



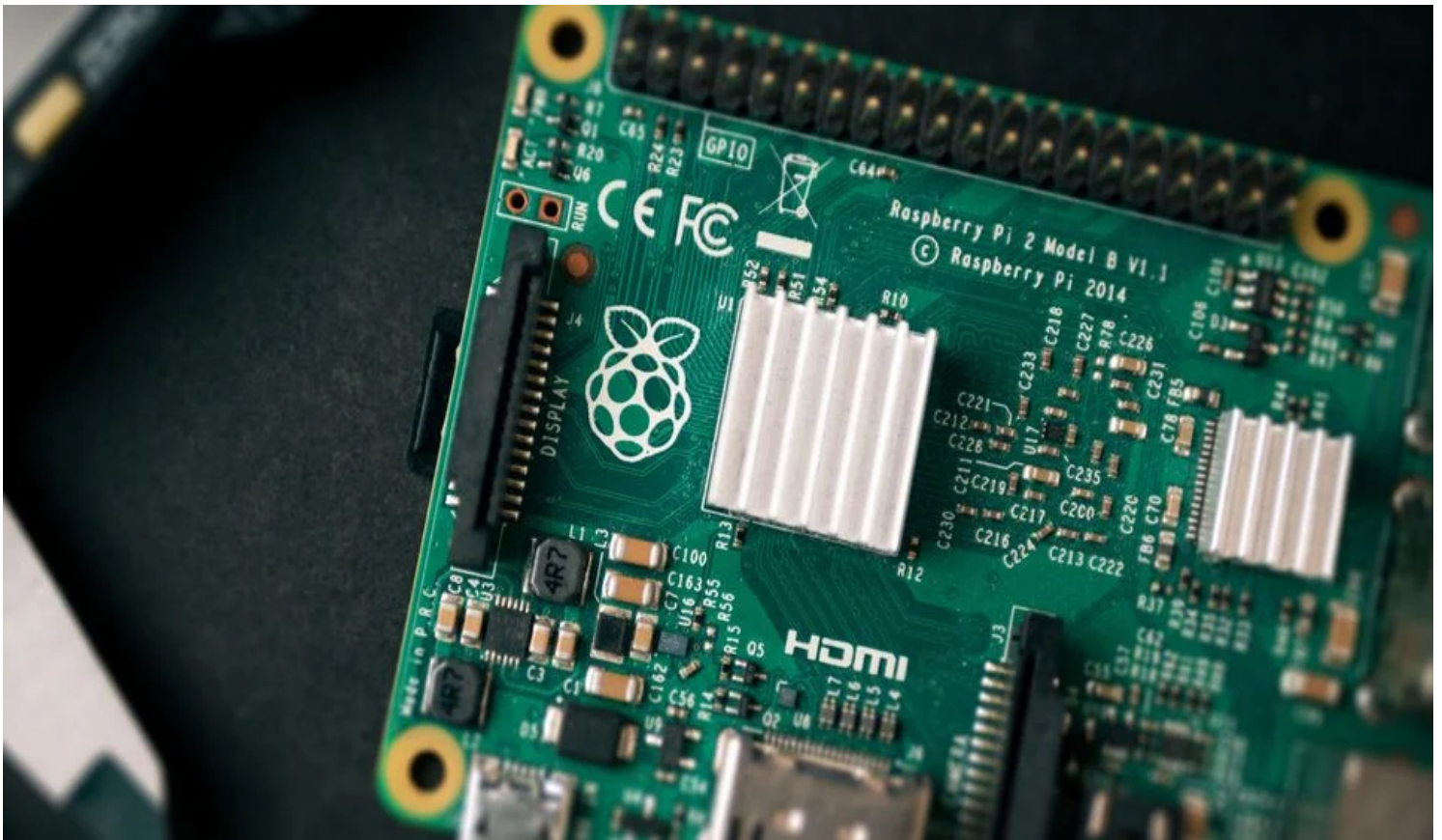
Home > Linux > The Ultimate Raspberry Pi Commands Cheat Sheet

The Ultimate Raspberry Pi Commands Cheat Sheet

Want to get more from your Raspberry Pi? Here's loads of helpful commands to navigate the Pi's terminal and program its GPIO pins.

BY IAN BUCKLEY

AUG 01, 2020



The Raspberry Pi is the most popular single-board computer (SBC) available. Just \$35 gets you a powerful, fully-fledged Linux computer with Wi-Fi, Bluetooth, and 40 connected general-purpose input/output (GPIO) pins. It's no surprise that the Raspberry Pi is equally popular between professional engineers, computer hobbyists, makers, and in the education sector.

While the Raspberry Pi runs on Linux, there are a few more features that you'd find in a common Linux distribution. The addition of GPIO pins, along with the two main libraries supporting them, means much more to remember!

That's why we've prepared this handy cheat sheet for day-to-day Raspberry Pi usage.

FREE DOWNLOAD: This cheat sheet is available as a **downloadable PDF** from our distribution partner, TradePub. You will have to complete a short form to access it for the first time only.

Download [**The Ultimate Raspberry Pi Commands Cheat Sheet**](#).

The Ultimate Raspberry Pi Commands Cheat Sheet

Command	Result
Raspbian OS Terminal	
cat [name]	Show the contents of the file [name]
cd ..	Change to parent directory
cd [path]	Move to the directory at [path]
cd /	Change to root directory
cd ~	Change to your home directory - usually "/home/"
chmod [who][+,-,=][permissions] [name]	Change the permissions for a file
chmod 777 [name]	Allow all users to read, write and execute the file [name]
chmod u+x [name]	Allow the user to execute [name]
cp -r [from] [to]	Copy all files and subdirectories from source [from] to destination [to]
cp [from] [to]	Copy a file from source [from] to destination [to]
find	Search for files and their contents
grep 'string' [name]	Search inside one or more files for occurrences of 'string'
head [name]	Return all occurrences of 'string' within file [name]
ls	List the contents of the current directory
ls -a	List all files including hidden files
ls -l	List the contents of the current directory with more file information
ls [path]	List the contents of the directory found at [path]
man [command]	Open the manual/help page for [command]
man man	Open the manual/help page for the 'man' command (helpception)

Command	Result
mkdir [name]	Create a directory called [name] in the current working directory
mv -r [from] [to]	Move all files and directories from source [from] to destination [to]
mv [from] [to]	Move a file from source [from] to destination [to]
pwd	Show the name of the current working directory
python/python3 --version	Shows you what version of Python you currently have installed
rm -r *	Remove all files and directories from the current working directory
rm [name]	Remove the specified file
rm *	Remove all files from the current working directory
rmdir [name]	Remove the empty directory [name] from the current working directory
sudo [command]	Superuser do. Execute [command] with elevated privileges (Allows you to do things you usually wouldn't have access to)
sudo apt-get install [package]	Install a package
sudo apt-get update	Update the list of packages
sudo apt-get upgrade	Upgrade the installed packages - must be run after sudo apt-get update
sudo chown pi:root [name]	Change the owner of the file [name] to user 'pi' and set the group to 'root'
sudo raspi-config	Launch the Raspberry Pi configuration menu
sudo reboot	Safely restart your Pi
sudo shutdown -h now	Safely shutdown your Pi immediately
sudo su	Places you in the root directory with root user access - be careful with this!
tail [name]	Show the end of file [name]
tar -cvzf [name] [path]	Create compressed file [name] from the contents of [path]
tar -xvzf [name]	Extract the contents of the compressed file [name]
wget [uri]	Download the file found at [uri] on the internet
RPi.GPIO Library	
import RPi.GPIO as GPIO	Import the RPi.GPIO module into the python sketch

Command	Result
<code>GPIO.setmode(GPIO.BCM)</code>	Use Broadcom pin numbers (GPIO 14, GPIO 15 etc)
<code>GPIO.setmode(GPIO.BOARD)</code>	Use board pin numbers (4,5, 8 etc)
<code>GPIO.getmode()</code>	Returns current pin numbering mode (BCM, BOARD, or None)
<code>GPIO.setup([pin number], GPIO.IN)</code>	Set up the pin at [pin number] to be an input
<code>GPIO.setup([pin number], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)</code>	Set up the pin at [pin number] to be an input with internal pull down resistance
<code>GPIO.setup([pin number], GPIO.IN, pull_up_down=GPIO.PUD_UP)</code>	Set up the pin at [pin number] to be an input with internal pull up resistance
<code>GPIO.setup([pin number], GPIO.OUT)</code>	Set up the pin at [pin number] to be an output
<code>GPIO.setup([pin number], GPIO.OUT, initial=1)</code>	Set up the pin at [pin number] to be an output with the initial value '1'
<code>GPIO.output([pin number], 1)</code>	Set [pin number]'s value to 1. Note that 1, GPIO.HIGH and True are the same thing
<code>GPIO.output([pin number], 0)</code>	Set [pin number]'s value to 0. Note that 0, GPIO.LOW and False are the same thing
<code>i = GPIO.input([pin number])</code>	Set the variable i to the value of [pin number]
<code>if GPIO.input([pin number]):</code>	Use the value of [pin number] as a boolean in code
<code>GPIO.cleanup()</code>	Reset all GPIO pins (good practice to call before leaving any program)
<code>GPIO.VERSION</code>	Returns current RPi.GPIO version
GPIO Zero Library	
LEDs	
<code>from gpiozero import LED</code>	Import the LED section of the gpiozero library
<code>led = LED(17)</code>	Assign the 'led' variable to an LED on pin GPIO 17
<code>led.on()</code>	Turn on the LED stored in the 'led' variable
<code>led.off()</code>	Turn off the LED stored in the 'led' variable
<code>led.toggle()</code>	Toggle the LED stored in the 'led' variable (if it's off, turn it on and vice versa)
Motors	
<code>from gpiozero import Motor</code>	Import the Motor section of the gpiozero library

Command	Result
<code>motor = Motor(17, 18)</code>	Assign the variable 'motor' to a Motor object containing the forward and backward drive pin numbers
<code>motor.forward()</code>	Activate the forward pin of the variable 'motor'
<code>motor.backward()</code>	Activate the backward pin of the variable 'motor'
<code>motor.reverse()</code>	Reverse the current motor direction
<code>motor.stop()</code>	Stop the motor
Buzzer	
<code>from gpiozero import Buzzer</code>	Import the Buzzer section of the gpiozero library
<code>bz = Buzzer(3)</code>	Assign the variable bz to a Buzzer on pin GPIO3
<code>bz.on()</code>	Turn the buzzer on
<code>bz.off()</code>	Turn the buzzer off
<code>bz.toggle()</code>	Toggle the buzzer's state (if it's off, turn it on and vice versa)
Servo	
<code>from gpiozero import Servo</code>	Import the Servo section of the gpiozero library
<code>servo = Servo(17)</code>	Assign the 'servo' variable to a Servo on GPIO 17
<code>servo.min()</code>	Move the servo to its minimum value
<code>servo.mid()</code>	Move the servo to its middle value
<code>servo.max()</code>	Move the servo to its maximum value
<code>servo.value = 0.5</code>	Move the servo to a set numerical point (min = -1, max = 1)
Raspi Camera Image	
<code>raspistill</code>	Command to take a still image with attached camera, modify with arguments below
<code>--width, -w</code>	Set image width
<code>--height, -h</code>	Set image height
<code>--quality, -q</code>	Set JPEG quality <0 to 100> (75 is most common)
<code>--raw, -r</code>	Inserts raw Bayer data from the camera into the JPEG metadata
<code>--output, -o</code>	Output filename (required for saving)
<code>--latest, -l</code>	Add latest frame to filename
<code>--verbose, -v</code>	Verbose debugging information during run
<code>--timeout, -t</code>	Set a time to wait before capturing an image.
<code>--encoding, -e</code>	Encoding to use for output file - jpg, gif, bmp, or png

Command	Result
Raspi Camera Video	
raspivid	Command to take a video using attached camera, modify with arguments below
--width, -w	Set image width (between 64px - 1920px)
--height, -h	Set image height (between 64px - 1080px)
--bitrate, -b	Set bitrate in bits per second (i.e 15 Mbits/s = 15000000)
--output, -o	Output filename (required for saving)
--verbose, -v	Verbose debugging information during run
--timeout, -t	Set a time to wait before capturing video
--framerate, -fps	Specify the frames per second for recording

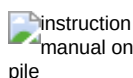
Expand Your Knowledge With Linux

These commands will help you navigate the Pi's terminal, and program its GPIO pins. That said, this cheat sheet doesn't even begin to scratch the surface. There is a huge depth of support for hobby components, cameras and screens for the Raspberry Pi. Listing them in one place is almost impossible!

The good thing is, since the Raspberry Pi is a Linux computer, you can [reference a cheat sheet for general Linux systems](#) to expand your knowledge even further.

The Linux Commands Reference Cheat Sheet

This simple cheat sheet will help you get comfortable with the Linux command line terminal in no time.



How to Find Any Instruction Manual for Free Online

You don't know when you might need an instruction manual to run a device. Here are some of the best available resources to find free manuals anytime.



RELATED TOPICS