Using Linux

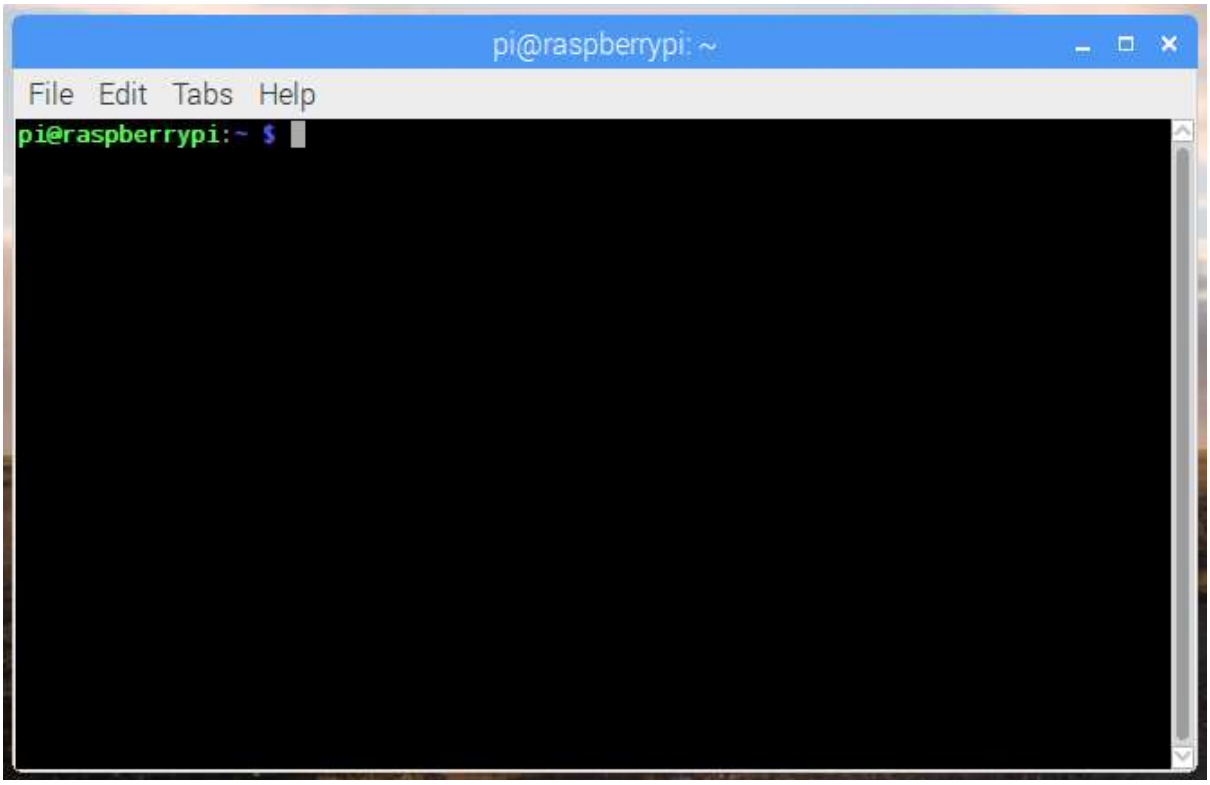
Terminal

Edit this on GitHub

The terminal (or 'command-line') on a computer allows a user a great deal of control over their system. Users of Windows may already have come across `Command Prompt` or `Powershell`, while mac OS users may be familiar with `Terminal`. All of these tools allow a user to directly manipulate their system through the use of commands. These commands can be chained together and/or combined together into complex scripts that can potentially complete tasks more efficiently than much larger traditional software packages.

Opening a Terminal window

On the Raspberry Pi OS, the default terminal application is called `LXTerminal`. This is known as a 'terminal emulator', this means that it emulates the old style video terminals — from before Windowing systems were developed — inside a graphical environment. The application can be found on the Raspberry Pi desktop, and when started will look something like this:



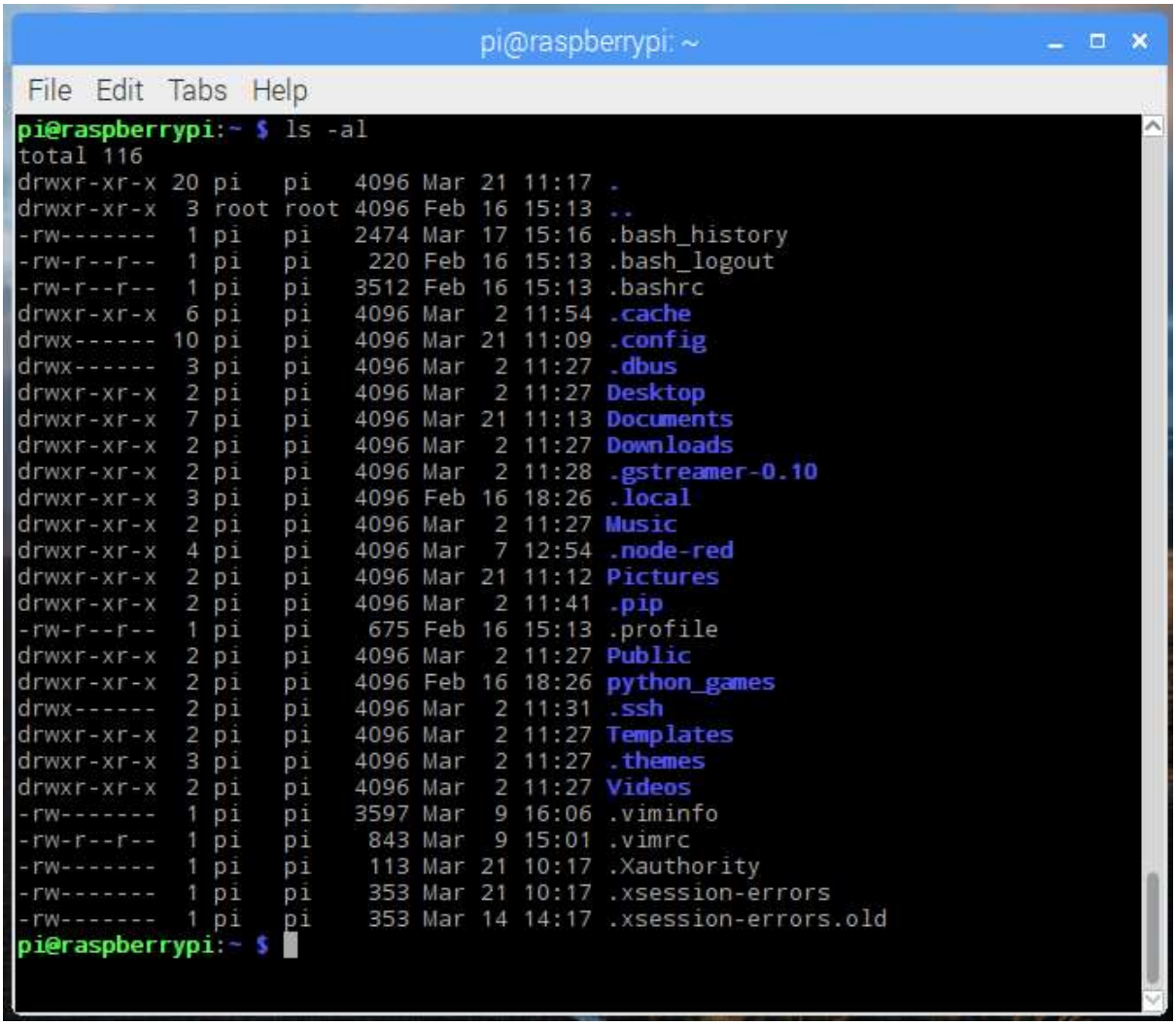
In the terminal window you should be able to see the following prompt:

```
pi@raspberrypi ~ $
```

This shows your username and the hostname of the Pi. Here the username is `pi` and the hostname is `raspberrypi`.

Navigating and browsing your Pi

One of the key aspects of using a terminal is being able to navigate your file system. Go ahead and type `ls -la` into the Terminal window, and then hit the RETURN key. You should see something similar to:



The `ls` command lists the contents of the directory that you are currently in (your present working directory). The `-la` component of the command is what's known as a 'flag'. Flags modify the command that's being run. In this case the `l` displays the contents of the directory in a list, showing data such as their sizes and when they were last edited, and the `a` displays all files, including those beginning with a `.`, known as 'dotfiles'. Dotfiles usually act as configuration files for software and as they are written in text, they can be modified by simply editing them.

In order to navigate to other directories the change directory command, `cd`, can be used. You can specify the directory that you want to go to by either the 'absolute' or the 'relative' path. So if you wanted to navigate to the `python_games` directory, you could either do `cd /home/pi/python_games` or just `cd python_games` (if you are currently in `/home/pi`). There are some special cases that may be useful: `~` acts as an alias for your home directory, so `~/python_games` is the same as `/home/pi/python_games`; `.` and `..` are aliases for the current directory and the parent directory respectively, e.g. if you were in `/home/pi/python_games`, `cd ..` would take you to `/home/pi`.

History and auto-complete

Rather than type every command, the terminal allows you to scroll through previous commands that you've run by pressing the `up` or `down` keys on your keyboard. If you are writing the name of a file or directory as part of a command then pressing `tab` will attempt to auto-complete the name of what you are typing. For example, if you have a file in a directory called `aLongFileName` then pressing tab after typing `a` will allow you to choose from all file and directory names beginning with `a` in the current directory, allowing you to choose `aLongFileName`.

The `sudo` command

Some commands that make permanent changes to the state of your system require you to have root privileges to run. The command `sudo` temporarily gives your account (if you're not already logged in as root) the ability to run these commands, provided your user name is in a list of users ('sudoers'). When you append `sudo` to the start of a command and press `enter`, the command following `sudo` will be run using root privileges. Be very careful: commands requiring root privileges can irreparably damage your system! Note that on some systems you will be prompted to enter your password when you run a command with `sudo`.

Further information on `sudo` and the root user can be found on the linux root page.

Installing software using `apt`

You can use the `apt` command to install software in Raspberry Pi OS. This is the 'package manager' that is included with any Debian-based Linux distributions, including Raspberry Pi OS. It allows you to install and manage new software packages on your Raspberry Pi.

In order to install a new package, you would type `sudo apt install <package-name>`, where `<package-name>` is the package that you want to install.

Running `sudo apt update` will update a list of software packages that are available on your system. If a new version of a package is available, then `sudo apt full-upgrade` will update any old packages to the new version.

Finally, `sudo apt remove <package-name>` removes or uninstalls a package from your system.

Other useful commands

There are a few other commands that you may find useful, these are listed below:

- `cp` makes a copy of a file and places it at the specified location (essentially doing a 'copy-paste'), for example - `cp file_a /home/other_user/` would copy the file `file_a` from your home directory to that of the user `other_user` (assuming you have permission to copy it there). Note that if the target is a folder, the filename will remain the same, but if the target is a filename, it will give the file the new name.

- `mv` moves a file and places it at the specified location (so where `cp` performs a 'copy-paste', `mv` performs a 'cut-paste'). The usage is similar to `cp`, so `mv file_a /home/other_user/` would move the file `file_a` from your home directory to that of the specified user. `mv` is also used to rename a file, i.e. move it to a new location, e.g. `mv hello.txt story.txt`.

- `rm` removes the specified file (or directory when used with `-r`). **Warning:** Files deleted in this way are generally not restorable.

- `mkdir`: This makes a new directory, e.g. `mkdir new_dir` would create the directory `new_dir` in the present working directory.

- `cat` lists the contents of files, e.g. `cat some_file` will display the contents of `some_file`.

Other commands you may find useful can be found in the commands page.

Finding out about a command

To find out more information about a particular command then you can run the `man` followed by the command you want to know more about (e.g. `man ls`). The man-page (or manual page) for that command will be displayed, including information about the flags for that program and what effect they have. Some man-pages will give example usage.

The Linux File System

Edit this on GitHub

It is important to have a basic understanding of the fundamentals of the Linux file system: where your files are kept, where software is installed, where the danger zones are, and so on. For more information, please refer to the Linux Filesystem Hierarchy Standard.

Home

When you log into a Pi and open a terminal window, or you boot to the command line instead of the graphical user interface, you start in your home folder; this is located at `/home/pi`, assuming your username is `pi`.

This is where the user's own files are kept. The contents of the user's desktop is in a directory here called `Desktop`, along with other files and folders.

To navigate to your home folder on the command line, simply type `cd` and press `Enter`. This is the equivalent of typing `cd /home/pi`, where `pi` is your username. You can also use the tilde key (~), for example `cd ~`, which can be used to relatively link back to your home folder. For instance, `cd ~/Desktop/` is the same as `cd /home/pi/Desktop`.

Navigate to `/home/` and run `ls`, and you'll see the home folders of each of the users on the system.

Note that if logged in as the root user, typing `cd` or `cd ~` will take you to the root user's home directory; unlike normal users, this is located at `/root/` not `/home/root/`.

Linux Commands

Edit this on GitHub

Here are some fundamental and common Linux commands with example usage:

Filesystem

ls

The `ls` command lists the content of the current directory (or one that is specified). It can be used with the `-l` flag to display additional information (permissions, owner, group, size, date and timestamp of last edit) about each file and directory in a list format. The `-a` flag allows you to view files beginning with `.` (i.e. dotfiles).

cd

Using `cd` changes the current directory to the one specified. You can use relative (i.e. `cd directoryA`) or absolute (i.e. `cd /home/pi/directoryA`) paths.

pwd

The `pwd` command displays the name of the present working directory: on a Raspberry Pi, entering `pwd` will output something like `/home/pi`.

mkdir

You can use `mkdir` to create a new directory, e.g. `mkdir newDir` would create the directory `newDir` in the present working directory.

rmdir

To remove empty directories, use `rmdir`. So, for example, `rmdir oldDir` will remove the directory `oldDir` only if it is empty.

rm

The command `rm` removes the specified file (or recursively from a directory when used with `-r`). Be careful with this command: files deleted in this way are mostly gone for good!

cp

Using `cp` makes a copy of a file and places it at the specified location (this is similar to copying and pasting). For example, `cp ~/fileA /home/otherUser/` would copy the file `fileA` from your home directory to that of the user `otherUser` (assuming you have permission to copy it there). This command can either take `FILE FILE` (`cp fileA fileB`), `FILE DIR` (`cp fileA /directoryB/`) or `-r DIR DIR` (which recursively copies the contents of directories) as arguments.

mv

The `mv` command moves a file and places it at the specified location (so where `cp` performs a 'copy-paste', `mv` performs a 'cut-paste'). The usage is similar to `cp`. So `mv ~/fileA /home/otherUser/` would move the file `fileA` from your home directory to that of the user otherUser. This command can either take `FILE FILE` (`mv fileA fileB`), `FILE DIR` (`mv fileA /directoryB/`) or `DIR DIR` (`mv /directoryB /directoryC`) as arguments. This command is also useful as a method to rename files and directories after they've been created.

touch

The command `touch` sets the last modified time-stamp of the specified file(s) or creates it if it does not already exist.

cat

You can use `cat` to list the contents of file(s), e.g. `cat thisFile` will display the contents of `thisFile`. Can be used to list the contents of multiple files, i.e. `cat *.txt` will list the contents of all `.txt` files in the current directory.

head

The `head` command displays the beginning of a file. Can be used with `-n` to specify the number of lines to show (by default ten), or with `-c` to specify the number of bytes.

tail

The opposite of `head`, `tail` displays the end of a file. The starting point in the file can be specified either through `-b` for 512 byte blocks, `-c` for bytes, or `-n` for number of lines.

chmod

You would normally use `chmod` to change the permissions for a file. The `chmod` command can use symbols `u` (user that owns the file), `g` (the files group) , and `o` (other users) and the permissions `r` (read), `w` (write), and `x` (execute). Using `chmod u+x filename` will add execute permission for the owner of the file.

chown

The `chown` command changes the user and/or group that owns a file. It normally needs to be run as root using sudo e.g. `sudo chown pi:root filename` will change the owner to pi and the group to root.

ssh

`ssh` denotes the secure shell. Connect to another computer using an encrypted network connection. For more details see SSH (secure shell)

scp

The `scp` command copies a file from one computer to another using `ssh`. For more details see SCP (secure copy)

sudo

The `sudo` command enables you to run a command as a superuser, or another user. Use `sudo -s` for a superuser shell. For more details see Root user / sudo

dd

The `dd` command copies a file converting the file as specified. It is often used to copy an entire disk to a single file or back again. So, for example, `dd if=/dev/sdd of=backup.img` will create a backup image from an SD card or USB disk drive at /dev/sdd. Make sure to use the correct drive when copying an image to the SD card as it can overwrite the entire disk.

df

Use `df` to display the disk space available and used on the mounted filesystems. Use `df -h` to see the output in a human-readable format using M for MBs rather than showing number of bytes.

unzip

The `unzip` command extracts the files from a compressed zip file.

tar

Use `tar` to store or extract files from a tape archive file. It can also reduce the space required by compressing the file similar to a zip file.

To create a compressed file, use `tar -cvzf filename.tar.gz directory/` To extract the contents of a file, use `tar -xvzf filename.tar.gz`

pipes

A pipe allows the output from one command to be used as the input for another command. The pipe symbol is a vertical line `|`. For example, to only show the first ten entries of the `ls` command it can be piped through the head command `ls | head`

tree

Use the `tree` command to show a directory and all subdirectories and files indented as a tree structure.

&

Run a command in the background with `&`, freeing up the shell for future commands.

wget

Download a file from the web directly to the computer with `wget`. So `wget https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf` will download the Raspberry Pi 4 datasheet and save it as `raspberry-pi-4-datasheet.pdf`.

curl

Use `curl` to download or upload a file to/from a server. By default, it will output the file contents of the file to the screen.

man

Show the manual page for a file with `man`. To find out more, run `man man` to view the manual page of the man command.

Search

grep

Use `grep` to search inside files for certain search patterns. For example, `grep "search" *.txt` will look in all the files in the current directory ending with .txt for the string search.

The `grep` command supports regular expressions which allows special letter combinations to be included in the search.

awk

`awk` is a programming language useful for searching and manipulating text files.

find

The `find` command searches a directory and subdirectories for files matching certain patterns.

whereis

Use `whereis` to find the location of a command. It looks through standard program locations until it finds the requested command.

Networking

ping

The `ping` utility is usually used to check if communication can be made with another host. It can be used with default settings by just specifying a hostname (e.g. `ping raspberrypi.com`) or an IP address (e.g. `ping 8.8.8.8`). It can specify the number of packets to send with the `-c` flag.

nmap

`nmap` is a network exploration and scanning tool. It can return port and OS information about a host or a range of hosts. Running just `nmap` will display the options available as well as example usage.

hostname

The `hostname` command displays the current hostname of the system. A privileged (super) user can set the hostname to a new one by supplying it as an argument (e.g. `hostname new-host`).

ifconfig

Use `ifconfig` to display the network configuration details for the interfaces on the current system when run without any arguments (i.e. `ifconfig`). By supplying the command with the name of an interface (e.g. `eth0` or `lo`) you can then alter the configuration: check the manual page for more details.

Text Editors

Edit this on GitHub

On Linux, you have a choice of text editors. Some are easy-to-use but have limited functionality; others require training to use and take a long time to master, but offer incredible functionality.

Text Editors on Desktop

Text Editor

When using Raspberry Pi OS Desktop, in the accessories menu there is an option to run a Text Editor. This is a simple editor which opens in a window like a normal application. It allows use of the mouse and keyboard, and has tabs and syntax highlighting.

You can use keyboard shortcuts, such as `Ctrl + S` to save a file and `Ctrl + X` to exit.

Thonny

Thonny is a Python REPL and IDE, so you can write and edit Python code in a window and run it directly from the editor. Thonny has independent windows, and syntax highlighting, and uses Python 3.

Geany

A fast and lightweight IDE, supporting many different file types, including C/C++ and Python. It is installed by default on Raspberry Pi OS.

Text Editors in the Terminal

Nano

GNU Nano is at the easy-to-use end of command-line editors. It's installed by default, so use `nano somefile.txt` to edit a file, and keyboard shortcuts like `Ctrl + O` to save and `Ctrl + X` to exit.

Vi

Vi is a very old (c. 1976) command-line editor, which is available on most UNIX systems and is pre-installed on Raspberry Pi OS. It's succeeded by Vim (Vi Improved), which requires installation.

Unlike most editors, Vi and Vim have a number of different modes. When you open Vi with `vi somefile.txt`, you start in command mode which doesn't directly permit text entry. Press `i` to switch to insert mode in order to edit the file, and type away. To save the file you must return to command mode, so press the `Escape` key and enter `:w` (followed by `Enter`), which is the command to write the file to disk.

To search for the word 'raspberry' in a file, make sure you're in command mode (press `Escape`), then type `/raspberry` followed by `n` and `N` to flick forwards/backwards through the results.

To save and exit, enter the command `:wq`. To exit without saving, enter the command `:q!`.

Depending on your keyboard configuration, you may find your cursor keys don't work. In this case, you can use the H-J-K-L keys (which move left, down, up, and right respectively) to navigate the file in command mode.

Vim

Vim is an extension of Vi and works in much the same way, with a number of improvements. Only Vi is installed by default so to get the full features of Vim, install it with APT:

```
sudo apt install vim
```

You can edit a file in Vim with `vim somefile.txt`.

Emacs

Emacs is a GNU command-line text editor; it's powerful, extensible, and customisable. You can install it with APT:

```
sudo apt install emacs
```

You can use keyboard combination commands, such as `Ctrl + X Ctrl + S` to save and `Ctrl + X Ctrl + C` to close.

Linux Users

Edit this on GitHub

User management in Raspberry Pi OS is done on the command line. The default user is `pi`, and the password is `raspberry`. You can add users and change each user's password.

Changing your Password

Once you're logged in as the `pi` user, it is highly advisable to use the `passwd` command to change the default password to improve your Pi's security.

Enter `passwd` on the command line and press `Enter`. You'll be prompted to enter your current password to authenticate, and then asked for a new password. Press `Enter` on completion and you'll be asked to confirm it. Note that no characters will be displayed while entering your password. Once you've correctly confirmed your password, you'll be shown a success message (`passwd: password updated successfully`), and the new password will apply immediately.

If your user has `sudo` permissions, you can change another user's password with `passwd` followed by the user's username. For example, `sudo passwd bob` will allow you to set the user `bob`'s password, and then some additional optional values for the user such as their name. Just press `Enter` to skip each of these options.

Remove a User's Password

You can remove the password for the user `bob` with `sudo passwd bob -d`. Without a password the user will not be able to login to a Terminal.

NOTE

> This is useful for users that need to exist for system reasons, but you don't want it to be possible to login to the account for security reasons.

Creating a New User

You can create additional users on your Raspberry Pi OS installation with the `adduser` command.

Enter `sudo adduser bob` and you'll be prompted for a password for the new user `bob`. Leave this blank if you don't want a password.

Your Home Folder

When you create a new user, they will have a home folder in `/home/`. The `pi` user's home folder is at `/home/pi/`.

The `skel` Command

Upon creating a new user, the contents of `/etc/skel/` will be copied to the new user's home folder. You can add or modify dot-files such as the `.bashrc` in `/etc/skel/` to your requirements, and this version will be applied to new users.

Deleting a User

You can delete a user on your system with the command `userdel`. Apply the `-r` flag to remove their home folder too:

```
sudo userdel -r bob
```

Root and Sudo

Edit this on GitHub

The Linux operating system is a multi-user operating system which allows multiple users to log in and use the computer. To protect the computer (and the privacy of other users), the users' abilities are restricted.

Most users are allowed to run most programs, and to save and edit files stored in their own home folder. Normal users are not normally allowed to edit files in other users' folders or any of the system files. There's a special user in Linux known as the **superuser**, which is usually given the username `root`. The superuser has unrestricted access to the computer and can do almost anything.

The `sudo` Command

You won't normally log into the computer as `root`, but you can use the `sudo` command to provide access as the superuser. If you log into your Raspberry Pi as the `pi` user, then you're logging in as a normal user. You can run commands as the `root` user by using the `sudo` command before the program you want to run.

For example, if you want to install additional software on Raspberry Pi OS then you normally use the `apt` tool. To update the list of available software, you need to prefix the `apt` command with sudo:

```
sudo apt update
```

You can also run a superuser shell by using `sudo su`. When running commands as a superuser there's nothing to protect against mistakes that could damage the system. It's recommended that you only run commands as the superuser when required, and to exit a superuser shell when it's no longer needed.

The Sudo'ers List

The default `pi` user on Raspberry Pi OS is a member of the `sudo` group. This gives the ability to run commands as root when preceded by `sudo`, and to switch to the root user with `sudo su`.

To add a new user to the `sudo` group, use the `adduser` command:

```
sudo adduser bob sudo
```

Note that the user `bob` will be prompted to enter their password when they run `sudo`. This differs from the behaviour of the `pi` user, since `pi` is not prompted for their password. If you wish to remove the password prompt from the new user, create a custom sudoers file and place it in the `/etc/sudoers.d` directory.

1. Create the file using `sudo visudo /etc/sudoers.d/010_bob-nopasswd`.
2. Insert the following contents on a single line: `bob ALL=(ALL) NOPASSWD: ALL`
3. Save the file and exit.

Once you have exited the editor, the file will be checked for any syntax errors. If no errors were detected, the file will be saved and you will be returned to the shell prompt. If errors were detected, you will be asked 'what now?' Press the 'enter' key on your keyboard: this will bring up a list of options. You will probably want to use 'e' for '(e)dit sudoers file again', so you can edit the file and fix the problem.

NOTE

Choosing option 'Q' will save the file with any syntax errors still in place, which makes it impossible for *any* user to use the sudo command.

NOTE

It is standard practice on Linux to have the user prompted for their password when they run `sudo`, since it makes the system slightly more secure.

The `.bashrc ` File

Edit this on GitHub

In your home folder you will find a hidden file called `.bashrc` which contains some user configuration options. You can edit this file to suit your needs. Changes made in this file will be actioned the next time a terminal is opened, since that is when the `.bashrc` file is read.

If you want your changes to take place in your current terminal, you can use either `source ~/.bashrc` or `exec bash`. These actually do slightly different things: the former simply re-executes the `.bashrc` file, which may result in undesirable changes to things like the path, the latter replaces the current shell with a new bash shell, which resets the shell back to the state at login, throwing away any shell variables you may have set. Choose whichever is most appropriate.

Some useful adaptions are provided for you; some of these are commented out with a by default. To enable them, remove the and they will be active next time you boot your Pi or start a new terminal.

For example, some `ls` aliases:

```
alias ls='ls --color=auto'
#alias dir='dir --color=auto'
#alias vdir='vdir --color=auto'

alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
```

Aliases like these are provided to help users of other systems like Microsoft Windows (`dir` is the `ls` of DOS/Windows). Others are to add colour to the output of commands like `ls` and `grep` by default.

More variations of `ls` are also provided:

```
# some more ls aliases
#alias ll='ls -l'
#alias la='ls -A'
#alias l='ls -CF'
```

Ubuntu users may be familiar with these as they are provided by default on that distribution. Uncomment these lines to have access to these aliases in future.

The `.bash_aliases` File

`.bashrc` also contains a reference to a `.bash_aliases` file, which does not exist by default. You can add it to provide a handy way of keeping all your aliases in a separate file.

```
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
```

The `if` statement here checks the file exists before including it.

Then you just create the file `.bash_aliases` and add more aliases like so:

```
alias gs='git status'
```

You can add other things directly to this file, or to another and include that file like the `.bash_aliases` example above.

Shell Scripts

Edit this on GitHub

Commands can be combined together in a file which can then be executed. As an example, copy the following into your favourite text editor:

```
#!/usr/bin/bash

while :
do
echo Raspberry Pi!
done
```

Save this with the name `fun-script`.

Before you can run it you must first make it executable; this can be done by using the change mode command `chmod`. Each file and directory has its own set of permissions that dictate what a user can and can't do to it. In this case, by running the command `chmod +x fun-script`, the file `fun-script` will now be executable.

You can then run it by typing `./fun-script` (assuming that it's in your current directory).

This script infinitely loops and prints `Raspberry Pi!`; to stop it, press `Ctrl + C`. This kills any command that's currently being run in the terminal.

Scheduling Tasks with Cron

Edit this on GitHub

Cron is a tool for configuring scheduled tasks on Unix systems. It is used to schedule commands or scripts to run periodically and at fixed intervals. Tasks range from backing up the user's home folders every day at midnight, to logging CPU information every hour.

The command `crontab` (cron table) is used to edit the list of scheduled tasks in operation, and is done on a per-user basis; each user (including `root`) has their own `crontab`.

Editing the `crontab` File

Run `crontab` with the `-e` flag to edit the cron table:

```
crontab -e
```

NOTE

> The first time you run `crontab` you'll be prompted to select an editor; if you are not sure which one to use, choose `nano` by pressing `Enter`.

Adding a Scheduled Task

The layout for a cron entry is made up of six components: minute, hour, day of month, month of year, day of week, and the command to be executed.

```
# m h  dom mon dow   command
```

```
# * * * * *  command to execute
# ┬ ┬ ┬ ┬ ┬
# │ │ │ │ │
# │ │ │ │ └──── day of week (0 - 7) (0 to 6 are Sunday to Saturday, or use nam
es; 7 is Sunday, the same as 0)
# │ │ │ └─────── month (1 - 12)
# │ │ └────────── day of month (1 - 31)
# │ └─────────── hour (0 - 23)
# └──────────── min (0 - 59)
```

For example:

```
0 0 * * *  /home/pi/backup.sh
```

This cron entry would run the `backup.sh` script every day at midnight.

Viewing Scheduled Tasks

View your currently saved scheduled tasks with:

```
crontab -l
```

Erase scheduled tasks

Delete all currently scheduled tasks:

```
crontab -r
```

Running a Task on Reboot

To run a command every time the Raspberry Pi starts up, write `@reboot` instead of the time and date. For example:

```
@reboot python /home/pi/myscript.py
```

This will run your Python script every time the Raspberry Pi reboots. If you want your command to be run in the background while the Raspberry Pi continues starting up, add a space and `&` at the end of the line, like this:

```
@reboot python /home/pi/myscript.py &
```

The `systemd` Daemon

Edit this on GitHub

In order to have a command or program run when the Pi boots, you can add it as a service. Once this is done, you can start/stop enable/disable from the linux prompt.

Creating a Service

On your Pi, create a `.service` file for your service, for example: `myscript.service`

```
[Unit]
Description=My service
After=network.target

[Service]
ExecStart=/usr/bin/python3 -u main.py
WorkingDirectory=/home/pi/myscript
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi
```

```
[Install]
WantedBy=multi-user.target
```

So in this instance, the service would run Python 3 from our working directory `/home/pi/myscript` which contains our python program to run `main.py`. But you are not limited to Python programs: simply change the `ExecStart` line to be the command to start any program or script that you want running from booting.

Copy this file into `/etc/systemd/system` as root, for example:

```
sudo cp myscript.service /etc/systemd/system/myscript.service
```

Once this has been copied, you have to inform `systemd` that a new service has been added. This is done with the following command:

```
sudo systemctl daemon-reload
```

Now you can attempt to start the service using the following command:

```
sudo systemctl start myscript.service
```

Stop it using following command:

```
sudo systemctl stop myscript.service
```

When you are happy that this starts and stops your app, you can have it start automatically on reboot by using this command:

```
sudo systemctl enable myscript.service
```

The `systemctl` command can also be used to restart the service or disable it from boot up.

NOTE

The order in which things are started is based on their dependencies — this particular script should start fairly late in the boot process, after a network is available (see the After section). You can configure different dependencies and orders based on your requirements.