Set keyboard layout
    change_locale      Set locale
    change_timezone    Set timezone
    memory_split       Change memory split
    ssh                Enable or disable ssh server
    finish             Finish config and don't start raspi-config at boot


                  <Ok>                          <Exit>
```

3. The resizing operation takes just a few seconds, and is followed by a message telling you that the process will complete when the Pi is next restarted. Press Enter to dismiss this message.

4. Press the Tab key twice to highlight Exit, and then press Enter to quit `raspi-config`.

5. Type `sudo reboot` to restart the Pi. The reboot process will take longer than usual, because the file system will need to be resized. This process happens only once per resize—the next time the Pi is rebooted, it will take no longer than usual.

When the Pi has fully rebooted, the root file system will now be as large as the SD card allows. To verify this, type `df -h` at the terminal to list the free space on all connected storage devices.

Manual Resizing

The most reliable way to resize Linux partitions on a desktop or laptop computer is to use a tool called Parted Magic, a free bootable CD that is designed specifically for adjusting file systems. The disc works on both PCs and Macs, and operates entirely from memory. As a result, it won't try to replace your existing operating system. It's also compatible with any distribution for the Raspberry Pi, unlike the Debian-specific `raspi-config` tool.

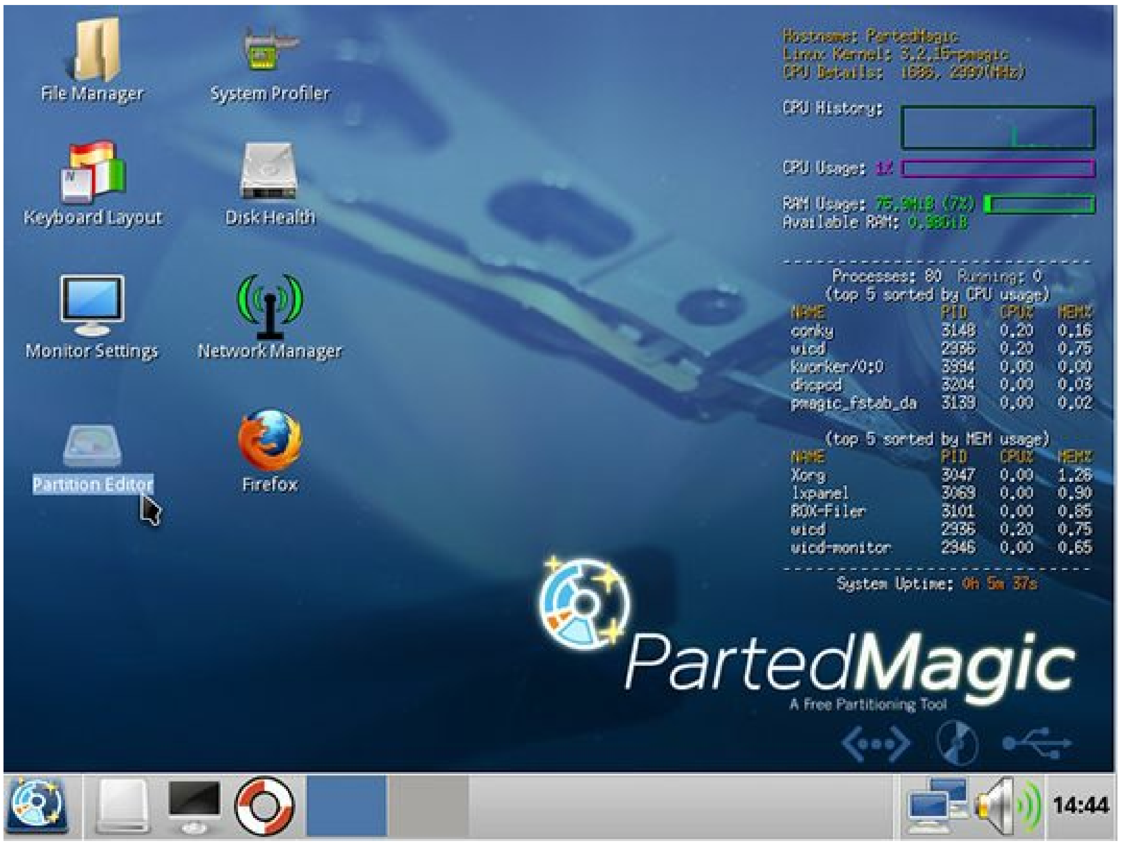
TIP

If you're a Linux user, you can install gparted—the graphical partitioning tool used in Parted Magic—instead of having to boot from the CD. For Debian-based distributions, you just type sudo apt-get install gparted followed by sudo gparted to load the program.

Download the Parted Magic ISO image file from http://partedmagic.com and write it to a CD or DVD using the CD writing program provided on your PC. With the disc still in the drive, reboot your computer and it will load into the Parted Magic menu system. From here, choose Standard Settings to load the software itself.

Parted Magic is a customised Linux operating system that includes tools specifically designed for managing storage devices. Connect the Raspberry Pi's SD card to your computer and load Partition Editor from the desktop by double-clicking the icon (see Figure 5-4).

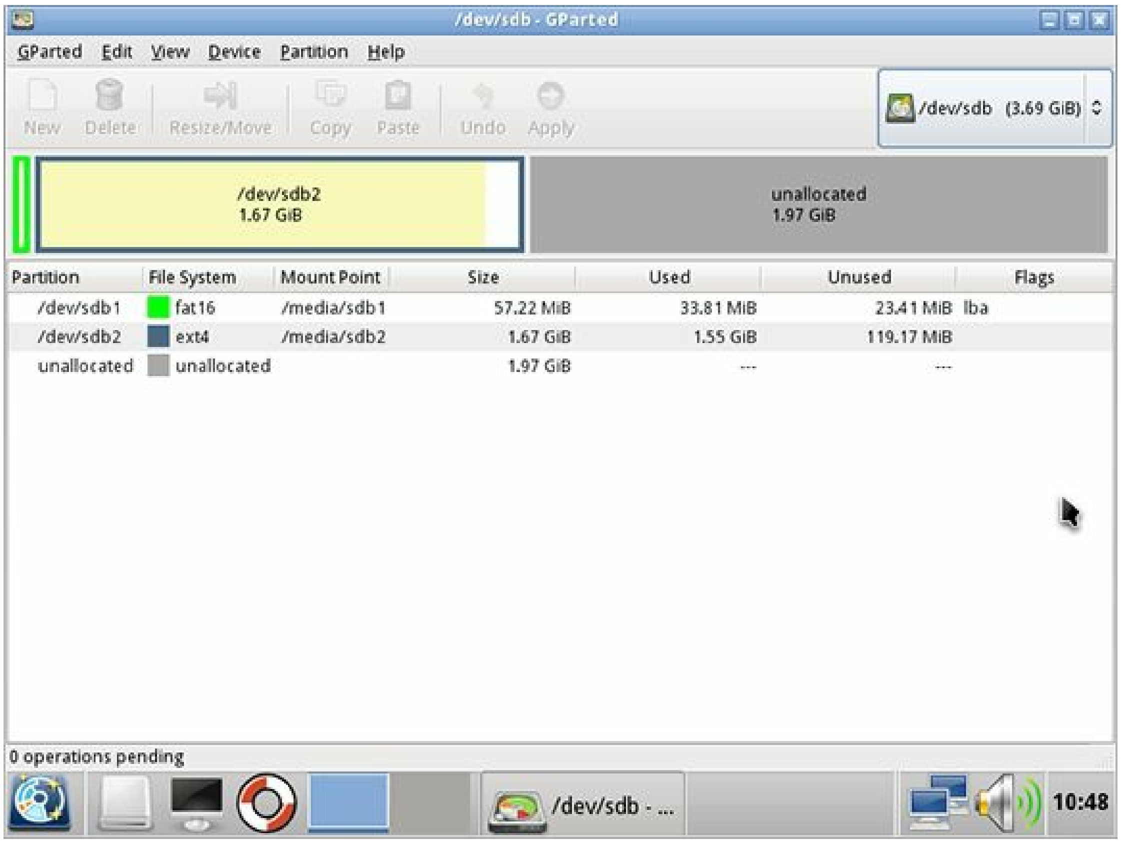
Figure 5-4: The Parted Magic desktop



```
Hostname: PartedMagic
Linux Kernel: 3.2.15-pmagic
CPU Details:  1686, 2397(MHz)

CPU History:

CPU Usage: 1%

RAM Usage: 75.9MiB (7%)
Available RAM: 0.98GiB

-----------------------------------
     Processes: 80  Running: 0
    (top 5 sorted by CPU usage)
 NAME               PID   CPU%   MEM%
 conky              3148  0.20   0.16
 wicd               2936  0.20   0.75
 kworker/0:0        3394  0.00   0.00
 dhcpcd             3204  0.00   0.03
 pmagic_fstab_da    3139  0.00   0.02

    (top 5 sorted by MEM usage)
 NAME               PID   CPU%   MEM%
 Xorg               3047  0.00   1.28
 lxpanel            3069  0.00   0.90
 ROX-Filer          3101  0.00   0.85
 wicd               2936  0.20   0.75
 wicd-monitor       2946  0.00   0.65
-----------------------------------
     System Uptime: 0h 5m 37s
```

By default, the partition editor will look at the first drive it finds in your system, which is usually your PC's hard drive. You don't want to make changes to that, so make sure to click on the device selector in the top-right corner and choose the device corresponding to the SD card. On a single-drive system, this will usually be `/dev/sdb` (see Figure 5-5).

Figure 5-5: Parted Magic's Partition Editor tool, before resizing the partition



Partition	File System	Mount Point	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	/media/sdb1	57.22 MiB	33.81 MiB	23.41 MiB	lba
/dev/sdb2	■ ext4	/media/sdb2	1.67 GiB	1.55 GiB	119.17 MiB	
unallocated	▨ unallocated		1.97 GiB	---	---	

0 operations pending

WARNING

Resizing and moving partitions is a risky process. If the SD card is removed while the resize is in progress, or the PC loses power, it will corrupt the contents of the card. Always make sure you've backed up any irreplaceable personal files from the card before editing the partitions.

The exact partitions that need to be resized and moved will differ according to the distribution chosen. In the case of Debian, you will need to resize the second partition, which will usually be `sdb2`. With the partition editor loaded, do the following:

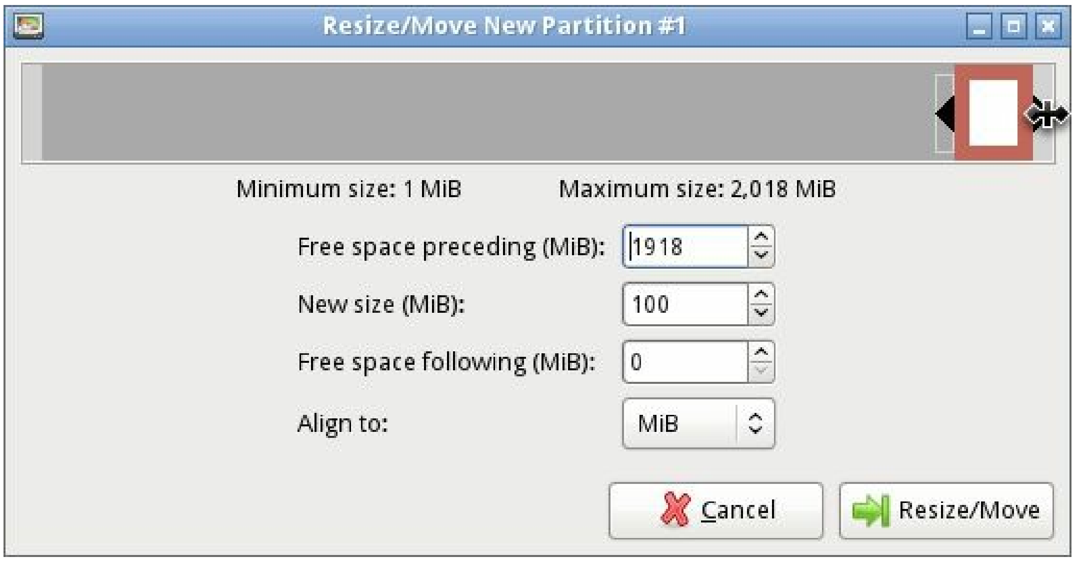
1. Some Linux distributions include a swap partition at the end of the image. This appears as a small partition of type `linux-`

`swap` in the partition editor. If this is present, continue with these instructions; if not, skip straight to step 5.

2. Click on the swap partition, which will be the last partition in the list, and choose Resize/Move from the toolbar.

3. In the dialogue box that appears, click and drag the box at the top left over to the top right (see Figure 5-6). Once complete, the Free Space Following box should read `0`.

Figure 5-6: Moving the swap partition in the partition editor



Minimum size: 1 MiB Maximum size: 2,018 MiB

Free space preceding (MiB): 1918

New size (MiB): 100

Free space following (MiB): 0

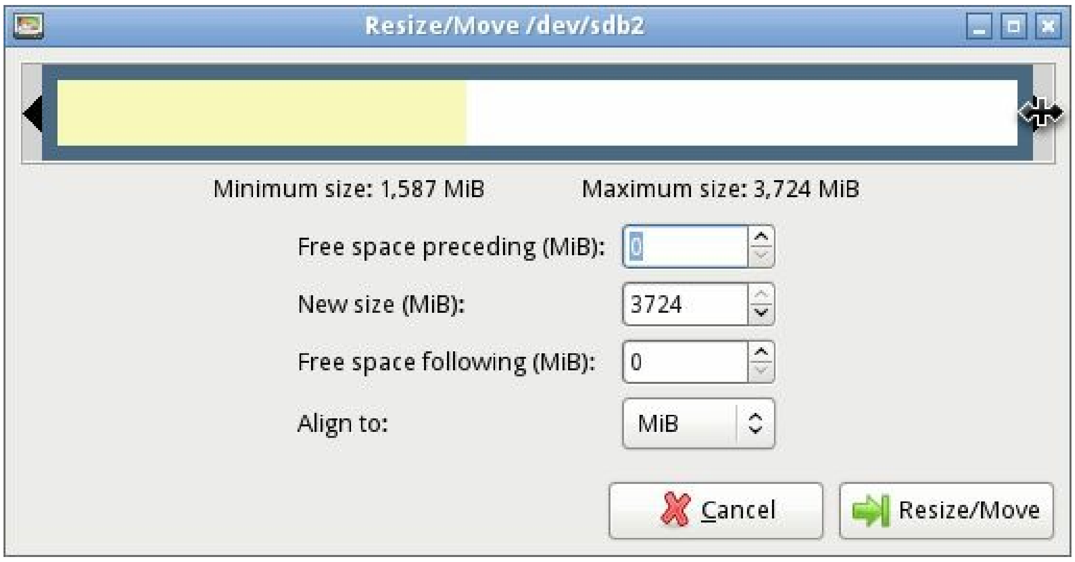
Align to: MiB

4. Click the Resize/Move button to confirm the change. Moving the partition will trigger a warning about the potential for the new partition table to lead to booting problems. That doesn't apply to this change, because you're not moving the boot partition, so just click OK.

5. Click on the largest partition in the list, which is typically labelled `sdb2`, and click Resize/Move on the toolbar again.

6. This time, click and hold the right arrow on the coloured box and drag it to the right edge of the grey box (see Figure 5-7). This will make the partition larger, rather than just moving it.

Figure 5-7: Resizing the root file system in the partition editor



Minimum size: 1,587 MiB Maximum size: 3,724 MiB

Free space preceding (MiB): 0

New size (MiB): 3724

Free space following (MiB): 0

Align to: MiB

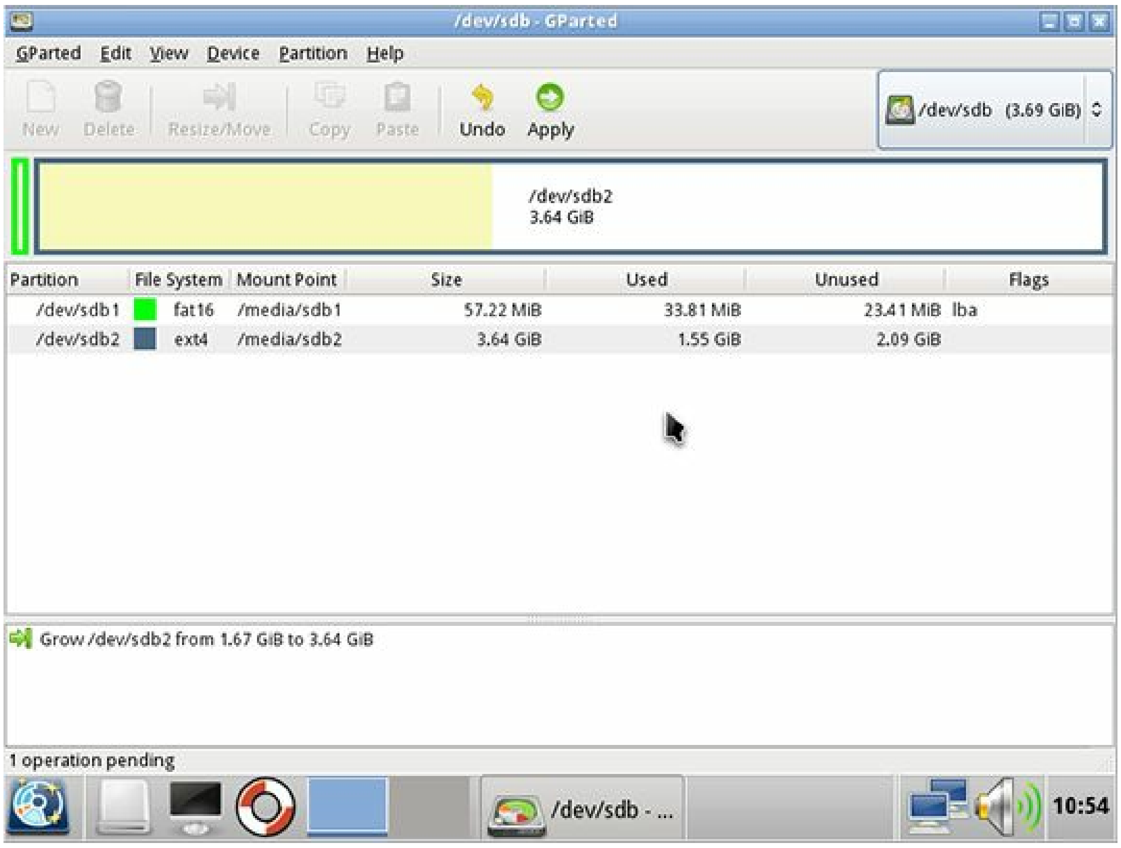
7. Click the Resize/Move button to confirm your change, and again click OK on the warning box that appears.

8. Click Apply on the toolbar, and again on the dialogue box that appears. Depending on the speed of your SD card, the resizing process may take a few minutes to complete.

9. When the resize has completed, you can turn off your PC and put the SD card back into your Pi.

Thanks to the changes made in the partition editor (see Figure 5-8), the main partition on the SD card is now as big as the card will allow it to be. As this is where most Linux distributions store both their own files and the users' files, the Pi should now have plenty of space available to use.

Figure 5-8: Parted Magic's Partition Editor tool, after resizing the partition


/dev/sdb - GParted
GParted Edit View Device Partition Help
New Delete Resize/Move Copy Paste Undo Apply
/dev/sdb (3.69 GiB)
/dev/sdb2
3.64 GiB

Partition	File System	Mount Point	Size	Used	Unused	Flags
/dev/sdb1	🟩 fat16	/media/sdb1	57.22 MiB	33.81 MiB	23.41 MiB	lba
/dev/sdb2	■ ext4	/media/sdb2	3.64 GiB	1.55 GiB	2.09 GiB	

Grow /dev/sdb2 from 1.67 GiB to 3.64 GiB

1 operation pending

Moving to a Bigger SD Card

If you've been using the Pi for a while, you may find that the 4 GB SD card you thought would be large enough for your needs has become full. Buying a new SD card with 8 GB, 16 GB or even more storage is cheap enough, but you don't want to lose your files.

Thankfully, it's pretty straightforward to move the contents of your existing SD card across to a bigger card. The latter half of the process is no different to how you flashed the SD card back in Chapter 1, "Meet the Raspberry Pi". Where it differs, however, is that you will be using the existing SD card as the source rather than a downloaded image file.

WARNING

The process of cloning an SD card is nondestructive, and will result in both cards having the same data on them. If you have personal files on the old SD card, make sure to wipe it clean before passing it on to a third party.vIn addition to your original SD card and the newer, larger card, you'll need access to a PC or Mac, an SD card reader and—if you're a Windows user—the Parted Magic disc used to resize the partitions earlier in this chapter.

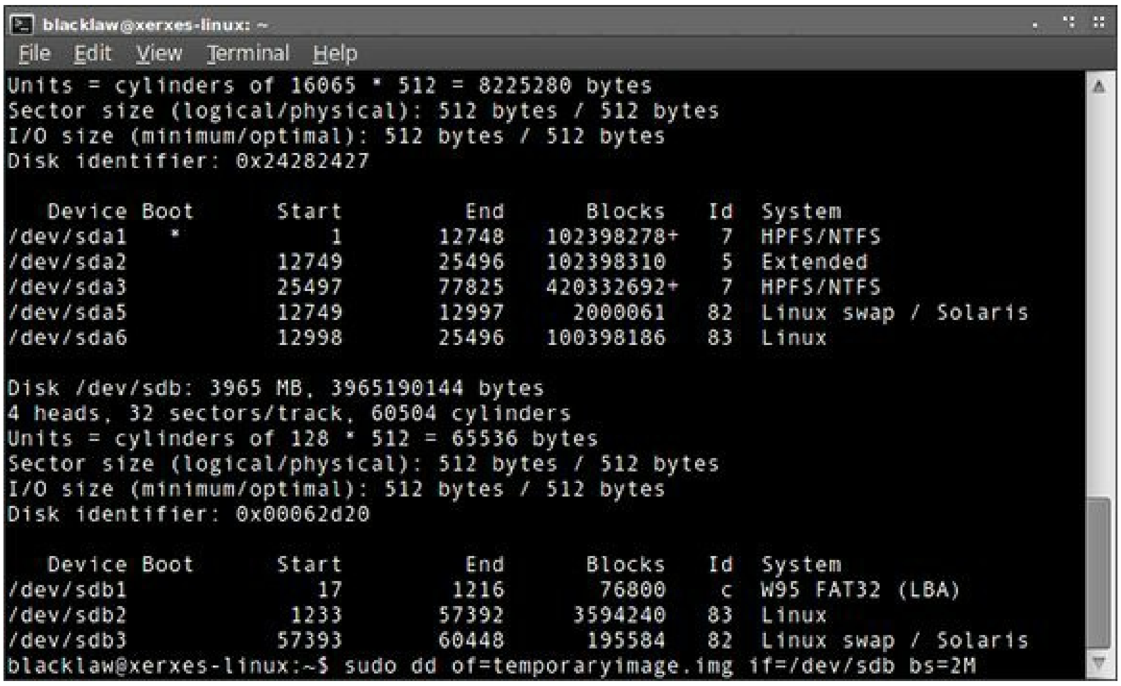
The first step to moving to a larger SD card is to create an image of the existing card. If you have access to two SD card readers, you can skip this step and, in the later instructions, replace the name of the image file with the device address of the SD card reader with the Pi's original card inserted.

Imaging from Linux

Creating an image of the SD card under Linux is no more difficult than flashing the card was in Chapter 1. Before starting, make sure you have enough disk space on your computer to hold a file the size of the SD card. Then follow these steps:

1. Open a terminal from your distribution's applications menu.
2. Plug your Pi's smaller SD card into a card reader connected to the PC.
3. Type `sudo fdisk -l` to see a list of disks. Find the SD card by its size, and note the device address (`/dev/sd`X, where X is a letter corresponding to the device. For some computers with in-built SD card readers, this may appear as `/dev/mmcblk`X where X is a letter corresponding to the device. If so, use that address in the following instructions.).
4. Type sudo dd of=temporaryimage.img if=/dev/sdX bs=2M to read the contents of the SD card and write it to a file called `temporaryimage.img` (see Figure 5-9).

Figure 5-9: Creating an image of an existing SD card using `dd`



```
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x24282427

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1       12748   102398278+   7  HPFS/NTFS
/dev/sda2           12749       25496   102398310    5  Extended
/dev/sda3           25497       77825   420332692+   7  HPFS/NTFS
/dev/sda5           12749       12997     2000061   82  Linux swap / Solaris
/dev/sda6           12998       25496   100398186   83  Linux

Disk /dev/sdb: 3965 MB, 3965190144 bytes
4 heads, 32 sectors/track, 60504 cylinders
Units = cylinders of 128 * 512 = 65536 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00062d20

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1              17        1216       76800    c  W95 FAT32 (LBA)
/dev/sdb2            1233       57392     3594240   83  Linux
/dev/sdb3           57393       60448      195584   82  Linux swap / Solaris
blacklaw@xerxes-linux:~$ sudo dd of=temporaryimage.img if=/dev/sdb bs=2M
```

Imaging from OS X

Imaging the Pi's SD card on OS X is almost exactly the same as flashing the SD card was back in Chapter 1. Again, make sure you have enough hard disk space to hold a file the size of the SD card. Then follow these steps:

1. Select Utilities from the Application menu, and then click on the Terminal application.

2. Plug your Pi's smaller SD card into a card reader connected to the PC.

3. Type `diskutil list` to see a list of storage devices. Find the SD card by its size, and note the device address (`/dev/disk`X, where X is a letter corresponding to the device).

4. If the SD card has been automatically mounted and appears on the desktop, type `diskutil unmountdisk /dev/disk`X to unmount it before proceeding.

5. Type `dd of=temporaryimage.img if=/dev/disk`X `bs=2M` to read the contents of the SD card and write it to a file called `temporaryimage.img`.

Imaging from Windows

The Windows Image Writer tool that you used to flash the SD card in Chapter 1, "Meet the Raspberry Pi", doesn't support the creation of images. Instead, you'll need to use the Parted Magic disc to gain access to the Linux `dd` utility, as follows:

1. Insert the Parted Magic CD into your PC, reboot and choose Standard Settings.

2. Open a terminal window using the third icon from the left on the bottom tool bar, which looks like a computer monitor (see Figure 5-10).

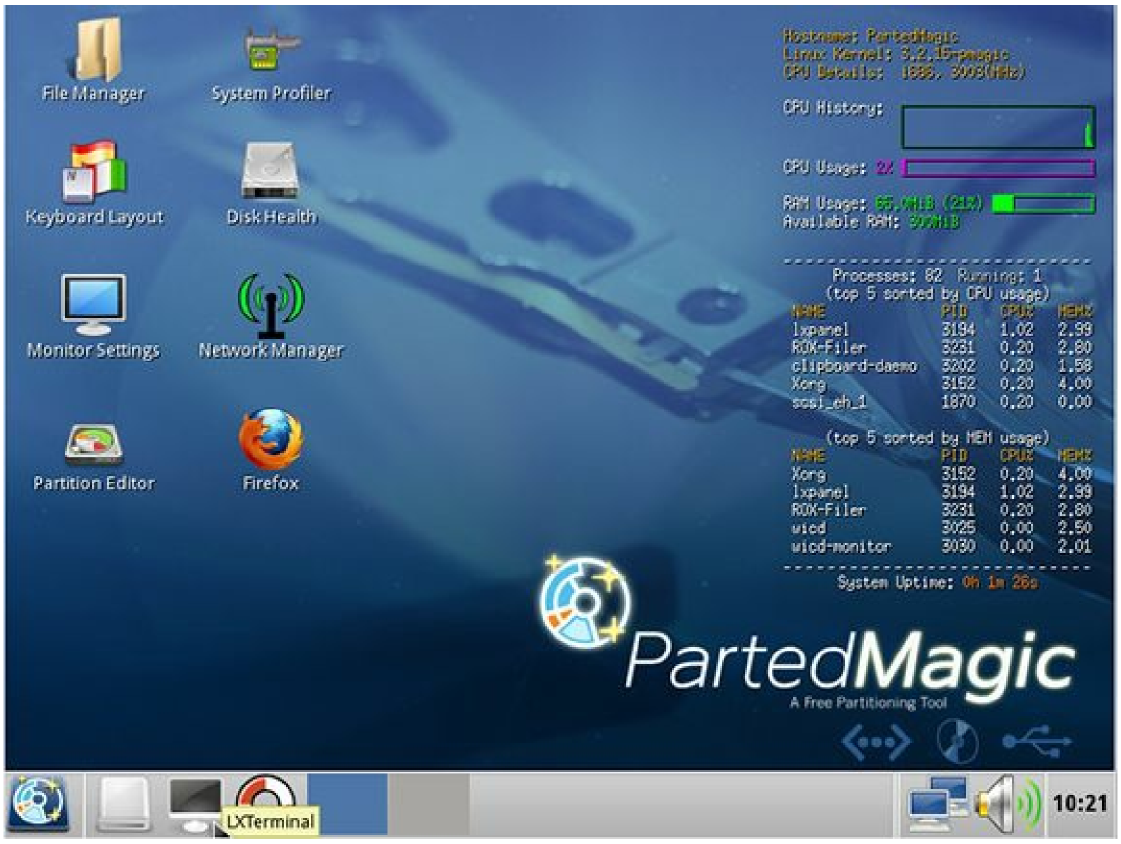
3. Type `fdisk -l` to get a list of drives on your PC, and find your main hard drive by size. Note the device name: `/dev/sd`XN, where X is the drive letter and N the partition number. For some computers with in-built SD card readers, this may appear as `/dev/mmcblk`X where X is a letter corresponding to the device. If so, use that address in the following instructions.

4. Create a mount point for your PC's hard drive by typing `mkdir /media/harddrive`, and then mount the drive with `mount /dev/sd`XN `/media/harddrive -o=rw` to gain access.

5. Insert your SD card reader with the Pi's smaller SD card into the PC, and then use `fdisk -l` to find its device node (`/dev/sd`Y where Y is the drive letter).

6. Type `dd of=/media/harddrive/temporaryimage.img if=/dev/sd`Y `bs=2M` to read the contents of the SD card and write it to a file called `temporaryimage.img` on your hard drive.

Figure 5-10: The terminal icon in Parted Magic



```
Hostname: PartedMagic
Linux Kernel: 3.2.15-pmagic
CPU Details:  i686, 3003(MHz)

CPU History:

CPU Usage: 2%

RAM Usage: 65.0MiB (21%)
Available RAM: 309MiB

--------------------------------
      Processes: 82  Running: 1
     (top 5 sorted by CPU usage)
 NAME               PID   CPU%   MEM%
 lxpanel            3194  1.02   2.99
 ROX-Filer          3231  0.20   2.80
 clipboard-daemo    3202  0.20   1.58
 Xorg               3152  0.20   4.00
 scsi_eh_1          1870  0.20   0.00

     (top 5 sorted by MEM usage)
 NAME               PID   CPU%   MEM%
 Xorg               3152  0.20   4.00
 lxpanel            3194  1.02   2.99
 ROX-Filer          3231  0.20   2.80
 wicd               3025  0.00   2.50
 wicd-monitor       3030  0.00   2.01
--------------------------------
       System Uptime: 0h 1m 26s
```

Now that you have your disk image, use the instructions on flashing an SD card from Chapter 1, "Meet the Raspberry Pi", to write it to the new card. Remember that writing an image takes time, so be patient and let it finish fully.

When the image writing has finished, you'll have two SD cards containing the exact same data, including the same partition table. This means that while the new card might be 16 GB or 32 GB, the Pi will only be able to access the same 2 GB or 4 GB of the original card.

To ensure the Pi can make use of the space on the new card, follow the instructions in "Resizing Existing Partitions" earlier in this chapter.