$ Welcome to the Matrix Unix/Linux II

Computational Biology

Lecture 2

Dr. Chris Bird

Substituting Characters Using & Predefined Characters

tr

[:upper:]

```
# change all a to b
$ echo "aaaabbb" | tr "a" "b"
bbbbbbb

$ echo "123456789" | tr 1-5 0
000006789

$ echo "ACtGGcAaTT" | tr actg ACTG
ACTGGCAATT

$ echo "ACtGGcAaTT" | tr [:lower:] [:upper:]
ACTGGCAATT

$ echo "aabbccddee" | tr a-c 1-3
112233ddee
```

Substituting Characters Using & Predefined Characters

tr

[:upper:]

```
# delete all occurrences of a
$ echo "aaaaabbbb" | tr -d a
bbbb

# remove consecutive duplicate occurrences of a
$ echo "aaaaabbbb" | tr -s a
Abbbb

# move to sandbox and list files
cd ../sandbox; ls
```

; is equivalent to end of line

tr does not accept a file as an argument, always use pipe

Make a new file BodyMass.csv in sandbox dir based on Pacifici2013_data.csv, columns 2-6, remove header, sort lines according to body mass (large to small), change ; to spaces

```
# 1. View header row to refresh your memory
$ head -n1
# 2. Start building pipe, use less to view
$ cut -d ";" -f2-6 ../data/Pacifici2013_data.csv | less -S
# 3. Add to pipe, use less to view
$ cut -d ";" -f2-6 ../data/Pacifici2013_data.csv | \
> tr ";" "\t" | less -S
# 4. Add to pipe, figure out sort options, use less to view
$ cut -d ";" -f2-6 ../data/Pacifici2013_data.csv | \
> tr ";" " " | tail -n+2 | sort -nrk6 | less -S
# 5. Instead of piping to less, redirect output to file
$ cut -d ";" -f2-6 ../data/Pacifici2013_data.csv | \
> tr ";" " " | tail -n+2 | sort -nrk6 > BodyMass.csv
```

\ is actually the “escape character”, what follows the \ is treated differently

Wildcards are Symbols that Represent Multiple Characters

*

- Zero or more characters, except leading dot

?

- Any single character, except leading dot

```
# goto miRNA dir inside data dir
$ cd ~/CSB/unix/data/miRNA

# count the numbers of lines in all the .fasta files
$ wc -l *.fasta

# print the first two lines of each file
# whose name starts with pp
$ head -n 2 pp*

# determine the type of every file that has
# an extension with exactly three letters
$ file *.???
```

Selecting lines with matching pattern using
grep [options] [pattern] filename

- Every line that matches pattern is returned
- Many options to increase functionality
- Regular Expressions are used for pattern matching in text files
  - A language of wildcards
  - 2 syntaxes: POSIX, Perl

```
$ cd ~/CSB/unix/sandbox

# how many wombats (fam Vombatidae)?
$ grep "Vombatidae" BodyMass.csv
$ grep -c "Vombatidae" BodyMass.csv

# which cattle are in file?
$ grep "Bos" BodyMass.csv

# Only match whole words
$ grep –w "Bos" BodyMass.csv

# Make search case insensitive
$ grep –i "Bos" BodyMass.csv
```

Selecting lines with matching pattern using

grep [options] [pattern] filename

```
# which mammals have body weight most similar to the gorilla?
# option –B lines before match, option –A lines after match
$ grep -B 2 -A 2 "Gorilla gorilla" BodyMass.csv
# show line number of gorilla
$ grep -n "Gorilla gorilla" BodyMass.csv

# -v means match anything except pattern
$ grep Gorilla BodyMass.csv | grep -v gorilla

# return all lines with Gorilla or Pan, note use of escape char \
$ grep -w "Gorilla\|Pan" BodyMass.csv

# return all lines with Gorilla for all files in data dir
# and it’s subdirs. -r recursive, searches subdirs
$ grep -r "Gorilla" ../data
```

Searching for files with `find [dir] [options] [pattern]`

```
# current directory is the unix sandbox
$ find ../data
# how many files are in data?
$ find ../data | wc -l
# find file named n30.txt in data
$ find ../data -name "n30.txt"
# use wildcards to find all files in data that contain about
$ find ../data -iname "*about*“
# count all files that end in .txt in data, then
# do same but don’t include subdirs
$ find ../data -name "*.txt" | wc –l
$ find ../data -maxdepth 1 -name "*.txt" | wc –l
# count files in data that don’t include about
$ find ../data -not -name "*about*" | wc –l
# find directories with data in path or name
$ find ../data -type d
```

Mind Expander 1.4

Computational Biology

Lecture 1

Dr. Chris Bird

Permissions

- Three types of permissions
  - Read, Write, Execute
  - Program won't run if x is not set
- Three types of users
  - User, Group, Global
- View with `ls -l`
- Change with `chmod`

```
-rwxrwxrwx 1 cbird cbird 515 Jul 10  2018
-rw-rw-rw- 1 cbird cbird 146 Jul 10  2018
-rw-rw-rw- 1 cbird cbird  39 Aug  2  2018
-rw-rw-rw- 1 cbird cbird  42 Jan 11  2019
-rw-rw-rw- 1 cbird cbird  61 Feb 15  2019
-rw-rw-rw- 1 cbird cbird  93 Jun 21 06:46
drwxrwxrwx 1 cbird cbird 512 Aug 24 10:57
drwxrwxrwx 1 cbird cbird 512 Aug 24 11:25
(base) cbird@LAPTOP-URS0LRPO:~$
```

Interpreting Output of `ls -l`

Permissions

	User	Group	Global
File	-rw-	rw-	rw-
Dir	drwx	rwx	rwx

Write

Read

Execute

Setting File Permissions with
chmod [options] ### filename

- Setting permissions using “octal” numeric system
  - read = 4
  - write = 2
  - execute =1
- Simply add numbers together for different combos of permissions
- Each combo is only represented by one number

```
# create a file in the unix sandbox
$ touch permissions.txt
$ ls -l
# change permissions so that user can r,w,x;
# group can r,x; and global can r
$ chmod 754 permissions.txt
$ ls -l
# give everybody full permissions
$ chmod 777 permissions.txt
$ ls -l
# give yourself full permissions, but only let
# others read your files
$ chmod 744 permissions.txt
$ ls -l
```

Super User Do to Execute Command as Administrator: **sudo**
Change Owners With **chown**

- Use **sudo** when computer tells you no
  - Make sure you are certain that you are right and computer is wrong to not execute your command
- You'll need **sudo** for installing software

```
# create a directory with a subdirectory
$ mkdir -p test_dir/test_subdir
$ ls -l
$ ls -l test_dir

# list valid users
$ cut –d: -f1 /etc/passwd

# change owner of dir, -R includes subdirs
$ chown -R ValidUserName test_dir/
$ sudo chown -R ValidUserName test_dir/
$ ls –l
$ ls –l test_dir
# change owner back to you
$ sudo chown -R $USER test_dir/
```

$ Welcome to the Matrix

1.7 Scripting

Computational Biology

Lecture 1

Dr. Chris Bird

Scripting

- A script is a file with a list of commands
- Commands are executed sequentially
- Here we create a simple script

```
# create a script in the unix sandbox
$ touch ExtractBodyM.sh
# open ExtractBodyM.sh in GUI text editor
gedit ExtractBodyM.sh
open -a bbedit ExtractBodyM.sh
open ExtractBodyM.sh
```

ctrl-c will quit a command running in the terminal

```
# open ExtractBodyM.sh in CLI text editor
$ nano ExtractBodyM.sh
```

Either type in or copy and paste the pipeline we made previously to make `BodyMass.csv` into `ExtractBodyM.sh`

```
cut -d ";" -f 2-6 ../data/Pacifici2013_data.csv | tr ";" " " | tail -n+2 | sort -nrk6 > BodyMass.csv
```

Scripting

- It is important to write comments in English to describe what the script is doing
  - You'll forget
  - Makes it easier for others to figure out what's happening
  - Easier to identify errors

ctrl-x to exit nano, then y, then enter

```
# run ExtractBodyM.sh script
$ bash ExtractBodyM.sh
```

I noticed that there had to be spaces after the options for this to run correctly

```
$ ls -ltrh
$ nano ExtractBodyM.sh
```

Add the following comments to the script before the code using nano

```
# isolate columns 2-6 of csv using cut
# translate the ; to " " using tr
# remove the header row using tail
# sort by sixth column, descending order
# save to file
```

ctrl-o, then enter to save changes made in nano without closing

Don't forget to use the tab key to autocomplete file names

cbird@LAPTOP-URS0LRPO: ~/CSB/unix/sandbox

```
  GNU nano 2.5.3                    File: ExtractBodyM.sh

# isolate columns 2-6 of csv using cut
# translate the ; to “ “ using tr
# remove the header row using tail
# sort by sixth column, descending order
# save to file

cut -d ";" -f 2-6 ../data/Pacifici2013_data.csv | tr ";" " " | tail -n+2 | sort -nrk6 > BodyM.csv

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     ^Y Prev Page
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter   ^_ Go To Line  ^V Next Page
```

Scripting

- Script is hardcoded
  - Only works with one input and one output file

- In nano replace:

../data/Pacifici2013_data.csv with $1

BodyM.csv with $2

- $1 and $2 are variables

nano

```
# isolate columns 2-6 of csv using cut
# translate the ; to “ “ using tr
# remove the header row using tail
# sort by sixth column, descending order
# save to file

cut –d “;” -f 2-6 $1| \
tr ";" " " | \
tail -n+2 | \
sort -nrk6 > $2
```

I added escape characters \ because my code wouldn’t fit in 1 line. Here they allow 1 line of code to be written across several lines

Ctrl-x, then y, then enter to exit nano

Scripting

- Now we must include arguments to run the ExtractBodyM.sh script
  - In file
  - Out file

- We can make the script executable by changing permissions with chmod and adding a shebang! To the first line of the script
  - A shebang is tells the computer which language the script is in

```
# run ExtractBodyM.sh script
$ bash ExtractBodyM.sh \
> ../data/Pacifici2013_data.csv \
> BodyM.csv
# change permissions so script is
# executable
$ chmod 777 ExtractBodyM.sh
# add shebang! to beginning of script
# cat glues files together. The $() opens
# an invisible shell and runs
# echo “#!/bin/bash”, producing a line of
# text that is added to ExtractBodyM.sh
$ cat $(echo “#!/bin/bash”) ExtractBodyM.sh
# you could also add the shebang! in nano
$ nano ExtractBodyM.sh
```

- Edit script in GUI to match script to the right

```
gedit ExtractBodyM.sh
open ExtractBodyM.sh
```

  - Add 2nd & 3rd lines
  - Use escape characters to separate pipeline by command
  - Copy comments from lines 4-8 and paste above the appropriate command in the pipeline
  - Save and close

Open ▾ | ExtractBodyM.sh ~/CSB/unix/sandbox | Save

File Edit View Search Tools Documents Help

```
#!/bin/bash

# to run do this:
# ./ExtractBodyM.sh [infile] [outfile]

# isolate columns 2-6 of csv (first argument) using cut
# translate the ; to “ “ using tr
# remove the header row using tail
# sort by sixth column, descending order
# save to file (second argument)

# isolate columns 2-6 of csv (first argument) using cut
cut -d ";" -f 2-6 $1 | \
 # translate the ; to “ “ using tr
 tr ";" " " | \
 # remove the header row using tail
 tail -n+2 | \
 # sort by sixth column, descending order
 # save to file (second argument)
 sort -nrk6 > $2
```

sh ▾ Tab Width: 8 ▾ Ln 20, Col 2 ▾ INS

Scripting

- Make sure the script works

- Now that it's executable, we can use `./` to run it rather than **bash**

```
# run ExtractBodyM.sh script
$ ./ExtractBodyM.sh \
> ../data/Pacifici2013_data.csv \
> BodyMass.csv
$
```

$ Welcome to the Matrix

1.8 For Loops

Computational Biology

Lecture 1

Dr. Chris Bird

For Loops: `for [variableName] in [list]`

- For loops automate repetitive tasks
  - 1 task, 100 files
  - Same task, many different arguments
- In the examples to the right the variable is "`file`", the list is composed of either two or all of the fasta files
- When the for loop starts, `file` takes on the value of the first item in the list, `ggo_miR.fasta`
- In the second line of the loop, the command is run on the value in `$file`
- The `done` means goto first line of `for` loop
- `file` takes on the value of the second item in the list, `hsa_miR.fasta`
- Etc…

```
$ cd ~/CSB/unix/data/miRNA
$ ls

# display first two lines of two fastas
$ for file in ggo_miR.fasta hsa_miR.fasta
> do head -n 2 $file
> done

# display first two lines of all fastas
$ for file in *.fasta
> do head -n 2 $file
> done
```

When setting a variable equal to a value, don't use a $.
When calling the value held in the variable, use a $

*.fasta is a list of all files that end with .fasta in the present dir

For Loops: `for [variableName] in [list]`

- Example 2: isolating DNA sequences from particular types of micro RNA
  - `miR-208a`, `miR-564`, `miR-3170`
- saving them into one file per type of miRNA
- Recall that `grep` returns lines that match a pattern
  - Pattern is `$miR`
  - What is the `-A1` argument doing?
- View resulting files

```
# display first two lines of two fastas
$ for miR in miR-208a miR-564 miR-3170
> do grep $miR -A1 *.fasta > $miR.fasta
> done

# Look at one of the files created
$ less -S miR-564.fasta
hsa_miR.fasta:>hsa-miR-564 MIMAT0003228
hsa_miR.fasta-AGGCACGGUGUCAGCAGGC
--
ppy_miR.fasta:>ppy-miR-564 MIMAT0016009
ppy_miR.fasta-AGGCACGGUGGCAGCAGGC
--
ptr_miR.fasta:>ptr-miR-564 MIMAT0008243
ptr_miR.fasta-AGGCACGGUGUCAGCAGGC
```

File Name

Sequence Name

Separator

Sequence

$ Welcome to the Matrix

1.9 Tips & Tricks

Computational Biology

Lecture 1

Dr. Chris Bird

$PATH

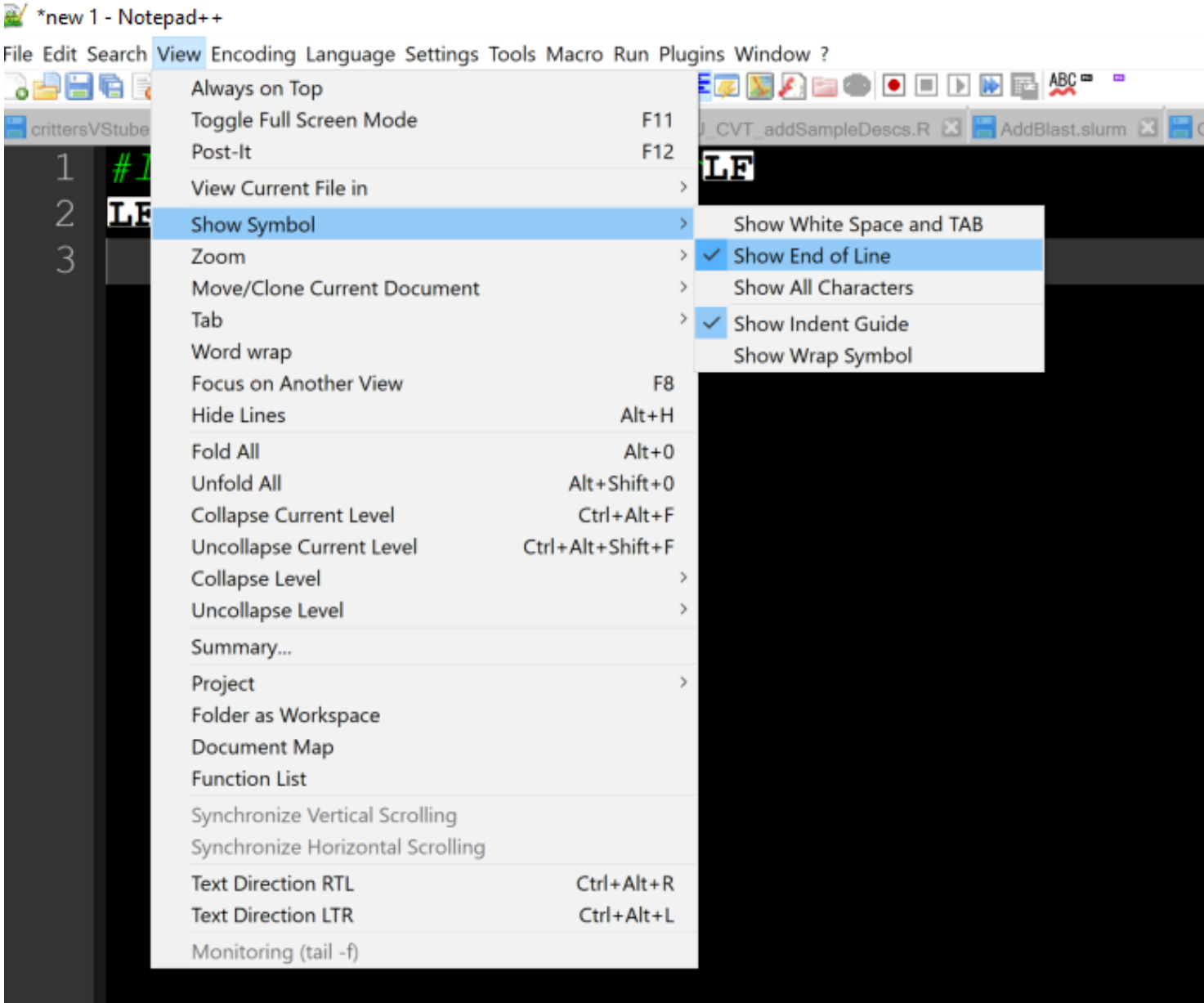
- A variable that holds all paths to directories where executable commands and scripts are located
- When you type `ls`, `bash` looks at `$PATH` to find the `ls` command file
- If you compile and install software manually, you need to move it to a $PATH dir `/usr/local/bin`

```
# show path variable
$ echo $PATH

# identify the path to the ls command
$ which ls
/bin/ls
```

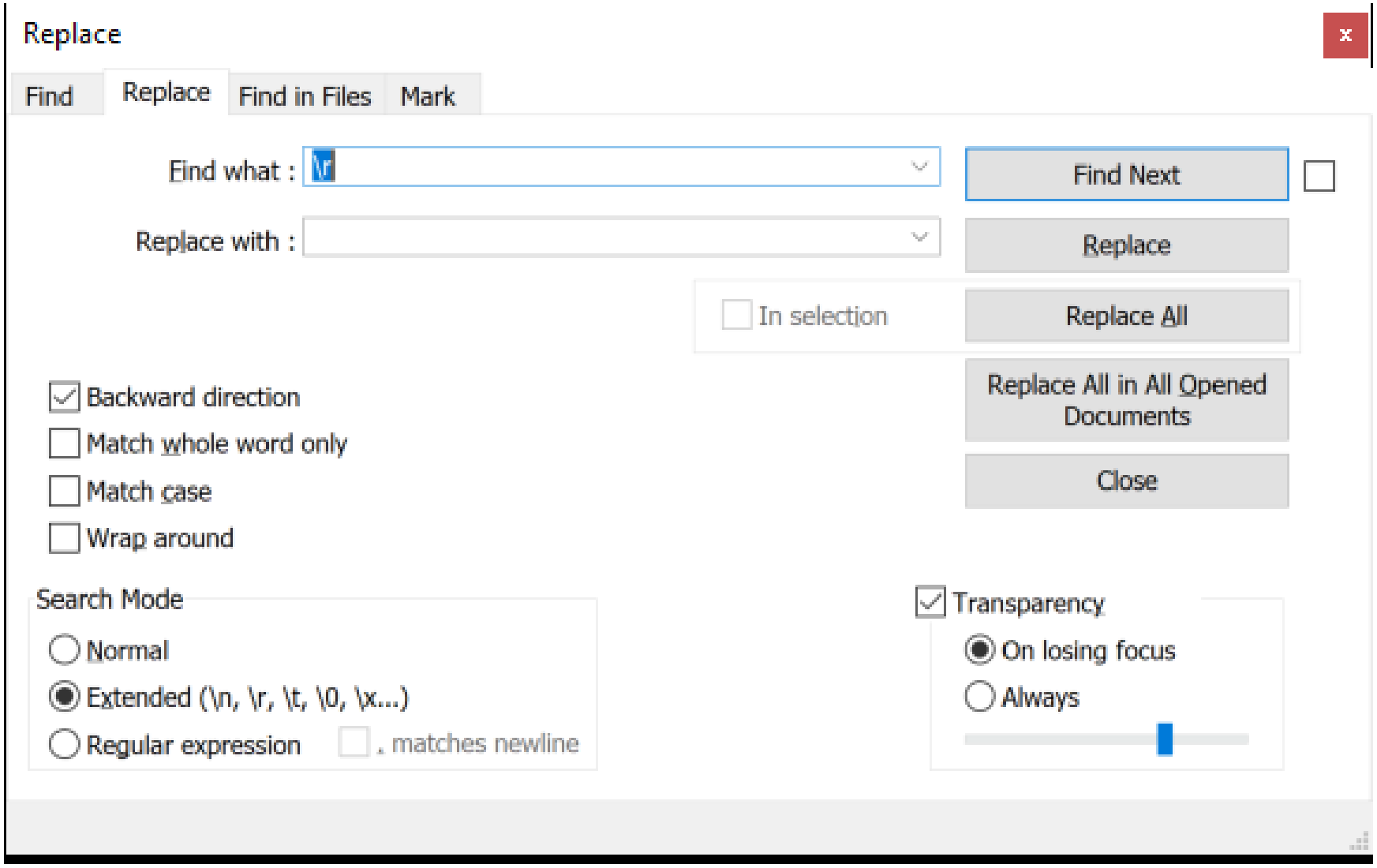
Line Terminators

- There are invisible characters at the end of every line in a text document
  - **Carriage Return (CR or `\r`)**
  - **Line Feed (LF or `\n`)**
- Unix, Linux, Mac systems end lines with LF
- Win systems end lines with CR LF
- Make sure your remove CR from files made in Windows
- This is one of many reasons why we use a Notepad++ or BBedit



Line Terminators

- There are invisible characters at the end of every line in a text document
  - **Carriage Return (CR or \r)**
  - **Line Feed (LF or \n)**
- Unix, Linux, Mac systems end lines with LF
- Win systems end lines with CR LF
- Make sure your remove CR from files made in Windows



Miscellaneous Useful Commands

- Note that some of the commands need to be installed on Macs

`history`	List the last commands you executed.[13]
`time [COMMAND]`	Time the execution of a command.
`wget [URL]`	Download the web page at `[URL]`.[14]
`open`	Open file or directory with default program; use `xdg-open` in Ubuntu or `start` in Windows Git Bash.
`rsync`	Synchronize files locally or remotely.
`tar` and `zip`	(Un)compress and package files and directories.
`awk` and `sed`	Powerful command-line text editors for much more complex text manipulation than `tr`.

13. In Git Bash all commands are listed.

14. Available in Ubuntu; for OS X look at `curl`, or install `wget` (see computingskillsforbiologists.com/wget).

`xargs`	Pass a list of arguments to other commands; for example, create a file for each line in `files.txt`: `cat files.txt | xargs touch`

$ Welcome to the Matrix

Questions?

Computational Biology

Lecture 2

Dr. Chris Bird

Exercises 1.10.2 & 1.10.3

Computational Biology

Lecture 2

Dr. Chris Bird