

(Re) introduction to Linux

Sarah Medland

Boulder 2023

# Getting the most out of the workshop

- Ask questions!!!
- Don't sit next to someone you already know
- Work with someone with a different skillset and different experience level
- You will have access to your files after you leave
- Come to the social functions
- Ask questions!!!

# I work in Brisbane at QIMR

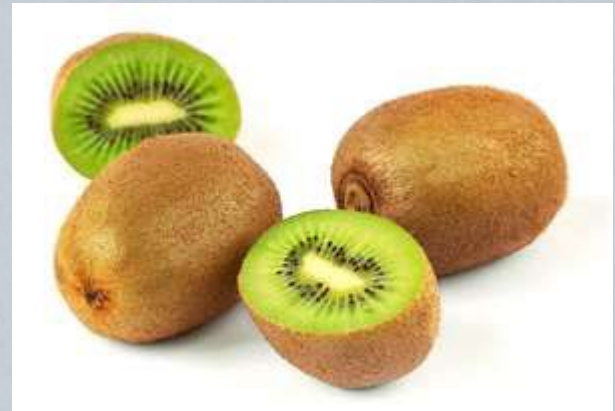


Sarah Medland





not



# Superfast intro to Linux



# File hygiene is very important

- Files are stored in Unix format not DOS or Mac
  - Changes the line ending characters
  - Use `dos2unix`, `unix2dos`, `mac2unix`, `unix2mac` to change formats
  - Can use the `file` command to check format
- Unix systems are case sensitive!
- NO SPACES in your file/directory names!!
- Wildcards ie `dos2unix *.dat`

# Working in the terminal

## Input .... Output

- Input

- Most commands don't need input signifiers
- < can be used to specify

- Output

- Without specifying most output will print to the screen
- > can be used to direct

- type: echo 'this is a dummy file'

- echo 'this is a dummy file' > dummy.txt

- | (pipe) | more pauses the output after a screen worth of text has appeared hit the space bar to get the next screens worth

# The manual

- The man command can be used in conjunction with other commands to put up some basic instructions
- type: man ls
  - ls is the list command it pulls up a list of the files in the directory

Also many many helpful webpages w examples



# Permissions

the ability to read, write and execute files

- type: `ls -l`

```
Integlio@Lapis /cygdrive/c/wedtemp
$ ls -l
total 32
-rw-r--r-- 1 Integlio mkpasswd 21 Mar  4 13:25 dummy.txt
```



- These are the permissions
- 1st a directory flag (d or -)
- then 3 letters to define the owners permissions
- 3 letters to define the groups permissions
- 3 letters to define the everyone else's permissions

## Permissions

the ability to read, write and execute files

- read access
- write access
- execute
  - to 'run' script or a program the file must be made executable

## Permissions

the ability to **read**, **w**rite and **ex**ecute files

- To change the mode/permissions use chmod
  - a number of ways to do this
  - **type:** echo “this is a test” > dummy.txt
  - ls -l
  - chmod +x dummy.txt
  - ls -l
  - chmod -x dummy.txt
  - ls -l
  - what happened?

# Useful 'one liners'

- cp copy
- mv move = rename
- rm remove
- ls list
- echo
- head looks at the top 10 lines
- tail looks at the last 10 lines
- wc counts number of lines, words, characters
- sed find and replace
- grep find and report
- awk restructure files
- pwd find where you are
- ~/ get to your home directory
- file reports type of file

# Grep

- search **g**lobally for lines matching the **r**egular **e**xpression, and **p**rint them
  - For association output for chromosome 2
  - To extract the result for snp rs59831
  - Type: `grep 'rs59831' output.txt > summary.txt`

# Grep

- Useful flags
  - -v
    - reverse grep select line that does not have the pattern
  - -C x
    - To x rows before and after the target
  - -n
    - Print the line number before the line
  - Many more...



# Awk

- derived from the surnames of its authors — Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan
- Many functions
- Very useful for restructuring data

# Awk

- Ozbmi2.rec

115	0	0.21	1	2	58	57	1.7	1.7	20.0692	19.7232	20.9943	20.8726
121	0	0.24	1	2	54	53	1.6299	1.6299	20.3244	19.9481	21.0828	20.9519
158	0	0.21	1	2	55	50	1.6499	1.6799	20.202	17.7154	21.0405	20.121
172	0	0.21	1	2	66	76	1.5698	1.6499	26.7759	27.9155	23.0125	23.3043
182	0	0.19	1	2	50	48	1.6099	1.6299	19.2894	18.0662	20.7169	20.2583
199	0	0.26	1	2	60	60	1.5999	1.5698	23.4375	24.3418	22.0804	22.3454
221	0	0.23	1	2	65	65	1.75	1.7698	21.2245	20.7476	21.3861	21.227
239	0	0.29	1	2	40	39	1.5598	1.5298	16.4366	16.6603	19.5966	19.6912
246	0	0.24	1	2	60	57	1.7598	1.7698	19.3698	18.194	20.746	20.3076

- `awk '{ print $1, $10, $11, $4, $5 }' ozbmi2.rec > new.rec`

```
115 20.0692 19.7232 1 2
121 20.3244 19.9481 1 2
158 20.202 17.7154 1 2
172 26.7759 27.9155 1 2
182 19.2894 18.0662 1 2
199 23.4375 24.3418 1 2
221 21.2245 20.7476 1 2
239 16.4366 16.6603 1 2
246 19.3698 18.194 1 2
```

# Awk

- \$1 = column 1
- Print \$0 = print whole line
- add subtract multiply etc
- change number of decimals
- Many functions

# Sort

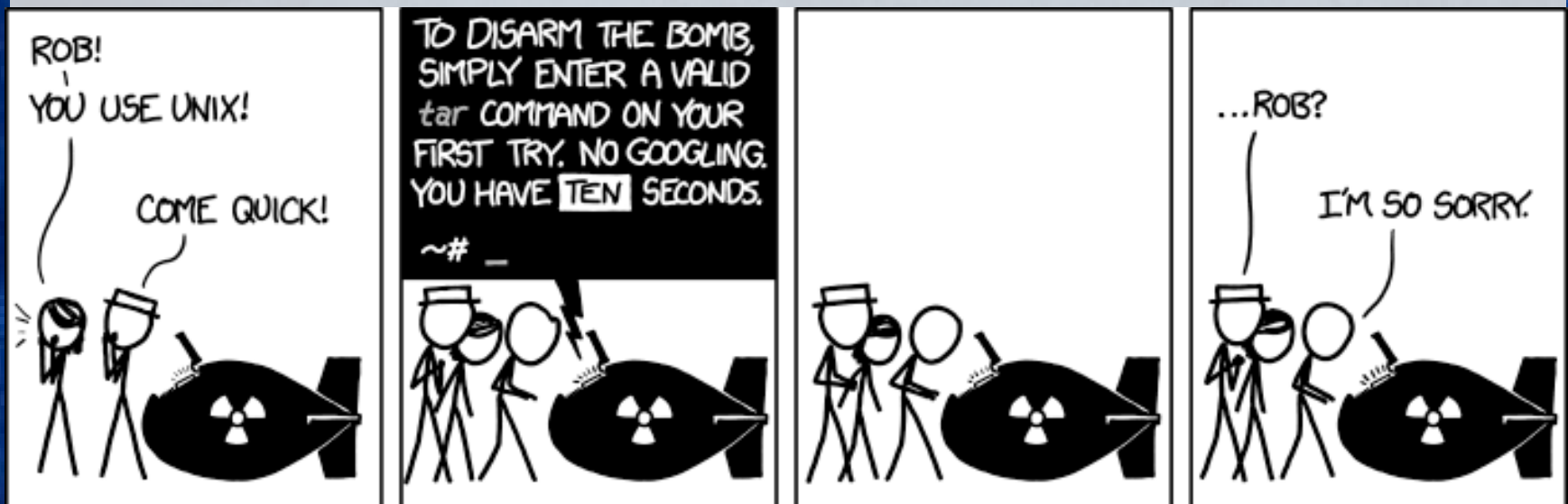
- Useful flags
  - -f ignore case
  - -n numeric sort
  - -r reverse
  - -c check if a file is sorted
  - -u prints only unique lines
  - -k2 sort starting at column 2
  
  - `sort -fg -k 3` (sort in numeric order on column 3)

# Zipping and unzipping

- zip
  - `zip my1st.zip *txt`
  - `zip -mTr my1st.zip *txt`
- unzip
  - `unzip my1st.zip`
- gzip
  - `gzip example.txt`
- Un-gzip
  - `gzip -d example.txt.gz`

# tar

- Unzipping tar.gz files
  - `tar -xzvf example.tar.gz`
- Make Tar files
  - `tar cvf MyProject.tar MyProject`
- List contents
  - `tar tvf my-archive.tar`
  - `tar tzvf my-archive.tar.gz`





# Looking at your data

- `less filename`
  - Allows you to scroll through your data
- `less -S filename`
  - Shows a screen width of data (stops text wrapping)
- `zless -S filename`
  - Allows you to look at a gz file without unzipping

# Nano (text editor)

- nano *filename*
  - Commands at bottom of screen
  - Save = ctrl+O
  - Exit = ctrl +X

# Putting it together

- Making a 'shell' script to automate analyses

<contents of imaginary file inefficient.sh>

```
pedstats -p 1.ped -d 1.dat -pdf --prefix:1
```

```
merlin -p 1.ped -d 1.dat -m 1.map --vc --pdf --prefix:1
```

```
pedstats -p 2.ped -d 2.dat -pdf --prefix:2
```

```
merlin -p 2.ped -d 2.dat -m 2.map --vc --pdf --prefix:2
```

```
pedstats -p 3.ped -d 3.dat -pdf --prefix:3
```

```
merlin -p 3.ped -d 3.dat -m 3.map --vc --pdf --prefix:3
```

To run this make inefficient.sh executable then type `./inefficient.sh`

# Loops 1

<contents of imaginary file loop\_a.sh>

```
for $i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
  21 22
do
  pedstats -p $i.ped -d $i.dat --pdf --prefix:$i
  merlin -p $i.ped -d $i.dat -m $i.map --vc --pdf --prefix:$i
done
```

# Loops 2

<contents of imaginary file loop\_b.sh>

```
for (( i = 1 ; i <= 22 ; i++ ))
```

```
do
```

```
    pedstats -p $i.ped -d $i.dat --pdf --prefix:$i
```

```
    merlin -p $i.ped -d $i.dat -m $i.map --vc --pdf --prefix:$i
```

```
done
```

# Other bits

- When working on servers
  - `bg &`
  - `fg`
  - `nohup`
  - `ctrl+c`
  - `ctrl+z`
  - `which`



# Shutting down you unix session

- exit
- logout
- quit
- q