Unix for Bioinformaticians

A survival guide
Agenda

- Unix
- Folders
- Files
- Processes
- Redirection
Unix
What is Unix?
What is Unix?

Operating system stable, multi-user, multi-tasking for servers, desktops and laptops.
Types of Unix

- Solaris
- OS-X
- Linux!
Unix Operating System

Parts of the UNIX System

- Kernel
- Shell
- Tools & Apps
What’s in Unix?

Files (data)

Processes (actions)
Terminal

It makes the difference!

Powerful
Transparent
unfriendly
Folders: read and write
Change Directory

To know where you are
>pwd

Let's move!
>cd /home/people/
Or
>cd ..
Back home!
>cd
>cd ~
Folder content

Let's check what's in the folder:

>ls

Better

>ls -la

Cheat: parameters are useful!

>ls -lart
Wildcards

Extremely useful!

>ls -la *heavy*
>ls -la heavy.?asta

*= 0,1,2,... occurences of whatever
?= exactly 1 occurence of whatever
Commands

To know more about a command:

>man ls
>whatis ls
>apropos ls

Or Google!
Unix Permissions

- **type**: d
- **users**: rwx
- **group**: rwx
- **others**: rwx

**Can Execute, List file**: Can Execute, List file
**Can Write, Create files**: Can Write, Create files
**Can Read, Read files**: Can Read, Read files
**Can Execute, List file**: Can Execute, List file
**Can Write, Create files**: Can Write, Create files
**Can Read, Read files**: Can Read, Read files
**Can Execute, List file**: Can Execute, List file
**Can Write, Create files**: Can Write, Create files
**Can Read, Read files**: Can Read, Read files
**Is a symbolic link**: Is a symbolic link
**Is a directory**: Is a directory
## Folders - summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls</code></td>
<td>list files and directories</td>
</tr>
<tr>
<td><code>ls -a</code></td>
<td>list all files and directories</td>
</tr>
<tr>
<td><code>mkdir</code></td>
<td>make a directory</td>
</tr>
<tr>
<td><code>cd directory</code></td>
<td>change to named directory</td>
</tr>
<tr>
<td><code>cd</code></td>
<td>change to home-directory</td>
</tr>
<tr>
<td><code>cd ~</code></td>
<td>change to home-directory</td>
</tr>
<tr>
<td><code>cd ..</code></td>
<td>change to parent directory</td>
</tr>
<tr>
<td><code>pwd</code></td>
<td>path of the current directory</td>
</tr>
</tbody>
</table>
Files: read and write
Copy the data

Create a directory

>mkdir immu00

and copy the files

>cp -R /home/peopme/pmar/immu00/* immu00/

>cd immu00
Backup original data

Create a directory
>mkdir backup

Copy all the files in the directory
>cp heavy.fasta backup/
>cp heavy.fasta heavy_copy.fasta
>mv heavy_copy.fasta backup/
heavy_backup.fasta
Read a file

Let's see…

>cat heavy.fasta
Read a file

Let’s see...
>cat heavy.fasta

Mmh...
>less heavy.fasta

**Hint**: q to quit...

but if you are stuck try `ctrl+c`
Look for text

If we want to look for something while in less:

/  
CAR (enter)

another search
CAA (enter)
n (next match)
Look for text

How many lgs in each file?
We can count the lines!
>wc heavy.fasta
Grep

# of proteins = # of “>” in a file

>grep “>” heavy.fasta
>grep -c “>” heavy.fasta
Files: summary

- `cp file1 file2`: copy file1 and call it file2
- `mv file1 file2`: move or rename file1 to file2
- `rm file`: remove a file
- `cat file`: display a file
- `less file`: display a file a page at a time
- `head file`: display the first few lines
- `tail file`: display the last few lines
- `grep 'key' file`: search a file for keywords
- `wc file`: number of lines
Processes
Run a process

Process = execution of some instructions
Usually the instructions are in a file.

e.g. less -> /usr/bin/less

So

>/usr/bin/less heavy.fasta
Run - It’s simple

Ok, so let’s try

>./loop.pl
Run - It’s simple

Ok, so let’s try

>./loop.pl

Ok, and now?
Controlling processes

Ctrl+z -> go to sleep

Now you’ve got the control again!
But it’s not still dead…

bg

Ctrl+C
Controlling processes

Or:
>./loop.pl
Ctrl+z
>ps
>top (q to exit)
>kill the_number_that_you_have_just_read
Redirection

Control the force
Inputs and outputs

- Keyboard
- Mouse
- Tablet
- Display
- Printer
- File
Write into a file

`>Cat > list.txt`

Heavy 29061
Ctrl+d

And
`>less list.txt`
Append to a file

> Cat >> list.txt

Kappa 7476
Ctrl+d

And
> less list.txt
The big one!

>cat heavy.fasta > bigone.fasta

>cat lambda.fasta >> bigone.fasta

>grep -c ">" bigone.fasta
Extract headers

>grep “>” bigone.fasta > headers.head
>sort headers.head > headers_sort.head

Do we need the first file?
Extract headers

>grep ">" bigone.fasta > headers.head
>sort headers.head > headers_sort.head

Do we need the first file? No!
>grep ">" bigone.fasta | sort > headers_sorted.head
| is called pipe, and it’s something
Redirect everything!

```bash
>grep "->" biggone.fasta 2> err.log
>less err.log
```
Redirect - summary

cmd > file  redirect std output to a file
cmd >> file  append std output to a file
cmd < file  redirect std input from a file
cmd1 | cmd2  pipe the output of cmd1 to the input of cmd2
cat f1 f2 > f0  concatenate f1 and f2 to f0
sort  sort data
Redirect - summary

command > file   redirect std output to a file
command >> file  append std output to a file
command < file   redirect std input from a file

cmd1 | cmd2  pipe the output of cmd1 to the input of cmd2

cat f1 f2 > f0  concatenate f1 and f2 to f0

sort             sort data
Quiz

- What does this command perform? 
  `>grep -v "">" heavy.fasta`
- How many proteins in each file?
- How many residues in each file? 
  (hint: `wc -m` counts the # of char in a line)
- Average length of proteins in each file?
- Average content of Prolines in `heavy.fasta`? 
  (hint: look at grep parameters)
Regular Expressions in a nutshell
Regular Expressions

What if we want to do something like “check for a valid email address”…
Regular Expressions

What if we want to do something like “check for a valid email address”…

1. There must be some letters or numbers
2. There must be a @
3. Other letters
4. .something

paolo.marcatili@gmail.com is good

paolo.marcatili@.com is not good
Regular Expressions

/\^ATOM/\nLine starts with ATOM

/\^ATOM\s+/\nLine starts with ATOM, then there are some spaces

/\^ATOM\s+[\-][0-9]+/\nLine starts with ATOM, then there are some spaces, then there are some digits or –

/\^ATOM\s+\?-[0-9]+/\nLine starts with ATOM, then there are some spaces, then there can be a minus, then some digits
Regular Expressions

```
cat | awk '/^[a-z1-9._]+@[^\.@]+\.[a-z]+$/{print "Ok\n"}'
```

This means:
Check if the input string has some chars at the beginning, then @,
then some characters (no point), then a point, then some letters
Regular Expressions

The pattern has to appear at the beginning of a string.
- The pattern has to appear at the end of a string.
- Matches any character.
- Bracket expression. Matches one of any characters enclosed.
- Negates a bracket expression. Matches one of any characters EXCEPT those enclosed.
- Range. Matches any characters within the range.
- Preceding item must match one or zero times.
- Preceding item must match one or more times.
- Preceding item must match zero or more times.
- Parentheses. Creates a substring or item that metacharacters can be applied to.
- Bound. Specifies exact number of times for the preceding item to match.
- Bound. Specifies minimum number of times for the preceding item to match.
- Bound. Specifies minimum and maximum number of times for the preceding item to match.
- Alternation. One of the alternatives has to match.

*cat matches any string that begins with cat
- cat matches any string that ends with cat
- cat matches cat and cat2 but not caty
- gr[a|y] matches gray OR grey
- [^02] matches 13 but not 10 or 12
- [1-9] matches any single digit EXCEPT 0
colour matches color OR colour but not colour
be*r matches be or bee but not b
be* matches b or be or beeeeee
a(bee)? matches at or abed but not abet
[0-9] matches any three digits
[0-9]{3} matches any three or more digits
[0-9][3,5] matches any three, four, or five digits
July (first|last|1) will match July 1st but not July 2

POSIX Character Classes

- alphanumeric character
- alphabetic character, any case
- space and tab
- digits
- lowercase alphabetic
- punctuation characters
- all whitespace characters, including newline and carriage return
- uppercase alphabetic

Peri-Style Metacharacters

// Default delimiters for pattern
/ A word boundary, the spot between word (\w) and non-word (\W) characters
\b A non-word boundary
\d A single digit character
\D A single non-digit character
\n The newline character. (ASCII 10)
\v The carriage return character. (ASCII 13)
\t A single whitespace character
\s A single non-whitespace character
\r The tab character. (ASCII 9)
\w A single word character - alphanumeric and underscore
\W A single non-word character

/colour/ matches color OR colour
/colour/l matches color OR colour
/Fred/b matches Fred but not Alfred
/Alfred/b matches Fred but not Alfred
/Fred\bmatches Fred but not Fred
/a\d/b matches a2b but not a0b
/a\d0/b matches a2b but not a0b
/a\d\b matches a2b but not a0b
/\d\b matches a2b but not a0b
/\d\b matches a2b but not a0b
/\d\b matches a2b but not a0b
/\d\b matches a2b but not a0b
/\d\b matches a2b but not a0b