

Day 4: Basic I/O

Suggested reading: *Learning Perl* (4th Ed.)

Chapter 5, Input and Output

Chapter 11, File Tests

Chapter 12, Directory Operations

Turn In Homework

Homework Review

Please consider enrolling

**Write code.
At least a little.
Every day.**

Files

Opening & Closing

- open file and make filehandle: **open()**
- extra argument is for *mode*

```
open(FILE, 'input.txt');  
open(INPUT, '<', $my_filename);
```

- close filehandle: **close()**

```
close(INPUT);
```

Failing

`open()` returns false upon failure:

```
unless (open(FILE, $filename)) {  
    die "Could not open '$filename': $!\n";  
}
```

or the Perl way:

```
open(FILE, $filename) or  
    die "Could not open '$filename': $!\n";
```


Reading

- read one line using `<>`

```
my $line = <INPUT>;
```

- returns false when file is done
- use **chomp** to strip trailing newline

```
while (my $line = <INPUT>) {  
    chomp $line;  
    # do something with $line  
}
```

Interlude: Scalar vs. List Context

```
my $foo =  ;  
my @foo =  ;
```

```
1 + 2 +  + 3 + 4;
```

```
push @stack,  ;
```

Reading Whole Files

- in *list* context, `<>` yields one list element per line

```
my @lines = <INPUT>;
```

- `chomp` works on arrays

```
my @lines = <INPUT>;  
chomp @lines;
```

Writing

- open for (over)writing:

```
open(OUTPUT, '> output.txt');  
open(OUTPUT, '>', 'output.txt');
```

- open for appending:

```
open(OUTPUT, '>> output.txt');  
open(OUTPUT, '>>', 'output.txt');
```

- write to filehandle:

```
print OUTPUT "What could be easier?\n";
```



File Example

```
#!/usr/bin/perl
use strict; use warnings;

open(INPUT, 'input.txt') or
    die "Could not open 'input.txt': $!\n";
open(OUTPUT, '>', 'output.txt') or
    die "Could not open 'output.txt': $!\n";
my $number = 1;
while (my $line = <INPUT>) {
    print OUTPUT "$number: $line";
    $number++;
}
close(INPUT); close(OUTPUT);
```

Automatic Filehandles

- **STDIN** = standard input (default for `<>`)
- **STDOUT** = standard output (default for `print`)
- **STDERR** = standard error (output)

```
chomp(my $user_input = <STDIN>);  
chomp(my $user_input = <>);
```

```
print STDOUT "Hello, world!\n";  
print "Hello, world!\n";
```

```
print STDERR "Critical error\n";
```

The Magic of \$_

- automatically created (e.g., by `<>`)
- used as default (e.g., by `chomp`, `print`, ...)
- explicit use OK
- check perldoc for details

```
while (<>) {  
    chomp;  
    print "your last input: $_\n";  
}
```

we will see \$_ again another day...

Directories

File Test Operators

-r	readable	-e	exists	-f	regular file
-w	writable	-z	zero length	-d	directory
-x	executable	-s	size (<i>bytes</i>)	-l	symbolic link

```
if (-r $path) {
  if (-d $path) {
    print "skipping directory: $path\n";
  } elsif (-f $filename) {
    open(INPUT, $filename);
    # ...
  }
}
```

Directory Contents I

- `glob('...')` lists files matching a (shell) pattern
- `<...>` syntax works, but can be confusing

```
foreach my $path (glob("$dir/*.html")) {  
    open(INPUT, $path) or die "...";  
    # do stuff  
    close(INPUT);  
}
```

```
foreach my $path (<$dir/*.html>)
```

Directory Contents II

- can read directory contents directly

```
opendir(DIR, $dirname) or  
  die "Could not open $dirname: $!\n";  
foreach my $path (readdir(DIR)) {  
  # do something with $path  
}  
closedir(DIR);
```

Shell-Like Operations

- **chmod, chown** : change file permissions, owner
- **mkdir, rmdir** : make and remove directories
- **rename, unlink, symlink** : move, remove, and link

```
chmod 0644, $filename if -x $filename;
```

```
mkdir $new_directory_name;
```

```
if (-e $old_name) {  
    rename $old_name, $new_name;  
    symlink $new_name, $old_name;  
}
```

Last 2 Slides!

Other Scripting Languages

- Most have similar I/O operations
- Check for different or additional:
 - Operation names (**-d** vs. **isdir()** vs. **directory?()**)
 - Syntax
 - Operations
- Some environments are more restricted

Homework

- Do a word frequency count and report
 - What collection type works best here?
 - Consider subsetting the full dataset for testing
- BE SURE TO LABEL YOUR PRINTOUT!!!

```
#!/usr/bin/perl
```

```
# Homework for CS 368-1
```

```
# Assigned on Day 04, 2010-07-16
```

```
# Written by Your Name Here
```

```
use strict;
```

```
use warnings;
```