Linux II

Douglas Scofield
Evolutionary Biology Centre and UPPMAX
douglas.scofield@ebc.uu.se

Creating directories and files

- mkdir

```
milou-b: ~ $ cd course
-bash: cd: course: No such file or directory
milou-b: ~ $ mkdir course
milou-b: ~ $ cd course
```

Creating directories and files

- touch
  - if file does not exist, creates it with 0 size
  - if file exists, update its modification time

```
milou-b: ~/course $ touch a
milou-b: ~/course $ ls -l
total 0
-rw-rw-r-- 1 douglas douglas 0 Aug 24 11:00 a
milou-b: ~/course $ date
Sun Aug 24 11:01:01 CEST 2014
milou-b: ~/course $ touch a
milou-b: ~/course $ ls -l
total 0
-rw-rw-r-- 1 douglas douglas 0 Aug 24 11:01 a
```

```
milou-b: ~/course $ touch b c d
milou-b: ~/course $ ls -l
total 0
-rw-rw-r-- 1 douglas douglas 0 Aug 24 11:01 a
-rw-rw-r-- 1 douglas douglas 0 Aug 24 11:03 b
-rw-rw-r-- 1 douglas douglas 0 Aug 24 11:03 c
-rw-rw-r-- 1 douglas douglas 0 Aug 24 11:03 d
```

Creating directories and files

- cat
  - con"cat"enate : dumps the contents of a file
  - can also be used to quickly create a short file

type this, then Return, then

Ctrl-D = EOF = "End Of File"

```
milou-b: ~/course $ cat > e
this is a short file
milou-b: ~/course $ cat e
this is a short file
milou-b: ~/course $ cat a
milou-b: ~/course $ ls -l
total 32
-rw-rw-r-- 1 douglas douglas  0 Aug 24 11:01 a
-rw-rw-r-- 1 douglas douglas  0 Aug 24 11:03 b
-rw-rw-r-- 1 douglas douglas  0 Aug 24 11:03 c
-rw-rw-r-- 1 douglas douglas  0 Aug 24 11:03 d
-rw-rw-r-- 1 douglas douglas 21 Aug 24 13:14 e
```

Redirecting input: <

- *command* < file
  - give *command* input from 'file'
  - for *command*, input comes from 'standard input', 'stdin'

```
milou-b: ~/course $ cat < e
this is a short file
milou-b: ~/course $
```

  - for flexible commands, it is not quite the same as giving a file on the command line

```
milou-b: ~/course $ cat e
this is a short file
milou-b: ~/course $ cat e e
this is a short file
this is a short file
milou-b: ~/course $ cat e e e
this is a short file
this is a short file
this is a short file
milou-b: ~/course $
```

Redirecting input: <

- *command* < file
  - Files and stdin generally cannot be mixed using '<'

```
milou-b: ~/course $ cat e e < e
this is a short file
this is a short file
```

  - command line files can be more flexible

```
milou-b: ~/course $ cat > f
this file is a little longer
milou-b: ~/course $ cat f
this file is a little longer
milou-b: ~/course $ cat e f
this is a short file
this file is a little longer
```

  - Some commands understand the filename '-' to mean 'read from stdin'

```
milou-b: ~/course $ cat f - < e
this file is a little longer
this is a short file
milou-b: ~/course $
```

Here, cat first reads f, then reads stdin, which has e

Redirecting output: > and >>

- *command* > file
  - 'standard output, 'stdout', for *command* goes to 'file'

```
milou-b: ~/course $ cat e > ee
milou-b: ~/course $ cat ee
this is a short file
milou-b: ~/course $ ls -l e ee
-rw-rw-r-- 1 douglas douglas 21 Aug 24 13:14 e
-rw-rw-r-- 1 douglas douglas 21 Aug 24 17:00 ee
milou-b: ~/course $
```

- command >> file
  - ***appends*** stdout from *command* to 'file'

```
milou-b: ~/course $ cat f >> ee
milou-b: ~/course $ cat ee
this is a short file
this file is a little longer
milou-b: ~/course $ ls -l e f ee
-rw-rw-r-- 1 douglas douglas 21 Aug 24 13:14 e
-rw-rw-r-- 1 douglas douglas 50 Aug 24 17:04 ee
-rw-rw-r-- 1 douglas douglas 29 Aug 24 13:47 f
milou-b: ~/course $
```

Connecting stdout to stdin with the pipe: |

Swedish Mac: Alt-7

- *command1* | *command2*
  - stdout of *command1* is connected to stdin of *command2*

```
milou-b: ~/course $ cat f ee | cat > ff
milou-b: ~/course $ cat ff
this file is a little longer
this is a short file
this file is a little longer
milou-b: ~/course $ cat f | wc -l
1
milou-b: ~/course $ cat ee | wc -l
2
milou-b: ~/course $ cat ff | wc -l
3
```

- 'wc -l' counts the number of lines in stdin, or a file or set of files, and prints the result to stdout
- 'wc' counts the number of lines, words and characters

```
milou-b: ~/course $ cat ff | wc
       3      17      79
```

The "other" output: standard error, 'stderr'

- Commands which use pipes or redirect stdout may still produce output to the terminal:

```
milou-b: ~/course $ bwa mem -a -E3 -t 8 ref.fa reads.fq | samtools view -Sb - > aln.bam
[M::main_mem] read 32 sequences (4916 bp)...
[M::mem_process_seqs] Processed 32 reads in 0.014 CPU sec, 0.005 real sec
[main] Version: 0.7.10-r789
[main] CMD: bwa mem -a -E3 -t 8 ref.fa[samopen] SAM header is present: 1 sequences.
 reads.fq
[main] Real time: 0.056 sec; CPU: 0.019 sec
```

- This is a second output stream: standard error, 'stderr'
- Not all tools use it
- To capture standard error, use **2>** (for stdout, **1>** equals **>**)

```
milou-b: ~/course $ bwa mem -a -E3 -t 8 ref.fa reads.fq 2> bwa.stderr | samtools view
-Sb - > aln.bam
[samopen] SAM header is present: 1 sequences.
milou-b: ~/course $ bwa mem -a -E3 -t 8 ref.fa reads.fq 2> bwa.stderr | samtools view
-Sb - > aln.bam 2> samtools.stderr
milou-b: ~/course $
```

Putting stderr together with stdout

- To capture *all* output of a command, not just stdout or stderr
- Reassign stderr to be directed to stdout (or vice versa) and then capture the combined output stream

command > file **2>&1** To append: >> file 2>&1

  - when directing to a file, order is important: 2>&1 *after* >file

- When piping between tools, this is usually not a good idea because downstream tools usually expect one output stream or the other, but not both

```
milou-b: ~/course $ bwa mem -a -E3 -t 8 ref.fa reads.fq 2>&1 | samtools view -Sb - > aln.bam
[samopen] no @SQ lines in the header.
[sam_read1] missing header? Abort!
milou-b: ~/course $
```

This sends both stdout and stderr through the pipe.

With Bash 4+, you can use '&>' and '&>>' to redirect/append both to a file, and '|&' as a pipe that redirects both, but these are not portable so don't use them.

Shell wildcards: ? *

- Using wildcards, filenames can be specified using expressions
- 0, 1 or more than 1 filename may match the expression
- Bash wildcards are similar but not identical to grep, sed, etc.

```
milou2: ~/course $ ls
a  b  c  d  e  ee  f  ff
```

- '?' matches any 1 character

```
milou2: ~/course $ ls ?
*  a  b  c  d  e  f
milou2: ~/course $ ls f?
ff
```

- '*' matches 0 or more of any character

```
milou2: ~/course $ ls e*
e  ee
milou2: ~/course $ ls *
*  a  b  c  d  e  ee  f  ff
```

Quote to match literally:

```
milou2: ~/course $ touch "*"
milou2: ~/course $ ls "*"
*
```

Shell wildcards: character groups with [ ] - ^

- You can specify character groups using **[ ] - ^**

```
milou2: ~/course $ ls
*  a  b  c  d  e  ee  f  ff
```

Match specific character: **[a]** Two characters: **[af]** Range: **[a-f]**

```
milou2: ~/course $ ls [a]
a
milou2: ~/course $ ls [af]
a  f
milou2: ~/course $ ls [a-f]
a  b  c  d  e  f
milou2: ~/course $ ls [a-f]?
ee  ff
```

Anything but specific characters: **[^a]** **[^a-d]**

```
milou2: ~/course $ ls [^a]
*  b  c  d  e  f
milou2: ~/course $ ls [^a-d]
*  e  f
milou2: ~/course $ ls [^a-d]?
ee  ff
```

Shell wildcards: locale settings affect ordering

- Linux locales specifies language, numeric, monetary, sort, and other conventions
- List available locales with **locale -a**
- The results of wildcards are affected by locale

```
rackham3: ~/course $ export LC_ALL="C"
rackham3: ~/course $ locale > /dev/null
rackham3: ~/course $ ls *
A  B  a  b  c  d  e
rackham3: ~/course $ export LC_ALL="en_GB.utf8"
rackham3: ~/course $ locale > /dev/null
rackham3: ~/course $ ls *
a  A  b  B  c  d  e
```

```
rackham3: ~/course $ locale
LANG=en_US.utf-8
LC_CTYPE="C"
LC_NUMERIC="C"
LC_TIME="C"
LC_COLLATE="C"
LC_MONETARY="C"
LC_MESSAGES="C"
LC_PAPER="C"
LC_NAME="C"
LC_ADDRESS="C"
LC_TELEPHONE="C"
LC_MEASUREMENT="C"
LC_IDENTIFICATION="C"
LC_ALL=C
```

> /dev/null sends the standard output of **locale** to 'nowhere'

- Including the contents of character ranges

; separates commands within the same line

```
rackham3: ~/course $ export LC_ALL="C"; locale > /dev/null
rackham3: ~/course $ ls [a-d]
a  b  c  d
rackham3: ~/course $ export LC_ALL="en_GB.utf8"; locale > /dev/null
rackham3: ~/course $ ls [a-d]
a  A  b  B  c  d
```

Shell wildcards: groups of terms using { , }

- You can specify term groups using **{ , }**

```
milou-b: ~/course $ ls
a  b  c  d  e  ee  f  ff
```

There must be at least two terms: **{e,f}**

```
milou-b: ~/course $ ls {e,f}
e  f
```

Terms can contain wildcards: **{c,??}**

```
milou-b: ~/course $ ls {c,??}
c  ee  ff
```

Using terms can save a lot of typing:

```
milou-b: ~ $ cp /some/really/long/directory/run-1.{log,out,err,pdf,sh} .
```

Man(ual) pages

- Uncertain about a command, or want to know what more it can do?
- Type 'man *command*'

- man wc

```
WC(1)                          User Commands                          WC(1)

NAME
       wc - print newline, word, and byte counts for each file

SYNOPSIS
       wc [OPTION]... [FILE]...
       wc [OPTION]... --files0-from=F

DESCRIPTION
       Print  newline,  word, and byte counts for each FILE, and a total line if more than
       one FILE is specified.  With no FILE, or when FILE is -, read standard input.

       -c, --bytes
              print the byte counts

       -m, --chars
              print the character counts

       -l, --lines
              print the newline counts

       --files0-from=F
              read input from the files specified by NUL-terminated names in file F; If  F
              is - then read names from standard input

       -L, --max-line-length
              print the length of the longest line

       -w, --words
              print the word counts

       --help display this help and exit

       --version
              output version information and exit

AUTHOR
       Written by Paul Rubin and David MacKenzie.
```

Man pipes output through **less**. Press SPACE to continue, 'q' at any time to quit, and 'h' to get help on searching, etc.

Using ‘find’ to search a directory tree

- **find** *location list-of-file-attributes optional-actions*

```
rackham3: ~/course $ mkdir directory
rackham3: ~/course $ mv ? ?? directory/
rackham3: ~/course $ find . -name d
./directory/d
rackham3: ~/course $ find . -name '??'
./directory/ee
./directory/ff
rackham3: ~/course $ find . -iname B -ls
470961739    0 -rw-rw-r--   1 douglas  douglas         0 Jan 13 14:29 ./directory/b
rackham3: ~/course $ find . -type d
.
./directory
rackham3: ~/course $ find . -type f -name '*.fa' -exec grep '^>seq3$' {} \;
>seq3
rackham3: ~/course $ find . -type f -name '*.fa' -exec grep -Hn '^>seq3$' {} \;
./ref.fa:19:>seq3
```

Most wildcards work, use within quotes

The `-iname` option is case-insensitive and `-ls` runs ‘`ls -l`’ on each file

Look for specific type of file

Run commands on found files

- Other options for size, ownership, modification times, etc.
- See the (long) man page and online tutorials for more

Create symbolic links to clear things up

- Use ‘ln -s’ … do not forget the ‘-s’ !
- Symbolic links indicate the location of another file/directory

```
milou-b: ~/course $ ln -s f sf
milou-b: ~/course $ ls -li f sf
1105098318 -rw-rw-r-- 1 douglas douglas 22 Jan 27  2015 f
1915648234 lrwxrwxrwx 1 douglas douglas  1 Aug 22 11:37 sf -> f
```

‘Hard links’ (ln *without* -s) are rarely necessary

- Hard links are truly another name for the same file

```
milou-b: ~/course $ ln f hf
milou-b: ~/course $ ls -li f hf sf
1105098318 -rw-rw-r-- 2 douglas douglas 22 Jan 27  2015 f
1105098318 -rw-rw-r-- 2 douglas douglas 22 Jan 27  2015 hf
1915648234 lrwxrwxrwx 1 douglas douglas  1 Aug 22 11:37 sf -> f
milou-b: ~/course $ rm f
rm: remove regular file `f'? y
milou-b: ~/course $ ls -li f hf sf
ls: cannot access f: No such file or directory
1105098318 -rw-rw-r-- 1 douglas douglas 22 Jan 27  2015 hf
1915648234 lrwxrwxrwx 1 douglas douglas  1 Aug 22 11:37 sf -> f
milou-b: ~/course $ mv hf f
milou-b: ~/course $ ls -li f hf sf
ls: cannot access hf: No such file or directory
1105098318 -rw-rw-r-- 1 douglas douglas 22 Jan 27  2015 f
1915648234 lrwxrwxrwx 1 douglas douglas  1 Aug 22 11:37 sf -> f
```