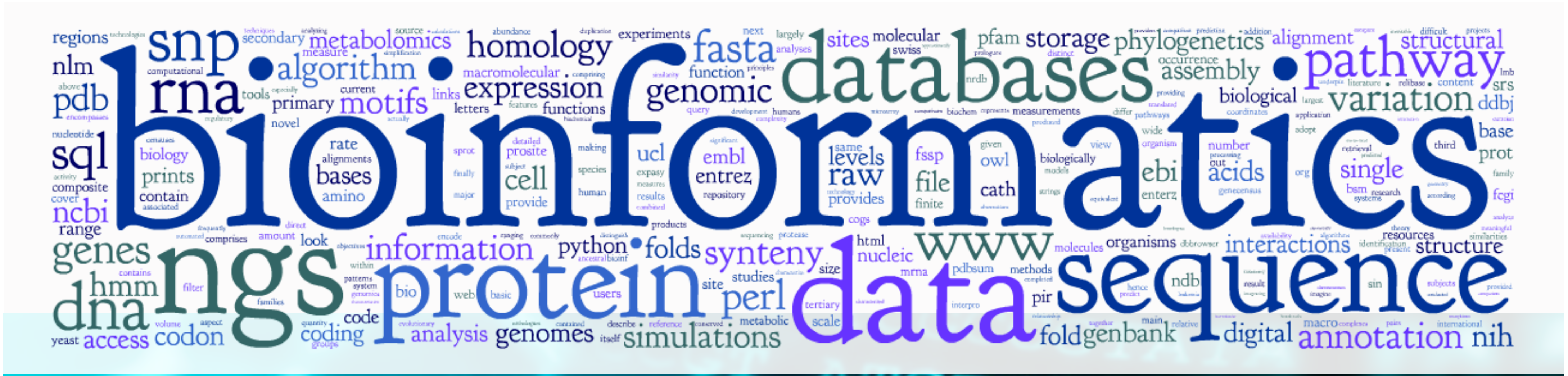


Bioinformatics
Unix

Hannes Svardal

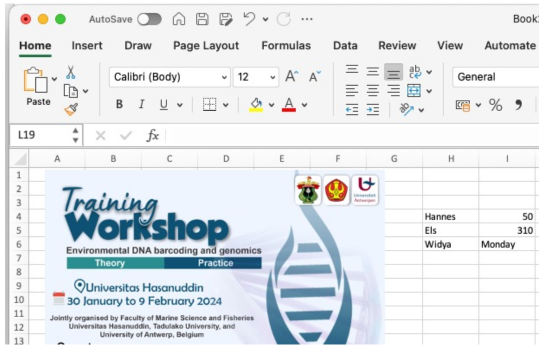
hannes.svardal
@uantwerpen.be

University of Antwerp
Evolutionary Ecology Group

Why not use Excel for Bioinformatics?

Two types of Computer programs

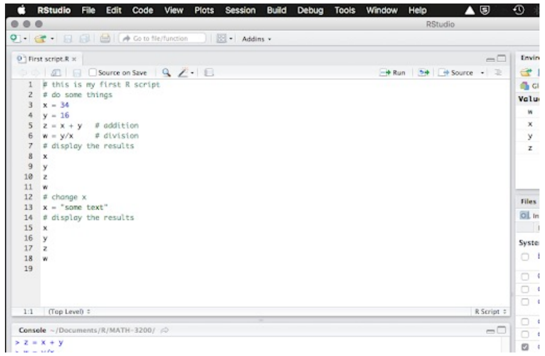
1. with a **Graphical User Interface (GUI)**
   Interaction with program by clicking on things or choosing from menus.
   Most normal users use GUI programs, e.g., LibreOffice, Microsoft Excel, PowerPoint, Adobe Illustrator, Inkscape, Google Chrome


AutoSave
Home
Insert
Draw
Page Layout
Formulas
Data
Review
View
Automate
Paste
Calibri (Body)
12
General
L19
Training Workshop
Environmental DNA barcoding and genomics
Theory
Practice
Universitas Hasanuddin
30 January to 9 February 2024
Jointly organised by Faculty of Marine Science and Fisheries Universitas Hasanuddin, Tadulako University, and University of Antwerp, Belgium

Hannes	50
Els	310
Widya	Monday

2. taking **programmatic commands/code (= text instructions)**

```
example: bcftools query -f '%CHROM\t%POS\t%REF\t%ALT[\t%SAMPLE=%GT]\n' file.vcf.gz
```



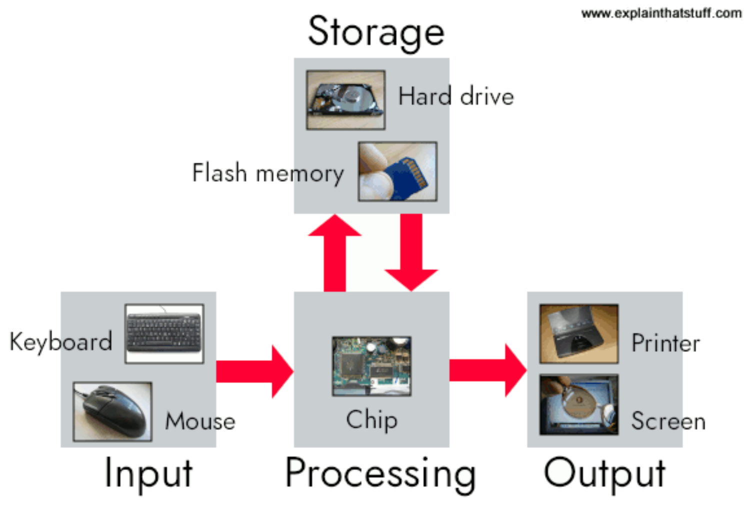
```
# this is my first R script
# do some things
x = 34
y = 16
z = x + y   # addition
w = y/x     # division
# display the results
x
y
z
w
# change x
x = "some text"
# display the results
x
y
z
w
```

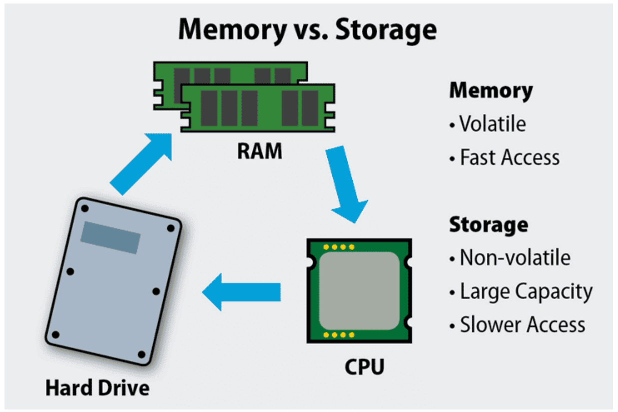
```
> z = x + y
```

R studio has a GUI, but allows you to enter R commands

Why GUI programs are not ideal for Bioinformatics

- Bioinformatic uses lots of data. For example, DNA sequencing data.
  - GUI programs tend to load all the data in a file into the memory (RAM) of the computer.
  - This can crash a computer.
- Research has lots of repetitive tasks. In GUI programs it is hard to automate things.
- Research needs to be documented and reproducible
  - "Clicking" in a program is hard to document and reproduce





Why coding is useful in Bioinformatics

Computer code is

- **Good for Big Data**
  You don't need to load whole files into memory. Code can be applied line by line.
- **Good to automate**
  Automatically analyse similar data with the same code
- **Good to reproduce**
  Good code allows you and other to understand and reproduce the research

What is Unix?

- Unix is a family computer operating systems
- Linux and MacOS are Unix-based operating systems
- Microsoft Windows, the operating system most of you use, is not based on Unix

Unix has advantages over Windows for Bioinformatics

1. Many bioinformatic programs run only on Unix-based operating systems
2. Unix systems can be operated by a **Terminal**
   A **terminal** is a text input and output environment that takes text instructions. These instructions are interpreted by a **shell (e.g., BASH).** Think of this as a type of command/programming language.
   (Note: Unix systems can still have GUIs on top, e.g., Linux Gnome, KDE etc., or MacOS)
3. Most servers and high performing compute (HPC) clusters run on Linux
   (e.g. University clusters or Amazon Web Services). You can log into and interact with such servers using a Unix Terminal.
   Using such HPC clusters is necessary for larger genomics data sets.

How to get a Unix Terminal

- On Linux, MacOS
  Should be installed. Search for a program called "Terminal", "Console", or similar
- On Windows: Install the WSL (Windows subsystem for Linux), e.g. Ubuntu

  https://genomicsworkshop.slack.com/archives/C06D8EP6HPG

  https://learn.microsoft.com/en-us/windows/wsl/install

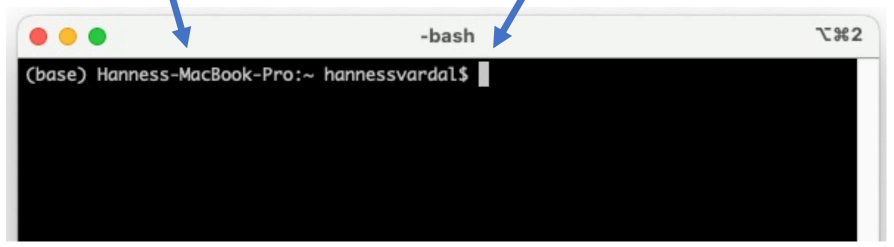
  Then open it by hitting windows key and typing Ubuntu.

(Note: Don't do this now. If you have not done it, it will be part of your practical instructions.)

Learning goals

- Navigate in the UNIX terminal environment
- Create, move and delete directories
- Create, move, delete and edit files
- Use basic UNIX commands
- Know where to find help

First steps in the Terminal

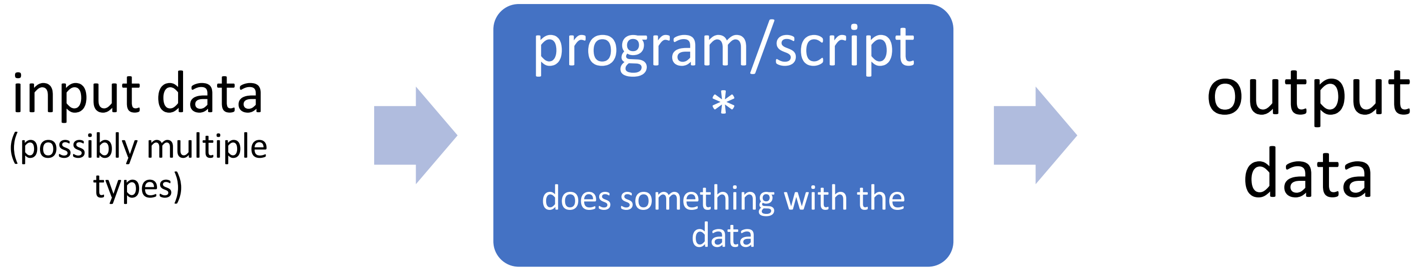

Some info (username, server)
This is where you type your command
-bash
⌥⌘2

```
(base) Hanness-MacBook-Pro:~ hannessvardal$ 
```

- In the Unix terminal you are faced by a *command line*
- You type your *command*
- The command is interpreted by a *shell*
- This means that there is a program running that understands your commands and does what you have instructed
- You need to write *command* that the *shell* understands
- We will be using *BASH*, the most common shell

We use Terminal to interact with data

- Your genomic data (sequencing reads, alignments, variant data, etc.) will generally be stored in (large) text files
- We will mainly use a Terminal
  - interact with data files
  - install programs
  - run programs on data files
  - write and run scripts (R, python, etc.) on data files
  - write and run bioinformatic workflows (= automate the above)

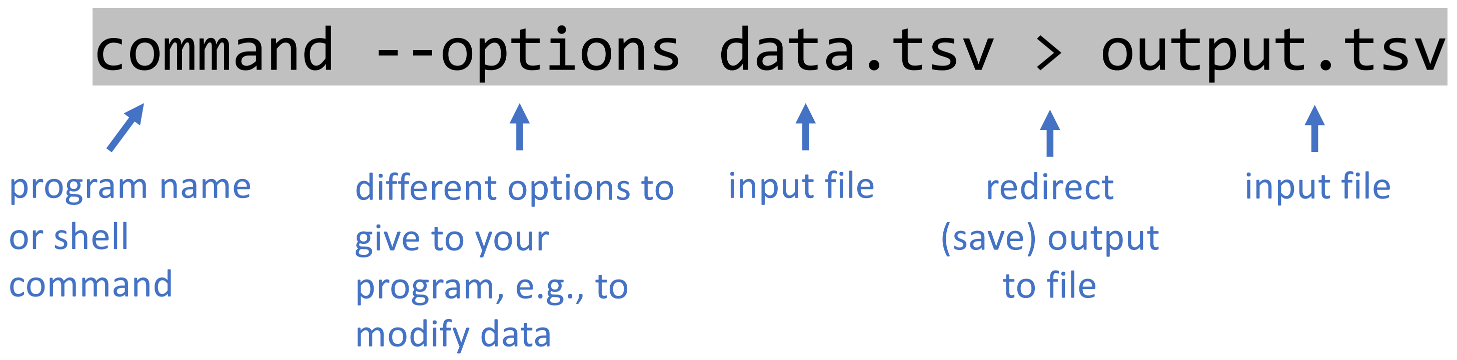


Shell command structure

A typical windows GUI workflow

- Open MS Excel
- Load data from a file data.tsv
- Modify the data
- Save the output to file output.tsv

A typical Shell command typed in a unix terminal

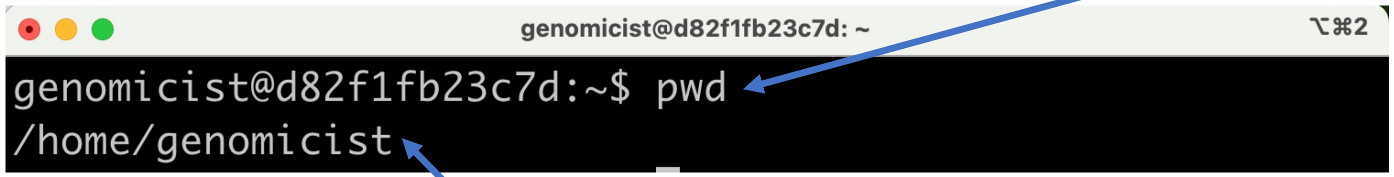


```
command --options data.tsv > output.tsv
```

program name or shell command
different options to give to your program, e.g., to modify data
input file
redirect (save) output to file
input file

Your first shell commands

- pwd ... display me the current directory path

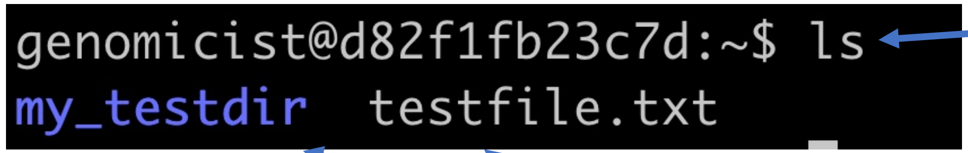


```
genomicist@d82f1fb23c7d:~$ pwd
/home/genomicist
```

type command and press *Enter*

this is the output, printed to the terminal

- ls ... list all files and folders in the current directory

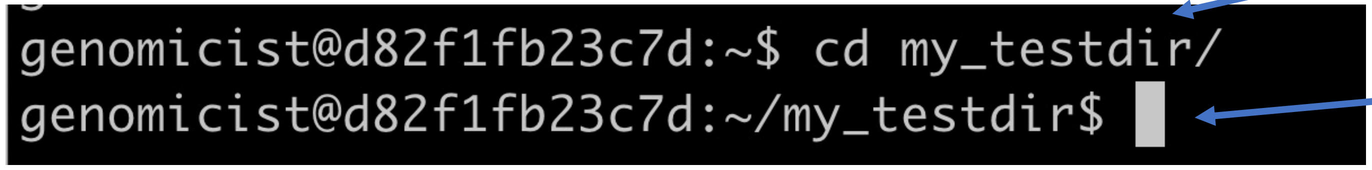


```
genomicist@d82f1fb23c7d:~$ ls
my_testdir  testfile.txt
```

type command and press *Enter*

this is the output, there are two files/folders in the current directory

- cd ... change directory

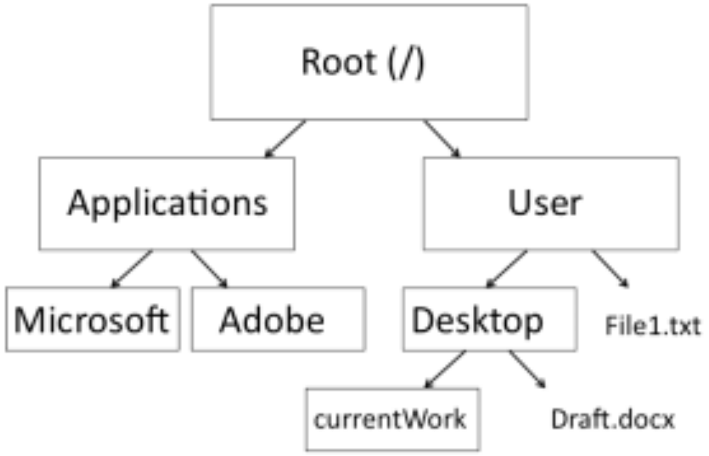


```
genomicist@d82f1fb23c7d:~$ cd my_testdir/
genomicist@d82f1fb23c7d:~/my_testdir$
```

the command cd takes a folder path as argument

there is no output, but the current directory is changed

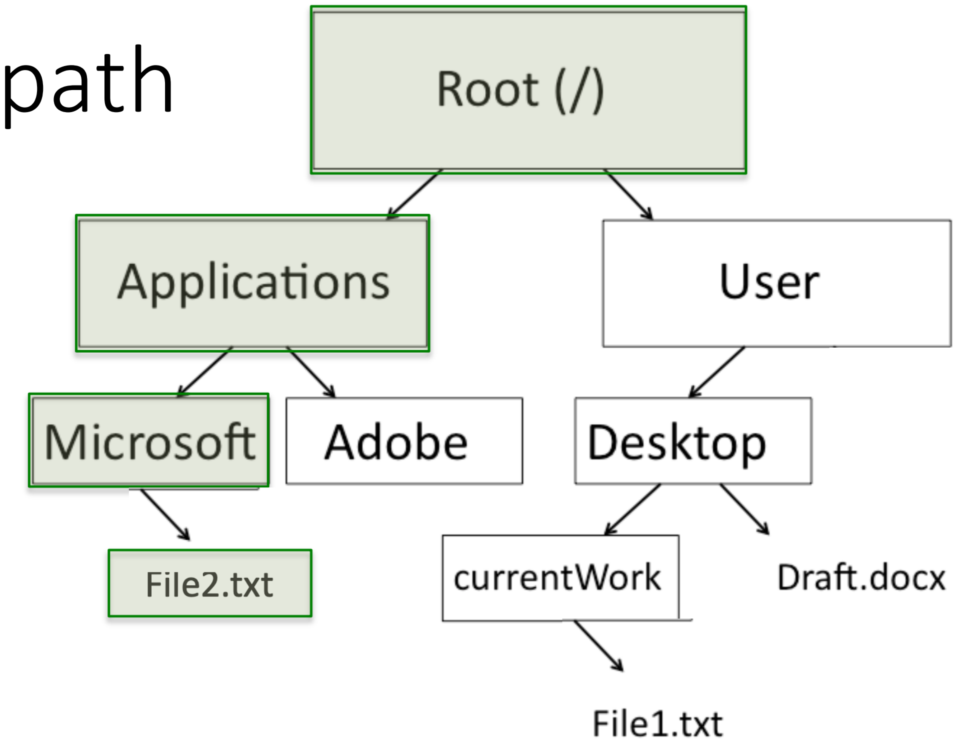
The Unix file system



- In Unix, there is a folder structure just like in Windows
- Folders are separated by slash /
- The origin of the file system is called root. The file path of the root is just a slash `/`
- So, the path `/Users/Hannes` points to a folder *Hannes* that is a subfolder of a folder *Users*. The folder *Users* is directly below the root.
- **IMPORTANT:** In the terminal, you are always in a specific folder. When you login, you are in your home folder, e.g., `/home/username`

Absolute path vs relative path

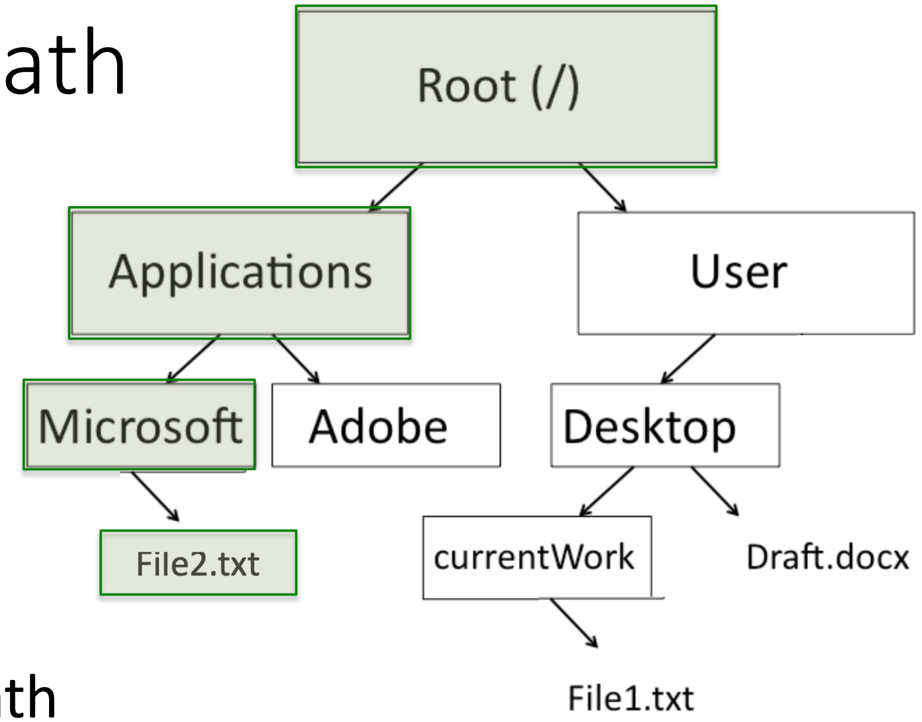
- The **path** is the location of a file or folder on the file system
- Each file has an absolute path, that is the path from the filesystem origin (root) to the file



- The absolute file path of *File2.txt* is `/Root/Applications/Microsoft/File2.txt`

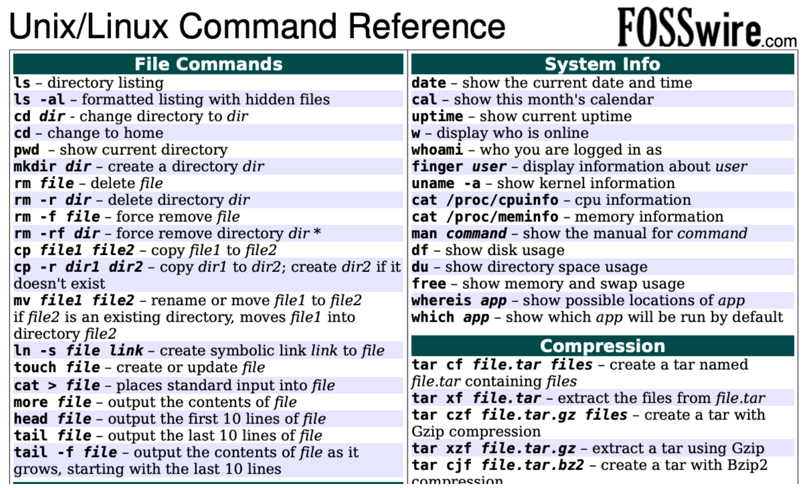
Absolute path vs relative path

- If you are in the folder Microsoft, the relative path to *File2.txt* is just `File2.txt`
- What is the path to *File2.txt*, if you are in the folder Applications? `Microsoft/File2.txt`
- You can always also use the absolute path



How to get help

- If you don't know which command to use
  - use a cheat sheet
  - google or ask chatGPT
- I don't know how to use a command/program, say "bcftools"
  - bcftools --help
  - bcftools -h
  - man bcftools
  - bcftools
  - google "bcftools user manual", "how do I select variants with bcftools"
  - ask chatGPT "How do I select variants with bcftools?"
- If your command/program does not work
  - the shell will return an **error message**
  - try to understand the error message
  - google the error message
  - ask your colleagues, ask us



Unix/Linux Command Reference FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2* if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression

Some important concepts

- Unix is case sensitive
    - `cd Myfolder` is not the same command as `cd myfolder`
- There are two main types of files : text files and binary files
- Know about *stdout and stderr*.
- `~` is a shortcut for your home folder
- Commands are often given as `command <filename>`, the `<>` just means that you should replace the content by any appropriate name. You should not actually type the symbols `<` and `>`.