CALLING EXTERNAL COMMANDS

Brandon Seah

Perl Course 2017

WAIT, PERL DOESN'T DO *EVERYTHING*?

- Scripting language
- Feed input to other programs ...
- ... and process their output

BASH IS A SCRIPTING LANGUAGE

```
for FILE in *.fasta
do
COUNT=$(grep '>' $FILE | wc)
echo "File: $FILE"
echo "Headers: $COUNT"
done
```

Why not just use Bash?

TWO CHOICES

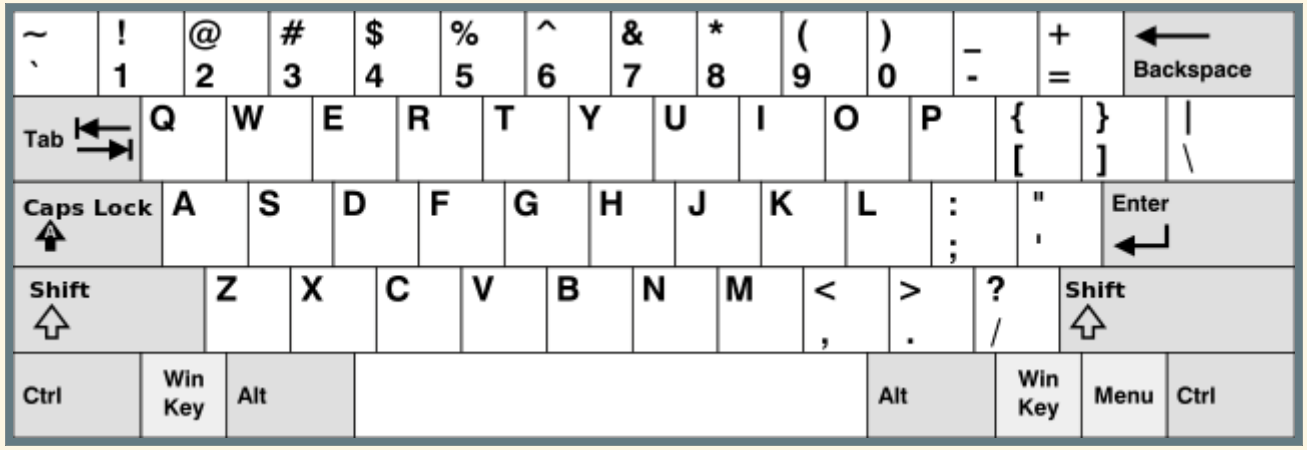
- Capture STDOUT
- Backticks
- Don't capture STDOUT
- `system()`

METHOD 1 - BACKTICKS

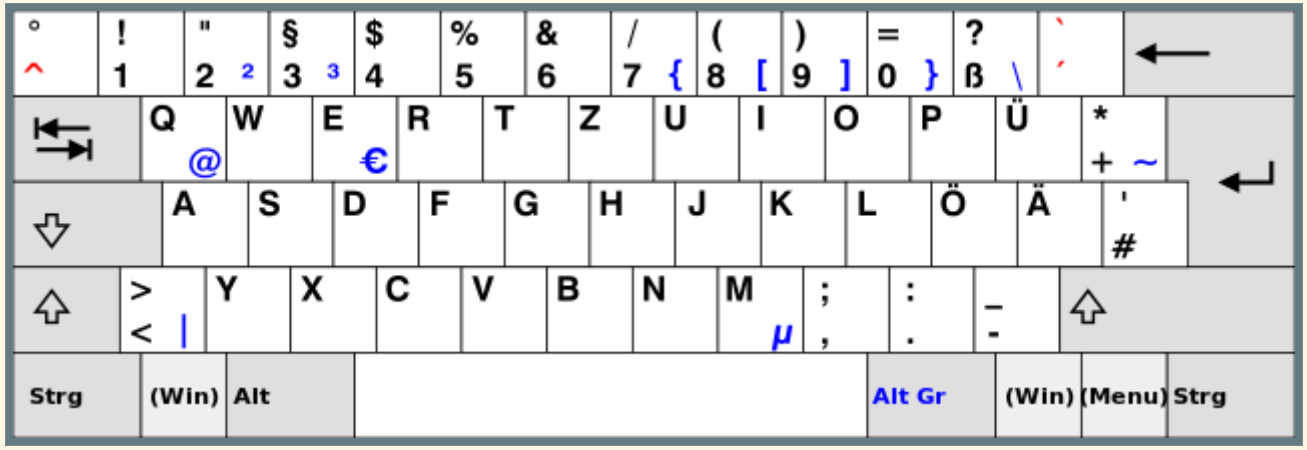
```
my $run_ls = `ls`;
```

- Captures STDOUT of command (Why?)

WHERE IS THE BACKTICK?



WHERE IS THE BACKTICK?



CAPTURE STDOUT

```
my $ls_output = `ls`;
print $ls_output;
```

Multiple lines "slurped" into a string

How can we separate?

CONVERT CAPTURED OUTPUT TO ARRAY

```
my $ls_output = `ls`;
my @ls_split = split "\n", $ls_output;
print $ls_split[0];
print "\n";
print $ls_split[1];
print "\n";
```

STDERR IS NOT CAPTURED

```
my $ls_output = `ls nonexistent_file`;
```

WHAT IF THE COMMAND FAILS?

```
my $failure = `caringtoomuch`;
```

What happens?

WHAT IF THE COMMAND FAILS?

```
use warnings;
my $failure = `caringtoomuch`;
```

Now what happens?

METHOD 2 - `SYSTEM()`

```
my $run_ls = system("ls");
```

- Doesn't capture STDOUT (Why not?)
- Returns *exit status*

EXIT STATUS

- 0 - success!
- Non-0 - not success!

EXIT STATUS EXAMPLE

```
my $status1 = system("ls");
my $status2 = system("ls nonexistentfile");
my $status3 = system("nonexistentcommand");
print "First exit status: $status1\n";
print "Second exit status: $status2\n";
print "Third exit status: $status3\n";
```

THINGS TO NOTE

- Script continues even if command fails
- STDOUT and STDERR "flow through" to user

SUMMARY

Backticks

- capture STDOUT

`system`

- return *exit status*

Remember

- Perl waits for command to finish
- Script continues if command fails
- `use warnings;`

EXERCISES

- Use `ls` and backticks to return a list of `.fasta` files in the current folders, and then open each of those files and count the number of sequences in each file. (Hints: `split`, `foreach`, Regex with `m/^>/`)