

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 character. 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.

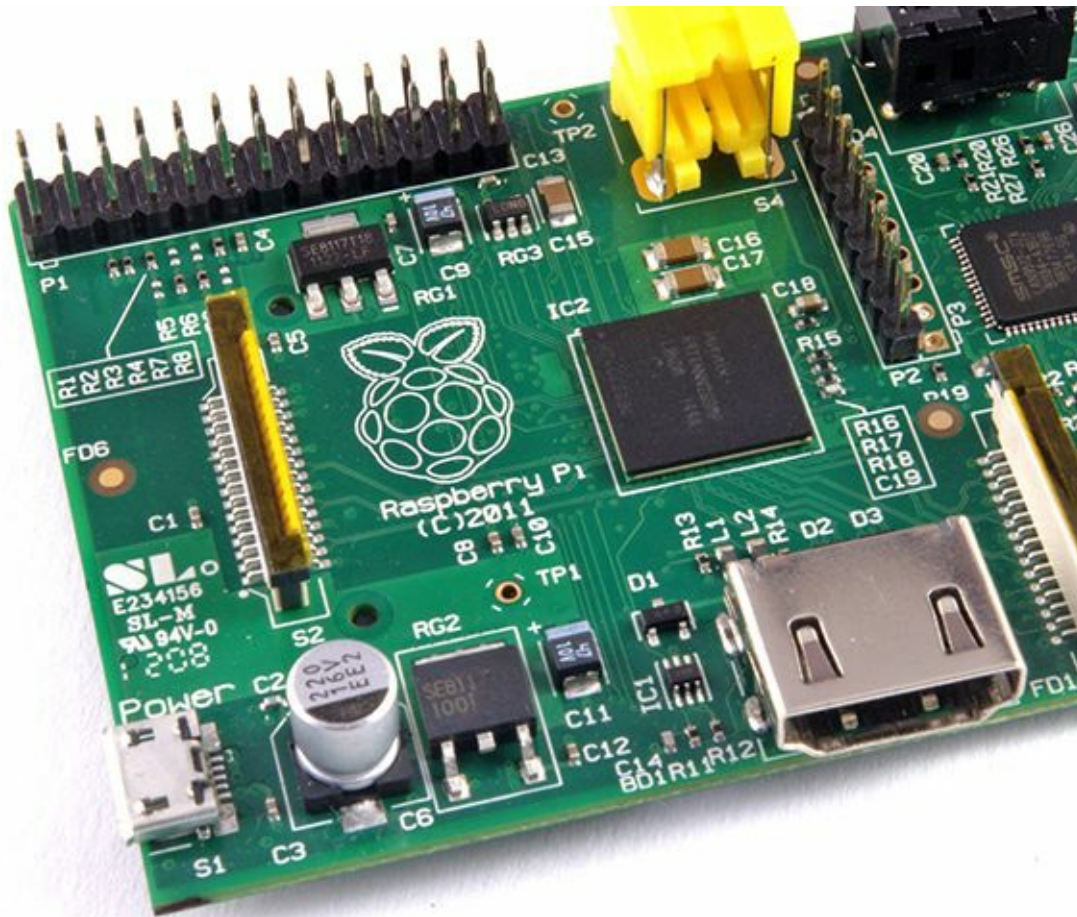
TO use the voltage test points, you'll need a voltmeter or multimeter with direct current (DC) voltage measuring capabilities. If your meter has multiple inputs for different voltages, use an appropriate setting.

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 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 or

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`.

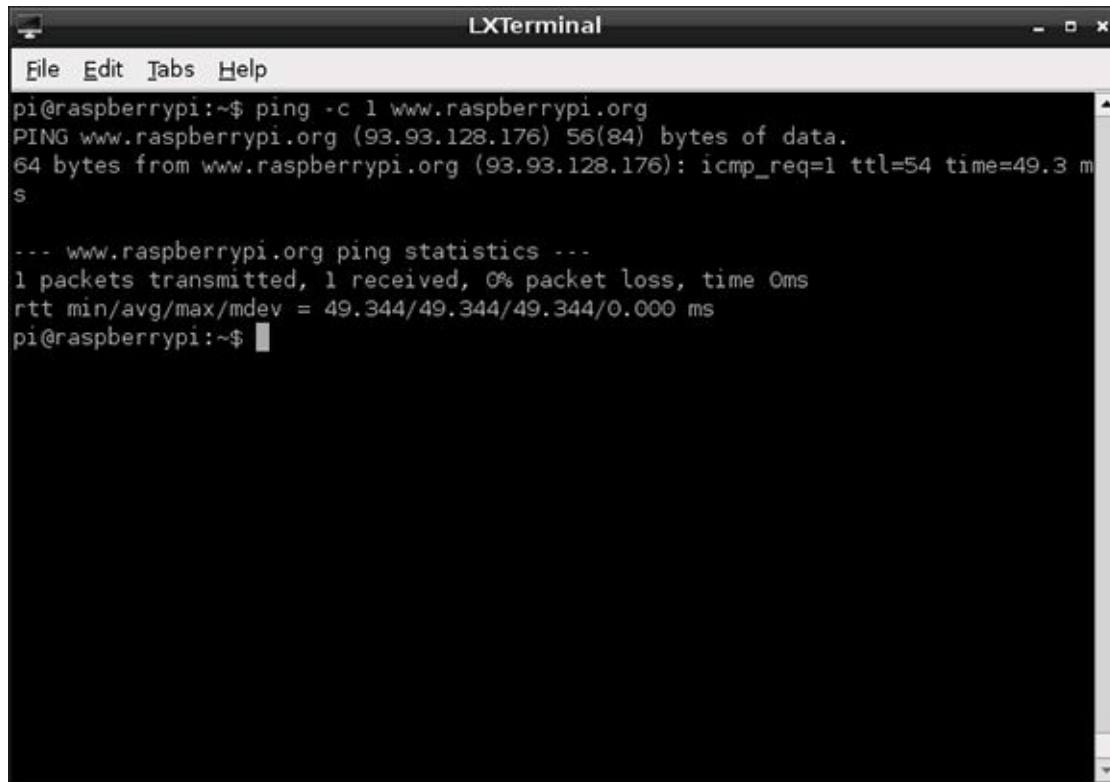
If the network is up, but not working correctly—for example, if `ifconfig` doesn't list anything in the `inet addr` section—start by disabling the network port. From the terminal, type the following command:

```
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:

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

A screenshot of an LXTerminal window titled "LXTerminal". The terminal shows the command `pi@raspberrypi:~$ ping -c 1 www.raspberrypi.org` and its output. The output indicates a successful ping to `www.raspberrypi.org` (IP `93.93.128.176`) with 56(84) bytes of data. It shows 64 bytes from the server with `icmp_req=1 ttl=54 time=49.3 ms`. Below this, it displays ping statistics: 1 packet transmitted, 1 received, 0% packet loss, time 0ms, and rtt min/avg/max/mdev = 49.344/49.344/49.344/0.000 ms. The prompt `pi@raspberrypi:~$` is visible at the bottom.

```
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 ms

--- 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.txt` file (located in the `/boot` directory) by adding the

This was the first time it should load the recommended `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".

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 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`

```
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:

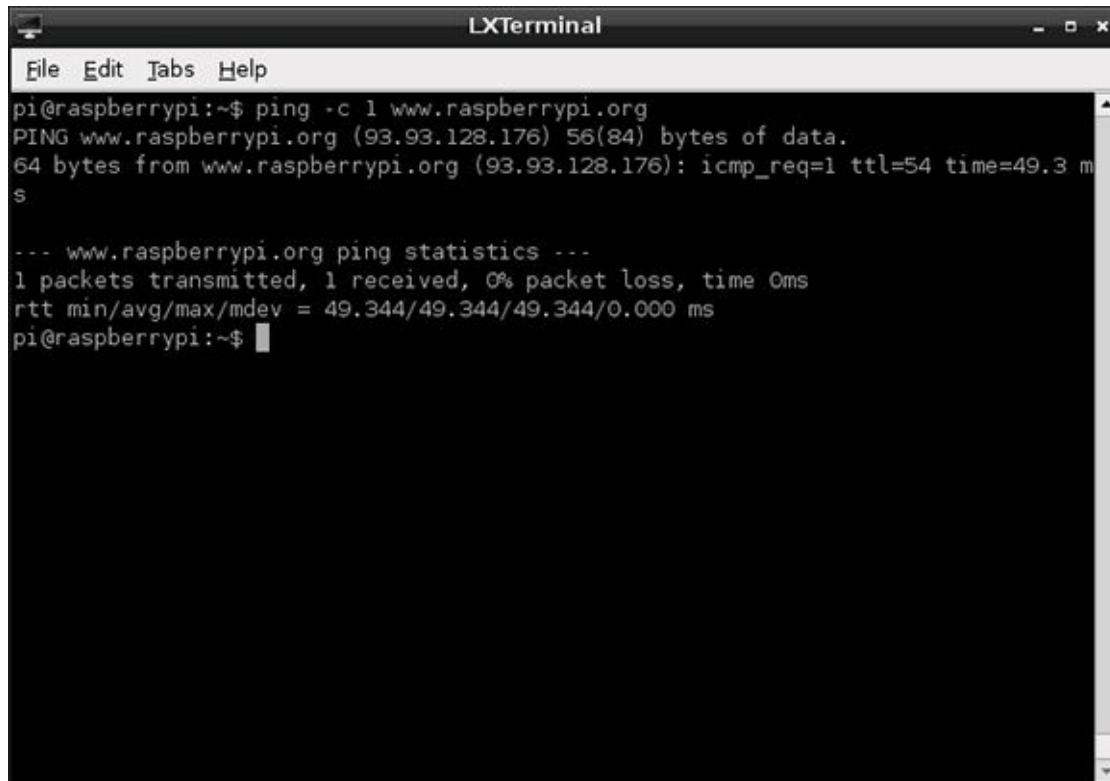
```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

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:


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

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

A screenshot of an LXTerminal window titled "LXTerminal". The terminal shows the command `pi@raspberrypi:~$ ping -c 1 www.raspberrypi.org` and its output. The output indicates a successful ping to `www.raspberrypi.org` (93.93.128.176) with 56(84) bytes of data. It shows 64 bytes from the server with `icmp_req=1 ttl=54 time=49.3 ms`. Below this, it displays ping statistics: 1 packet transmitted, 1 received, 0% packet loss, time 0ms, and rtt min/avg/max/mdev = 49.344/49.344/49.344/0.000 ms. The prompt `pi@raspberrypi:~$` is shown at the bottom.

```
File Edit Tabs Help
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 ms

--- 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 installed 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,

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
```

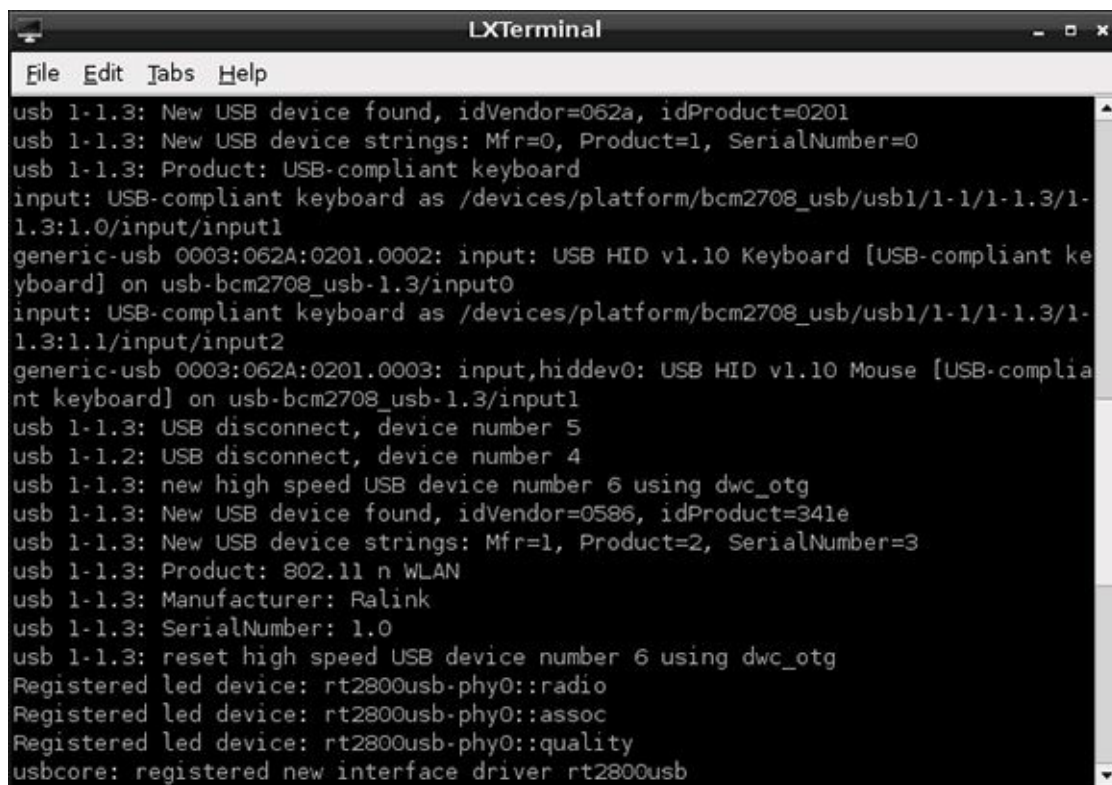
This will print out the entire kernel ring buffer, which will contain all messages output by the kernel since the Pi was switched on. If the Pi has been running a while, that can be a lot of text. To locate error message particular to the wireless adapter, it can help to send the output of `dmesg` through 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



```
LXTerminal
File Edit Tabs Help
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 keyboard] 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-compliant 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: 802.11n 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 the chapter. For the example Zyxel NWD2015 adapter, the `apt-cache` command would be