Cloud Computing and Unix: An Introduction

Dr. Sophie Shaw
University of Aberdeen, UK
s.shaw@abdn.ac.uk
What We’re Going To Do

• Why Unix?
• Cloud Computing
• Connecting to AWS
• Introduction to Unix Commands
Etiquette

• PowerPoint interspersed with Challenges
• Ask me questions
• Ask demonstrators
• Work together
• Cheat!
<table>
<thead>
<tr>
<th>File Commands</th>
<th>System Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls - directory listing</td>
<td>date - show the current date and time</td>
</tr>
<tr>
<td>ls -al - formatted listing with hidden files</td>
<td>cal - show this month's calendar</td>
</tr>
<tr>
<td>cd dir - change directory to dir</td>
<td>uptime - show current uptime</td>
</tr>
<tr>
<td>cd - change to home</td>
<td>w - display who is online</td>
</tr>
<tr>
<td>pwd - show current directory</td>
<td>whoami - who you are logged in as</td>
</tr>
<tr>
<td>mkdir dir - create a directory dir</td>
<td>finger user - display information about user</td>
</tr>
<tr>
<td>rm file - delete file</td>
<td>uname -a - show kernel information</td>
</tr>
<tr>
<td>rm -r dir - delete directory dir</td>
<td>cat /proc/cpuinfo - cpu information</td>
</tr>
<tr>
<td>rm -rf dir - force remove directory dir*</td>
<td>cat /proc/meminfo - memory information</td>
</tr>
<tr>
<td>cp file1 file2 - copy file1 to file2</td>
<td>man command - show the manual for command</td>
</tr>
<tr>
<td>cp -r dir1 dir2 - copy dir1 to dir2; create dir2 if it doesn't exist</td>
<td>df - show disk usage</td>
</tr>
<tr>
<td>mv file1 file2 - rename or move file1 to file2 if file2 is an existing directory, moves file1 into directory file2</td>
<td>du - show directory space usage</td>
</tr>
<tr>
<td>ln -s file link - create symbolic link link to file</td>
<td>free - show memory and swap usage</td>
</tr>
<tr>
<td>touch file - create or update file</td>
<td>whereis app - show possible locations of app</td>
</tr>
<tr>
<td>cat &gt; file - places standard input into file</td>
<td>which app - show which app will be run by default</td>
</tr>
<tr>
<td>more file - output the contents of file</td>
<td>Compression</td>
</tr>
<tr>
<td>head file - output the first 10 lines of file</td>
<td>tar cf file.tar files - create a tar named file.tar containing files</td>
</tr>
<tr>
<td>tail file - output the last 10 lines of file</td>
<td>tar xf file.tar - extract the files from file.tar</td>
</tr>
<tr>
<td>tail -f file - output the contents of file as it grows, starting with the last 10 lines</td>
<td>tar czf file.tar.gz files - create a tar with Gzip compression</td>
</tr>
<tr>
<td>Process Management</td>
<td>tar xzf file.tar.gz - extract a tar using Gzip</td>
</tr>
<tr>
<td>ps - display your currently active processes</td>
<td>tar cjf file.tar.bz2 - create a tar with Bzip2 compression</td>
</tr>
<tr>
<td>top - display all running processes</td>
<td>tar xjf file.tar.bz2 - extract a tar using Bzip2</td>
</tr>
<tr>
<td>kill pid - kill process id pid</td>
<td>gzip file - compresses file and renames it to file.gz</td>
</tr>
<tr>
<td>killall proc - kill all processes named proc *</td>
<td>gzip -d file.gz - decompresses file.gz back to file</td>
</tr>
<tr>
<td>bg - lists stopped or background jobs; resume a stopped job in the background</td>
<td>Network</td>
</tr>
<tr>
<td>fg - brings the most recent job to foreground</td>
<td>ping host - ping host and output results</td>
</tr>
<tr>
<td>fg n - brings job n to the foreground</td>
<td>whois domain - get whois information for domain</td>
</tr>
<tr>
<td>File Permissions</td>
<td>dig domain - get DNS information for domain</td>
</tr>
<tr>
<td>chmod octal file - change the permissions of file to octal, which can be found separately for user, group, and world by adding:</td>
<td>dig -x host - reverse lookup host</td>
</tr>
<tr>
<td>4 - read (r)</td>
<td>wget file - download file</td>
</tr>
<tr>
<td>2 - write (w)</td>
<td>wget -c file - continue a stopped download</td>
</tr>
<tr>
<td>1 - execute (x)</td>
<td></td>
</tr>
<tr>
<td>Examples:</td>
<td><strong>Shortcuts</strong></td>
</tr>
<tr>
<td>chmod 777 - read, write, execute for all</td>
<td>Ctrl+C - halts the current command</td>
</tr>
<tr>
<td>chmod 755 - rwx for owner, rx for group and world</td>
<td>Ctrl+Z - stops the current command, resume with fg in the foreground or bg in the background</td>
</tr>
<tr>
<td>For more options, see man chmod</td>
<td>Ctrl+D - log out of current session, similar to exit</td>
</tr>
<tr>
<td>Sub</td>
<td>Ctrl+W - erases one word in the current line</td>
</tr>
<tr>
<td>ssh user@host - connect to host as user</td>
<td>Ctrl+U - erases the whole line</td>
</tr>
<tr>
<td>ssh -p port user@host - connect to host on port port as user</td>
<td>Ctrl+K - type to bring up a recent command</td>
</tr>
<tr>
<td>ssh-copy-id user@host - add your key to host for user to enable a keyed or passwordless login</td>
<td>11 - repeats the last command</td>
</tr>
<tr>
<td>Searchline:</td>
<td>exit - log out of current session</td>
</tr>
</tbody>
</table>
| grep pattern files - search for pattern in files | **Use with extreme caution.**
| grep -r pattern dir - search recursively for pattern in dir |
What is Unix?

- Operating System
Why Unix?

• Bioinformatics software designed to run on Unix platforms.
• Large amounts of data.
• Much faster than your Windows PC.
How Can We Use Unix?

• Linux computers or servers.
• Compute clusters.
• The cloud.
  – What we’re going to use this week
So What is Cloud Computing?
Cloud Computing Solutions

Amazon Web Services

Google Compute Engine

Microsoft Azure
AWS “Availability Zones” and Data Centres
How it Works

AMI ("Amazon Machine Image")
Base computer with all data and software
How it Works

Own copy of the AMI = Instance (Virtual Machine or VM)
Terminology

- Creating an instance – *buying a brand new computer with software already installed.*
- Starting an instance – *turning that computer on.*
- Stopping an instance – *turning that computer off.*
- Terminating an instance – *setting that computer on fire and throwing it out of the window.*
The Rules

• Only create one instance each.
• Stop your instance at the end of each day (unless you have software running).
• Name your instance (with YOUR name! No Bruce Waynes please)
• Only start or stop your own instance.
• Only terminate your own instance.
Connecting to Your Instance

Remote Desktop Software
e.g. X2Go

Secure Shell – “SSH”
e.g. SSH or PuTTY
Now What?!

• You’re each going to create, start and connect to your own instance.

***INSERT LIVE DEMO***
You’re now connected to your instance and you’re ready to learn some Unix!
Any Questions So Far?
The Terminal Window

The Command Line, The Shell, The Prompt

Where you see this “$” followed by text, I want you to type the text on your command line
Location is Important

First Task – Where am I?

$ pwd

This is your “present working directory”
genomics@harvard_ami:~$ pwd
/home/genomics
genomics@harvard_ami:~$
This location is also known as your Home Directory.

Tilde is shorthand for Home:

~
Now let’s create some directories and files

Make a directory

```
$ mkdir Data
```

Change into this directory

```
$ cd Data
```

Now what is your present working directory?

NOTE! Directory names (and file names for the matter) can not contain spaces. Underscores are often used instead if you want to separate words.
mkdir Data
cd Data
pwd
/home/genomics/Data
Now let’s create some directories and files

Make an empty file

$ touch rags

And another two

$ touch Earth Heaven

Now let’s list the contents of the current directory (Data)

$ ls
Now list ALL of the files

```
$ ls -a
```

genomics@harvard_ami:~:/Data$ ls -a
.
..  Earth  Heaven  rags

genomics@harvard_ami:~:/Data$
Now list ALL of the files

```
$ ls -a
.
..  Earth  Heaven  rags
```
These special files are in every directory:

- `..` Points to one directory above
- `.` Points to the current directory
These special files are in every directory:
- .. Points to one directory above
- . Points to the current directory
ABSOLUTE AND RELATIVE PATHS: GETTING FROM ONE PLACE TO ANOTHER
Absolute and Relative Paths

ABSOLUTE PATH

$ cd /home/genomics/

Moving from Data to More_Data

ABSOLUTE PATH

$ cd /home/genomics/More_Data
Absolute and Relative Paths

Moving from Data to More_Data

RELATIVE PATH

$ cd ../More_Data
Absolute and Relative Paths

Moving from Data to users

ABSOLUTE PATH

$ cd /home/

Data

More_Data

genomics

master

ubuntu

bin

lib
Absolute and Relative Paths

Moving from Data to users

RELATIVE PATH

$ cd ../..
Let’s put this to practice

Where am I right now? (Should be the Data directory)

$ pwd

Change to the directory above

$ cd ..

Let’s list the contents of the Data directory

$ ls ./Data
Let’s put this to practice

Where am I right now? (Should be the Data directory)

```
$ pwd
```

Change to the directory above

```
$ cd ..
```

Let’s list the contents of the Data directory

```
$ ls ./Data
```

**CHALLENGE 1!**

1. Move into the Data directory and list the contents of your home directory
2. In Data, make a new directory and move into this location
3. From this new directory, move into your home directory IN ONE COMMAND and check your location
If You’re Typing, You’re Doing Something Wrong!

Tab complete is a nice trick to save you typing paths

For this examples we are going to list everything in directory /var/run

Start by typing:

```
$ ls /
```

Followed by tab twice quickly

This shows the contents of the root directory
If You’re Typing, You’re Doing Something Wrong!

Now type:

```bash
$ ls /v
```

Followed by tab once. The path to the `/var/` directory has filled in.

```bash
$ ls /var/
```

Now type:

```bash
$ ls /var/r
```

Followed by tab once. The path to the `/var/run/` directory has filled in.

```bash
$ ls /var/run
```

Tab complete will fill in paths, save you time in typing and prevent typos!
If You’re Typing, You’re Doing Something Wrong!

Two more tricks for less typing!

* Represents a special character
  For example:
  
  $ ls /home/genomics/* .txt

  Will list everything in my home directory ending .txt

  The up arrow can be used to re-run commands

  Press your up arrow and see

  If you want all of these commands listed, simply type
  
  $ history
Any Questions So Far?
Binary programs

These are all programs installed on the Unix machine.

They can be found in /bin

```
$ ls /bin
```

These include pwd, mkdir, ls ...
Every binary program has a manual

To view the manual page, type man followed by the name of the program

$ man <PROGRAMME>

Open the manual page for ls

Scroll through (enter) and find the options for: long listing format, human-readable sizes and sort by modification time

Exit the manual page (type q) and give these ls options a go in your Data directory
Every binary program has a manual

To view the manual page, type `man` followed by the name of the program.

```
$ man <PROGRAMME>
```

Open the manual page for `ls`.

```
$ man ls
```

Scroll through (enter) and find the options for: long listing format (`-l`), human-readable sizes (`-h`), and sort by modification time (`-t`).

Exit the manual page (type `q`) and give these `ls` options a go in your Data directory.

```
$ ls -l OR $ ls -h OR $ ls -t
```
Examples!

First I need you to make a new directory called “Working” within your home directory.

Afterwards your file structure should look like this!
Moving Files

Let's move Heaven and Earth from Data to Working

```
$ cd ~/Data
$ mv Earth ../Working/
```

File  Location

Now move Heaven too

REMEMBER TAB COMPLETE!
Moving Files

mv can also be used to rename files
Let’s change rags to riches

$ mv rags riches

File
Location
Deleting Files

Now let’s delete Heaven
(Check your present working directory is Data)

$ rm -i ..:/Working/Heaven

When prompted type y for yes and press enter
Deleting Files

Now let’s delete the entire Working directory Including Earth

$ rm -r -i ../Working/
Deleting Files

Deleting files is very dangerous! There is no recycle bin in Unix! Once gone, files are gone for ever!

Therefore try to ALWAYS use `rm -i`
Let’s make a copy of riches within the home directory
(Make sure your present working directory is Data)

```
$ cp riches ../
```
Copying Files

You can also copy entire directories and use this function to rename files/directories

Move to home

```
$ cd ~
```

Make a copy of the Data directory here and call it Backup

```
$ cp -r ./Data ./Backup
```
Typical File Sizes

One Sequencing Sample on the Illumina NextSeq
3,000,000 reads = 1 Gb

But typically you will sequence more than one sample
You may have different patients, different locations, replicates etc...

The size of the sequencing data file can easily become 100s of Gb

(or even bigger depending on the sequencer used)
Archived/Compressed Files

Commonly, people will compress large files so that they are easier to store or share

Here’s an example:

sequences.tar.gz

.tar – means that it is a tape archive
.gz – means that it is gzipped

These can be used alone or in combination

To uncompressed
A Tar Archive

$ tar -xvf <filename>
(x = extract, v = verbose, f = all files)

A Gzipped file

$ gunzip <filename>

A Gzipped Tar archive

$ tar -xzvf <filename>
Any Questions So Far?
Challenge 2!

1. Change to the unix_workshop directory at the following path:

   `$ cd ~/workshop_materials/unix_workshop`

   You should find a compressed directory:

   `Sequences.tar`

2. Make a copy of this file in a Backup directory

3. Unarchive the directory

4. Unzip the read files

4. Rename the unarchived files – `sequence_1.fq` and `sequence_2.fq`

5. Delete the original .tar file

   `tar`  `gunzip`
   `cp`  `mv`
   `rm`  `mv`
   `cd`  `mkdir`
Paired Reads

Illumina Adaptors ➔ DNA for Sequencing ➔ Illumina Adaptors

Fragment
R1  Insert  R2
Forward Read  Insert  Reverse Read

An example: 300 bp paired end reads with a 700 bp fragment size
R1 = 300 bp, R2 = 300 bp, Insert = 100bp
# Looking at File Contents

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>head</strong></td>
<td>Shows the top lines of a file</td>
</tr>
<tr>
<td><strong>tail</strong></td>
<td>Shows the bottom lines of a file</td>
</tr>
<tr>
<td><strong>more</strong></td>
<td>Shows the file one full screen at a time</td>
</tr>
<tr>
<td><strong>less</strong></td>
<td>Shows the file one full screen at a time</td>
</tr>
<tr>
<td><strong>cat</strong></td>
<td>Shows an entire file all at once</td>
</tr>
</tbody>
</table>

- **n** specifies the number of lines (default 10)
- Enter to scroll one line
- Space to scroll a page
- q to quit
- Ctrl + C to stop

Use these command line programs to look at the sequence files
Let’s put this to use

Fastq File Format:
- Header
- Sequence
- Second Header (often +)
- Phred Quality Score

Lot’s of analysis software like paired reads to be in the same order

Use head to check that the top three headers are in the same order in sequence_1.fq and sequence_2.fq
Sequencing Stats

How many reads?
Count the number of lines

$ wc -l sequence_1.fq

742640 lines THEREFORE 185660 reads

Are there the same number of reverse reads?

How about just counting the header lines?
Each header line starts @E

$ grep -c "@E" sequence_1.fq

219153

BUT the numbers from the two programs don’t match?!

$ grep "@E" sequence_1.fq

How about with this

$ grep -c "^@E" sequence_1.fq

185660

^ matches this pattern at the start of the line – this is an example of a regular expression
Any Questions So Far?
AND NOW FOR A BRIEF SEGWAY INTO SCRIPTS...
Shell Scripts

Imagine you have a complicated command to run. Take this as an example:

`ref_map.pl -o ./stacks_gsnap/ -T 4 -O ./popmap -B middleton2_radtags -b 1 -s ./aligned_gsnap/s13_an_01.bam -s ./aligned_gsnap/s13_an_02.bam -s ./aligned_gsnap/s13_an_03.bam -s ./aligned_gsnap/s13_an_04.bam -s ./aligned_gsnap/s13_an_05.bam -s ./aligned_gsnap/s13_an_06.bam -s ./aligned_gsnap/s13_an_07.bam -s ./aligned_gsnap/s13_an_08.bam -s ./aligned_gsnap/s13_fw_01.bam -s ./aligned_gsnap/s13_fw_02.bam -s ./aligned_gsnap/s13_fw_03.bam -s ./aligned_gsnap/s13_fw_04.bam -s ./aligned_gsnap/s13_fw_05.bam -s ./aligned_gsnap/s13_fw_06.bam -s ./aligned_gsnap/s13_fw_07.bam -s ./aligned_gsnap/s13_fw_08.bam`

But what if you make a mistake?
Or want to run this command 10 times?
You have to type it out every time 😞
Shell Scripts

Instead we can put this command inside a script.

Then it can easily be edited and ran multiple times

To understand shell scripts, we’re going to look at a few topics:
  • Shell scripting languages
    • Text editors
  • How to write a script
  • How to run a script

Scripts make a great record of what you’ve done, when and with what. You should also aim to keep a computational biology lab book.
What is a Shell Script?

A computer program designed to be run by the Unix shell, the command line interpreter.

There are various types of shell scripts. These are scripting languages.

Today we are going to look at bash

First, let’s run a simple bash command:

```
$ echo Hello World
Hello World
```

Try using echo with a different phrase
Text Editors

These are pieces of software which can be used to write your script. Think of them as Unix versions of Notepad.

Some have an interactive user interface – E.G. gedit

Some work from within the command line – E.G. nano, vim, emacs

Today we are going to work with nano but have a play around with the others when you have a chance. Emacs and vim are notoriously difficult to use for the first time, so look up a cheat sheet.
Your First Script

Let’s start by opening nano

$ nano
Key Nano Commands

Ctrl + O – This saves the file. You will be asked for a file name. Type the name and press enter.

Ctrl + X – This exits nano. If the file is unsaved, you will be asked at this point if you’d like to save it.
Your First Script

#!/bin/bash tells the computer that this script is in the language bash. It always needs to go at the top of any bash script.

# Your script code goes here

echo "This is my first bash script."
Your First Script

#!/bin/bash

echo "This is my first bash script."

Then use Ctrl + O to save and give the file the name firstscript.sh.

Then use Ctrl + X to exit.
Now Run Your Script

Simply Type:

```bash
$ bash firstscript.sh
```

Reopen the same script:

```
$ nano firstscript.sh
```

Change the phrase, save it and run the script again
Bash Scripts

Bash scripts can be used to run binary programs like cd, mv, cp etc...

```bash
#!/bin/bash

echo "Hello Sophie"
echo "I'm going to rename some files"

mv File1.txt Results.txt

echo "Now I'm going to make back up copies"

mkdir Backup
cp Results.txt ./Backup/
```
PIPELINES
(TIME DEPENDING)
Pipelines

STDIN → PROGRAM 1 → STDOUT + STDERR
Pipelines

STDIN  PROGRAM 1  STDOUT

STDIN  PROGRAM 2  STDOUT

etc...
Pipelines

STDIN → PROGRAM 1 → PIPE | → PROGRAM 2 → STDOUT

etc...
Let’s put this to practice: Building Pipelines

Count the number of files and folders in your home directory

Let’s build the first part of the pipeline, listing the files:

```
$ ls -l /home/genomics/
```

PIPE this into `wc -l` to count the number of lines:
(i.e. the number of files and folders)

```
$ ls -l /home/genomics/ | wc -l
```
Let’s put this to practice: Building Pipelines

How many base pairs in first sequence?

Firstly let’s get the top two lines of the sequence file:

```
$ head -n 2 sequence_1.fq
```

Now let’s PIPE this into tail to get just the sequence line

```
$ head -n 2 sequence_1.fq | tail -n 1
```

Finally PIPE this into word count of characters to count the base pairs

```
$ head -n 2 sequence_1.fq | tail -n 1 | wc -c
```

Is the first reverse read the same length?
Some More Examples

Within the Unix Workshop directory you should find a file called scientists.txt

```
$ cd ~/workshop_materials/unix_workshop/

Take a look at the contents
```

```
$ more scientists.txt
```

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charles</td>
<td>Darwin</td>
<td>12 February 1809</td>
</tr>
<tr>
<td>Marie</td>
<td>Curie</td>
<td>07 November 1867</td>
</tr>
<tr>
<td>Stephen</td>
<td>Hawking</td>
<td>08 January 1942</td>
</tr>
<tr>
<td>Rosalind</td>
<td>Franklin</td>
<td>25 July 1920</td>
</tr>
<tr>
<td>Isaac</td>
<td>Newton</td>
<td>04 January 1643</td>
</tr>
<tr>
<td>Richard</td>
<td>Dawkins</td>
<td>26 March 1941</td>
</tr>
</tbody>
</table>
Some More Examples

$ cat scientists.txt | cut -f 1

<table>
<thead>
<tr>
<th>First</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charles</td>
</tr>
<tr>
<td>Marie</td>
</tr>
<tr>
<td>Stephen</td>
</tr>
<tr>
<td>Rosalind</td>
</tr>
<tr>
<td>Isaac</td>
</tr>
<tr>
<td>Richard</td>
</tr>
</tbody>
</table>

Now take a look at the original file

$ more scientists.txt
### Some More Examples

```bash
$ cat scientists.txt | cut -f 1,3
```

<table>
<thead>
<tr>
<th>First</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charles</td>
<td>12 February 1809</td>
</tr>
<tr>
<td>Marie</td>
<td>07 November 1867</td>
</tr>
<tr>
<td>Stephen</td>
<td>08 January 1942</td>
</tr>
<tr>
<td>Rosalind</td>
<td>25 July 1920</td>
</tr>
<tr>
<td>Issac</td>
<td>04 January 1643</td>
</tr>
<tr>
<td>Richard</td>
<td>26 March 1941</td>
</tr>
</tbody>
</table>
Some More Examples

$ cat scientists.txt | cut -f 1 | sort

<table>
<thead>
<tr>
<th>Charles</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
</tr>
<tr>
<td>Isaac</td>
</tr>
<tr>
<td>Marie</td>
</tr>
<tr>
<td>Richard</td>
</tr>
<tr>
<td>Rosalind</td>
</tr>
<tr>
<td>Stephen</td>
</tr>
</tbody>
</table>

What if you wanted to keep the sorted list?

$ cat scientists.txt | cut -f 1 | sort > newfile.txt

Redirects
CHALLENGE 3!

Looking at the *Saccharomyces cerevisiae* gff3 file

**GFF = general feature format**

This is a file which lists all of the genome features, their coordinates, and info about them (genes, tRNAs, exons etc...)

---

**SAMPLE GFF3 DATA**

```
##gff-version 3
##comment

---
```

---

---

---

---

### CHALLENGE 3!

Lines that start # are comments – just run information

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Feature/Type</th>
<th>Start Location</th>
<th>Stop Location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column 1 = Chromosome  
Column 3 = Feature/Type e.g. gene, chromosome, exon  
Column 4 = Start Location  
Column 5 = Stop Location
CHALLENGE 3!

1. In the Unix workshop directory you should find a gff3 file.

   $ cd ~/Unix_Workshop/Challenge5

   Saccharomyces_cerevisiae.R64-1-1.85.gff3.gz

2. Unzip the file.

3. How many feature entries are there?

4. List and count all the different types of features

5. Which chromosome is the longest?
Hints!

• Use head to work with 10 lines whilst testing what your pipe does!
  • This is a tab delimited file with a column layout.
  • Google “gff3 format” to find out what each of the columns are.
  • Remember that cat opens an entire file at once.
• There are a number of info lines at the start which begin with a hash. Look into grep with invert matches to skip these.
• Cut can be used to isolate certain columns. You’ll want the field option.
  • The programs sort and uniq may be helpful.
  • Sort must be used before uniq.
  • Uniq has a counting option.
  • Sort uses the key option to sort by a column.

more  gunzip  head  uniq
cp     mv     grep  wc
rm –i  mkdir  cut   |
cd     cat    sort  man
Any Questions So Far?