Practical Bioinformatics

Basic Linux

Part 3

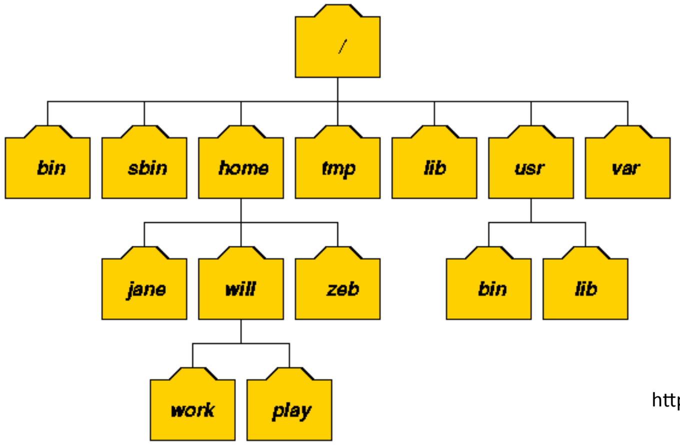
Stefan Wyder
stefan.wyder@uzh.ch
URPP Evolution
www.evolution.uzh.ch

Universität Zürich^UZH

URPP

Shell - the iron ration 1

- Directory structure
  everything is under the root: /



http://www.doc.ic.ac.uk/~wjk/

/bin
/var
/home
/home/wyder

- Absolute and relative paths
  cd /home/wyder/ vs cd ..

  ~ for home
  .. for the directory up 1 level
  . for the current directory

```
mv *.txt ~/data/
mv *.txt ..
```

- The Mac OS X shell differs from the typical Linux shell
  (directory structure, less powerful BSD commands, end-of-line \r)

Shell - the iron ration 2

- Many tools that only do 1 thing
- Command -Option(s) Parameter(s)
  ls -lh ~/data
- Working with files & directories
  ls, cd, mkdir, cp, mv, rm
- Working with text files
  head, tail, less, cat, grep, cut, sort, tr, wc, uniq
- Pipe output to another command with |

  ```
  ls | wc -l
  ```
- Send output to a file

  ```
  ls > ListofFiles.txt
  ls .. >> ListofFiles.txt
  ```

What we will be learning today

• Automatizing tasks (scripting)	Heidi
• Making use of multiple cores	Heidi & Stefan
• Introduction into awk	Stefan
• Search patterns: regular expressions	Stefan

awk

Several kinds of tasks occur repeatedly when working with text files. You might want to extract certain lines and discard the rest. Or you may need to make changes wherever certain patterns appear, but leave the rest of the file alone. Writing single-use programs for these tasks in languages such as C, C++, or Java is time-consuming and inconvenient. Such jobs are often easier with awk. The awk utility interprets a special-purpose programming language that makes it easy to handle simple data-reformatting jobs.

https://www.gnu.org/software/gawk/manual/html_node/Preface.html

awk

- A small, fast and simple programming language for text processing
- meant for processing column-oriented data (e.g. tables):
  compare, replace, filter, modify, ... text files
- works well together with other shell tools (piping)
- can handle very large files
- you could also use a full programming language (e.g. python or perl)

awk: blocks

pattern {action}

awk 'pattern {action}' FILENAME

```
awk '$1=="chr1" {print $0}' file.gff
```

1. Read in file line by line:
   $1: first column, $2: second, ...
   $0: whole line
2. For each line check if pattern is true then do action

multiple patterns

awk 'pattern1 {action1} pattern2 {action2} ' FILENAME

awk is a Unix filter

```
zcat file.gff.gz | awk '$1=="chr1" && $2>30 && $2<400 {print $0} ' | wc -l
```

Regular Expressions (regex/regexp)

- a way to describe *set of strings*

- can be used to find/replace patterns and to extract/parse
  (format conversion, parsing, format checking)

- available in
  - many Linux tools (g**re**p, sed, awk)
  - most programming languages
  - many text editors/OpenOffice

- case-sensitive!

- the simplest regular expressions are literal characters:
  the pattern N matches the character 'N'
  the pattern Nick matches 'Nick'

Wildcards

5th
3rd
2nd
4th

A wildcard is a special character that represents a specific set of character
\w: matches any letter (A-z) or digit (0-9) or underscore (_) [A-z0-9_].

\w\w\w

Regexps are **non-overlapping**
(\w\w would match 5t 3r 2n)

Capturing text with ()

5th
3rd
2nd
4th

Search:	(\w)\w\w	Capture portions of the search with ()
Replace:	\1	Reuse captured text with \1

5
3
2
4

Matching once or more

\w+ matches until the next non-word character (e.g. space, punctuation, end of line)

```
Agalma elegans
Frillagalma vitiazi
Mus musculus
```

Search: (\w)\w+ (\w+)

Replace: \1. \2

```
A. elegans
F. vitiazi
M. musculus
```

Shortcuts

	means
.	any character [-.?+%$A-Za-z0-9...]
\d	digit [0-9]
\w	word character (alphanumerics or underscore)
\s	white space (space, tab, end-of-line)
\S	complement of \s: any non-whitespace character
\t	tab

[Nn]ick matches 'Nick', 'nick'
[Nn]+ick matches 'Nick', 'nick', 'NNick', 'Nnick', 'nnick', 'NNNick', ...
[Nn]{2}ick matches 'Nick', 'nick', 'NNick', 'Nnick', 'nnick'

Quantifiers

	means
*	zero or more times
+	one or more times
{n}	exactly n times
{m,n}	at least m times but no more than n times

[Nn]ick matches 'Nick', 'nick'
[Nn]+ick matches 'Nick', 'nick', 'NNick', 'Nnick', 'nnick', 'NNNick', ...
[Nn]{2}ick matches 'Nick', 'nick', 'NNick', 'Nnick', 'nnick'

Character classes []

	Matches
[abcde]	exactly one of the characters listed
[a-e]	exactly one character in the given range
[!abcde]	any character not listed
[!a-e]	any character that is not in the given range
{URPP,evolution}	exactly one entire word from the options given

Range limits are defined according to the ASCI values

[Nn]ick matches 'Nick' or 'nick'

Regexps match the first instance

```
Agalma,A. elegans,hydrozoan,316164
Frillagalma,F. vitiazi,hydrozoan,645341
Mus,M. musculus,rodent,10088
```

```
([^,]+),([^,]+),[^,]+,([^,]+)
\3\t\1\t\2
```

```
hydrozoan  Agalma       316164
hydrozoan  Frillagalma  645341
rodent     Mus          10088
```

4 columns: this regexp will match all 4

5 columns: leaves 5th column untouched

<4 columns: no match

8 columns: this regexp will match twice

* and + are greedy

They match the maximum number of characters they can (from left to right)

abcdefgabc

Search: (a.*c)
Replace: \1

abcdefgabc **NOT abc !!**

Use the lazy quantifier '?' so that the expression tries the minimal match first

Search: (a.+?c)
Replace: \1

abc

Some examples

Regex	chr	chr[1-5]	chr.	AAF12\.[1-3]	AT[1,5]G[:digit:]+\.[1,2]
	chr1	chr1	chr1	AAF12.1	AT5G08160.1
	chr2	chr2	chr2	AAF12.2	AT5G08160.2
	chr3	chr3	chr3	AAF12.3	AT5G10245.1
	chr4	chr4	chr4		AT1G14525.1
	chr5	chr5	chr5		
	chr6		chr6		

Parallelizing jobs

- **GNU parallel**

  ```
  find *.bam | parallel samtools index {}
  ```

  elegant
  can handly any number of jobs
  even on remote computers

- Heidi presented another way today by submitting a controlled number of jobs in the bg (&)

See exercises

Sources & Links

General (incl Linux, Python, regexps, databases)

- Haddock&Dublin. Practical Computing for Biologists. Sinauer Associates 2011.
- Tips for Mac Users http://practicalcomputing.org/
- Cheatsheet practicalcomputing.org/files/PCfB_Appendices.pdf

awk

- to learn http://www.grymoire.com/Unix/Awk.html
- comprehensive manual http://www.gnu.org/software/gawk/manual/gawk.html
- example one-liners http://www.pement.org/awk/awk1line.txt

regular expressions

- online tool to build&learn http://www.regexr.com/
- Cheatsheet practicalcomputing.org/files/PCfB_Appendices.pd
- http://stackoverflow.com/questions/4736/learning-regular-expressions

GNU parallel

- http://www.gnu.org/software/parallel
- https://www.biostars.org/p/63816/

Tips&Tricks for using the shell on Mac OS

- http://furbo.org/2014/09/03/the-terminal/