
Geospatial Linux Documentation

Samuel Bowers

Mar 07, 2019

Contents

1	Aims	3
2	Instructions	5
3	Contents	7
3.1	The Linux command line	7
3.2	The Geospatial Data Abstraction Library	18
3.3	Refresher for Linux	29
3.4	To be continued...	36
4	Search	39

Welcome to this very short introduction to using Linux for geospatial applications.

We will start with the very basics of the Linux, and then look at how Linux command line tools can be used to process, analyse and interpret geospatial data.

Our aim is to introduce you to some of the fundamental concepts of the Linux command line. This is **not** designed to be a thorough tutorial, but will introduce you to the most important concepts to be able to get started using Linux.

We will aim to build up this documentation over time to give you a record of the methods we've covered. Take note of the URL, these notes may be something that you'll refer back to in future.

CHAPTER 1

Aims

By the end of this tutorial, we hope that you will be able to:

- Navigate the Linux command line
- Execute programs from the Linux command line
- Use some key GDAL command line tools to manipulate geospatial data
- Understand how to ssh to other PCs or servers

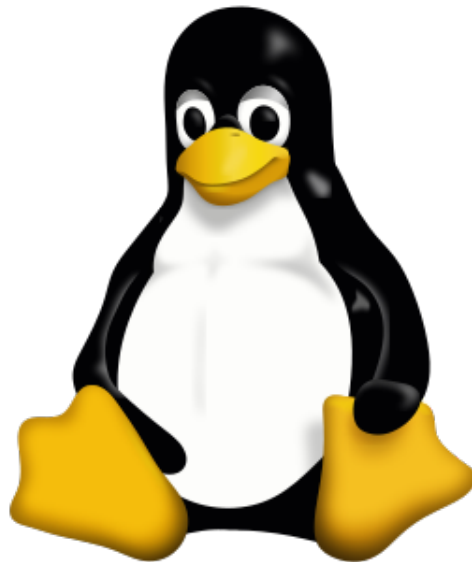
CHAPTER 2

Instructions

- Work through each input, and make it run on your own PC
- Do not copy-paste. You'll learn a lot by entering each command manually
- When you encounter an error, in the first instance try and work out the issue yourself. If you're still stuck, do ask for some pointers
- Don't worry about making mistakes, you can't break Linux
- Using Google to help with exercises is encouraged
- Go at your own pace; we don't expect you to reach the end of this tutorial today.

Note: For those of you with some previous Linux command line experience, we recommend skipping forwards to the [*Refresher for Linux*](#) page.

3.1 The Linux command line



3.1.1 Why Linux?

Linux is an operating system, the set of system software that manages computer hardware and software. You'll be familiar with other common operating systems, such as Windows, macOS, and Android.

In this tutorial we'll refer to Linux throughout, but be aware that Linux is an offshoot of the UNIX operating system. For our purposes, we can consider the two terms interchangeable.

There are lots of very good reasons to use a Linux operating system. For the purposes of geospatial data processing the main reasons are:

1. It's open-source

Linux is free to use and open-source. You own it 100%, and you are not beholden to a commercial company.

2. It's great for science

Linux hosts a large ecosystem of high-quality open-source tools, many of which are equal to or better than commercial alternatives. In this tutorial we'll look at [GDAL](#), an unrivalled library for processing geospatial data which is freely available on Linux systems.

3. Distributed and large-scale computing

Linux systems are particularly suitable for tasks involving the processing of large volumes of data.

You may be used to processing data on a local desktop machine, and know the limitations of this. Linux allows multiple users to share a single powerful machine, or a single user to operate multiple smaller machines. It's very common for a Linux user to be actually operating a computer in a different location. For example, files may be stored on a remote server and processing run at the same location as the data. This machine can easily be on another continent operating as a 'cloud platform'. This paradigm means that some operations that are limiting in remote sensing (e.g. data download and storage) become less of an issue.

4. It's easy to use

Seriously, it is! Linux does come with a steep learning curve, but once you get used to it, it really will improve your efficiency. For flexibility and large-scale processing tasks it's hard to beat.

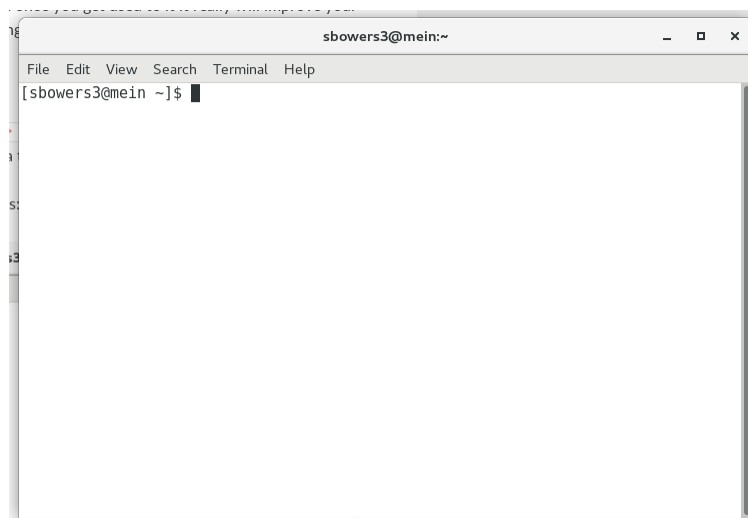
3.1.2 The Linux terminal

There are many different [distributions](#) of Linux, most of which come with a graphical user interface (GUI). Whilst most of the operations we'll cover here can be performed using the GUI, Linux is at its most powerful when operated through the command line.

First we'll open up a new terminal window.

How to open the terminal depends on your Linux distribution. You'll usually find the terminal at Applications -> System Tools -> Terminal. Or, right click the Desktop and there is usually a shortcut to open a terminal window.

The terminal window will look something like this:



The terminal shows your username (in my case, `showers3`), the name of the computer you're connected to (`mein`), the directory you're in (`~`) and has a space for you to type commands (`()`).

Let's try to run a basic a command. Type `echo Hello World!` into ther terminal, and hit `return` to execute it:

```
[username@linuxpc ~] echo Hello World!  
Hello World!
```

The `echo` command is very simple, it prints text to the terminal. Get used to typing commands into the terminal using the `echo` command.

3.1.3 Navigation

Where am I?

The Linux file system is based on a series of directories. In Windows, these are called 'folders'.

You can get the find out where you are with the `pwd` (print working directory) command:

```
[username@linuxpc ~] pwd  
/home/username
```

On most systems when you open a new terminal window you'll be located in your home directory. The home directory is located at `/home/username`, which can also be reached with the shortcut `~`. The home space is the part of the system that you have control over, and is the default location for file storage.

The root of the directory system is `/`, with each subsequent `/` separating directory names.

What's here?

The command `ls` (list) will tell you what files and directories are in the working directory:

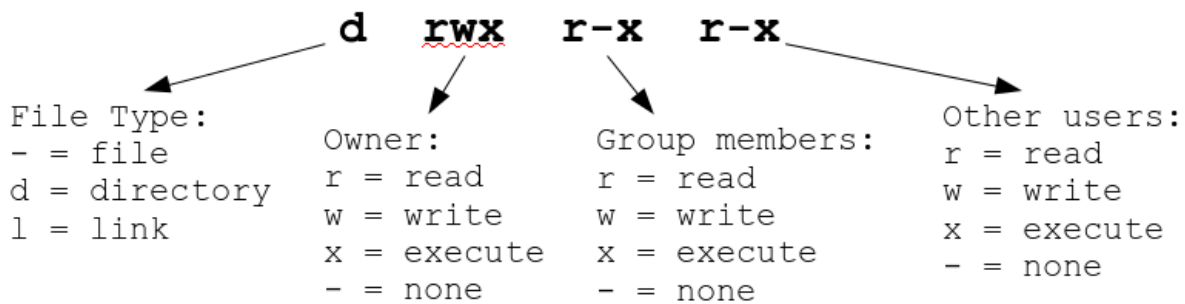
```
[username@linuxpc ~] ls  
Desktop  
Documents  
Downloads  
...
```

To show more details about the contents of the working directory, you can use the `-lh` (long, human readable) flags:

```
[username@linuxpc ~] ls -lh  
drwxr-xr-x  4 username username 4.0K Sep  7 11:32 Desktop  
drwxr-xr-x 12 username username 4.0K Sep 12 11:35 Documents  
drwxr-xr-x  9 username username 12K Sep 13 13:14 Downloads  
...
```

This additionally gives you infomrmation the about file permissions (e.g. `drwxr-xr-x`), file ownder (username), the file size (e.g. `4.0K`), modification date and time (e.g. `Sep 12 11:35`).

It's worth taking a little time to understand file permissions. This string is split into four sections:



The first character gives the file type, and the remaining nine characters the read, write and execute permissions for the owner, group members and others users.

Moving around

You can move into other directories of the system with the `cd` (change directory) command:

```
[username@linuxpc ~] cd Documents
[username@linuxpc Documents] pwd
/home/username/Documents
```

You can move back up the directory tree with `..`:

```
[username@linuxpc Documents] cd ..
[username@linuxpc ~] pwd
/home/username
```

Similarly, you can move two steps up the directory tree with `../..`:

```
[username@linuxpc ~] cd ../../
[username@linuxpc /] pwd
/
```

With the above commands we've been using 'relative' file paths. They refer to a location that relative to the present working directory. We can also use absolute pathnames which do not take account of the working directory. Absolute paths always start from the root `/` directory, for example:

```
[username@linuxpc ~] cd /home/username/Documents
```

You can move back to the home directory with the shortcut `'~'`:

```
[username@linuxpc /] cd ~
[username@linuxpc ~] pwd
/home/username
```

Exercises

1. What happens when you run the command `cd .`? What location do you think that `.` refers to?
2. Who has permission to read and write file with the following permissions:

- (a) `-rwx-----`
- (b) `-rwxr--r--`
- (c) `-rwxrwxrwx`

3. Are the following relative or absolute paths?

- (a) `Documents/DATA/`
- (b) `/home/username`
- (c) `/`
- (d) `~/Documents`

3.1.4 Making and moving files

Manipulating files

First, we'll make a test file to play with using the command `touch`:

```
[username@linuxpc ~] cd ~
[username@linuxpc ~] touch file1
[username@linuxpc ~] ls
...
file1
...
```

We can also make a new directory using the command `mkdir` (make directory):

```
[username@linuxpc ~] mkdir directory1
[username@linuxpc ~] ls
...
directory1
file1
...
```

We can move our file into the directory using the `mv` command:

```
[username@linuxpc ~] mv file1 directory1
```

Change directory to `directory1` and verify that `file1` is there.

We can use the `cp` command to copy a file:

```
[username@linuxpc directory1] cp file1 file2
[username@linuxpc directory1] ls
file1
file2
```

Files can be deleted with the `rm` (remove) command:

```
[username@linuxpc directory1] rm file2
```

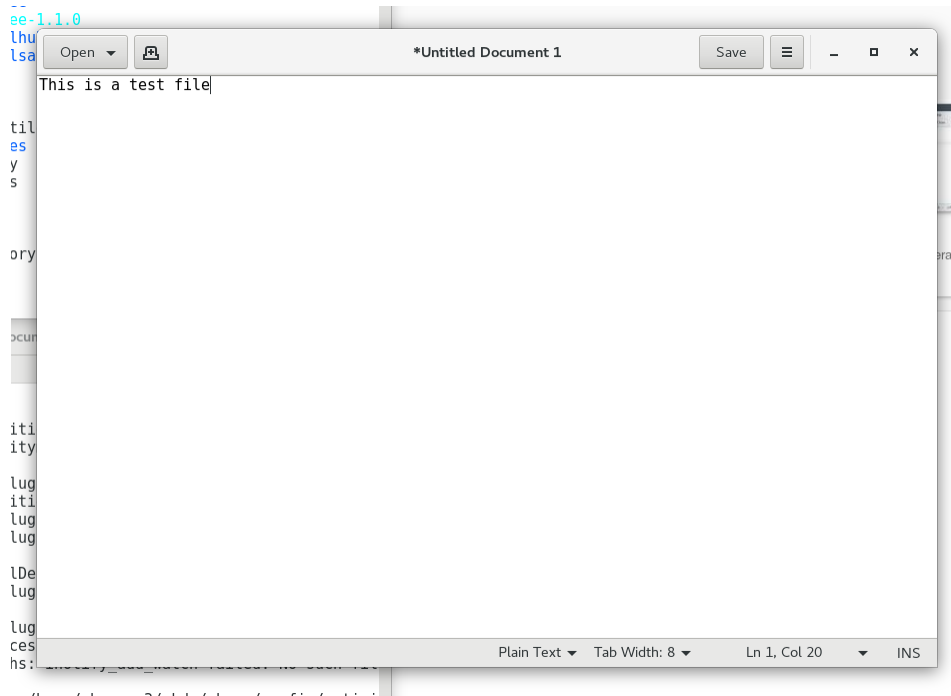
We can run similar commands to `cp` and `rm` entire directories. Note in these cases we need to use the `-r` flag, which stands for recursive. This means perform the command on the directory and all its contents:

```
[username@linuxpc directory1] cd ..  
[username@linuxpc ~] cp -r directory1 directory2  
[username@linuxpc ~] rm -r directory2
```

Creating a text file

We can launch programs from the command line. A common example of this will be to create text files. There exist a lot of text editors for this purposes (e.g. vim, emacs, kate), here we'll use gedit. Use gedit to create a new file called newfile.txt and save it to the home directory. Close gedit when you're done.:

```
[username@linuxpc ~] gedit
```



To edit newfile.txt in gedit, you can call gedit with the filename::

```
[username@linuxpc ~] gedit newfile.txt
```

Changing file permissions

Recall that you can use `ls -l` to view file permissions. Imagine we wanted to give other users permission to read our text file. For this we can use the command `chmod` (change mode).

To use `chmod` we need to specify three options relating to (i) what person or group, (ii) whether the permission should be added or removed, and (iii) which permission should be changed. These are encoded as:

Who?	What?	Which?
u = owner	+ add permission	r = read
g = group	- remove permission	w = write
o = others		x = execute
a = all		

For example, to give other users permission to read our file:

```
[username@linuxpc ~] chmod o+r newfile.txt
```

In this case `o` refers to others, `+` to add the specified mode, and `r` to read-access.

Exercise: ASCII art

Early computers and printers lacked the ability to render graphics, and were limited to text outputs. The result was the use of text characters to build images. For example:

```
_ {v}_      (v)      ('>      ( )
 /-\      //-\\      /V\      // \\\
(\_/_/      (\_/_/      <(_      (\=/)
 ^ ^      ^ ^      ~ ~      ~ ~
By: Axel Poque aka apx
```

Here we'll use the Linux terminal to follow in the footsteps of these computer art pioneers:

1. Create a new directory in your home space (`~`) named `ascii_art/`.
2. Open a new text file called `art.txt`
3. Draw a (quick!) ASCII picture, and save it to the directory `ascii_art/`
4. Give read permission to all users to your file

You can display your image in the terminal with the command:

```
cat ascii_art/art.txt

 | \_/_/ |
 / @ @ \
( > ° < )
`>>x<<`
 /  O  \
```

3.1.5 Running programs

The command line can be used to execute commands or scripts. You've already run a few commands from the command line (`ls`, `pwd`, `gedit`, `chmod`, `cat`), so you already know how to execute programs

Simple programs

Some simple programs can be executed with a single word. What do the following programs do?:

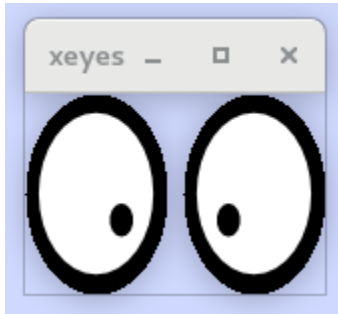
```
[username@linuxpc ~] whoami
[username@linuxpc ~] fortune
[username@linuxpc ~] xeyes
```

Programs with options

Lots of programs have options, which we can modify with `--flags`.

Let's take the `xeyes` program as an example. By default, `xeyes` operates as follows:

```
[username@linuxpc ~] xeyes
```



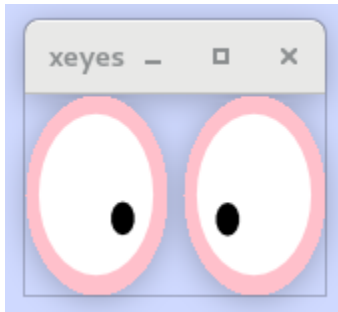
We can change the colour of the eyes using `-fg` (foreground colour) and choosing a colour:

```
[username@linuxpc ~] xeyes -fg green
```



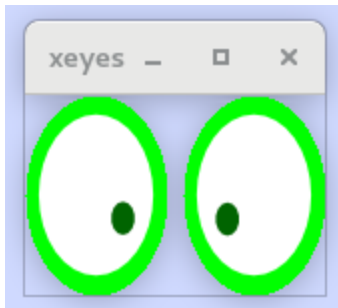
Similarly, with the option `-outline` we can choose an outline colour:

```
[username@linuxpc ~] xeyes -outline pink
```



If we wish, we can specify multiple options:

```
[username@linuxpc ~] xeyes -fg darkgreen -outline gren
```



Getting help

Flags are a very common way of specifying input options. But how do you know what flags exist?

There is a command called `man` (manual). We can use it to get help for most command line programs. For the example of `xeyes`:

```
[username@linuxpc ~] man xeyes
XEYES(1)                      General Commands Manual                      XEYES(1)

NAME
    xeyes - a follow the mouse X demo

SYNOPSIS
    xeyes [-option ...]

DESCRIPTION
    Xeyes watches what you do and reports to the Boss.

OPTIONS
    -fg foreground color
        choose a different color for the pupil of the eyes.

    -bg background color
        choose a different color for the background.

    -outline outline color
        choose a different color for the outline of the eyes.

    -center center color
        choose a different color for the center of the eyes.

    ...
```

Exercise

1. Using `xeyes` and the instructions given by `man`, can you produce a set of eyes that look like this?



1. Using `man`, can you work out how to `ls` only directories?
2. What does `man man` do?

3.1.6 Shortcuts and tricks

Wildcards

Often we will want to refer to multiple filenames or directories. We do this with the wildcard symbols `*` and `?`. The `?` a single character, and the `*` symbol refers to zero or more characters.

For example, we might want to list everything beginning with `D` in the home directory:

```
[username@linuxpc ~] ls -l D*
drwxr-xr-x  4 username username 4.0K Sep  7 11:32 Desktop
drwxr-xr-x 12 username username 4.0K Sep 12 11:35 Documents
drwxr-xr-x  9 username username 12K Sep 13 13:14 Downloads
```

We'll return to using wildcard symbols later.

Auto complete

We can use the `tab` key to auto complete commands in the terminal. For example, if we wanted to `cd` to the `Documents` directory we could start with:

```
[username@linuxpc ~] cd Doc
```

... hit `tab`, and as there are no other directories fitting that pattern, the field will autofill to `Documents`:

```
[username@linuxpc ~] cd Documents/
```

Where there is more than one possible program, directory, or files, the `tab` key will display possible options:

```
[username@linuxpc ~] cd Do
Documents/ Downloads/
```

Fill in more letters and press `tab` again to proceed.

You'll find that use of the `tab` key to auto complete will speed up navigation around the command line substantially.

Get previous commands

You can use the `up` key to scroll through a command history. This will save you having to re-type out commands repeatedly. Once you find the command you're looking for, you can edit it with the `left` and `right` keys. Give it a try!

3.1.7 Remote access

Within Linux

Note: These instructions are specific to the University of Edinburgh School of Geosciences, and are very unlikely to work elsewhere. If not working at The School of Geosciences you'll need to replace the destination to connect to another PC.

One of the strengths of linux is that you can control computers remotely. We can do this using `ssh`. In The School of Geosciences we have two large servers that can cope with large jobs named `burn` and `achray`. To connect to these servers, use the following command:

```
[username@linuxpc ~] ssh burn
-----
Unauthorised access is a criminal offence under The Computer Misuse Act 1990.
      If you are not an authorised user, disconnect NOW.
-----
Last login: Thu Sep 13 16:11:17 2018
[username@burn ~]$
```

You're now controlling a remote PC. To log out and return to the local machine, type `logout`:

```
[username@burn ~]$ logout
Connection to burn closed.
[username@linuxpc ~]$
```

If you want to use graphical programs, such as `gedit` or `QGIS`, when you `ssh` you'll need to specify the option `-X`:

```
[username@linuxpc ~] ssh -X burn
```

You can test whether you have access to graphical windows (known as X11) by trying to open a graphical program (e.g. `gedit`, `xeyes`).

From Windows or Mac

This is more complicated, but you can connect to a remote Linux server from a Windows or Mac PC. Exactly how this works will depend on the remote server, but to make a start you'll need access to a Linux terminal.

If you're on a Mac, you'll find the command line under Applications -> Utilities -> Terminal.

If you're on Windows, you can also get a terminal window when connected to a remote PC, but you'll need to download an ssh client. For example, 'PuTTY' _

3.1.8 Summary

We learned how to use the following commands. There are many thousands of other commands in Linux which you'll learn over time, but these are the main commands you'll need to navigate and manipulate files on the command line.

- `echo`: Print to the terminal
- `pwd`: Print working directory.
- `ls`: List contents of working directory
- `cd`: Change directory
- `mkdir`: Make directory
- `mv`: Move
- `cp`: Copy
- `rm`: Remove
- `chmod`: Change file permissions
- `gedit`: Create a text file in the gedit editor
- `man`: View manual
- `ssh`: Connect to another PC

3.2 The Geospatial Data Abstraction Library

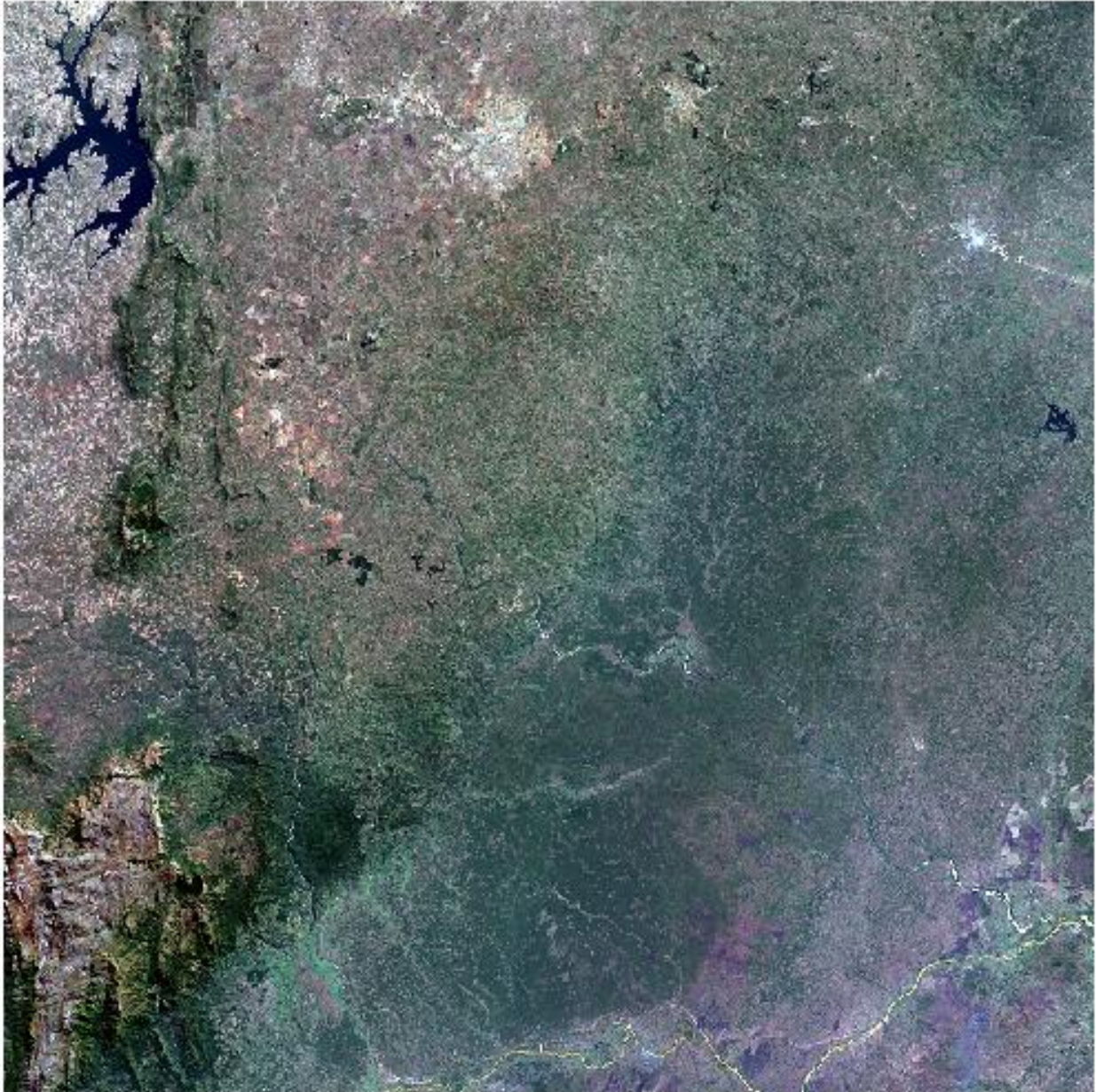
3.2.1 What is GDAL?

The Geospatial Data Abstraction Library (GDAL) is a set of tools for processing geospatial data on the command line. For bulk-processing of large satellite datasets, it's an indispensable tool.

3.2.2 Setting up test data

Getting Sentinel-2 data

We'll use data from Sentinel-2 as an example dataset here. The tile we'll use is `36KWD` imaged on the 2nd August 2018, which covers an area of Central Mozambique:



You can download this image [here](#). Make sure that you've first signed up to the [Copernicus Open Access Hub](#).

Save the Sentinel-2 file to an appropriate location. Once downloaded, `cd` to the file's location and `unzip` it:

```
cd download_location/  
unzip S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.zip
```

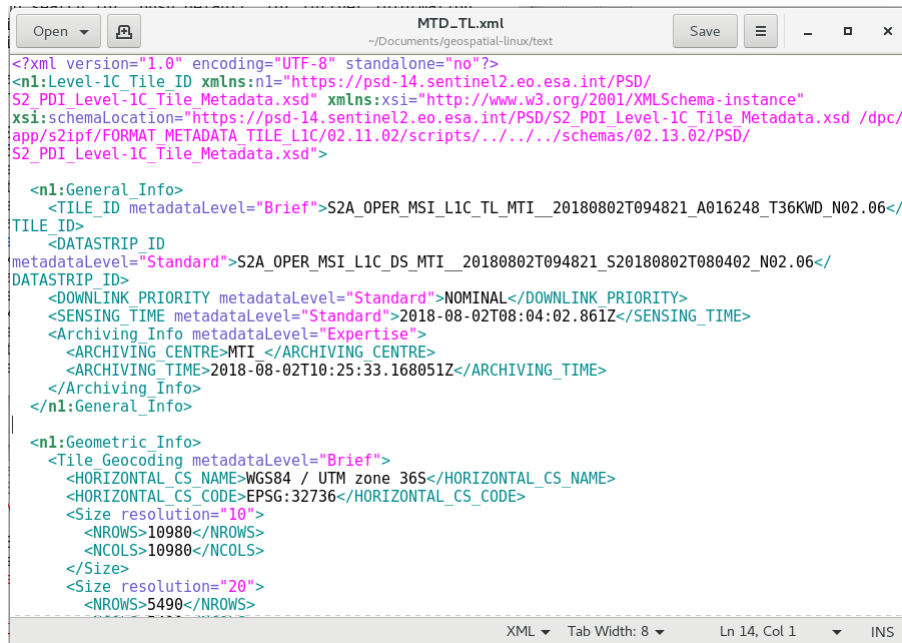
Note: Feel free to download a different Sentinel-2 image that covers your area of interest. This will make the tutorial tougher as you'll need to modify filenames and coordinates where appropriate. If you do decide to do this, we recommend finding a cloud-free Sentinel-2 image acquired after 6th December 2016, as prior to this date Sentinel-2 data have a slightly different file format. A good resource for downloading Sentinel-2 data is the Sentinel Hub [EO Browser](#).

Sentinel-2 data format

Data from Sentinel-2 are delivered in the tricky-to-understand .SAFE format. Each .SAFE file is actually a directory, containing imagery and metadata. The .SAFE file can be navigated in the same manner as any other directory.

Here we'll just aim to understand the most important elements:

- S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.SAFE/MTD_MSIL1C.xml: An XML file containing metadata including acquisition time and quality assurance information. You can examine this data in a text editor such as gedit:

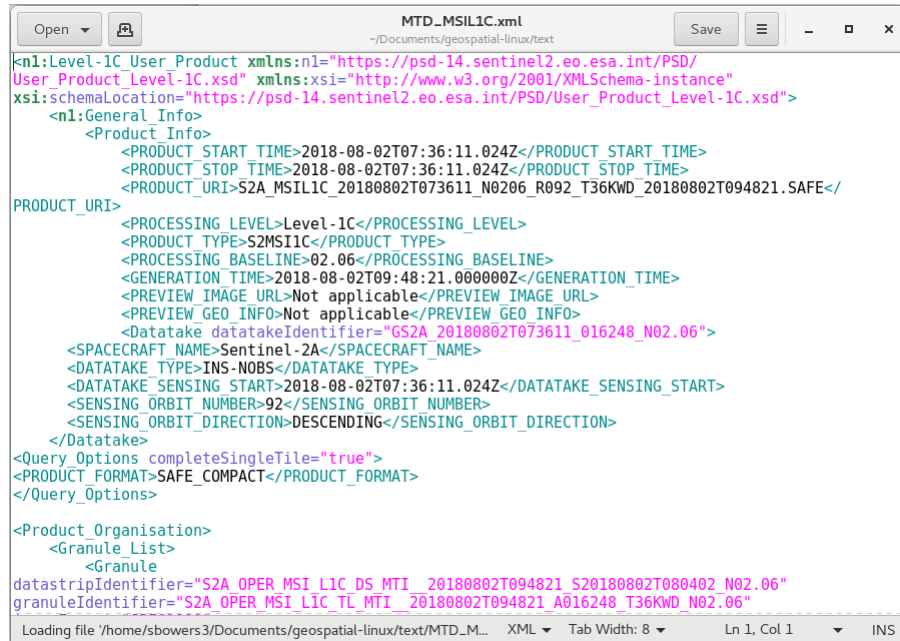


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<n1:Level-1C Tile_ID xmlns:n1="https://psd-14.sentinel2.eo.esa.int/PSD/
S2_PDI_Level-1C Tile_Metadata.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://psd-14.sentinel2.eo.esa.int/PSD/S2_PDI_Level-1C Tile_Metadata.xsd /dpc/
app/s2ipf/FORMAT_METADATA_TILE_L1C/02.11.02/scripts/../../schemas/02.13.02/PSD/
S2_PDI_Level-1C Tile_Metadata.xsd">

  <n1:General_Info>
    <TILE_ID metadataLevel="Brief">S2A_OPER_MSI_L1C_TL_MTI_20180802T094821_A016248_T36KWD_N02.06</
TILE_ID>
    <DATASTRIP_ID
metadataLevel="Standard">S2A_OPER_MSI_L1C_DS_MTI_20180802T094821_S20180802T080402_N02.06</
DATASTRIP_ID>
    <DOWNLINK_PRIORITY metadataLevel="Standard">NOMINAL</DOWNLINK_PRIORITY>
    <SENSING_TIME metadataLevel="Standard">2018-08-02T08:04:02.861Z</SENSING_TIME>
    <Archiving_Info metadataLevel="Expertise">
      <ARCHIVING_CENTRE>MTI </ARCHIVING_CENTRE>
      <ARCHIVING_TIME>2018-08-02T10:25:33.168051Z</ARCHIVING_TIME>
    </Archiving_Info>
  </n1:General_Info>

  <n1:Geometric_Info>
    <Tile_Geocoding metadataLevel="Brief">
      <HORIZONTAL_CS_NAME>WGS84 / UTM zone 36S</HORIZONTAL_CS_NAME>
      <HORIZONTAL_CS_CODE>EPSG:32736</HORIZONTAL_CS_CODE>
      <Size resolution="10">
        <NROWS>10980</NROWS>
        <NCOLS>10980</NCOLS>
      </Size>
      <Size resolution="20">
        <NROWS>5490</NROWS>
      </Size>
    </Tile_Geocoding>
  </n1:Geometric_Info>
</n1:Level-1C Tile_ID>
</n1:Level-1C Tile_Metadata>
```

- S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.SAFE/GRANULE/: Sentinel-2 data are distributed as a fixed set of tiles or 'granules' (see image below). The GRANULE directory will contain one or more (before December 2016) directories for each of the tiles contained in the file.
- S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.SAFE/GRANULE/L1C_T36KWD_A016248_20180802T080402/MTD_TL.xml: An XML file containing metadata for the Sentinel-2 granule, including projection, resolution, extent, and quality assurance information. You can examine this data in a text editor such as gedit:



- S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.SAFE/GRANULE/L1C_T36KWD_A016248_20180802T080402/IMG_DATA/: The actual image data, in JPEG2000(.jp2) format. There is one file for each of the Sentinel-2 bands

Sentinel-2 data are distributed in two forms:

- Level 1C: Top Of Atmosphere reflectance without a cloud mask.
- Level 2A: Bottom Of Atmosphere reflectance and pixel classification including cloud-cover.

For the purposes of this tutorial we'll use Level 1C data, but for many remote sensing tasks level 2A data are more appropriate (e.g. time series analysis). Level 1C data can be converted to level 2A with the [sen2cor](#) program.

For those interested, there's a lot more information on the format of Sentinel-2 data on the [ESA website](#).

3.2.3 Showing image metadata

We can use GDAL to query image details using the command `gdalinfo`:

```
cd S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.SAFE/GRANULE/L1C_
→T36KWD_A016248_20180802T080402/IMG_DATA/
gdalinfo T36KWD_20180802T073611_B8A.jp2
```

```
Driver: JP2OpenJPEG/JPEG-2000 driver based on OpenJPEG library
Files: T36KWD_20180802T073611_B8A.jp2
      T36KWD_20180802T073611_B8A.jp2.aux.xml
Size is 5490, 5490
Coordinate System is:
PROJCS["WGS 84 / UTM zone 36S",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84", 6378137, 298.257223563,
        AUTHORITY["EPSG", "7030"]],
      AUTHORITY["EPSG", "6326"]],
    PRIMEM["Greenwich", 0,
      AUTHORITY["EPSG", "8901"]],
```

(continues on next page)

(continued from previous page)

```

UNIT["degree",0.0174532925199433,
    AUTHORITY["EPSG","9122"]],
AXIS["Latitude",NORTH],
AXIS["Longitude",EAST],
AUTHORITY["EPSG","4326"]],
PROJECTION["Transverse_Mercator"],
PARAMETER["latitude_of_origin",0],
PARAMETER["central_meridian",33],
PARAMETER["scale_factor",0.9996],
PARAMETER["false_easting",500000],
PARAMETER["false_northing",1000000],
UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
AXIS["Easting",EAST],
AXIS["Northing",NORTH],
AUTHORITY["EPSG","32736"]]
Origin = (499980.0000000000000000,7900000.0000000000000000)
Pixel Size = (20.000000000000000,-20.000000000000000)
Corner Coordinates:
Upper Left  ( 499980.000, 7900000.000) ( 32d59'59.32"E, 18d59'33.08"S)
Lower Left  ( 499980.000, 7790200.000) ( 32d59'59.31"E, 19d59' 5.30"S)
Upper Right ( 609780.000, 7900000.000) ( 34d 2'34.55"E, 18d59'22.50"S)
Lower Right ( 609780.000, 7790200.000) ( 34d 2'57.49"E, 19d58'54.13"S)
Center      ( 554880.000, 7845100.000) ( 33d31'22.67"E, 19d29'16.52"S)
Band 1 Block=640x640 Type=UInt16, ColorInterp=Gray
Overviews: 2745x2745, 1372x1372, 686x686, 343x343
Overviews: arbitrary
Image Structure Metadata:
  COMPRESSION=JPEG2000
  NBITS=15

```

With `gdalinfo -stats` we can also get summary statistics of the contents of the file:

```

gdalinfo T36KWD_20180802T073611_B8A.jp2

...
Band 1 Block=640x640 Type=UInt16, ColorInterp=Gray
Minimum=0.000, Maximum=10335.000, Mean=2140.866, StdDev=406.668
Overviews: 2745x2745, 1372x1372, 686x686, 343x343
Overviews: arbitrary
Metadata:
  STATISTICS_MAXIMUM=10335
  STATISTICS_MEAN=2140.8663662695
  STATISTICS_MINIMUM=0
  STATISTICS_STDDEV=406.66771229522
Image Structure Metadata:
  COMPRESSION=JPEG2000
  NBITS=15

```

Exercises

Using `gdalinfo` to answer the following questions:

1. What file format are Sentinel-2 images provided in?
2. What is the projection and extent of this Sentinel-2 tile?

3. Is the resolution of all Sentinel-2 bands the same? Why?

3.2.4 Translating rasters

`gdal_translate` is a command line tool to convert between raster data formats and extents.

Converting between formats

The JPEG2000 format of Sentinel-2 data can be difficult to work with, so we might want to convert it to a GeoTiff:

```
gdal_translate -of GTiff T36KWD_20180802T073611_B8A.jp2 T36KWD_20180802T073611_B8A.tif
```

`gdal_translate` can be used to convert between many file formats: https://gdal.org/formats_list.html.

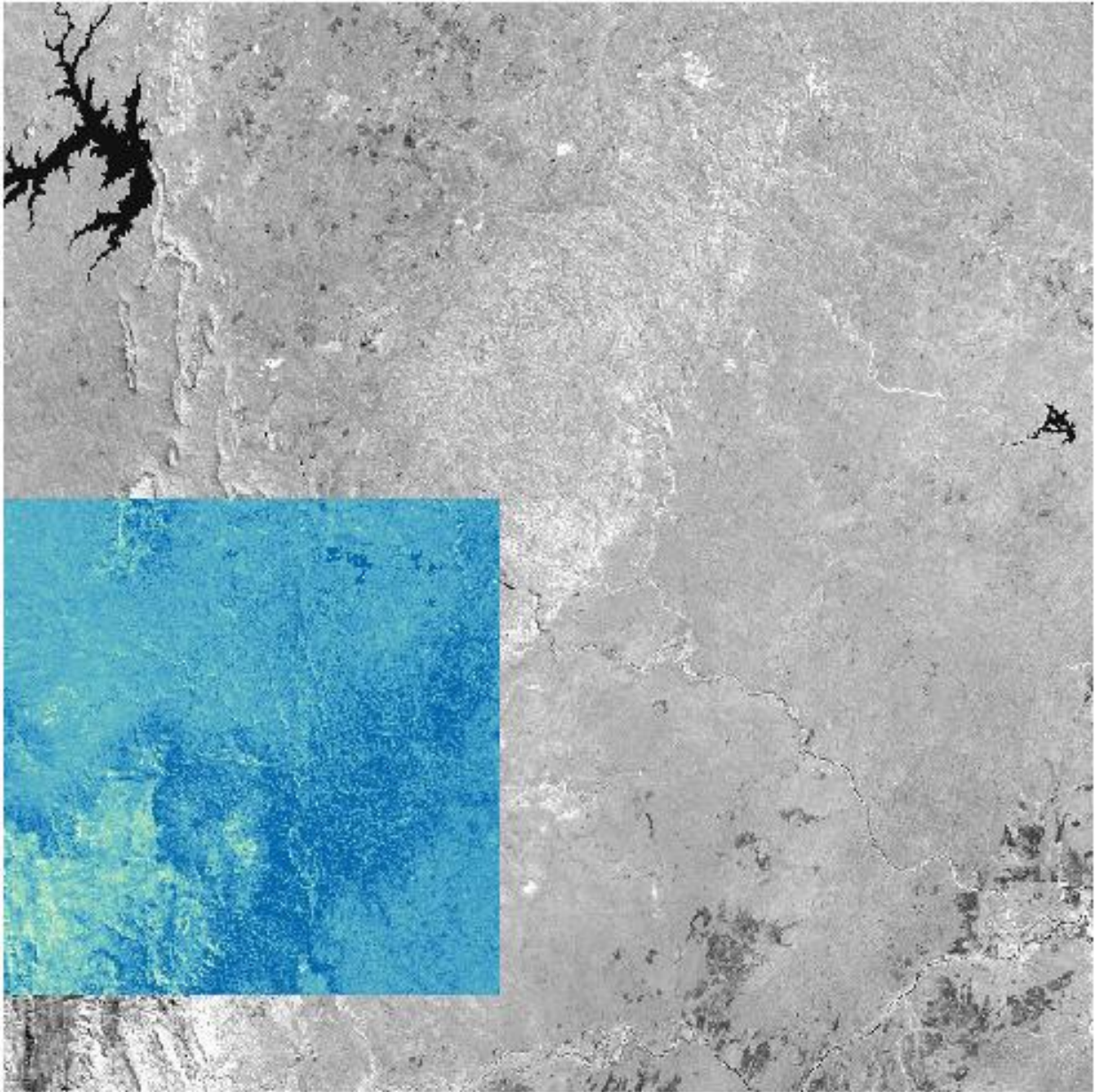
Clipping rasters

`gdal_translate` can also be used to clip a raster to a set of extents. Here, we'll produce a raster of reduced size. We'll output this as a GeoTiff.

```
gdal_translate -projwin 500000 7850000 550000 7800000 -of GTiff T36KWD_
↳20180802T073611_B8A.jp2 T36KWD_20180802T073611_B8A_clipped.tif
```

In this example, the `--projwin` flag refers to a window specified in the same projection of the data given in the format `<xmin ymax xmax ymin>`.

Take a look at the output of `gdalinfo` (`gdalinfo T36KWD_20180802T073611_B8A_clipped.tif`) to confirm it's worked. You can also open the newly produced image in QGIS: here I've displayed the reduced image in colour overlaid on the original image in greyscale:



Resampling

Spectral bands in Sentinel-2 data are not all collected as the same spatial resolution, with images variously provided at 10m, 20m and 60m. If we wanted to combine images with different resolutions (for example to build a classified image), we might want to resample one spectral band to match the resolution of another. For example, resampling our image to 60 m resolution:

```
gdal_translate -tr 60 60 -of GTiff T36KWD_20180802T073611_B8A.jp2 T36KWD_
→20180802T073611_B8A_resampled.tif
```

When resampling images, it's a good idea to carefully consider the type of resampling. By default `gdal_translate` uses nearest-neighbor resampling, but in the case such as this we might prefer to use the mean average of input pixels. We can specify this with `gdal_translate` as follows:

```
gdal_translate -tr 60 60 -of GTiff T36KWD_20180802T073611_B8A.jp2 T36KWD_
↳20180802T073611_B8A_resampledavearge.tif
```

Use `gdalinfo` and QGIS to verify that the output image has been resampled appropriately. We'll return to resampling later.

Exercises

Using `gdal_translate`:

1. **Translate the green Sentinel-2 band to an image with the following properties:** Extent: X: 600000-700000, Y: 7850000-7900000 Format: GeoTiff Resolution: 20 m
2. Can you perform this operation in a single command?

3.2.5 Building a colour composite image

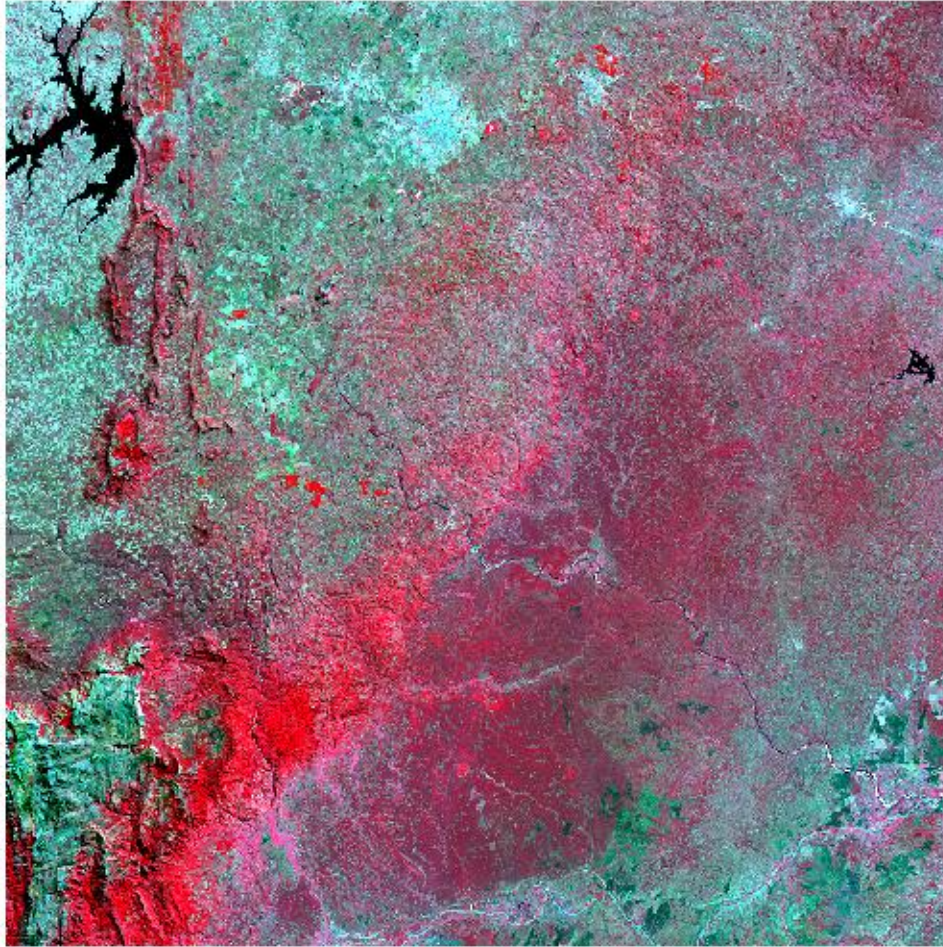
Having individual bands is useful, but often we want to view a colour image. For example, to build a red/green/blue natural colour image, we can use the following command:

```
gdalbuildvrt -separate -o T36KWD_20180802T073611_RGB.vrt T36KWD_20180802T073611_B04.
↳jp2 T36KWD_20180802T073611_B03.jp2 T36KWD_20180802T073611_B02.jp2
```

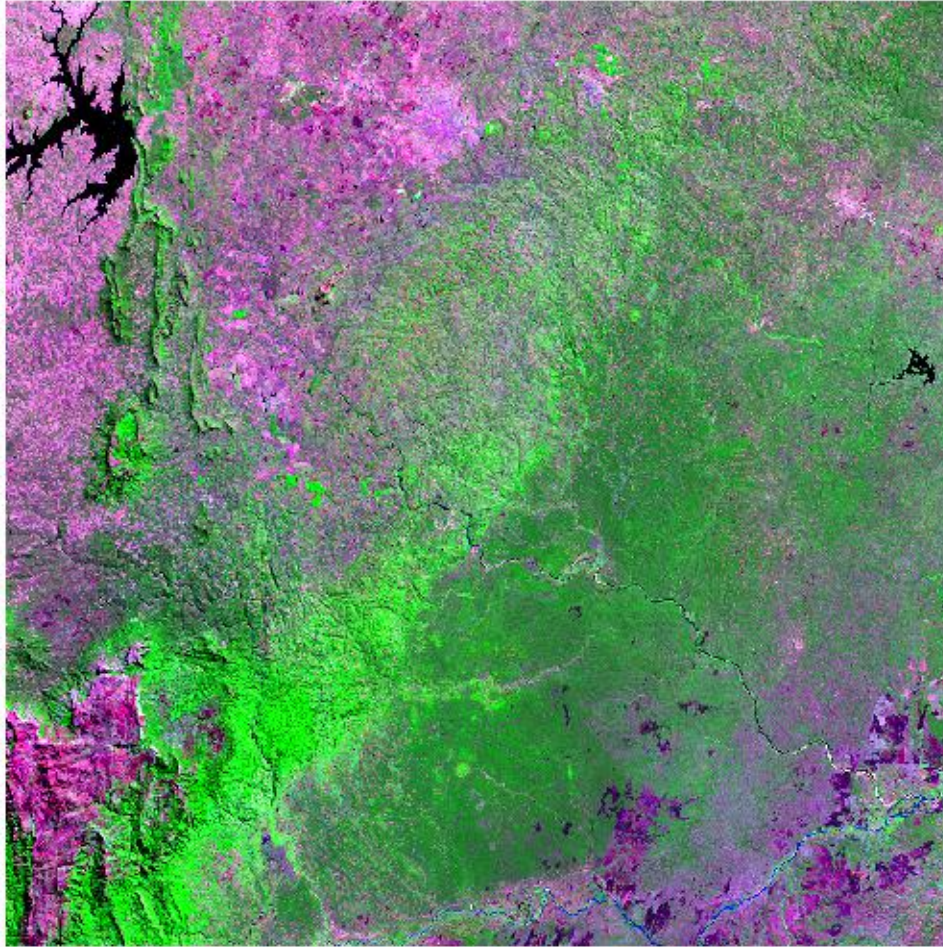
This generates a `.vrt` file, which is a 'virtual raster'. You can open this file as a raster in QGIS to view it.

Exercises

1. Generate a false colour composit .vrt file with: band 8 as red, band 4 as green, band 3 as blue. What property does this image emphasise?



2. Generate a false colour composit .vrt file with: band 12 as red, band 8 as green, band 4 as blue. What property does this image emphasise? (Hint: What do you see to the lower-right of the image?)



3.2.6 Changing raster projections

`gdalwarp` is similar to `gdal_translate`, but can also be used to convert between projections.

`gdalwarp` can replicate much of the functionality of `gdal_translate`, including converting between formats, clipping and resampling. In most cases you'll want to use `gdalwarp`, as `gdal_translate` is limited to images that are already aligned.

As a reprojection tool, GDAL can understand coordinate reference systems in a range of formats. Perhaps the simplest format is in the form of 'EPSG codes', a collection of numbers that refer to commonly used coordinate reference systems. For example, WGS84 has the code 4326, the British National Grid is 27700, and UTM36S/WGS84 is 32736. Find your EPSG code at spatialreference.org.

For example, Sentinel-2 data for the tile 36KWD are provided in UTM 36S. If we wanted to convert this to lat/lon coordinates (WGS84), we could run the following:

```
gdalwarp -t_srs EPSG:4326 -of GTiff T36KWD_20180802T073611_B8A.jp2 T36KWD_
↳ 20180802T073611_B8A_wgs84.tif
```

The `-t_srs` flag is used to specify target coordinate reference set. Use `gdalinfo` to confirm that the image has been reprojected.

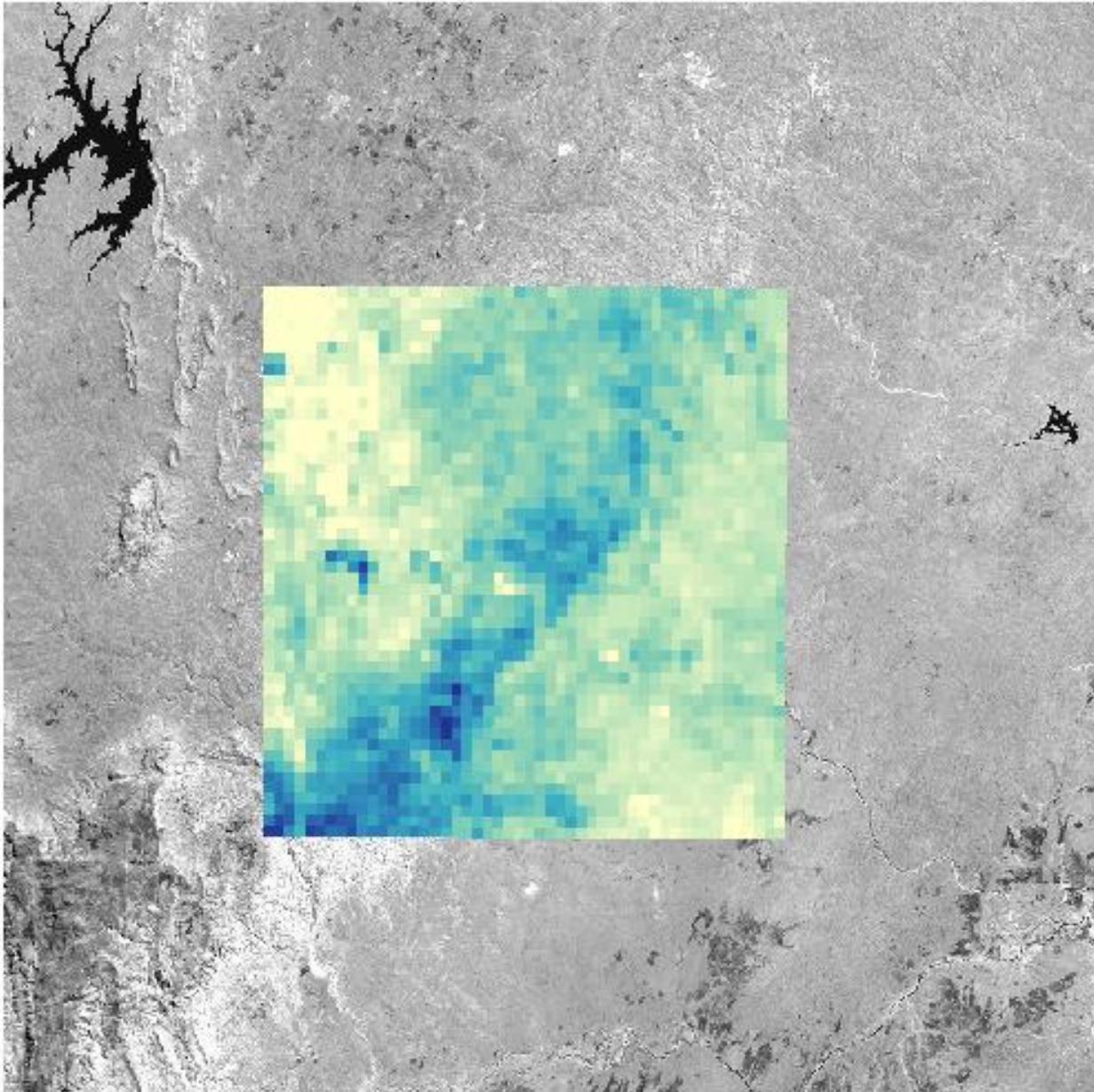
We can also clip the image with `gdalwarp` using the `-te` flag. This operates very similarly to the `--projwin` flag in `gdal_translate`, but note that the order of bounds in `gdalwarp` should be `<xmin ymin xmax ymax>`:


```
gdalwarp -t_srs EPSG:4326 -te 33.25 -19.75 33.75 -19.25 -of GTiff T36KWD_  
↳20180802T073611_B8A.jp2 T36KWD_20180802T073611_B8A_wgs84clipped.tif
```

We can combine reprojection with clipping and resampling (average), as follows:

```
gdalwarp -t_srs EPSG:4326 -te 33.25 -19.75 33.75 -19.25 -tr 0.01 0.01 -r average -of_  
↳GTiff T36KWD_20180802T073611_B8A.jp2 T36KWD_20180802T073611_B8A_  
↳wgs84clippedresampled.tif
```

See:



3.2.7 Mosaicking data

Remote sensing images such as those from Sentinel-2 are commonly provided in tiles. If we want to generate a wall-to-wall land cover map, we need to stitch these images together. Here we'll do that with two adjacent Sentinel-2 tiles.

First, download some data. As an example, we'll use data from the tile 36KWC, which is to the south of the 36KWD tile which we've been looking at up to this point. You can download the tile from ESA at this [link](#).

For this task we'll use `gdal_merge.py`. Unfortunately, `gdal_merge.py` doesn't understand JPEG2000 files, so first we'll have to convert inputs to GeoTiffs:

```
gdal_translate -of GTiff S2A_MSIL1C_20180802T073611_N0206_R092_T36KWC_20180802T094821.
↪SAFE/GRANULE/L1C_T36KWC_A016248_20180802T080402/IMG_DATA/T36KWC_20180802T073611_B8A.
↪jp2 T36KWC_20180802T073611_B8A.tif
gdal_translate -of GTiff S2A_MSIL1C_20180802T073611_N0206_R092_T36KWD_20180802T094821.
↪SAFE/GRANULE/L1C_T36KWD_A016248_20180802T080402/IMG_DATA/T36KWD_20180802T073611_B8A.
↪jp2 T36KWD_20180802T073611_B8A.tif

gdal_merge.py -o B8A_combined.tif -of GTiff T36KWC_20180802T073611_B8A.tif T36KWD_
↪20180802T073611_B8A.tif
```

Use `gdalinfo` or QGIS to confirm that you've generated a seamless mosaic.

Exercise (advanced!)

Using `gdalwarp`, `gdal_translate`, `gdal_merge.py`, and `gdalbuildvrt`, build a false colour composite image in `.vrt` format from tiles 36KWC and 36KWD. Give it the following properties:

- CRS: WGS84
- Extent: X: 33.25 to 33.75, Y: -20.25 to -19.75
- Resolution: 0.001 degrees
- Red = Band 8, Green = Band 4, Band 3 = Band 3

3.3 Refresher for Linux

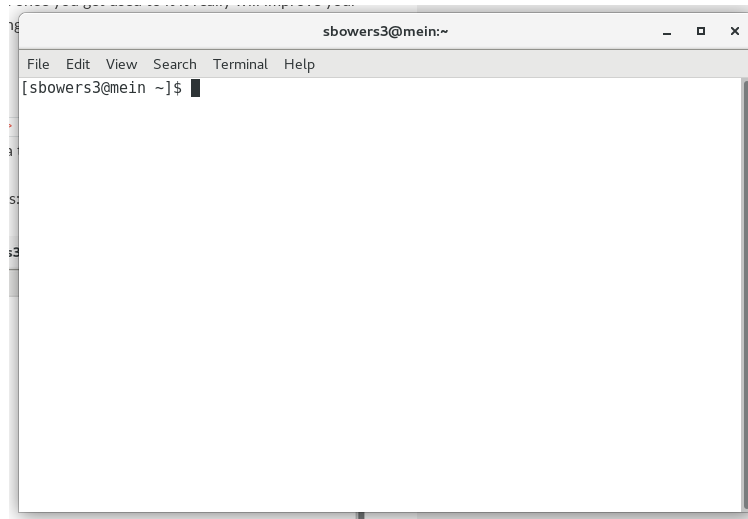
These instructions are a stripped-down version of those from *The Linux command line*, designed to refresh the memory of those who have previously used the Linux command line. If you've recently run through *The Linux command line*, you can skip this page. If this page moves too quickly, return to *The Linux command line* to run through this in more detail.

3.3.1 The Linux terminal

Recall that Linux has both a graphical interface, similar to those you'll be used with Windows or Mac operating systems, but also a more powerful text interface. We call this text interface the 'terminal'

How to open the terminal depends on your Linux distribution. You'll usually find the terminal at Applications -> System Tools -> Terminal. Or, right click the Desktop and there is usually a shortcut to open a terminal window.

The terminal window will look something like this:



The terminal shows your username (in my case, `sbowers3`), the name of the computer you're connected to (`mein`), the directory you're in (`~`) and has a space for you to type commands (`~`).

Recall that you can type commands into the terminal, and execute them with the `return` key. For instance, type `echo Hello World!` into the terminal, and execute it:

```
[username@linuxpc ~] echo Hello World!  
Hello World!
```

The `echo` command is very simple, it prints text to the terminal. There exist a large number of command line tools within Linux, take a look at some of [these](#) and try out a few that you think might be useful.

3.3.2 Navigation

Where am I?

The Linux file system is based on a series of directories. In Windows, these are called 'folders'.

You can get the find out where you are with the `pwd` (print working directory) command:

```
[username@linuxpc ~] pwd  
/home/username
```

On most systems when you open a new terminal window you'll be located in your home directory. The home directory is located at `/home/username`, which can also be reached with the shortcut `~`. The home space is the part of the system that you have control over, and is the default location for file storage.

The root of the directory system is `/`, with each subsequent `/` separating directory names.

What's here?

The command `ls` (list) will tell you what files and directories are in the working directory:

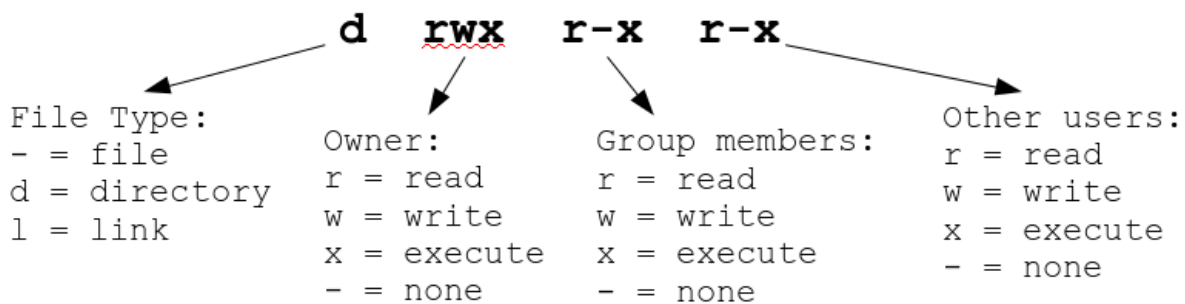
```
[username@linuxpc ~] ls
Desktop
Documents
Downloads
...
```

To show more details about the contents of the working directory, you can use the `-lh` (long, human readable) flags:

```
[username@linuxpc ~] ls -lh
drwxr-xr-x  4 username username 4.0K Sep  7 11:32 Desktop
drwxr-xr-x 12 username username 4.0K Sep 12 11:35 Documents
drwxr-xr-x  9 username username 12K Sep 13 13:14 Downloads
...
```

This additionally gives you information about file permissions (e.g. `drwxr-xr-x`), file owner (username), the file size (e.g. 4.0K), modification date and time (e.g. Sep 12 11:35).

It's worth taking a little time to understand file permissions. This string is split into four sections:



The first character gives the file type, and the remaining nine characters the read, write and execute permissions for the owner, group members and other users.

Moving around

You can move into other directories of the system with the `cd` (change directory) command:

```
[username@linuxpc ~] cd Documents
[username@linuxpc Documents] pwd
/home/username/Documents
```

You can move back up the directory tree with `..`:

```
[username@linuxpc Documents] cd ..
[username@linuxpc ~] pwd
/home/username
```

Similarly, you can move two steps up the directory tree with `../..`:

```
[username@linuxpc ~] cd ../../
[username@linuxpc /] pwd
/
```

With the above commands we've been using 'relative' file paths. They refer to a location that relative to the present working directory. We can also use absolute pathnames which do not take account of the working directory. Absolute paths always start from the root `/` directory, for example:

```
[username@linuxpc ~] cd /home/username/Documents
```

You can move back to the home directory with the shortcut `'~'`:

```
[username@linuxpc ~] cd ~
[username@linuxpc ~] pwd
/home/username
```

3.3.3 Making and moving files

Manipulating files

First, we'll make a test file to play with using the command `touch`:

```
[username@linuxpc ~] cd ~
[username@linuxpc ~] touch file1
[username@linuxpc ~] ls
...
file1
...
```

We can also make a new directory using the command `mkdir` (make directory):

```
[username@linuxpc ~] mkdir directory1
[username@linuxpc ~] ls
...
directory1
file1
...
```

We can move our file into the directory using the `mv` command:

```
[username@linuxpc ~] mv file1 directory1
```

Change directory to `directory1` and verify that `file1` is there.

We can use the `cp` command to copy a file:

```
[username@linuxpc directory1] cp file1 file2
[username@linuxpc directory1] ls
file1
file2
```

Files can be deleted with the `rm` (remove) command:

```
[username@linuxpc directory1] rm file2
```

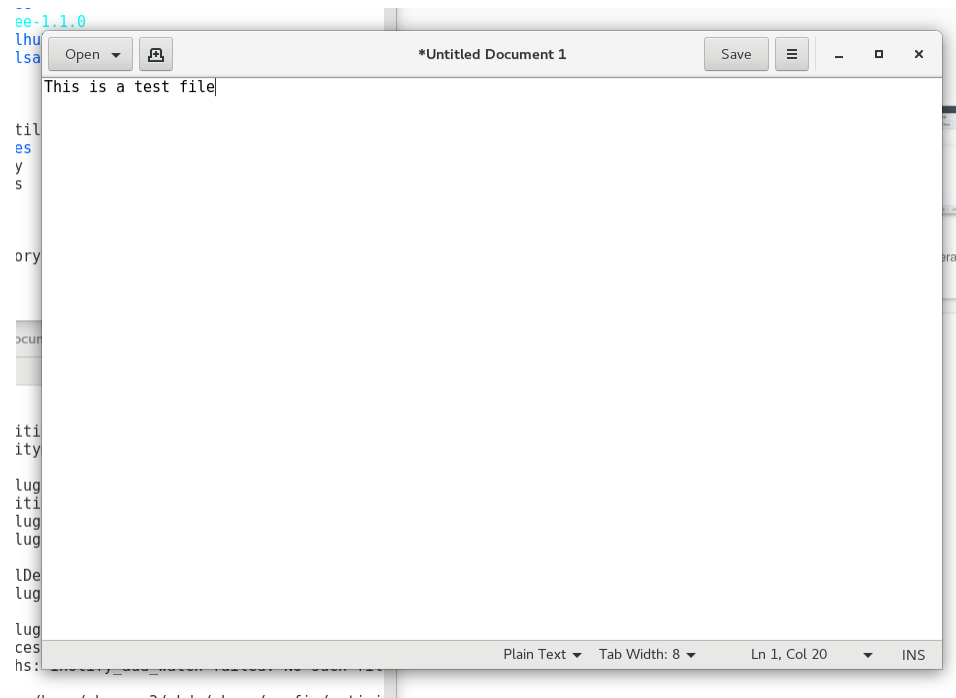
We can run similar commands to `cp` and `rm` entire directories. Note in these cases we need to use the `-r` flag, which stands for recursive. This means perform the command on the directory and all its contents:

```
[username@linuxpc directory1] cd ..
[username@linuxpc ~] cp -r directory1 directory2
[username@linuxpc ~] rm -r directory2
```

Creating a text file

We can launch programs from the command line. A common example of this will be to create text files. There exist a lot of text editors for this purposes (e.g. vim, emacs, kate), here we'll use `gedit`. Use `gedit` to create a new file called `newfile.txt` and save it to the home directory. Close `gedit` when you're done.:

```
[username@linuxpc ~] gedit
```



To edit `newfile.txt` in `gedit`, you can call `gedit` with the filename::

```
[username@linuxpc ~] gedit newfile.txt
```

Changing file permissions

Recall that you can use `ls -l` to view file permissions. Imagine we wanted to give other users permission to read our text file. For this we can use the command `chmod` (change mode).

To use `chmod` we need to specify three options relating to (i) what person or group, (ii) whether the permission should be added or removed, and (iii) which permission should be changed. These are encoded as:

Who?	What?	Which?
u = owner	+ add permission	r = read
g = group	- remove permission	w = write
o = others		x = execute
a = all		

For example, to give other users permission to read our file:

```
[username@linuxpc ~] chmod o+r newfile.txt
```

In this case `o` refers to others, `+` to add the specified mode, and `r` to read-access.

3.3.4 Running programs

The command line can be used to execute commands or scripts. You've already run a few commands from the command line (`ls`, `pwd`, `gedit`, `chmod`, `cat`), so you already know how to execute programs

Simple programs

Some simple programs can be executed with a single word. What do the following programs do?:

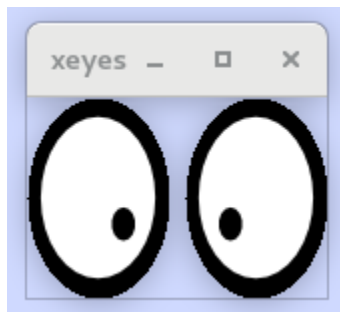
```
[username@linuxpc ~] whoami  
[username@linuxpc ~] fortune  
[username@linuxpc ~] xeyes
```

Programs with options

Lots of programs have options, which we can modify with `--flags`.

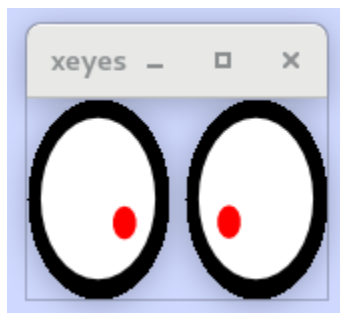
Let's take the `xeyes` program as an example. By default, `xeyes` operates as follows:

```
[username@linuxpc ~] xeyes
```



We can change the colour of the eyes using `-fg` (foreground colour) and choosing a colour:

```
[username@linuxpc ~] xeyes -fg green
```



Flags are a very common way of specifying input options. There is a command called `man` (manual). We can use it to get help for most command line programs. For the example of `xeyes`:

```
[username@linuxpc ~] man xeyes
XEYES(1)                                General Commands Manual                                XEYES(1)

NAME
    xeyes - a follow the mouse X demo

SYNOPSIS
    xeyes [-option ...]

DESCRIPTION
    Xeyes watches what you do and reports to the Boss.

OPTIONS
    -fg foreground color
        choose a different color for the pupil of the eyes.

    -bg background color
        choose a different color for the background.

    -outline outline color
        choose a different color for the outline of the eyes.

    -center center color
        choose a different color for the center of the eyes.

    ...
```

Try some of these options to remind yourself how flags work.

3.3.5 Shortcuts and tricks

Wildcards

Often we will want to refer to multiple filenames or directories. We do this with the wildcard symbols `*` and `?`. The `?` a single character, and the `*` symbol refers to zero or more characters.

For example, we might want to list everything beginning with `D` in the home directory:

```
[username@linuxpc ~] ls -l D*
drwxr-xr-x  4 username username 4.0K Sep  7 11:32 Desktop
drwxr-xr-x 12 username username 4.0K Sep 12 11:35 Documents
drwxr-xr-x  9 username username 12K Sep 13 13:14 Downloads
```

We'll return to using wildcard symbols later.

Auto complete

We can use the `tab` key to auto complete commands in the terminal. For example, if we wanted to `cd` to the `Documents` directory we could start with:

```
[username@linuxpc ~] cd Doc
```

... hit `tab`, and as there are no other directories fitting that pattern, the field will autofill to `Documents`:

```
[username@linuxpc ~] cd Documents/
```

Where there is more than one possible program, directory, or files, the `tab` key will display possible options:

```
[username@linuxpc ~] cd Do  
Documents/ Downloads/
```

Fill in more letters and press `tab` again to proceed.

You'll find that use of the `tab` key to auto complete will speed up navigation around the command line substantially.

Get previous commands

You can use the `up` key to scroll through a command history. This will save you having to re-type out commands repeatedly. Once you find the command you're looking for, you can edit it with the `left` and `right` keys. Give it a try!

3.3.6 Summary

We learned how to use the following commands. There are many thousands of other commands in Linux which you'll learn over time, but these are the main commands you'll need to navigate and manipulate files on the command line.

- `echo`: Print to the terminal
- `pwd`: Print working directory.
- `ls`: List contents of working directory
- `cd`: Change directory
- `mkdir`: Make directory
- `mv`: Move
- `cp`: Copy
- `rm`: Remove
- `chmod`: Change file permissions
- `gedit`: Create a text file in the gedit editor
- `man`: View manual
- `ssh`: Connect to another PC

3.4 To be continued...

3.4.1 Further reading

Getting used to Linux is tough. The best way to learn is by doing.

If you would like to learn more about Linux and GDAL, I can suggest the following resources:

- [Ryans Tutorials](#): A very nicely-written command line tutorial.
- [University of Surrey](#): A detailed guide to Linux
- [This GDAL cheat sheet](#): Learn GDAL by example
- [Google](#): honestly.

3.4.2 Stay in touch

Please do email either Sam (sam.bowers@ed.ac.uk) or Simone (simone-vaccari@ltsi.co.uk) if you'd like any assistance or any further tips. We'll be very happy to help.

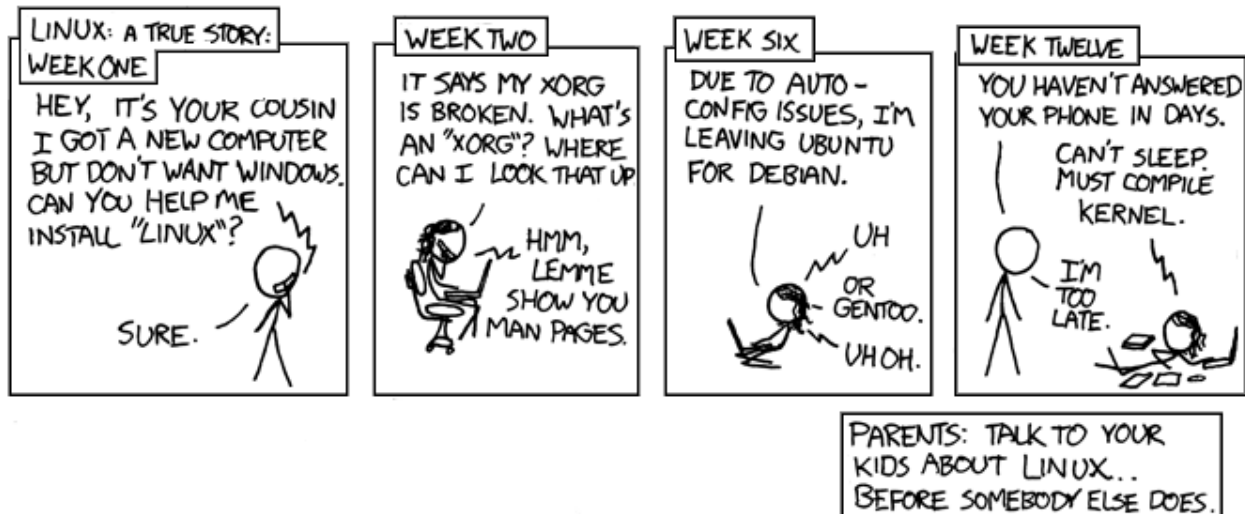


Fig. 1: Souce: XKCD.

CHAPTER 4

Search

• search



THE UNIVERSITY *of* EDINBURGH

