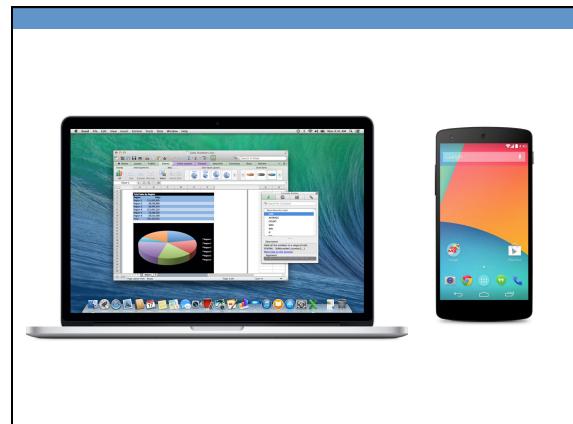


UNIX INTRODUCTION

Janos Binder - janos.binder@embl.de



Biology in 2014

- Omics Era
- Several Go of data / experiment
- Personalized Medicine

Why using Unix in Biomedical Research?

Free Fast
Ethics Open

Easy & documented!

What do you do with Unix in Biology?

- File manipulation
 - Sequences
 - Interaction networks
 - Research articles
 - Spectrum
 - ...
- Analysis
 - Calling programs (R, Perl, Python, Java, etc...)
 - Generating knowledge from raw information
 - Statistics
 - Filtering
 - ...

OSX terminal

```
james@volcano:~$  
Login as root? James  
james@volcano's password:  
Linux volcano 2.6.22-14-server #1 SMP Sun Oct 14 23:34:23 GMT 2007 1686  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
james@volcano:~-> [REDACTED]
```

Why do we use terminal

- To learn how to work on a server, because:
- Common collaborations/projects
- More computing power
- Safer (backups)

- File manipulation
 - Sequences
 - Interaction networks
 - Research articles
 - Spectrum
 - ...
- Analysis
 - Calling programs (R, Perl, Python, Java, etc...)
 - Generating knowledge from raw information
 - Statistics
 - Filtering
 - ...

Part 1

Part 2

The Shell – What the (s)hell is this?

- *Piece of software that provides an interface for users of an operating system*



A screenshot of a terminal window titled "johnec246:~". The window shows the command "python test.py" being run, followed by its output: "foobar" repeated eight times, and then the command "stty sane" being run. The terminal window has a dark background and light-colored text.

```
johnec246:~$ python test.py
foobar
foobar
foobar
foobar
foobar
foobar
foobar
foobar
stty sane
johnec246:~$
```

The Shell – Tips

- Previous command: 
- Auto-Complete: 
- Emergency Stop:  
-  
- Going forward/back:  
 

STEP 0

Getting help

info [command]

The info command provides help

man [command]

The man command is a more traditional way to get help

What's going on: top

Display the list of running processes
To kill a program type 'K' on the keyboard followed by the PID

STEP 1

Browsing around

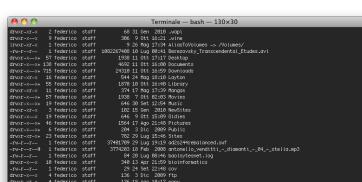
Everything's a file

ls [directory] (list)

The ls command lists the contents of your current working directory

A screenshot of a terminal window titled "Terminal" running on a Mac OS X desktop. The window has a dark gray background. At the top, there are three colored window control buttons (red, green, blue) on the left, followed by the title bar "Terminal" and "Terminal — bash — 130x30". Below the title bar is a small input field containing the command "clear". The main area of the terminal is completely black, indicating that the "clear" command has been successfully executed.

```
ls -al [directory] (long list)


```

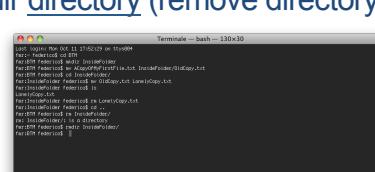
```
cd [directory]
[User-Login: Sat May 10 21:05:56 on ttyp002]
[Binders-Gadget:-] jbinder% cd Documents
[Binders-Gadget:/Documents] jbinder%
```

The cd command changes the current working directory to 'directory'
One level up: ..
This directory: .
Your home: ~



The screenshot shows a terminal window titled "Terminal - bash - 230x39". The user runs the command "mkdir directory" and then lists the contents of the current directory with "ls". The output shows the newly created "directory" folder.

```
root@centos: ~# mkdir directory
root@centos: ~# ls
Desktop  Documents  Downloads  directory  Pictures  Public  Templates
```



The screenshot shows a terminal window titled "Terminal - bash - 130x30". The command "rm -r /tmp/testdir" is being typed, and the output shows the directory being deleted.

```
user@user-MacBook-Pro: ~ $ rm -r /tmp/testdir
rm: removing directory '/tmp/testdir'
user@user-MacBook-Pro: ~ $
```

The screenshot shows a terminal window with the following text:

```
[user@host ~] $ cd /tmp/test1/test2/test3  
[user@host test3] $ ls  
file1 file2 file3  
[user@host test3] $ cd ..  
[user@host test2] $ ls  
file1 file2  
[user@host test2] $ cd ..  
[user@host test1] $ ls  
file1 file2  
[user@host test1] $ cd ..  
[user@host ~] $
```

pwd (print working dir)

```
Terminal — bash — 130x30
Last login: Mon Oct 11 21:50:05 on ttys000
Federico:~ Federico$ pwd
/Federico:~ Federico$
```

The `pwd` command prints the absolute pathname of the current working directory

STEP 2

Handling files

cp file1 file2 (copy)

```
Terminal — bash — 130x30
Last login: Mon Oct 11 21:50:05 on ttys000
Federico:~ Federico$ cp file1 file2
total 0
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 .
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 ..
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file1.txt
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file2.txt
total 0
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 .
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 ..
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file1.txt
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file2.txt
Federico:~
```

The `cp` command makes a copy of `file1`, naming it `file2`

mv file1 file2 (move)

```
Terminal — bash — 130x30
Last login: Mon Oct 11 21:50:05 on ttys000
Federico:~ Federico$ mv file1 file2
total 0
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 .
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 ..
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file1.txt
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file2.txt
total 0
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 .
drwxr-xr-x 2 Federico staff 240 10 Oct 17:50 ..
-rw-r--r-- 1 Federico staff 0 10 Oct 17:50 file2.txt
Federico:~
```

The `mv` command moves `file1` into `file2`
(if `file2` is in the same directory, `file1` is renamed)

rm file (remove)

```
Terminal — bash — 130x30
Last login: Mon Oct 11 21:50:05 on ttys000
Federico:~ Federico$ rm file1
rm: cannot remove 'file1': No such file or directory
Federico:~ Federico$ rm file1
rm: cannot remove 'file1': No such file or directory
Federico:~ Federico$ rm file1
rm: cannot remove 'file1': No such file or directory
Federico:~ Federico$ rm file1
rm: cannot remove 'file1': No such file or directory
Federico:~ Federico$
```

The `rm` command removes (a) `file`

scp server:file1 file2 (secure copy)

```
Terminal — tcsh — 80x24
[dhcp-]binder:~] bbindler% scp organisms.cbsi.dtu.dk:/PFI-analysis.tar.xz .
PFI-analysis.tar.xz                                100%   2 MB    0:14/0:5  88/64
[dhcp-]binder:~] bbindler%
```

`scp` copies a file from a server to a local place
`scp file1 server:file2` also works

wget url

```
Terminal — wget — 80x24
[Binders-Gadget:/Downloads] joinder% wget ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/completedb/uniprot_sprot.dat.gz
--2014-09-10 21:55:18--  ftp://ftp.uniprot.org/pub/databases/uniprot/current_rele
ease/knowledgebase/complete/uniprot_sprot.dat.gz
Resolving ftp.uniprot.org (ftp.uniprot.org)... 141.161.188.197
Connecting to ftp.uniprot.org (ftp.uniprot.org)|141.161.188.197|:21...
Connected to ftp.uniprot.org (ftp.uniprot.org)|141.161.188.197|:21...
...done.
Length: 491964024 (469M) (unknown)
[1%]
```

wget can download files

curl url > filename

```
Terminal — curl — 80x24
[Binders-Gadget:/Downloads] joinder% curl ftp://ftp.uniprot.org/pub/databases/u
niprotnet/current_release/knowledgebase/complete/uniprot_sprot.dat.gz > uniprot_dot
.gz
% Total % Received % Xferd Average Speed Time Time Time Current
0 469M 0 4413K 0 0 539k 0:14:50 0:08:08 0:14:42 813K
```

curl retrieves data from various internet services

STEP 3

Files content

less file1

```
Terminal — less — 130x30
[Binders-Gadget:/Downloads] joinder% less file1
file1: 1 Nel testo del poema di nostra vita
file1: 2 al ritorno per una delle nostre
file1: 3 vite, aveva scritto questo versetto:
file1: 4 Ah! quanto è dir vero cosa è così doloroso
file1: 5 sentire qualcosa e sperare e farne
file1: 6 niente, e poi non accadere nulla.
file1: 7 Tant'è secco che poco è più morto;
file1: 8 se poi trovi la vita in 'l frutto,
file1: 9 non ti interessa più nulla, non ti sente.
file1: 10 Io non so ben rite che l'è' v'interesse,
file1: 11 se tu senti qualcosa, e poi non accade
file1: 12 che tu venga via abbandonato.
file1: 13 Ma pot'ess' t'è' fui il più d'un colpo giunto,
file1: 14 e tu senti qualcosa, e poi non accade
file1: 15 che a fine di sconsigli il tuo compagno,
file1: 16 guardandoti, e visti le sue mosse
file1: 17 che non aveva fatto nulla, e poi non accade
file1: 18 che non scriva niente per ogni colpa.
file1: 19 E' stato lui lo scrittore a fare questo
file1: 20 che non accade, e poi non accade.
file1: 21 Non ti senti nulla, e poi non accade
file1: 22 che non accada nulla, e poi non accade.
file1: 23 Ma senti nulla, e poi non accade
file1: 24 che non accada nulla, e poi non accade.
file1: 25 Nel testo del poema di nostra vita
file1: 26 al ritorno per una delle nostre
file1: 27 vite, aveva scritto visto questo versetto:
```

The less command shows the contents of a file on the screen, one page at a time

head [-n] file

```
Terminal — bash — 130x30
[Binders-Gadget:/Downloads] joinder% head -10 file1
file1: 1 Nel testo del poema di nostra vita
file1: 2 al ritorno per una delle nostre
file1: 3 vite, aveva scritto visto questo versetto:
file1: 4 Ah! quanto è dir vero cosa è così doloroso
file1: 5 sentire qualcosa e sperare e farne
file1: 6 niente, e poi non accadere nulla.
file1: 7 Tant'è secco che poco è più morto;
file1: 8 se poi trovi la vita in 'l frutto,
file1: 9 non ti interessa più nulla, non ti sente.
```

The head command shows the first 10 (or n) lines of a file

tail [-n] file

```
Terminal — bash — 130x30
[Binders-Gadget:/Downloads] joinder% tail -10 file1
file1: 1 Nel testo del poema di nostra vita
file1: 2 al ritorno per una delle nostre
file1: 3 vite, aveva scritto visto questo versetto:
file1: 4 Ah! quanto è dir vero cosa è così doloroso
file1: 5 sentire qualcosa e sperare e farne
file1: 6 niente, e poi non accadere nulla.
file1: 7 Tant'è secco che poco è più morto;
file1: 8 se poi trovi la vita in 'l frutto,
file1: 9 non ti interessa più nulla, non ti sente.
```

The tail command shows the last 10 (or n) lines of a file

wc [-lwm] file (word count)

```
Terminal — bash — 330x30
wc -lwm file
 1 1 1
```

The wc command returns
lines/words/characters number of a file

grep pattern file

```
Terminal — ssh — 80x24
grep "Villebrand disease" file
... (many lines of text)
9606 ENSP00000231509 -25 BT010000798 UnprotKB-RC CURATED S T
9606 http://www.uniprot.org/uniprot/GCR_HUMAN UnprotKB-RC CURATED S T
9606 BT010000798 UnprotKB-RC CURATED S T
9606 ENSP00000231509 -25 BT010000395 UnprotKB-RC CURATED S T
9606 http://www.uniprot.org/uniprot/GCR_HUMAN UnprotKB-RC CURATED S T
9606 BT010000798 UnprotKB-RC CURATED S T
9606 ENSP00000231509 -25 BT010000472 UnprotKB-RC CURATED S T
9606 http://www.uniprot.org/uniprot/GCR_HUMAN UnprotKB-RC CURATED S T
9606 BT010000798 UnprotKB-RC CURATED S T
9606 ENSP00000231509 -25 BT010000709 UnprotKB-RC CURATED S T
9606 http://www.uniprot.org/uniprot/GCR_HUMAN UnprotKB-RC CURATED S T
9606 BT010000798 UnprotKB-RC CURATED S T
9606 ENSP00000231509 -25 BT010000376 UnprotKB-KV CURATED S T
9606 http://www.uniprot.org/uniprot/GCR_HUMAN UnprotKB-KV CURATED S T
9606 ENSP00000231509 -21 GO:0008211 Compara IEA 2 TRUE h
http://www.uniprot.org/uniprot/GCR_HUMAN ProtInc TAS 4 TRUE h
9606 ENSP00000231509 -22 GO:0016020 Compara IEA 2 TRUE h
http://www.uniprot.org/uniprot/GCR_HUMAN ProtInc TAS 4 TRUE h
```

Grep returns lines, which contain the pattern
-v flag return those line, which does not contain the pattern

cut -f[number] [file]

```
Terminal — tcsh — 110x32
cut -f1 file
... (many lines of text)
Cellular Component
```

Cut returns only the specified columns.

sort -u [file]

```
Terminal — tcsh — 98x25
sort -u file
... (many lines of text)
Cellular Component
```

Sort the lines in a file. -u specifies that only unique lines
should be listed (against duplication)

uniq [-c] [file] (unique)

```
Terminal — tcsh — 98x25
uniq -c file
... (many lines of text)
1 Cellular Component
```

Prints only unique lines (similar -u in sort). -c defines that it
counts also the occurrence

gawk “short program” file

```
Terminal — tcsh — 110x32
gawk '{print $1,$2,$3}' file
... (many lines of text)
Cellular Component UnIPRED CURATED S
```

(g)awk is a language for data manipulation – some examples
will come during the tutorial

Take home message

- Unix/Linux is the most widely used operating system in bioinformatics
- Unix/Linux is used to handle files and to run analysis on very large experiments, by calling external programs
 - You can manipulate your files through command lines
 - You can launch programs by the command line (next talk)
- Google is the best way to find information about particular command lines

STEP 4

Setting up programs

ssh

```
Terminal - ssh - 107x43
me@Dender:~$ ssh
```

ssh allows to log in to a remote machine

chmod a+x file

chmod changes permission of a file. Here it becomes executable

configure

```
Terminal - clang - 107x43
me@Dender:~$ ./configure
```

configure adopt a program package for the local environment

make

```
Terminal - clang - 107x43
me@Dender:~$ make
```

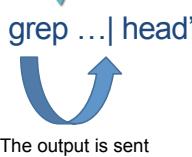
make compiles and set ups a program package

STEP 4

Pipelines and streams

CHAINING PROGRAMS

Pipes – Chaining programs



Checking whether Xorg is running:
top | grep 'Xorg'

```

top | grep "Xorg"
2483 root 29 0 114w 284 1084 5 0 0.0 8:40:42 Xorg
2483 root 29 0 114w 284 1084 5 0 0.0 8:40:42 Xorg
2483 root 29 0 114w 284 1084 5 0 0.0 8:40:42 Xorg
  
```

Output only the line of 'top' containing the string 'Xorg'

Redirecting standard input & output

ENSP000000000233 ARF5	GO:0005575	cellular_component	UniProtKB
URATED 5			
ENSP000000000233 ARF5	GO:0005575	cellular_component	UniProtKB
URATED 5			
ENSP000000000233 ARF5	GO:0005622	Intracellular	UniProtKB
URATED 5			
ENSP000000000233 ARF5	GO:0005622	Intracellular	UniProtKB
URATED 5			
ENSP000000000233 ARF5	GO:0005629	Cell	UniProtKB CURATED 5
ENSP000000000233 ARF5	GO:0005629	Cell	UniProtKB CURATED 5
ENSP000000000233 ARF5	GO:0005737	Cytoplasm	UniProtKB
URATED 5			
ENSP000000000233 ARF5	GO:0005737	Cytoplasm	UniProtKB

< reads in a file, > write the results into a file

Introduction to bash scripting

- Repeating steps is time-consuming and error-prone
- Reproducible research ->
- An automated solution is needed
- UNIX – shell script

First script – Hello World

- The first line defines what kind of program will interpret this script
- echo – this command prints out text
- “ denotes that it is a text – work also without it, but error prone

```
#!/bin/bash
echo "Hello World!"
```

Variables

- Storing often used texts, numbers
- Storing results of a calculation
- Remember to the \$

```
#!/bin/bash
HW="Hello World!"
echo $HW
echo "It is boring to print" $HW
```

Conditionals

- Let you decide whether to perform an action or not, which based on the evaluation of the expression

```
#!/bin/bash
FILE="3_basic_conditionals.sh"
if [ -f $FILE ]; then
    echo "$FILE: $FILE exists!"
fi
```

- An if-then-else construct:

```
if [ -f $FILE ]; then
    echo "$FILE: $FILE exists!"
else
    echo "$FILE: $FILE does not exists!"
fi
```

Streams and pipes - revisited

- Storing the names of the files in a directory

```
ls -1 > files_in_the_directory.txt
less files_in_the_directory.txt
```

- But it shows also lists files_in_the_directory.txt
- Piping the other way

```
grep -v files_in_the_directory.txt <
files_in_the_directory.txt | less
```

- A more clever solution:

```
ls -1 | grep -v files_in_the_directory.txt >
files_in_the_directory.txt
less files_in_the_directory.txt
```

Loops

- To do stuff the “same” stuff more than once

```
#!/bin/sh
echo "We count to ten!"
for (( i = 1 ; i <= 10 ; i++ )) ; do
    echo $i
done
```

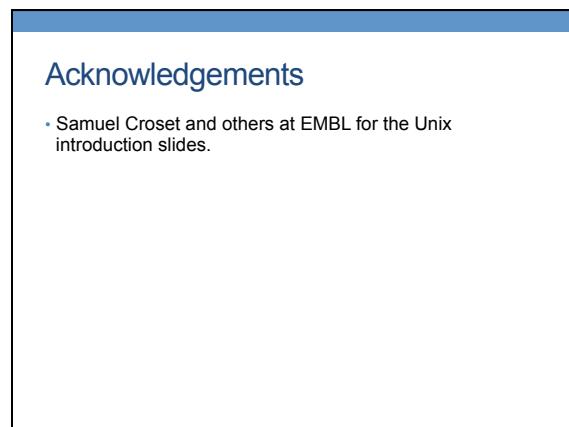
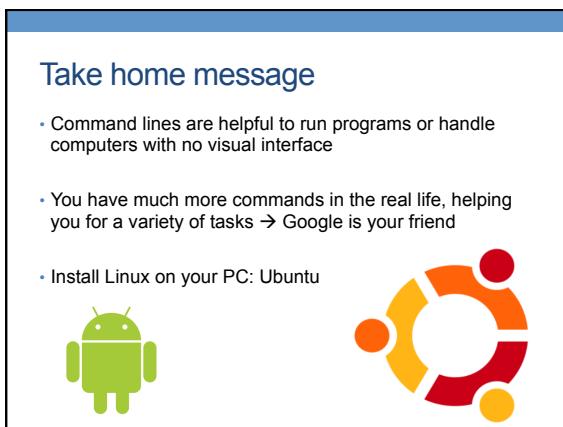
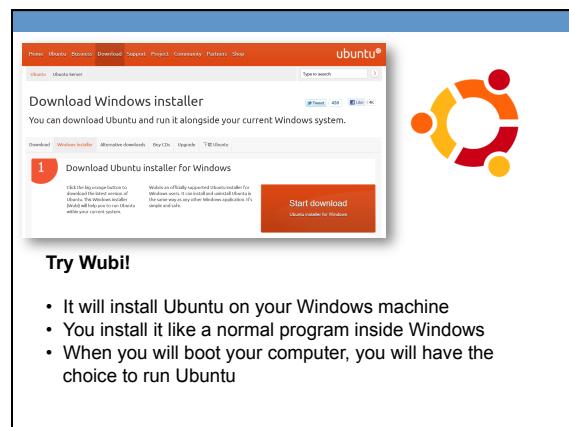
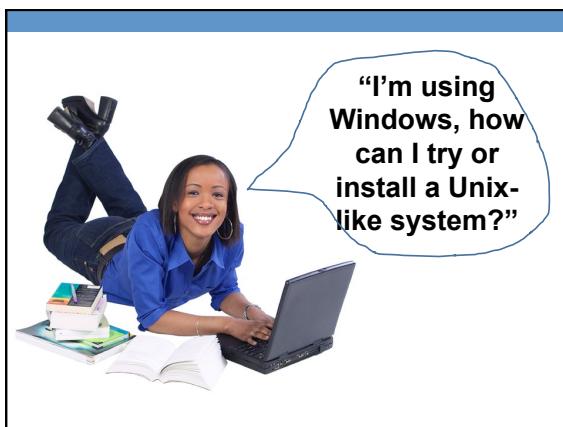
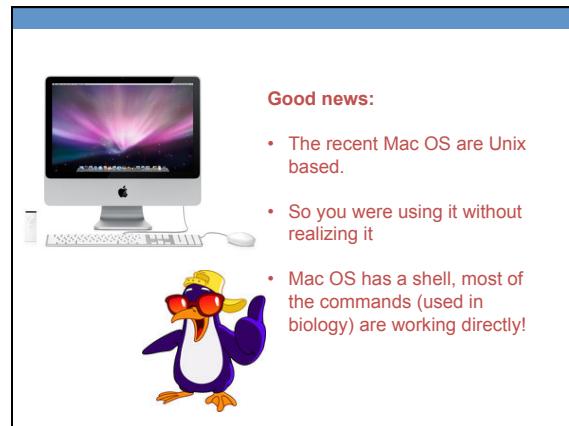
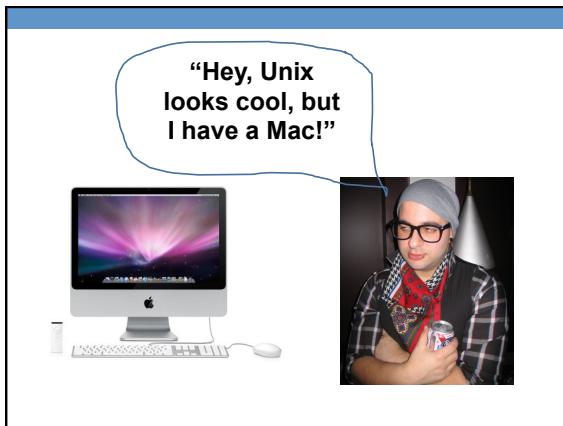
```
#!/bin/sh
(( i = 1 ))
while (( i <= 10 )) ; do
    echo $i
    (( i++ ))
done
```

How do I execute a series of commands? Bash scripting

- You can create scripts (e.g., “my_series_of_command.sh”) executing a series of Unix commands

```
#!/bin/bash
for i in $(seq 1 5)
do
    echo "i is equal to: $i"
    ls -l
done
```

You can execute the scripts via: sh my_series_of_command.sh
or chmod a+x my_series_of_command.sh; ./my_series_of_command.sh



Important parts

- Understand the file/directory structure of Unix (+ special characters ... ~ /)
- Knowledge of basic commands
- Pipes, redirection (| < >)
- Basic manipulation and filtering of text files
- Writing simple scripts (conditionals, loops, printing out things to the console)