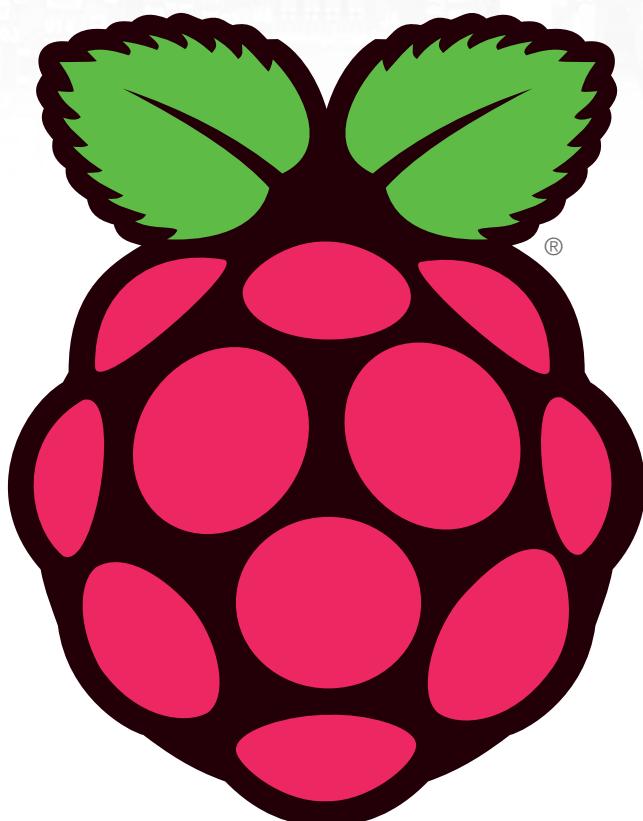


# Raspberry Pi

## User Guide



### Getting started

Operating Systems & Installation

### Learn the ropes

Linux & Packages

### Making the most of your Pi

Pi Projects

**element14**

[element14.com](http://element14.com)

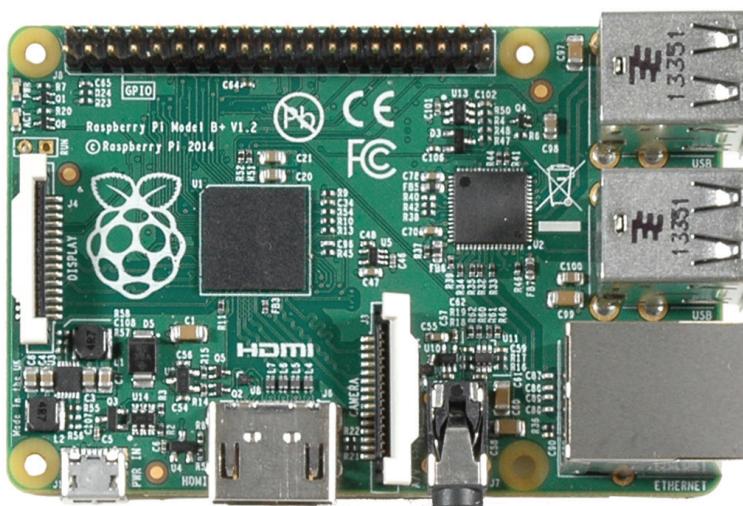
# CONTENTS

<b>Unpack the box</b>	3
<b>Operating Systems</b>	4
<b>Installation</b>	6
<b>Loading the O/S</b>	7
<b>Raspbian</b>	8
<b>Command line: Learn the ropes</b>	10
<b>Packages: How do they work?</b>	16
<b>RaspBMC</b>	20
<b>Camera controller</b>	22
<b>Sending output</b>	26
<b>Minecraft Pi</b>	30
<b>Postscript</b>	32

# Unpack the box

Inside the box you will find the Raspberry Pi board, a 5V Power Supply (optional) and a 8GB microSD card.

Depending on which bundle you might have purchased you might also receive a case or other accessories. Feel free to put those items together prior to starting up your Pi. The Pi requires 5V 1A power so if you did not purchase a power supply you will need to locate one that has a Micro USB tip on the end. Many smartphone and tablets use Micro USB charging cables, those will work just fine.



Raspberry Pi is a trademark of the Raspberry Pi Foundation. Parts of this manual are reproduced from 'Raspberry Pi Beyond the Manual' and is copyright of or licensed by Future Publishing Limited (a Future plc group company), UK 2013. All rights reserved. The rest of this manual is copyright of Premier Farnell UK Limited, 2014 and all rights are reserved. No part of this manual may be sold, licensed, transferred, copied or reproduced in whole or in part in any manner or form or in or on any media without the prior written consent of the copyright owner.

# Operating Systems

Let's take a look at a sweet selection of tasty operating systems for the Raspberry Pi.

The Raspberry Pi phenomenon appears to go from strength to strength; like a runaway train, it's ploughing ahead and forging itself a place in the record books.

It's hardly surprising – the hardware alone is developed perfectly for the goals of the Raspberry Pi Foundation, the pricing is pitched perfectly, and having the unique versatility of Linux as the operating system seals the deal nicely.

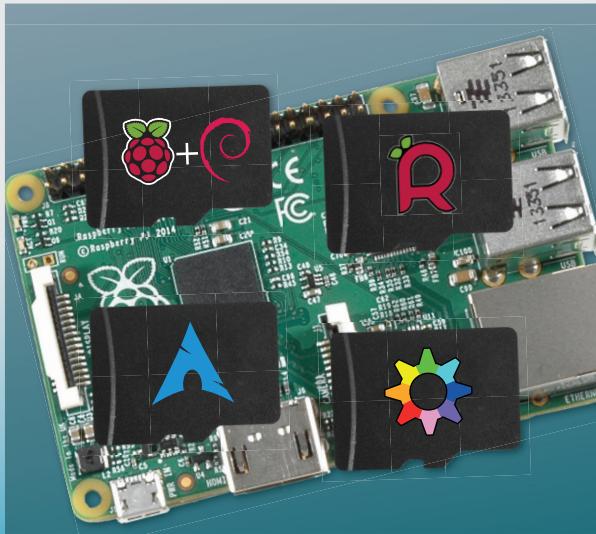
Most buyers, once they get their hands on their new RPi, make a move towards the official Raspberry Pi site and follow the getting started instructions therein; the end result is the user running Raspbian 'Wheezy', the Foundation's recommended

operating system, creating, learning and programming, and strapping the poor wee beast on to a weather balloon and sending it to the outer edge of the atmosphere. What many RPi users don't realize, though, is that there's a wealth of other operating systems available for their beloved Pi.

We thought, therefore, that those users who aren't aware of these other sweet toppings for the Raspberry Pi need to be informed, and what's more, they need to have a chocolate box selection presented to them.

## › OUR SELECTION

- Raspbian
- Risc OS
- Arch
- RaspBMC
- OpenELEC
- Pidora



## How we tested...

Therefore, to get a true all-round perspective, we took the time to install the operating systems on a fresh 8GB microSD Card card. The areas we're looking at are installation, default software, media playback (out-of-the-box), looks and usability, the community behind the OS and their respective attitudes toward software freedom. Basically, the very stuff that makes a Linux user decide on what system to use.

We also want to gauge this from the point of view of someone who's not as familiar with Linux as other people are, so that they can jump into the project without too much hassle, and not end up leaving it feeling disheartened.

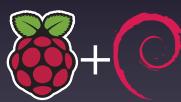
## Raspbian

This is the recommended distro by the Raspberry Pi Foundation. Unless you have good reason to use a different one, it's probably your best bet.

It's based on Debian Wheezy, so you can easily install anything from the huge Debian repositories.

The default desktop environment is LXDE, which is very lightweight, but a little basic for some tastes. Xfce is available for people who like a few more graphical niceties. It has the **raspi-config** program, which is probably the easiest way of configuring your Pi.

The Raspberry Pi was designed to get children into programming, and Raspbian was designed with this in mind. You'll find *Idle* (a Python IDE) and *Scratch* (a programming environment for young children) on the desktop. It's available from: [www.raspberrypi.org](http://www.raspberrypi.org).



## RaspBMC

The Raspberry Pi may have been designed as an educational tool, but hobbyists have been pretty quick to make it a toy. This distro is designed to turn your Pi into a media center that can be used to control your TV. It's based on XBMC, which allows you to play music and videos that you have as files, or stream them from the internet. The image can be downloaded from: [www.raspbmc.com](http://www.raspbmc.com). For details of how to install it and set it up, see the following pages. If you have a *MythTV* back-end set up, you can use XBMC Pi to provide a front-end interface.



## Arch Linux

While Raspbian has been created to try to shield users from the internal setup of the OS, Arch Linux is designed to help users understand how the system works.

The initial image, available from [www.raspberrypi.org](http://www.raspberrypi.org), includes just the basic system to get your Pi running and connected to the network. It doesn't include much of the software you may want to use, such as a graphical environment, for example. You should find the information you need at [bit.ly/9APmgA](http://bit.ly/9APmgA).

Taking it from this initial state to a working system will require a bit of work, but along the way you'll learn about how the internals of a Linux distribution fit together.

Whether or not this is worth all the work, does of course, depend on you.



## Risc OS

The difference with the Risc OS is that it is small and fast. Developed when the fastest desktop computer was an 8MHz ARM2 with 512KB of RAM. The core system including windowing system and a few apps fits inside 6MB. That means it's fast and responsive on modern hardware. The memory taken by apps is usually counted in the kilobytes.



To Risc OS a 700MHz 512MB Raspberry Pi is luxury, what to do with all that memory? Risc OS like Raspbian, takes you to a nice GUI at the startup. One feature of the Risc OS is that it boots to the desktop in 1080P by default. The Risc desktop is a little retro but is functional in its default mode. The Risc OS take a little getting used to, one caveat is that the Ethernet port is disabled at launch so it requires some configuration before using.

# Installation

Do you need a PhD to install the OS?

The NOOBS operating system installation is as easy as copying files onto your SD card.

The installation of an operating system image is fairly well documented, as per the area on the Raspberry Pi site titled *Guide for beginners* which can be found here: [goo.gl/53xgp](http://goo.gl/53xgp), along with the simple installation routine of using either **dd** on Linux, or with the new NOOBS image you simply unzip the image download and copy the files onto a blank formatted microSD card in Windows. The process is relatively painless, it's what happens after you insert the microSD card into the Raspberry Pi and apply some power that the fun starts.

The six operating systems on the NOOBS card – Raspbian, Risc OS, Arch Linux, OpenELEC, RaspBMC and Pidora – each have their own nuances, and methods by which to install and provide the user with a base working graphical desktop. While having a GUI isn't absolutely necessary, it does cover the large percentage of users who are new to Linux. That being the case, the definition of 'installation' must include getting to the point whereby the new user can recognize the operating system as they would a standard Linux desktop – in other words, be presented with a graphical user interface. In a world where easing the user into the bath water of Linux is paramount, **Raspbian** has the most user friendly desktop, but the other offerings have just as good a start for the user.

**Risc OS**, for example; once transferred to the SD card and booted, we are rapidly launched into a colorful and friendly GUI, with relatively detailed messages informing us of any issues during the initial boot and setup. From here, we can

simply click on the 'Configure' icon and begin to alter any settings we see fit.

**Arch Linux** for the RPi is a different beast, booting the user into a terminal environment and leaving them to download, install and configure their OS. Arch, once fully appreciated, is one of the best operating systems available, but it takes some tweaking to get to a standard desktop.

**OpenELEC** is the simplest version of XBMC, starting directly into the XBMC GUI without any additional configuration required.

**RaspBMC** makes it easy for you, once you select this operating system you are taken directly to the GUI, no scary terminal prompts to deal with. Once you have completed the configuration screens and RaspBMC has checked for updates, you will be ready to go.

**Pidora** which is a remix of the Fedora Linux distribution is fairly easy to configure. Once you select this O/S, you will be presented with several configuration screens and then the desktop will load. One thing to watch out for here is the terminal commands are a little different than the other operating systems (distros in Linux language). The foundation that compiled Pidora has a good tutorial here [bit.ly/1piC2LA](http://bit.ly/1piC2LA).

In summary, Raspbian would be the easiest to use and has the most support from the Raspberry Pi Foundation. OpenELEC offers the most media support if that is the route you would like to go. The other operating systems all have their advantages and disadvantages. It is up to you to decide which way you would like to go.

# Loading the O/S

The memory card is affectionately referred to as NOOBS.

NOOBS is a way to make setting up a Raspberry Pi for the first time much, much easier. You won't need network access, and you won't need to download any special imaging software.

Insert the microSD card in the memory slot underneath the board. Plug your power cord into the port next to the HDMI video port. When you boot up for the first time, you'll see a menu prompting you to install one of several operating systems into the free space on the card. The choice means you can boot the Pi with a regular operating system like Raspbian, or with a media-center specific OS like RaspBMC.

Once you've selected the O/S you'd like (see the O/S section for more information), make sure you check the dialogue box at the bottom of the page for the proper language that you prefer. The installer will build the O/S that you selected (a good time for a sandwich or a cup of coffee).

Once you've installed an operating system, your Pi will boot as normal. However, NOOBS stays resident on your card, by holding **Shift** down during boot you can return to the recovery interface. This allows you to switch to a different operating system, or overwrite a corrupted card with a fresh install of the current one; it also

provides a handy tool to let you edit the config.txt configuration file for the currently installed operating system, and even a web browser so you can visit the forums or Google for pointers if you get stuck.

Once the installation is complete your Pi will restart and a bunch of strange looking text will appear on your screen. If you are new to Linux this might take you back to the "DOS" days. Not to fear this is called the "command line" You will see at the bottom of the page the last line reads – raspberrypi login:

Next you will need this information.

The default username and password are:

**Username: pi**

**Password: raspberry**

When you hit the enter key you will see the following line

**Pi@raspberrypi ~ \$**

Type the following to launch the desktop:

**startx**

Congratulations! You now have started up your Pi! Now go explore.



■ The microSD card slot is located on the underside of the Raspberry Pi board.

# Raspbian

For the majority of people who use it, Raspbian will be the graphical face of the Raspberry Pi.

It can be obtained and installed on to an SD card by following the instructions on the previous page.

Once it's up and running, it's a good idea to grab the latest versions of all the software by connecting your Pi to the internet, opening a terminal and running:

```
sudo apt-get update  
sudo apt-get upgrade
```

The killer feature of Raspbian is the *raspi-config* program. This will start automatically the first time you boot, or can be run at any time by typing **sudo raspi-config** in a terminal. It has got quite a few options, but the most important are:

- **expand\_rootfs** Because of the way Raspbian is installed, it will only create a 2GB filesystem, so if you have a larger card, any remaining space will remain unused. You can use this option to expand the filesystem to take advantage of any wasted space.
- **memory\_split** The Raspberry Pi uses the same chunk of memory for both the main processor and the graphics chip. Using this option, you can change the amount allocated to each.

■ **overscan** This option can be used on some displays to make the graphics expand to fill the whole screen. You can safely ignore it unless you have problems.

■ **overclock** Get an extra 50 per cent performance at no extra cost! See 'Overclocking' for more details.

■ **boot\_behaviour** This rather cryptically named option changes whether your Pi boots into a graphical environment or a text one.

The installed software has been kept to a minimum. This is a good idea, but you may find that tools you use on other desktop distros aren't there. Fortunately, as Raspbian is linked to the Debian Armhf repositories, you have access to more software than you're ever likely to need. If you like using a mouse, you may want to install a graphical package manager. We recommend *Synaptic*. To install it, type:

```
sudo apt-get install synaptic
```

in a terminal. It can then be opened by going to the 'LXDE menu > Preferences > Synaptic Package Manager'. You can then install any software you want.



■ Like all good distros, Raspbian comes with a selection of addictive time-wasters. This is Squirrels from the Pygmes selection.

## 12 COOL RASPBERRY PI PROJECTS

- The complete channel one temperature monitor and alarm project
- The Raspberry Pi softball camera
- The Raspberry Pi karaoke machine
- The Drinkmotizer: a Raspberry Pi drink mixing robot
- New Year's Eve countdown timer with fireworks launching ability
- Raspberry Pi remote fish tank controls: AKA Project Goldie
- The scary door
- RaspiWatt: discover power consumption using a Kill-A-Watt & Pi
- Build your own Gertboard experimenter kit
- Raspberry Pi enabled Christmas lights
- Ultimate Raspberry Pi bundle security system
- Pumpkin Pi project for Halloween and a second helping of pumpkin Pi

[www.element14.com/raspberrypiprojects](http://www.element14.com/raspberrypiprojects)

## Overclocking

The processor at the heart of the Raspberry Pi is designed to run at 700MHz. That is, perform 700,000,000 operations per second. Of course, 'designed to run' doesn't mean 'has to run'. You can increase this speed. However, doing this will increase the power consumption, which in turn increases the amount of heat generated. If it gets too hot, you're liable to have a smoking pile of silicon rather than a functional processor.

Fortunately, Raspbian includes a tool to help you ramp up the speed while also keeping a careful eye on the temperature. And because this is an official tool, using it won't void your warranty (unlike earlier unofficial methods). Overclocking your Raspberry Pi is simply a matter of running **sudo raspi-config** and then selecting 'Overclocking'.

There are a few options to choose from, depending on how brave you're feeling.

If you find that your Pi becomes unstable, reboot with the Shift key held down to disable overclocking, then change the option in *raspi-config*. The maximum setting should give you a whopping 50 per cent extra speed, which we found makes a real difference to the desktop user experience, especially for web browsing.

If you want to keep an eye on your core temperature, you can add the Temperature widget to the LXDE panel. However, your Pi will automatically turn off overclocking once it reaches 85°C.

- Overclocking will increase the amount of power that your Pi draws, and so may become less stable if running a number of USB devices.

```
Raspi-config

  info           Information about this tool
  expand_rootfs  Expand root partition to fit
  overscan       Change overscan
  configure_keyboard Set keyboard layout
  change_pass    Change password for 'pi'
  change_locale  Set locale
  change_timezone Set timezone
  memory_split   Change memory split
  overclock      Configure overclocking
  ssh            Enable or disable ssh service
  boot_behaviour Start desktop on boot?
  update         Try to upgrade raspi-config

  <Select>           <Finish>
```

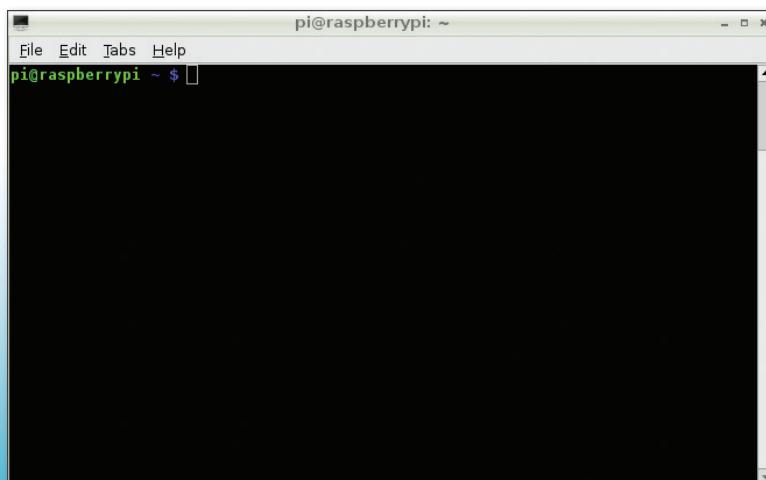
# Command line: Learn the ropes

Get to grips with your Raspberry Pi's command line interface and unleash its full power without using the mouse.

As you have no doubt discovered, Raspbian has a graphical interface similar to that of Windows or Mac OS X.

You can do most of your day-to-day tasks in this interface. There's a file manager, web browser, text editor and many other useful applications. However, sometimes you need an interface that's a bit more powerful, and this is where the command line interface (CLI) comes in. It's also known as the terminal or shell.

This is an entirely text-based interface, where you type in commands and get a response. We won't lie to you: it will seem confusing at first. Don't worry, though – once you've had a bit of practice, it will start to make sense, and spending a little time learning it now will pay dividends in the future.



■ LXTerminal running on the Raspbian desktop.

The first thing you need to do is open up a terminal.  
Click on 'LXTerminal' on the Raspbian desktop.

You should now see a line that looks like:

```
pi@raspberrypi ~ $
```

This is the command prompt. Whenever you see this, you know the system is ready to receive input. Now type **pwd**, and press **Enter**. You should see:

```
/home/pi
```

If you've changed your username, then you'll see a different line. The rather cryptically named **pwd** command stands for Print Working Directory, and the system simply outputs the directory you're currently in. When you start a terminal, it will go to your home directory.

Now we know where we are, the next logical thing to do is move about through the directories. This is done using the **cd** (change directory) command. Try entering:

```
cd ..
```

```
pwd
```

You should find that the system returns **/home**. This is because we've **cded** to

'..' and two dots always points to the parent directory. To move back to your home directory, you can enter **cd pi**. There is also another way you can do it. The ~ (pronounced tilda) character always points to your home directory, so wherever you are in the filesystem, you can enter **cd ~** and you'll move home.

Now type **ls** and hit **Enter**. This will list all the files in the current directory. One of the big advantages of commands is that we can tell them exactly how we want them to behave. This is done using flags, which come after the command and start with a '-'. For example, if we want to list all the files in the current directory (including hidden ones, which start with a '.' on Unix-based systems), we use the flag **-a**. So, in your terminal, type **ls -a**.

This time, you should see more files appear. Another flag for **ls** is **-l**. This gives us more information about each file. Try it out now by typing **ls -l**. You can even combine flags, such as in **ls -al**.

## Interactive programs

Most of the commands we're dealing with here are non-interactive. That means you set them running and then wait for them to finish. However, not all command line programs work like this. For example, when you first booted Raspbian, it started a config tool that ran in the terminal. There are a few other programs that work in a similar way. Traditionally, the most common has been text editors that allow you to work on files if you don't have a graphical connection.

There are a few quite complicated ones that are great if you spend a lot of time working from the command line, but they can be hard to learn. There's also an easy-to-use terminal-based text editor called **nano**. Enter **nano** followed by a filename at the command prompt to start it. You can then navigate around the text file and make any changes you need. Press **Ctrl+X** to save your work and exit back to the prompt.

# Knowing what commands to use

At this point, you're probably wondering how on earth you are supposed to know what commands and what flags you can use for a task.

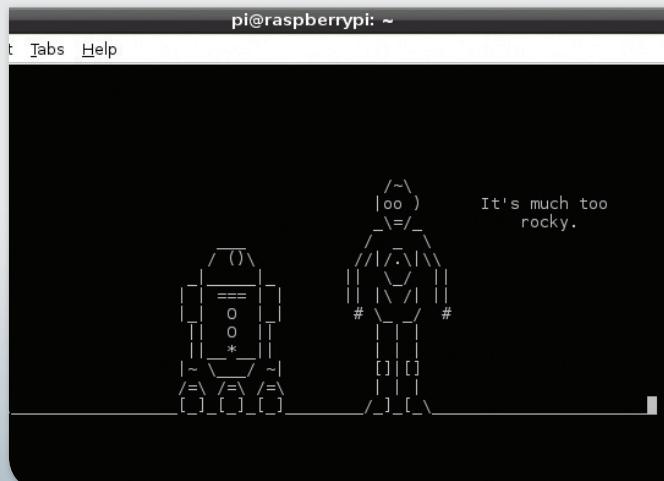
Well, there's good news and bad news. The good news is that it's usually not too difficult to find out the flags for a command. Most commands support the **-h** or **--help** flags, which should give some information about what flags a command can take and how to use it. For example, if you run **ls --help**, you'll see a long list of flags and what they all do, including:

```
-a, --all      do not ignore
entries starting with .
...
-l           use a long listing format
```

The second way of finding information on a command is using **man**. This is short for manual. It takes a single argument, that is, a word after the command that isn't preceded by a hyphen. It then displays information on

the command given as an argument. To see the man page for **ls**, type **man ls**. You can navigate through the page using the up and down arrows, or the page up and page down keys to scroll faster. To search for a word or phrase inside the man page, type **/**, then the phrase. For example, **/-l** will find all occurrences of **-l**. You can use the **N** key and **Shift+N** to scroll forwards and backwards through the occurrences of **-l**.

As we introduce more commands, it's good to take a look at the help and the man page to familiarize yourself with what they do. Of course, you can always Google a command if you find the text-based help a little off-putting, but staying in the terminal will help you become more familiar with the command line interface.



- You can even watch movies in the command line. To stream the classic, just enter `telnet towel.blinkenlights.nl` and put some popcorn on.

## Tab completion

When you're dealing with long filenames, it can be very annoying to have to type them out every time you want to run a command on them. To make life a bit easier, the terminal uses tab completion. This means that if you start typing a filename and press the Tab key, the system will try to fill in the rest of the name. If there's only one file that fits what you've typed so far, it will fill in the rest of the name for you (try typing `cd /h` then pressing Tab). If there are more than one, it will fill in as far as the two are the same. If you press Tab again, it will show the options (try typing `cd /m`, and then pressing Tab twice).

# How will I know?

Remember we said there's good news and bad news? Well, the bad news is that it can be quite tricky to find commands if you don't know what they're called.

One helpful feature is the **man** keyword search. This is done with the **-k** flag. To search for all the programs related to 'browser' on your system, run **man -k browser**. You'll notice that this lists graphical programs as well as command line ones. This is because there's no real difference between the two. You can launch windows from the terminal, and sometimes even control them.

If you've got *Iceweasel* (a rebranded version of *Firefox*) installed on your Pi (it's not on there by default), you can open [TuxRadar.com](http://TuxRadar.com) in a new tab in a currently running *Iceweasel* window with the command **iceweasel --new-tab www.tuxradar.com**.

We're now going to quickly introduce a few useful commands. **rm** deletes (ReMoves) a file. **mkdir** makes a new directory. **cp** copies a file from one place to another. This one takes two arguments, the original file and the new file. **cat** outputs the contents of one or more text files. This takes as many arguments as you want, each one a text file, and simply spits their contents out on to the terminal. **less** is a more friendly way of viewing text files. It lets you scroll up and down using the arrow keys. To exit the program back to the command line, press **Q**. We'll use all these commands in examples below, so we won't dwell on them for too long.

```
File Edit Tabs Help
LS(1)                               User Commands                               LS(1)
NAME
ls - list directory contents

SYNOPSIS
ls [OPTION]... [FILE]...

DESCRIPTION
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all
      do not ignore entries starting with .

-A, --almost-all
      do not list implied . and ..

--author
Manual page ls(1) line 1 (press h for help or q to quit)
```

- Probably the most important command in any Unix-like system is **man**, since it is the key to understanding every other command. Take time to become familiar with the structure and language used and it will make life easier in the future.

## find

**find** is a useful command for finding files on your computer. You use it in the format **find location flags**. For example, to find every file on your computer that's changed in the last day, run

**find / -mtime 1**

There are more details of what this means, and the different flags that can be used over the page.

## Power up

So far you're probably thinking to yourself, "I could have done all this in the graphical interface without having to remember obscure commands and flags."

You'd be right, but that's because we haven't introduced the more powerful features yet. The first of them are wildcards. These are characters that you can put in that match different characters. This sounds a bit confusing, so we're going to introduce it with some examples.

First, we'll create a new directory, move into it and create a few empty files (we use the command **touch**, which creates an empty file with the name of each argument). Hint – you can use tab completion to avoid having to retype long names, such as in the second line.

```
mkdir wildcards  
cd wildcards  
touch one two three four  
touch one.txt two.txt three.txt four.txt
```

Then run **ls** to see which files are in the new directory. You should see eight.

The first wildcard we'll use is **\***. This matches any string of zero or more characters. In its most basic usage, it'll match every file in the directory. Try running:

```
ls *
```

This isn't particularly useful, but we can put it in the middle of other characters. What do you think **\*.txt** will match? Have a think, then run:

```
ls *.txt
```

to see if you are right.

How about **one\***? Again, run

```
ls one*
```

to see if you were correct. The wildcards can be used with any command line programs. They're particularly useful for sorting files. To copy all the .txt files into a new directory, you could run:

```
mkdir text-files  
cp *.txt text-files
```

We can then check they made it there correctly with:

```
ls text-files/
```

The second wildcard we'll look at is **?**. This matches any single character. What do you think:

```
ls ???
```

will match? Have a guess, then run it to see if you're right.

We can also create our own wildcards to match just the characters we want. **[abc]** will match just a lower-case A, B and C. What do you think **ls [ot]\*** will match? Now try

```
ls [!ot]*
```

What difference did the exclamation mark make? It should have returned everything that didn't start with a lower-case letter O or T.

## sudo

When using the Raspberry Pi for normal use, you can work with files in your home directory (for example, **/home/pi**). You will also be able to view most other files on the system, but you won't be able to change them. You also won't be able to install software. This is because Linux has a permissions system that prevents ordinary users from changing system-wide settings. This is great for preventing you from accidentally breaking

your settings. However, there are times when you need to do this. You can use **sudo** to run a command as the super user (sometimes called root), which can do pretty much anything on the system. To use it, prefix the command with **sudo**. For example:

```
sudo apt-get install synaptic
```

will install the package **synaptic** and make it available to all users.

## Pipes and redirection

The commands that we've been looking at so far have all sent their output to be displayed in the terminal.

Most of the time this is what we want, but sometimes it's useful to do other things with it. In Linux, you can do two other things with a command: send it to a file, or send it to another program. To send it to a file, use the **>** character followed by the filename. Run:

```
ls > files  
cat files
```

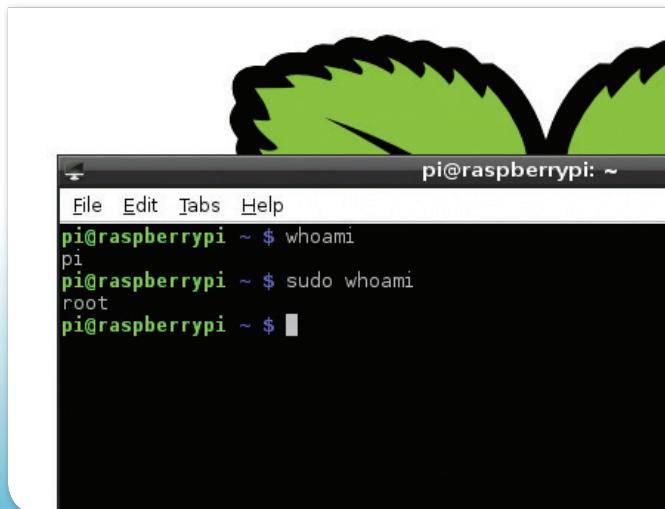
and you should see that it creates a new file called **files**, which contains the output of **ls**.

The second option, sending it to another program, is another of the really powerful features of the Linux command line, because it allows you to chain a series of commands together to make one super

command. There are a lot of commands that work with text that are designed to be used in this way. They're beyond the scope of this tutorial, but as you continue to use the command line, you'll come across them and start to see how they can be linked together. We'll take a look at a simple example. If you run **find /** (don't do it just yet!) it will list every file on the system.

This will produce a reel of filenames that will quickly go off the screen. However, rather than direct it to the screen, we can send (or 'pipe') it to another command that makes the output easier to read. We can use the **less** command that we looked at on page 13 for this. Run:

```
find / | less
```



- Use the **sudo** command to switch between the normal user 'pi', and the superuser 'root'.

### Take it further

We've only been able to touch on the basics of using the command line, but you should have enough knowledge now to get started, and hopefully you're beginning to see just how powerful the command line interface is once you get to know it.

If you want to know more (and you should!) there are loads of resources in print and online. [LinuxCommand.org](http://LinuxCommand.org) is a great place to start. Its book (*The Linux Command Line*) is available from bookshops, or for free online [www.linuxcommand.org/lc3\\_learning\\_the\\_shell.php](http://www.linuxcommand.org/lc3_learning_the_shell.php).

# Packages: How do they work?

Discover how Raspbian's package manager, **apt-get**, gets software from online repositories and manages it on your system.

If you're used to Windows, you may be used to each bit of software having its own installer, which gets the appropriate files and puts them in the appropriate places.

Linux doesn't work in the same way (at least, it doesn't usually). Instead, there is a part of the operating system called the package manager. This is responsible for getting and managing any software you need. It links to a repository of software so it can download all the files for you. Since Linux is built on open source software, almost everything you will need is in the repositories and free. You don't need to worry about finding the install files, or anything like that – the package manager does it all for you.

There are a few different package managers available for Linux, but the one used by Raspbian is **apt-get**. Arch uses a different one, so if you want to try this distribution on your Raspberry Pi, you'll need to familiarize yourself with the **pacman** software, which we won't cover here.

Before we get started, we should mention that since this grabs software from the online repositories, you will need to connect your Pi to the internet before following this section.



**apt-get** is a command line program, so to start with you'll need to open a terminal (see the command line interface section on page 10 for more information on this). Since package management affects the whole system, all the commands need to be run with **sudo**. The first thing you need to do is make sure you have the latest list of software available to you. Run:

```
sudo apt-get update
```

Since all the software is handled through the package manager, it can update all the software for you so you don't need to bother doing it for each program separately. To get the latest versions of all the software on your system, run:

```
sudo apt-get upgrade
```

This may take a little while, because open source software tends to be updated quite regularly. In Linux terms, you don't install particular applications, but packages. These are bundles of files. Usually, each one represents an application, but not always. For example, a package could be documentation, or a plug-in, or some data for a game. In actual fact, a package is just a collection of files that can contain anything at all.

In order to install software with **apt-get**, you need to know the package name. Usually this is pretty obvious, but it needs to be exactly right for the system to get the right package. If you're unsure, you can use **apt-cache** to search through the list of available packages. Try running:

```
apt-cache search iceweasel
```

This will spit out a long list of packages that all have something to do with the web browser (*iceweasel* is a rebranded version of *Firefox*).

To install *iceweasel*, run:

```
sudo apt-get install iceweasel
```

You will notice that **apt-get** then prompts you to install a number of other packages. These are dependencies. That means that *iceweasel* needs the files in these packages in order to run properly. Usually you don't need to worry about these – just press **Y** and the package manager will do everything for you.

However, if your SD card is running low on space, you may sometimes come across a program that has so many dependencies that they'll overflow the device. In these cases, you'll either need to free up some space or find another application that has fewer dependencies.

```
pi@raspberrypi: ~
File Edit Tabs Help
sis
xul-ext-requestpolicy - improve your browsing: more private, more secure
xul-ext-sage - Lightweight RSS and Atom feed reader for Iceweasel/Firefox
xul-ext-scrapbook - Iceweasel/Firefox extension to save and manage Web pages
xul-ext-searchload-options - tweak the searchbar's functionality
xul-ext-status4ever - Status bar widgets and progress indicators for Firefox 4+
xul-ext-syncplaces - synchronise Bookmarks and Passwords via WebDAV
xul-ext-tabmixplus - adds dozens of new capabilities to tabbed browsing
xul-ext-toggle-proxy - Toggle Proxy adds a status bar icon to toggle between two proxy settings
xul-ext-torbutton - Iceweasel/Firefox extension enabling 1-click toggle of Tor usage
xul-ext-treeestyletab - Show tabs like a tree
xul-ext-ubiquity - browser interface based on natural language input
xul-ext-uppity - toolbar button to "go up" on the web
xul-ext-useragentswitcher - Iceweasel/Firefox addon that allows the user to choose user agents
xul-ext-venkman - Javascript debugger for Mozilla based applications
xul-ext-webdeveloper - web developer extension for the Iceweasel/Firefox web browser
```

■ **apt-cache** in a terminal will give you a list of available packages.

If you then want to remove *Iceweasel*, you can do it with:

```
sudo apt-get purge iceweasel
```

The package manager will try to remove any dependencies that aren't used by other packages.

You'll often see packages with **-dev** at the end of package names. These are only needed if you're compiling software. Usually you can just ignore these.

**apt-get** is a great tool, but it isn't as user-friendly as it could be. There are a couple of graphical tools that make package management a bit easier. The first we'll look at is *Synaptic*. This isn't installed by default on Raspbian, so you'll have to get it using **apt-get** with:

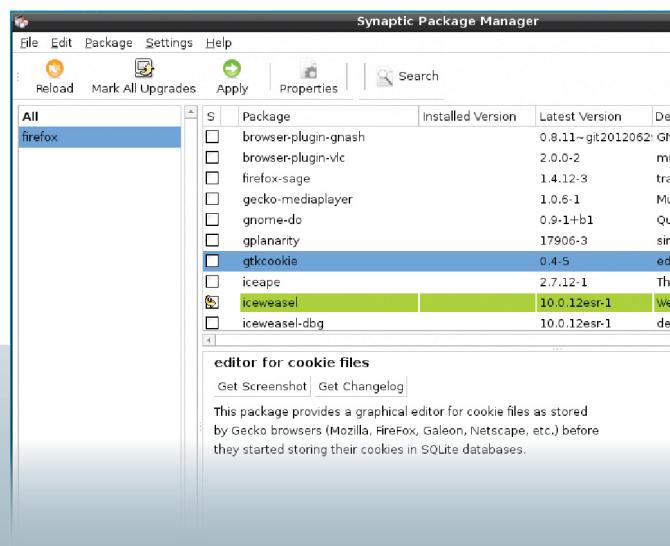
```
sudo apt-get synaptic
```

This works with packages in the same way as **apt-get**, but with a graphical interface. Once the package is installed, you can start it with:

```
--where is synaptic
```

See below for more information.

The second graphical tool is the Raspberry Pi App store. Unlike **apt-get** and *Synaptic*, this deals with commercial software as well as free software, so you have to pay to install some things. It comes by default on Raspbian, and you can get it by clicking on the icon on desktop. See 'Raspberry Pi Store' on the next page for more information.



## Synaptic

*Synaptic* lets you do everything you can with the command line **apt-get**, but the graphical interface is easier to use. We find it especially useful when searching, because the window is easier to look through than text on the terminal.

■ *Synaptic* provides a user-friendly front-end to the **apt** package management system.

## Compiling software

Sometimes, you'll find you need software that isn't in the repository, and so you can't get it using **apt-get**. In this case, you'll need to compile it.

Different projects package their source code in different ways, but usually, the following will work. Get the source code from the project's website, and unzip it. Usually, the filename will end in .tar.gz or .tgz. If this is the case, you can unzip it with:

```
tar zxvf <filename>
```

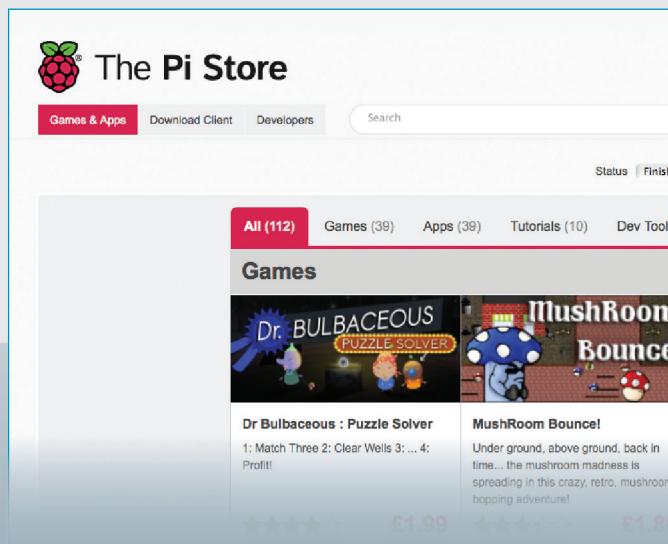
If the filename ends in .tar.bz2, you need to replace **zxvf** with **xjf**. This should now create a new directory which you need to **cd** into. Hopefully, there'll be a file called **INSTALL**, which you can read with **less INSTALL**. This should tell you

how to continue with the installation. Usually (but not always), it will say:

```
./configure  
make  
sudo make install
```

The first line will check you have all the necessary dependencies on your system. If it fails, you need to make sure you have the relevant -dev packages installed.

If that all works, you should have the software installed and ready to run.



### Raspberry Pi store

The Raspberry Pi store allows users to rate the software, so you can see how useful other people have found it. It also includes some non-free software. However, it doesn't have anywhere near the range that is available through **apt-get** or **Synaptic**.

■ The Pi Store is an app store for your Raspberry Pi.

# RaspBMC

You can install a media player, such as VLC, on to Raspbian, and use that to play videos.

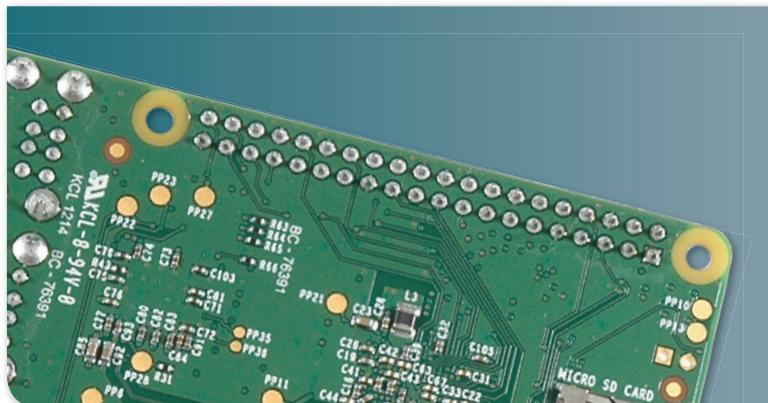
This works fine if you're using your Raspberry Pi as a general computer and giving it occasional multimedia functions.

However, the small size of the hardware, and the fact that it runs silently, makes it a good choice for building your own entertainment center. You could start from Raspbian and customize it to your needs, and this is a good idea if you have any unusual functions in mind. Fortunately for us (and you), a team of hackers have done all the difficult bits of building a media center for the Raspberry Pi, and packaged it for you to use. Follow the guide on page 6 to install the RaspBMC operating system from the NOOBS image.

This uses the popular XBMC media desktop, which is quite different from LXDE, which

you may have experienced on Raspbian. You can play media stored locally or, through the use of add-ons, stream content from the internet. Music and videos can be added to your setup either by attaching a USB device, or directly on to the SD card using FTP (username pi, password raspberry). See below for how to find the IP address.

Perhaps the only piece of configuration you will need is to make sure the sound is sent to the right place. Under '*System > System > Audio Output*', make sure that Output is set to Analog if you're using the jack, or HDMI if you're using this.



■ The Model B+ boards have mounting holes to help you keep your entertainment center tidied away.

This is all good, but it's a bit impractical to use a mouse and keyboard to control your telly. The good folks at XBMC thought of this and added support for remote controls. The simplest and most general-purpose way of doing this is via RaspBMC's web interface. Using this, any device with a web browser that's on the same network as the Pi can control the playback.

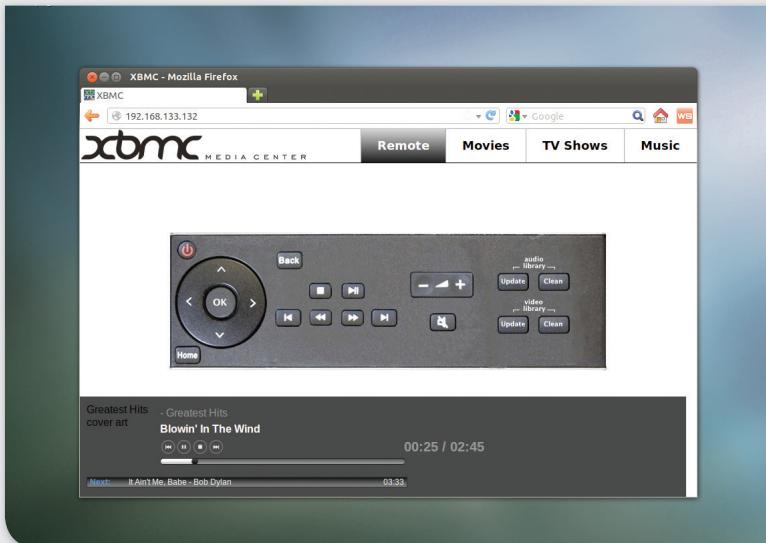
This is enabled by default, so all you need to do is find out the IP address of the Pi in 'System > System Info' (you should be able to set this to be a static IP on your router). Then, on any other device attached to a network, open a browser and point it to that IP.

If you're a modern sort, and using an Android or iOS mobile device, you should find a selection of apps in the various app stores that can do this with a nicer interface than the HTML pages. A new feature on the Model B+ allows a software switch to be set in the config.txt which allows additional power to be routed to the 4 USB ports (up to a maximum of 1.2Amps). This provides enough power to drive a 2.5 inch hard drive directly off the Pi. To enable this feature the Pi must be powered using a 1.8Amp or 2Amp micro USB power supply.

## Taking it further

It is possible to take complete control of your TV viewing using Linux, including watching live TV, and recording shows for later. This can be done using *MythTV* (available from [www.mythtv.org](http://www.mythtv.org)). You'll need a separate computer with the appropriate cable connections to act as the server.

You can play video files that you have stored on other computers on your network, for example those on a Network Attached Storage (NAS) box. The exact method for doing this will vary depending on how you share the files, but they are configured through the 'Add Sources' buttons. For more information about this, you should check out the wiki at: [bit.ly/OOvXb6](http://bit.ly/OOvXb6).



■ This is the default remote, but other more graphically stimulating ones are available in the RaspBMC settings.

# Camera controller

Back up your photos using your Pi.

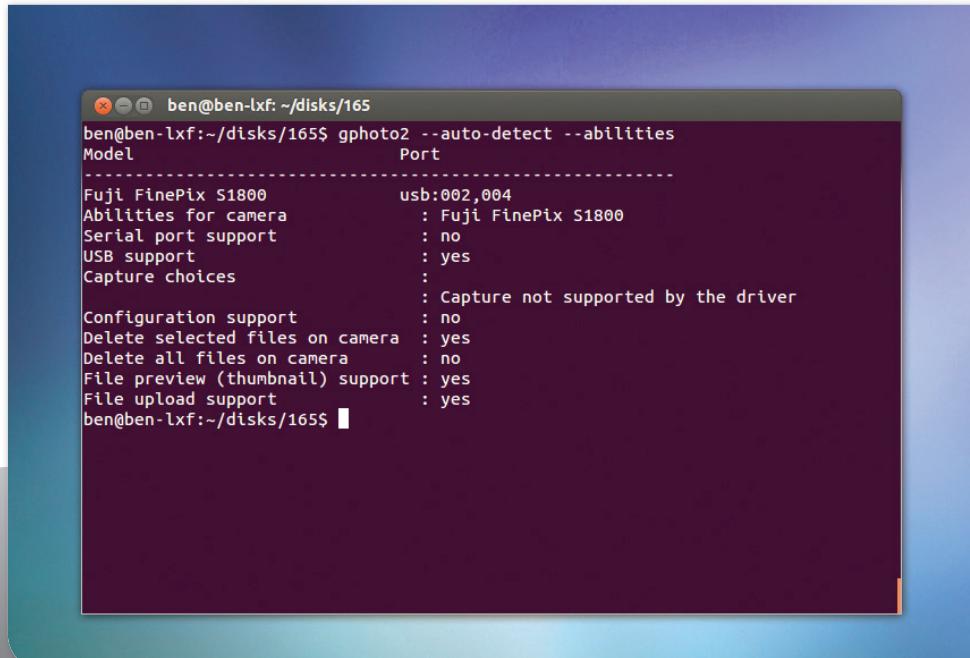
The size of the Raspberry Pi means we can use it to take control of other embedded devices.

This may seem a little redundant – the embedded devices obviously have some form of controller already – but it means we can script and extend them in ways that aren't possible (or are, at least, very difficult) without the extra device. Almost anything that you can plug in to a normal desktop can be scripted by a Pi, but we're going to look at cameras for a couple of reasons.

Firstly, there's support for most in Linux, and secondly there's a range of useful projects you can do once you've grasped the basics.

The best command-line tool for manipulating cameras in Linux is *Gphoto2*. Get it with:

```
apt-get install gphoto2
```



A screenshot of a terminal window titled "ben@ben-lxf: ~/disks/165". The window displays the output of the command "gphoto2 --auto-detect --abilities". The output shows various camera capabilities and their status. Key entries include:

Ability	Status
Model	Port
Fuji FinePix S1800	usb:002,004
Abilities for camera	: Fuji FinePix S1800
Serial port support	: no
USB support	: yes
Capture choices	:
Configuration support	: Capture not supported by the driver
Delete selected files on camera	: no
Delete all files on camera	: yes
File preview (thumbnail) support	: no
File upload support	: yes

- Gphoto2 has far more capabilities than we use here, including bindings for Java and Python. For full details, check out the project website: [www.gphoto.org](http://www.gphoto.org).

Before getting stuck in to the project, we'll take a look at this useful tool to see what it can do.

The desktop environment can try to mount the camera, and this can cause *Gphoto2* a few problems, so the easiest thing to do is run without it. Open a terminal and run **sudo raspi-conf**, and under Boot Behaviour, select 'No' to not start the windowing system, then reboot.

In the new text-only environment, plug in your camera and run:

**gphoto2 --auto-detect**

This will try to find any cameras attached to the Pi. Hopefully, it will pick up yours. While it does support an impressive array, there are a few cameras that won't work. If yours is one of the unlucky few, you'll need to beg, steal or borrow one from a friend before continuing.

Not all supported cameras are equal, so the next step is to see what the camera can do. To list the available actions, run:

**gphoto2 --auto-detect --abilities**

There are, broadly speaking, two main classes of abilities: capture, and upload/download. The former let you take photos with your scripts, and are present mostly

on higher-quality cameras. The latter let you deal with photos stored on the memory card, and are present on most supported cameras. In this project, we'll deal only with the second set of abilities.

The simplest command we can send to the camera is to get all the photos stored on it. This is:

**gphoto2 --auto-detect  
--get-all-files**

Running this will download all the files from the camera into the current directory. This would be fine on a normal computer, but you may not want to do it on a Pi, as you run the risk of filling up your memory card pretty quickly. Instead, we'll copy them on to a USB stick. To do this in an interactive session, you could simply use a GUI tool to mount the stick then run **df -h** to see where the USB stick is mounted, and **cd** to the directory. However, since this will run automatically, we need to know where the device will be. There are a few ways of doing this, but we'll keep it simple. We'll mount the first partition of the first serial disk, and store the photos there.

## Powering your Pi

The Raspberry Pi gets power from its micro USB port. This supplies 5V, and the Raspberry Pi foundation recommends an available current of at least 600mA. This can easily be delivered via a mains adaptor, or a USB cable from a computer. If you want your Raspberry Pi to be portable, however, there are

other options. Four AA batteries, for instance, should provide enough power, provided you have the appropriate housing and cables to get the power into the micro USB port. However, we found the best solution was to get a backup power supply for a mobile phone that plugs directly into the Pi.

Here, we're assuming that you're using the default user pi.  
If you're not, you'll need to adjust the script.

First, we need to create a mount point for the drive. This is just a folder, and can be put anywhere – we're going to spurn convention and put it in our home folder. So before running the script, run:

```
mkdir /home/pi/pic_mount
```

With this done, we're ready to go. The script to mount the drive and get the photos is:

```
#!/bin/bash

if mount /dev/sda1
/home/pi/pic_mount ; then
    echo "Partition mounted"
    cd /home/pi/pic_mount
    yes 'n' | gphoto2 --
auto-detect --get-all-files
    umount /dev/sda1
else
    echo "/dev/sda1 could not
be mounted"
fi
```

**yes 'n'** is a command that simply emits a stream of n characters. This means that when *Gphoto2* prompts to overwrite any previously downloaded files,

it will decline. The **umount** is essential, because it ensures that the drive is properly synced and can be removed.

We've called the script **get-pics.sh**, and saved it in Pi's home directory. To make it executable, run:

```
chmod +x /home/pi/get-pics.sh
```

You should now be able to run it manually. You'll need to use **sudo** because it needs to mount the drive.

The final piece of the puzzle is to get the script to run automatically. To do this, we add it to the file **/etc/rc.local**. This script runs when you boot up, and it executes as root, so there's no need to worry about permissions. Just open the file with a text editor as root. For example, with **sudo nano /etc/rc.local**, and add the line:

```
/home/pi/get-pics.sh
```

just before the line exit 0.

Now all you have to do is plug in your camera (making sure it's turned on, of course) and your USB stick, and it'll back up your photographs when you boot up your Raspberry Pi.



■ In this form, it's not very portable, but with a little bit of judicious DIY, you should be able to package it more conveniently.

## New directions

If you want to run the device headless, as will most likely be the case, you could attach LEDs to the GPIO pins, as shown over the page, and then use these to indicate statuses.

As well as saving them to the USB stick, you could upload them to an online service, such as Flickr. See the box on networking below for information on how to connect your Pi to your phone.

You could include some sort of switch to tell your Pi which photos to upload, and which to store on the USB stick – for example, upload low-resolution images, and store high-res ones. Or you could create low-resolution versions of the images, and

upload those. Of course, you don't have to stop there. If you have a wireless dongle in your Pi, you could use it to run an HTTP server. With some PHP (or other web language) scripting, you should be able to create an interface to *GPhoto2* that will allow you to connect from your mobile phone.

Taking it in a different direction, if your camera supports capture options, you could use your Pi to take photos as well as copy them.

## Networking

The Raspberry Pi comes with a wired Ethernet connection, which is fine for most occasions, but sometimes the cable just won't reach. You could use a USB wireless dongle – however, if you've got an Android phone, and your carrier hasn't disabled the feature, you can use this as your networking device. This has an extra advantage of not drawing as much power from the Pi, and so makes it easier when running from batteries.

You should be able to share your phone's connection to Wi-Fi as well as 3G, so it won't necessarily eat into your data allowance. Of course, it's best to check the connection type before downloading large files.

To do this, connect your phone to your Pi, and enable tethering in 'Settings > Wireless and Networks > Tethering and Portable Hotspots' (on the phone). Back

on the Pi, if you type **sudo ifconfig**, you should then see the interface **usb0** listed, but it won't have an IP address.

Networking interfaces are controlled by the **/etc/network/interfaces** file. By default, there isn't an entry in here for USB networking, so we need to set one up. Open the file with your favorite text editor as sudo. For example, with **sudo nano /etc/network/interfaces** and add the lines:

```
iface usb0 inet dhcp
nameserver 208.67.220.220
nameserver 208.67.222.222
```

This used the OpenDNS nameservers, but you could use others if you wish.

You can now either restart the interfaces or restart your Raspberry Pi to pick up the changes. You should have an internet connection up and running.

# Sending output

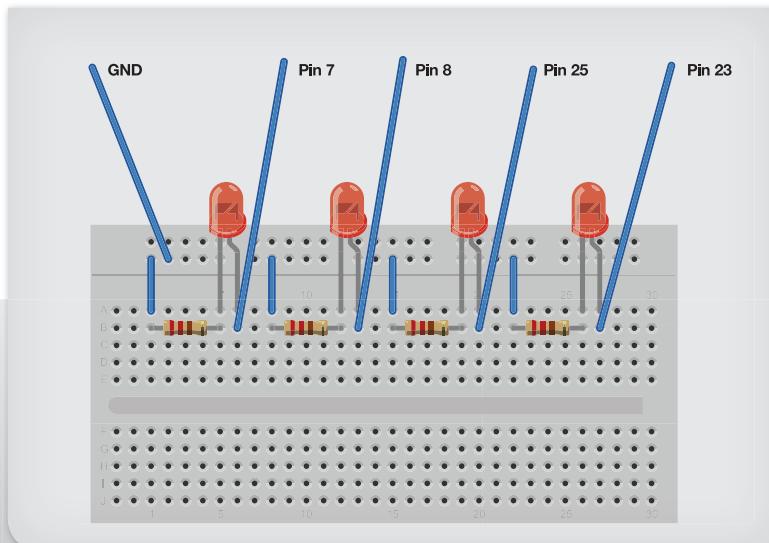
Use the GPIO pins to light up some LEDs.

The Pi's tiny size makes it ideal for making your own embedded devices. This can be a great way of creating small computing devices to solve specific problems, as we saw with the camera controller.

However, there is the slight problem that it can be hard to know what's going on inside your Pi without a screen. Fortunately, the designers of the Pi thought of this problem and have added the facility to get information on and off a Pi without the bulk of usual PC peripherals. This is done via General Purpose Input and Output (GPIO).

You may have wondered what the spiky pins near the SD card reader are for – well, you're about to find out. This basic circuit

can be used to display information from any source, but here we're going to use it to find the final byte of the IP address. This is useful if you want to remotely access your Pi, but can't configure it with a static IP because, for example, you have to move it between networks. Typically, you can find out the first three bytes from the netmask, but the final one can be elusive unless you have a monitor attached.



■ Figure 1. This shows how half of the LEDs are wired up. The additional ones are added in exactly the same manner.

We're going to use the **gpio** program, which is part of *WiringPi*. You can find out more about this from the website: [bit.ly/RP8UKJ](http://bit.ly/RP8UKJ).

The software comes as source code, so we'll have to unzip it and compile it with:

```
tar xvf wiringPi.tgz  
cd wiringPi/wiringPi  
make  
sudo make install  
cd ../gpio  
make  
sudo make install
```

We'll also use **bc**, so install it with:

```
sudo apt-get install bc
```

Now, that's enough about software – on with the hardware! Just a quick word of warning before we start: it is possible to break your Pi by connecting the wrong wires together, so make sure you double-check before powering up.

The circuit for this is very simple – you just have to connect each output to the positive leg of an LED, then the negative leg of the LED (shorter) to a 1 KOhm resistor, and finally the other leg of the resistor to the common ground. See figures 1, 2 and 3 for details.

Once you have your fully-set-up board connected to your Pi, you can make things happen. To start with, we'll just use the final pin. This is pin 7 (the layout of the pins doesn't follow a numbering pattern). Open up a terminal, and set it to output with:

```
gpio -g mode 7 out
```

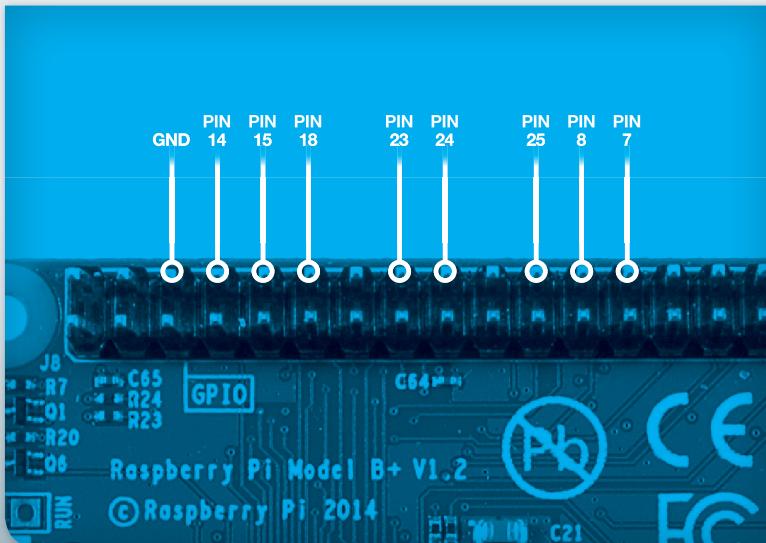
Then you can turn it on with:

```
gpio -g write 7 1
```

and off again with:

```
gpio -g write 7 0
```

If you're like us, you'll do that repeatedly until the novelty of it wears off. Once it has, you're ready to run the script. It contains four parts.



■ Figure 2. Connect the bread board to these pins. We used commercially-available single-pin connectors, but you could also solder connectors on, or use an old IDE cable.

The first just sets the pins to the right mode and makes sure they're turned off:

```
pins="7 8 25 24 23 18 15 14"
for x in $pins
do
    gpio -g mode $x out
    gpio -g write $x 0
done
```

The second grabs the IP address from **ifconfig**, converts it to binary, then pads it out with leading zeros, if necessary:

```
ipaddress=`ifconfig eth0 | grep
'inet ' | awk '{print $2}' | cut
-f4 -d'.'
binary=`echo
"ibase=10;obase=2;$ipaddress" | bc`  
paddedBinary=`printf %08d $binary`
```

The next part uses **cut** to extract the part we want from this binary string and outputs it to the appropriate pin.

```
bit=1
for x in $pins
do
    out=`echo $paddedBinary | cut
-b$bit`
    gpio -g write $x $out
    bit=$((bit+1))
done
```

And, finally, we tell the script to sleep for five minutes, then turn the LEDs off:

```
sleep 5m
for x in $pins
do
    gpio -g write $x 0
done
```

That's it! Save this into a file called **showIP.sh**, make it executable with:

```
chmod a+x showIP.sh
```

and type **sudo ./showIP.sh** to display your IP. To get this run automatically on boot, you just need to add the line:

```
/home/pi/showIP.sh &
```

to **rc.local**. See the 'Camera controller' section for details on how to do this.

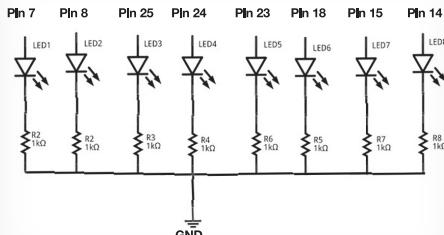
## Gertboards and Arduinos

Connecting directly to your Pi's GPIO pins can provide you with basic input and output control, but there are limitations. There are two additional items that you can obtain to help you interact more precisely with the world around you.

The Gertboard is a fairly complete expansion pack for connecting between your Pi and the real world, including a micro controller, and a range of input and output options.

Meanwhile, the Arduino is a micro controller that can connect to your Pi (or any other computer) via the USB port. Typically, it comes assembled, but kit forms are also available. In its raw form, it has fewer features than the Gertboard (which includes an Arduino microcontroller), but it can be expanded with a huge range of shields.

■ Figure 3. The simple circuit in all its glory.



We've shown you how to send output via the GPIO, but as the name suggests, they can also receive input.

With this, it's even more important to ensure that you don't send too much power into the pins. To get input, just set the mode to input with `gpio -g mode <pin number> in` and then read the value with `gpio -g read <pin number>`.

This hardware can display any eight bits of information, so you don't have to limit it to displaying just IP addresses. For example, you could make a modified version of the camera controller script to use the LEDs to indicate its progress.

You can find details on the full selection of GPIO pins at [bit.ly/JTIE3](http://bit.ly/JTIE3). If you design your

own circuits, or use ones off the web, make sure you use the right pins for your board.

You don't have to limit yourself to just switching pins on and off. The Pi supports a few methods of passing larger amounts of data through the GPIO. The two most common of these are Serial Peripheral Interface bus (SPI) and Inter-Integrated Circuit (I<sup>2</sup>C).

There are a number of devices available that use these, and plenty of information online to help get you started. So what's stopping you? Get out your soldering iron and build a robot army.

## Ohm's Law

There are two key ways of measuring electricity: voltage and current. Voltage (measured in volts) is the amount of energy a given quantity of electrons has, while current (measured in amps) is the amount of electrons flowing past a point.

The two are intimately connected by Ohm's law which states: Voltage = Current x Resistance, or  $V=IR$ . You can use this connection to make sure you don't accidentally toast your Raspberry Pi by pushing too much current into it. The exact setup of the Pi is a little complex. If you wish to delve into it, Gert van Loo (one of the designers) has put together an explanation, which can be found at: [bit.ly/Qp4PMI](http://bit.ly/Qp4PMI)

As a rough rule of thumb, you can expect to draw voltage out of a GPIO pin at 3.3V, and you shouldn't draw more than 16mA, or push more than this into an input pin.

This is the maximum current; you should aim to use less. So, with Ohm's law we know  $V=IR$ , so  $R=V/I$ . If we put in the data from the Pi, and want to ensure we don't damage it, we know that R must be greater than  $3.3/0.016$ , which is 206.25 Ohms.

Remember, this is the smallest amount of resistance it's safe to use with a GPIO output. You should aim for a margin of safety several times above this unless absolutely necessary. In our circuits, we've used 1,000 Ohms, which gives us a safety factor of almost 5.

# Minecraft Pi

## Minecraft on the Raspberry Pi

Minecraft: Pi Edition is a version of Minecraft that's designed to work on the Raspberry Pi.

Based on the "Pocket Edition", Minecraft Pi is smaller and faster as well as containing support for multiple programming languages. Minecraft Pi runs on the Raspbian Operating System for the Pi. This is the most common operating system used on the Pi. To get started, head over to [s3.amazonaws.com/assets.minecraft.net/pi/minecraft-pi-0.1.1.tar.gz](https://s3.amazonaws.com/assets.minecraft.net/pi/minecraft-pi-0.1.1.tar.gz) to download the package. Once that is finished, open a new terminal window or click on "LX Terminal" icon on the desktop.

To decompress your freshly downloaded package type:

```
tar -zxvf minecraft-pi-0.1.1.tar.gz
```

To start it:

```
Type cd mcpi
```

Then:

```
./minecraft-pi
```

At this point you can play Minecraft as a single user or over a network. What is really exciting is what comes next which is programming Minecraft to perform certain functions for you.



Minecraft Pi.

# Programming Minecraft Pi

Once you have launched the game then you can select Start Game followed by Create New. This should take you into a randomly generated world that you can start to modify with code.

You should still have your LX Terminal windows open. If not then click on the LX Terminal window again to open a command line prompt. To do this you will need to press escape to pause the game and allow you to move the mouse out of the game window. You can then launch another terminal from the desktop. One way is to have the programming terminal window and the Minecraft Window side by side

Once you have your terminal launched you need to go to the Python API directory (programming tools) by typing the following at the terminal window:

```
cd mcpi/api/python/mcpi
```

and then:

```
python
```

and pressing return.

Your terminal prompt should change to **>>>**.

Now we are ready to start programming. The first thing we need to do is to import the Minecraft library.

At the command prompt in the terminal windows type:

```
import minecraft
```

Now we need to initialize it which will create a connection to your game and allow you to start modifying the game world. So now type:

```
mc = minecraft.Minecraft.create()
```

You can now use the mc object you have created to control the world. Let's say hello to your Minecraft world:

So type at the terminal window:

```
mc.postToChat("Hello Minecraft  
World!")
```

You should then see your message come up on the chat window in your game. Feel free to change this message to what ever you want. You can use the postToChat method to post all sorts of information you want the player to see.

Congratulations, you have learned to simply program Minecraft Pi.

Now on to a more advanced project, that is Placing Block within your game. To place a block let's type the following at the command prompt:

```
import block  
mc.setBlock(0, 10, 0, block.STONE)
```

The first line which imports block simply allows us to use the block names when we position them. The name "block. STONE" can be changed to block. LAVA, block.MELON or block.WATER. The Minecraft Wikipage gives you many other types of blocks to use.  
[minecraft.wikia.com/wiki/Category:Blocks](http://minecraft.wikia.com/wiki/Category:Blocks)

The second line calls setBlock and passes in 3 numbers and the block type. These 3 numbers are the X, Y and Z coordinates in the game world. Just like you would find in maths, this coordinate system, called Cartesian Coordinates, allows you to map your blocks in a 3D world. From your player start position X is left and right, with numbers going down as you head left and up if you head right. The Y coordinates are up and down, with up increasing the number and down decreasing the number. This leaves the Z axis being forwards and backwards, with forward increasing the number and backwards decreasing. It's worth noting that decreasing numbers go to a negative value.

Have a go at setting different coordinates for blocks and see how they come out. If you don't see a block then it means you either put it very far away or it's underground. To avoid blocks going too far away try and keep the X, Y and Z values within 20 and -20.

You can view the source code for this tutorial here: [github.com/piprogramming/Minecraft/blob/master/tutorial1.py](https://github.com/piprogramming/Minecraft/blob/master/tutorial1.py)

If you have the git-core package installed then you can download it using: git clone [git://github.com/piprogramming/Minecraft.git](https://github.com/piprogramming/Minecraft.git)

# Postscript

## The making of Pi

The idea behind a tiny and affordable computer for kids came in 2006, when Eben Upton, Rob Mullins, Jack Lang and Alan Mycroft, based at the University of Cambridge's Computer Laboratory, became concerned about the year-on-year decline in the numbers and skills levels of the A-Level students applying to read Computer Science.

From a situation in the 1990s where most of the kids applying were coming to interview as experienced hobbyist programmers, the landscape in the 2000s was very different; a typical applicant might only have done a little web design.

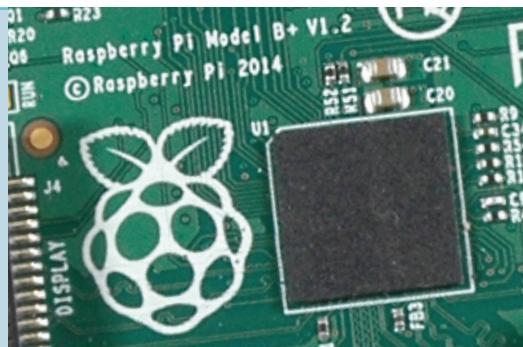
The project started to look very realizable. Eben (now a chip architect at Broadcom), Rob, Jack and Alan, teamed up with

Pete Lomas, MD of hardware design and manufacture company Norcott Technologies, and David Braben, co-author of the seminal BBC Micro game Elite, to form the Raspberry Pi Foundation to make it a reality. Three years later, the Raspberry Pi Model B entered mass production through licensed manufacture deals with element14, and within two years it had sold over three million units.

### More information

For more information about your Pi, the following links are the best places to start

- [www.element14/raspberrypi](http://www.element14/raspberrypi)
- [www.raspberrypi.org](http://www.raspberrypi.org)
- [www.mcmelectronics.com](http://www.mcmelectronics.com)
- [www.cpc.farnell.com/raspberrypi](http://www.cpc.farnell.com/raspberrypi)
- [www.element14.com/raspberrypi/projects](http://www.element14.com/raspberrypi/projects)



# Command Line reference

## Navigation and files

<b>cd</b>	Changes directory. For example, <code>cd movies</code> moves to the <code>movies</code> folder. <code>cd ~</code> moves to your home directory, <code>cd /</code> moves to the root directory, <code>cd ..</code> moves back one directory.
<b>ls</b>	Lists files. By itself, it lists the files in the current directory. <code>ls movies</code> lists the files in the directory <code>movies</code> . <code>ls -a</code> lists all files (including hidden ones), and <code>ls -l</code> lists more information about each file.
<b>cp</b>	Copies files. <code>cp orig-file new-file</code> copies <code>orig-file</code> to <code>new-file</code> .
<b>wget</b>	Downloads a file from the internet. To download the Google homepage to the current directory, use <code>wget www.google.com</code> .
<b>df -h</b>	Displays the amount of space left on the device.
<b>pwd</b>	Displays the current directory.
	The pipe symbol is used to pass information from one program to another.

## Finding files

<b>find &lt;location&gt; &lt;tests&gt;</b>	Useful flags include: <code>-mtime &lt;number&gt;</code> finds files modified in the last <code>&lt;number&gt;</code> days. <code>&lt;number&gt;</code> could be, for example, <code>2</code> (exactly two days ago), <code>-2</code> (less than two days ago) or <code>+2</code> (more than two days ago). <code>-name &lt;filename&gt;</code> finds files called <code>&lt;filename&gt;</code> . <code>-iname &lt;filename&gt;</code> matches files called <code>&lt;filename&gt;</code> but not case-sensitive. <code>-writable</code> finds files that are writable. There are many more options. See the man page for a detailed list. For example <code>find / -mtime -2 -writable</code> finds all files on the filesystem that were changed less than two days ago and are writable by the current user.
--	--

# Command Line reference

## Remote working

### ssh

Log in to a remote computer using Secure SHell (SSH protocol). `ssh pi@192.168.1.2` will log in as user pi on the computer at the IP address 192.168.1.2. Note, this will only work if the remote computer has an SSH server running.

### scp

Secure copy. `scp file pi@192.168.1.2 :/home/pi` will copy a file to the directory `home/pi` on the machine with 192.168.1.2. `scp pi@192.168.1.2:/home/pi/file .` will copy `/home/pi/file` from the machine 192.168.1.2 to the current directory. Note, this will only work if the remote machine has an SCP server running.

## Wildcards

**\*** Matches any string of characters, or no characters.

**?** Matches any single character.

**[abc]** Matches a, b or c.

**[!abc]** Matches any character except a, b or c.

**[A-Z]** Matches any character in the range A–Z (that is, any upper-case letter).

**[A-z]** Matches any character in the range A–z (that is, any upper- or lower-case letter).

**[one, two]** Matches the words one and two.

## Information about the computer

### top

Displays the programs that are currently using the most CPU time and memory.

### uname

Displays information about the kernel. `uname -m` outputs the architecture it's running on.

### lscpu

Lists information about the CPU.

### dmesg

Displays the kernel messages (can be useful for finding problems with hardware).

## Text files

<b>head</b>	Displays the first 10 lines of a text file. Change 10 to any number with the <code>-n</code> flag. For example, <code>dmesg   head -n 15</code> displays the first 15 lines of the kernel message log.
<b>less</b>	Allows you to scroll through a text file.
<b>cat</b>	Dumps the contents of a text file to the terminal.
<b>tail</b>	Displays the last 10 lines of a text file. Can use the <code>-n</code> flag like <code>head</code> . Can also keep track of a file as it changes with the <code>-f</code> (follow) flag. For example, <code>tail -n15 -f /var/log/syslog</code> will display the final 15 lines of the system log file, and continue to do so as it changes.
<b>nano</b>	A user-friendly command line text editor ( <code>Ctrl+X</code> exits and gives you the option to save changes).

## Special keys

<b>[Ctrl]+[C]</b>	Kills whatever is running in the terminal.
<b>[Ctrl]+[D]</b>	Sends the end-of-file character to whatever program is running in the terminal.
<b>[Ctrl]+[Shift]+[C]</b>	Copies selected text to the clipboard.
<b>[Ctrl]+[Shift]+[V]</b>	Pastes text from the clipboard.

## Installing software

<b>tar zxvf file.tar.gz</b> <b>tar xjf file.tar.bz</b> <b>./configure</b>	When you unzip a program's source code, it will usually create a new directory with the program in it. <code>cd</code> into that directory and run <code>./configure</code> . This will check that your system has everything it needs to compile the software.
<b>make</b>	This will compile the software.
<b>make install</b> (needs sudo)	This will move the newly compiled software into the appropriate place in your system so you can run it like a normal command.
<b>apt-get</b>	This can be used to install and remove software. For example, <code>sudo apt-get install iceweasel</code> will install the package <code>iceweasel</code> (a rebranded version of <code>Firefox</code> ). <code>sudo apt-get purge iceweasel</code> will remove the package. <code>apt-get update</code> will grab an up-to-date list of packages from the repository (a good idea before doing anything). <code>apt-get upgrade</code> will upgrade all packages that have a newer version in the repository.
<b>apt-cache search &lt;keyword&gt;</b>	Will search the repository for all packages relating to keyword.

