READING AND WRITING

Brandon Seah

Perl Course 2017

REVIEW: STANDARD STREAMS

Unix is a series of tubes

- `STDIN`
- `STDOUT`
- `STDERR`

REVIEW: STANDARD STREAMS

This is in the `bash` shell

```
ls | less # Redirect STDOUT to another program
ls > list_of_files # ... to a file
man foobar 2> error_message # Redirect STDERR
cat error_message # Write file to STDOUT
```

PRINT TO STDOUT & STDERR

Perl writes to STDOUT by default

```
print "Hello world!\n";
print STDOUT "Hello world!\n"; # Same thing
```

What about STDERR?

PRINT TO STDOUT & STDERR

Good guess...

```
print STDERR "Goodbye world...\n";
```

Useful for error messages

OPENING A FILE

Perl needs to know three things

- Where is the file? (*path*)
- Open for reading, writing, or appending? (*mode*)
- What's the alias for the stream? (*filehandle*)

OPENING A FILE

```
my $filename = "example.txt";

open (my $output, ">", $filename);
print $output "Hello world!\n";
print $output "Hello world again!\n";
close($output);
```

Identify the *path*, *mode*, and *filehandle*

WRITING VS APPENDING

```
open ($output, ">", $filename);
```

VS.

```
open ($output, ">>", $filename);
```

READING A FILE

Try this:

```
my $input_file = "example.txt" # Output from earlier

open (my $input, "<", $input_file);
print $input;
close($input);
```

PERL READS LINES BY DEFAULT

```
my $input_file = "example.txt";

open (my $input, "<", $input_file);
my $line = <$input>;
print $line;
close ($input);
```

LOOPING ACROSS A FILE

```
my $input_file = "example.txt";
my $counter = 0; # Counter for lines
open (my $input, "<", $input_file);
while (<$input>) {
    $counter++;
    print "line $counter is:\n";
    print $_;
}
close ($input);
```

CHOMP

```
my $input_file = "example.txt";
my $counter = 0; # Counter for lines
open (my $input, "<", $input_file);
while (<$input>) {
    chomp;
    $counter++;
    print "line $counter is:\n";
    print $_;
}
close ($input);
```

WHAT IF THE FILE ISN'T THERE?

```
my $input_file = "nonexistent.txt";
open (my $input, "<", $input_file);
close ($input);
```

ERROR MESSAGES

```
my $input_file = "nonexistent.txt";
open (my $input, "<", $input_file) or die;
close ($input);
```

ERROR MESSAGES

```
my $input_file = "nonexistent.txt";
open (my $input, "<", $input_file) or die ("$!");
close ($input);
```

What is reported by $! ?

ERROR MESSAGES

```
my $input_file = "nonexistent.txt";
open (my $input, "<", $input_file) or die ("Can't find $input: $!
close ($input);
```

This should be more informative

TASKS

- Read a Fasta file and extract only the header lines (Hint: regex)
- Read a Fasta file, rename the headers with sequential numbers, and write a new Fasta file (Hint: Use if/else + regex)

TASK 1 - HEADERS OF A FASTA FILE

- Fasta files have a *delimiter* for header lines

```
>sequence001
ATGGCTATACTACTGACTGACTGACCCACATGCTTTAGTCACTACTGTT
ATCGTCTTAAACTTTTGTGCATCTTATCTATGCGTCTCTACGTGTAGTC
ATCGATGCTACATCGTAGCTGAT
>sequence002
ACTCGTAGTCTACNTAGTCGTGATCGATCNTGCTAGTCTATTTCTATCG
...
```

TASK 1 - HEADERS OF A FASTA FILE

- Open file to stream with `open`
- Iterate over the lines of the file with `while`
- Check for matches with `if` and regex (`m//`)
- Close the stream

TASK 1 - HEADERS OF A FASTA FILE

```
my $file = "sequences.fasta";
open (my $input, "<", $file);
while (<$input>) {
  chomp;
  if (m/^>/) {
    print $_;
  }
}
close($input);
```

Quiz: What does ^ mean in the RegEx?