Finally, you can find out specific information about your RPi using the `hostnamectl` application, which can be used to query and change some system settings (e.g., the chassis description and hostname):

```
pi@erpi ~ $ sudo hostnamectl set-chassis server
pi@erpi ~ $ hostnamectl
   Static hostname: erpi
         Icon name: computer-server
           Chassis: server
        Machine ID: 3882d14b5e8d408bb132425829ac6413
           Boot ID: ea403b96c8984e37820b7d1b0b3fbd6d
  Operating System: Raspbian GNU/Linux 8 (jessie)
            Kernel: Linux 4.1.18-v7+
      Architecture: arm
```

Basic File System Commands

This section describes the basic commands that enable you to move around on, and manipulate, a Linux file system. When using Raspbian/Debian and Ubuntu user accounts, you often must prefix certain commands with the word `sudo`. That is because `sudo` is a program that allows users to run programs with the security privileges of the superuser. (User accounts are described in Chapter 3.) Table 2-4 lists the basic file system commands.

Table 2-4: Basic File System Commands

NAME	COMMAND	OPTIONS AND FURTHER INFORMATION	EXAMPLE(S)
List files	`ls`	`-a` shows all (including hidden files). `-l` displays long format. `-R` gives a recursive listing. `-r` gives a reverse listing. `-t` sorts last modified. `-S` sorts by file size. `-h` gives human readable file sizes.	`ls -alh`
Current directory	`pwd`	Print the working directory. `-P` prints the physical location.	`pwd -P`
Change directory	`cd`	Change directory. `cd` then Enter or `cd ~/` takes you to the home directory. `cd /` takes you to the file system root. `cd ..` takes you up a level.	`cd /home/pi` `cd /`

NAME	COMMAND	OPTIONS AND FURTHER INFORMATION	EXAMPLE(S)
Make a directory	`mkdir`	Make a directory.	`mkdir test`
Delete a file or directory	`rm`	Delete a file. `-r` recursive delete (use for directories; be careful) . `-d` remove empty directories.	`rm bad.txt` `rm -r test`
Copy a file or directory	`cp`	`-r` recursive copy. `-u` copy only if the source is newer than the destination or the destination is missing. `-v` verbose copy (i.e., show output).	`cp a.txt b.txt` `cp -r test testa`
Move a file or directory	`mv`	`-i` prompts before overwrite. No `-r` for directory. Moving to the same directory performs a renaming.	`mv a.txt c.txt` `mv test testb`
Create an empty file	`touch`	Create an empty file or update the modification date of an existing file.	`touch d.txt`
View content of a file	`more`	View the contents of a file. Use the Space key for the next page.	`more d.txt`
Get the calendar	`cal`	Display a text-based calendar.	`cal 04 2016`

That covers the basics but there is so much more! The next chapter describes file ownership, permissions, searching, I/O redirection, and other topics. The aim of this section is to get you up and running. Table 2-5 describes a few shortcuts that make life easier when working with most Linux shells.

Table 2-5: Some Time-Saving Terminal Keyboard Shortcuts

SHORTCUT	DESCRIPTION
Up arrow (repeat)	Gives you the last command you typed, and then the previous commands on repeated presses.
Tab key	Autocompletes the file name, the directory name, or even the executable command name. For example, to change to the Linux `/tmp` directory, you can type **`cd /t`** and then press Tab, which autocompletes the command to `cd /tmp/`. If there are many options, press the Tab key again to see all the options as a list.

Continues

Table 2-5 (*continued*)

SHORTCUT	DESCRIPTION
Ctrl+A	Brings you back to the start of the line you are typing.
Ctrl+E	Brings you to the end of the line you are typing.
Ctrl+U	Clears to the start of the line. Ctrl+E and then Ctrl+U clears the line.
Ctrl+L	Clears the screen.
Ctrl+C	Kills whatever process is currently running.
Ctrl+Z	Puts the current process into the background. Typing **bg** then leaves it running in the background, and **fg** then brings it back to the foreground.

Here is an example that uses several of the commands in Table 2-4 to create a directory called `test` in which an empty text file `hello.txt` is created. The entire `test` directory is then copied to the `/tmp/test2` directory, which is off the `/tmp` directory:

```
pi@erpi ~ $ cd /tmp
pi@erpi /tmp $ pwd
/tmp
pi@erpi /tmp $ mkdir test
pi@erpi /tmp $ cd test
pi@erpi /tmp/test $ touch hello.txt
pi@erpi /tmp/test $ ls -l hello.txt
-rw-r--r-- 1 pi pi 0 Dec 17 04:34 hello.txt
pi@erpi /tmp/test $ cd ..
pi@erpi /tmp $ cp -r test /tmp/test2
pi@erpi /tmp $ cd /tmp/test2
pi@erpi /tmp/test2 $ ls -l
total 0
-rw-r--r-- 1 pi pi 0 Dec 17 04:35 hello.txt
```

WARNING **Linux assumes that you know what you are doing! It will gladly allow you to do a recursive deletion of your root directory when you are logged in as root (I won't list the command).** ***Think before you type when logged in as root!***

NOTE **Sometimes it is possible to recover files that are lost through accidental deletion if you use the `extundelete` command immediately after the deletion. Read the command manual page carefully, and then use steps such as the following:**

```
pi@erpi ~ $ sudo apt install extundelete
pi@erpi ~ $ mkdir ~/undelete
pi@erpi ~ $ cd ~/undelete/
pi@erpi ~/undelete $ sudo extundelete --restore-all --restore-directory
 . /dev/mmcblk0p2
pi@erpi ~/undelete $ ls -l
drwxr-xr-x 6 root root 4096 Dec 17 04:39 RECOVERED_FILES
```

```
pi@erpi ~/undelete $ du -sh RECOVERED_FILES/
100M    RECOVERED_FILES/
```

In this example, 100 MB of files were recovered—typically temporary files that were deleted as a result of package installations.

Environment Variables

Environment variables are named values that describe the configuration of your Linux environment, such as the location of the executable files or your default editor. To get an idea of the environment variables that are set on the RPi, issue an `env` call, which provides a list of the environment variables on your account. Here, `env` is called on the Raspbian image:

```
pi@erpi ~ $ env
TERM=xterm
SHELL=/bin/bash
SSH_CLIENT=fe80::50b4:eb95:2d00:ac3f%eth0 2599 22
USER=pi
MAIL=/var/mail/pi
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:...
PWD=/home/pi
HOME=/home/pi  ...
```

You can view and modify environment variables according to the following example, which adds the `/home/pi` directory to the `PATH` environment variable:

```
pi@erpi ~ $ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
pi@erpi ~ $ export PATH=$PATH:/home/pi
pi@erpi ~ $ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/pi
```

This change will be lost on reboot. Permanently setting environment variables requires modifications to your `.profile` file when using sh, ksh, or bash shells; and to your `.login` file when using csh or tcsh shells. To do this, you need to be able to perform file editing in a Linux terminal window.

Basic File Editing

A variety of editors are available, but one of the easiest to use is also one of the most powerful: the *GNU nano editor*. You start the editor by typing **`nano`** followed by the name of an existing or new filename; for example, typing **`nano hello.txt`** displays the view captured in Figure 2-6 (after the text has been entered). Typing **`nano -c hello.txt`** also displays the current line number, which is useful for debugging. You can move freely around the file in the window by using the arrow keys and editing or writing text at the

cursor location. You can see some of the nano shortcut keys listed on the bottom bar of the editor window, but there are many more, some of which are presented in Table 2-6.

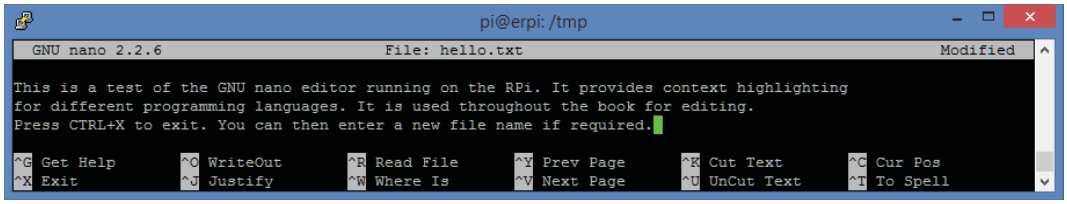

Figure 2-6: The GNU nano editor being used to edit an example file in a PuTTY Linux terminal window

Table 2-6: Nano Shortcut Keys: A Quick Reference

KEYS	COMMAND	KEYS	COMMAND
Ctrl+G	Help	Ctrl+Y	Previous page
Ctrl+C	Find out the current line number	Ctrl+_ or Ctrl+/	Go to line number
Ctrl+X	Exit (prompts save)	Alt+/	Go to end of file
Ctrl+L	Enable long line wrapping	Ctrl+6	Start marking text (then move with arrows to highlight)
Ctrl+O	Save	Ctrl+K or Alt+6	Cut marked text
Arrows	Move around	Ctrl+U	Paste text
Ctrl+A	Go to start of line	Ctrl+R	Insert content of another file (prompts for location of file)
Ctrl+E	Go to end of line	Ctrl+W	Search for a string
Ctrl+Space	Next word	Alt+W	Find next
Alt+Space	Previous word	Ctrl+D	Delete character under cursor
Ctrl+V	Next page	Ctrl+K	Delete entire line

NOTE **Ctrl+K appears to delete the entire line but it actually removes the line to a buffer, which can be pasted using Ctrl+U. This is a quick way of repeating multiple lines. Also, Mac users may have to set the meta key in the Terminal application to get the Alt functionality. Select Terminal ➪ Preferences ➪ Settings ➪ Keyboard, and then choose Use option as meta key.**

What Time Is It?

A simple question like "What time is it?" causes more difficulty than you can imagine. For example, typing **`date`** at the shell prompt might produce the following:

```
pi@erpi ~ $ date
Thu 17 Dec 16:26:59 UTC 2015
```

This result happens to be the correct time in this instance because the board is connected to a network. If it is wrong, why did the RPi manufacturer not set the clock time on your board? The answer is that they could not. Unlike a desktop computer, the RPi has no battery backup to ensure that the BIOS settings are retained; in fact, there is no BIOS! That topic is examined in detail in the next chapter, but for the moment, you need a way to set the time, and for that you can use the *Network Time Protocol* (NTP). The NTP is a networking protocol for synchronizing clocks between computers. If your RPi has the correct time, that is only because your RPi is obtaining it from your network using an NTP service that is running the board:

```
pi@erpi ~ $ systemctl status ntp
• ntp.service - LSB: Start NTP daemon
   Loaded: loaded (/etc/init.d/ntp)
   Active: active (running) since Sat 2015-12-19 07:18:04 GMT; 22h ago
  Process: 499 ExecStart=/etc/init.d/ntp start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/ntp.service
           └─544 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 107:112
```

The NTP service is configured using the file `/etc/ntp.conf`, and the lines beginning with the word `server` (hence the ^ in the call to `grep`) identify the servers to which your RPi is communicating to retrieve the current time:

```
pi@erpi ~ $ more /etc/ntp.conf | grep ^server
server 0.debian.pool.ntp.org iburst
server 1.debian.pool.ntp.org iburst
server 2.debian.pool.ntp.org iburst
server 3.debian.pool.ntp.org iburst
```

To be a good NTP citizen, you should adjust these entries to refer to the closest NTP server pool by going to `www.pool.ntp.org` (the closest server to me is `ie.pool.ntp.org` for Ireland) and updating the entries accordingly. If you want to test the settings first, you can install and execute the `ntpdate` command:

```
pi@erpi ~ $ sudo apt install ntpdate
pi@erpi ~ $ sudo ntpdate -b -s -u ie.pool.ntp.org
pi@erpi ~ $ date
Sun 20 Dec 16:02:39 GMT 2015
```

After setting the time, you can set your time zone. Use the following command, which provides a text-based user interface that allows you to choose your location. The RPi is set for Irish standard time (IST) in this example:

```
pi@erpi ~ $ sudo dpkg-reconfigure tzdata
Current default time zone: 'Europe/Dublin'
Local time is now:      Sun Dec 20 16:37:48 GMT 2015.
Universal Time is now:  Sun Dec 20 16:37:48 UTC 2015.
```

NOTE **If your RPi is not connected to the Internet, you can manually set the date using the `timedatectl` tool:**

```
pi@erpi ~ $ sudo timedatectl set-time '2017-1-2 12:13:14'
pi@erpi ~ $ date
Mon  2 Jan 12:13:16 GMT 2017
```

Unfortunately, this date and time will be lost when the RPi restarts. Chapter 8 describes how a battery-backed real-time clock (RTC) can be connected to the RPi to solve that problem.

Package Management

At the beginning of this chapter, a good package manager was listed as a key feature of a suitable Linux distribution. A *package manager* is a set of software tools that automate the process of installing, configuring, upgrading, and removing software packages from the Linux operating system. Different Linux distributions use different package managers: Ubuntu and Raspbian/Debian use *APT* (Advanced Packaging Tool) over *DPKG* (Debian Package Management System), and Arch Linux uses *Pacman*. Each has its own usage syntax, but their operation is largely similar. Table 2-7 lists some common package management commands.

Table 2-7: Common Package Management Commands (Using nano as an Example Package)

COMMAND	RASPBIAN/DEBIAN/UBUNTU
Install a package.	`sudo apt install nano`
Update the package index.	`sudo apt update`
Upgrade the packages on your system.	`sudo apt upgrade`
Is nano installed?	`dpkg-query -l | grep nano`
Is a package containing the string `nano` available?	`apt-cache search nano`
Get more information about a package.	`apt-cache show nano` `apt-cache policy nano`
Get help.	`apt help`

COMMAND	RASPBIAN/DEBIAN/UBUNTU
Download a package to the current directory.	`apt-get download nano`
Remove a package.	`sudo apt remove nano`
Clean up old packages.	`sudo apt-get autoremove` `sudo apt-get clean`

NOTE **Over time, the `apt` binary command is slowly integrating the features of the `apt-get` and `apt-cache` commands. This change should reduce the number of tools required to manage packages. However, older Linux distributions may require that you use the `apt-get` command in place of the `apt` command.**

Wavemon is a useful tool that you can use in configuring Wi-Fi connections (see Chapter 13). If you execute the following command, you will see that the package is not installed by default:

```
pi@erpi ~ $ wavemon
-bash: wavemon: command not found
```

You can use the platform-specific package manager to install the package, once you determine the package name:

```
pi@erpi ~ $ apt-cache search wavemon
wavemon - Wireless Device Monitoring Application
pi@erpi ~ $ sudo apt install wavemon
Reading package lists... Done
Building dependency tree ...
Setting up wavemon (0.7.6-2) ...
```

The `wavemon` command now executes, but unfortunately it will not do anything until you configure a wireless adapter (see Chapter 13):

```
pi@erpi ~ $ wavemon
wavemon: no supported wireless interfaces found
```

It is also worth noting that packages can be manually downloaded and installed. This method can be useful should you want to retain a specific version or need to distribute a package to multiple devices. For example, the Wavemon package can be removed, manually downloaded as a `.deb` file, and installed:

```
pi@erpi ~ $ sudo apt remove wavemon
pi@erpi ~ $ wavemon
-bash: /usr/bin/wavemon: No such file or directory
pi@erpi ~ $ apt-get download wavemon
pi@erpi ~ $ ls -l wavemon*
-rw-r--r-- 1 pi pi 48248 Mar 28  2014 wavemon_0.7.6-2_armhf.deb
pi@erpi ~ $ sudo dpkg -i wavemon_0.7.6-2_armhf.deb
pi@erpi ~ $ wavemon
wavemon: no supported wireless interfaces found
```