

File/Directory manipulation

Opening a File

- To read from a file, create a filehandle.
 - another scalar variable.
 - prior to perl 5.6, had to use global barewords
 - by convention, all caps
- open a file for reading:
 - `open my $fh, '<', 'myfile.txt';`
 - open file if exists, else return false;
- open file for writing:
 - `open my $outfh, '>', 'output.txt';`
 - clobber file if exists, else create new
- open file for appending:
 - `open my $appfh, '>>', 'append.txt';`
 - open file, positioned at end if exists, else create new

Always check for errors!

- You should always, yes, **always**, check the return value of system calls before proceeding
- `open my $fh, '<', 'file.txt' or die "Cannot open file: $!";`
- The \$! will tell you why the open failed.

Reading from a file

- `open my $fh, '<', 'myfile.txt' or ...;`
- `my $one_line = <$fh>;`
 - <...> in scalar context returns next line
- `my @all_lines = <$fh>;`
 - <...> in list context returns all remaining lines
 - `@all_lines` get all remaining lines from myfile.txt
 - puts each line of file into one member of array
- remember to chomp!
 - `chomp (my $next = <$fh>);`
- Rewind a file by "seeking" to the beginning:
 - `seek ($fh, 0, 0);`
 - see Camel for explanation
 - `perldoc -f seek`

printing to a file

- `open my $ofh, '>', 'output.txt' or ...;`
- `print $ofh "Hello World!\n";`
 - Note! No comma!!
- this can be tedious if all outputs are to same output file.
- `my $old_fh = select $ofh;`
 - make \$ofh the default file handle for all print statements.
 - store the originally `select`'ed file handle in `$old_fh`, so you can `select()` back to it later.

Reading a File at Once

- If you have the need to read the entire contents of a file into one large scalar, (ie, "slurp" the file)
- ***There Is More Than One Way To Do It***
- Read the file into an array, and join the elements:
 - `my @lines = <$file>;`
 - `my $all_lines = join ('', @lines);`
- Undefine `$/`
 - Controls Perl's notion of what a "line" is.
 - By default, `$/` is equal to `"\n"`, meaning that a line is terminated by a newline character. If `$/` is undefined, a "line" is terminated by the end-of-file marker.
 - `undef $/;`
 - `my $all_lines = <$file>;`
- Carefully consider whether or not you actually need to do this.
 - Usually not. Just process the file line-by-line, using loops

Close your files!

- `open my $fh, '<', 'file.txt' or ... ;`
`my @all_lines = <$fh>;`
`close $fh or die ...;`
- opening another file to the same filehandle will implicitly close the first one.
- If you don't explicitly close them, lexical filehandles will be closed as soon as they go out of scope

Directory manipulation

- directories can be opened, read, created, deleted, much like files.
- take care when doing these operations: you're affecting your directory structure
- many of the functions' success will depend on what permissions you have on the specified directories.

open, read, close

- `opendir my $dh, 'public' or ...;`
– `$dh` is a directory handle
- `my $nextfile = readdir $dh;`
– returns next file/directory name in scalar context
- `my @remaining_files = readdir $dh;`
– returns rest of file/directory names in list context
- Only returns name of file or directory, not full path!
– If you need to open the file, specify path:
– `open my $fh, '<', "public/$nextfile" or ...;`
- `closedir $dh or die ...;`

Rewind

- `opendir my $dh, '.' or die "...";`
- `my $first_file = readdir $dh;`
- `my $second_file = readdir $dh;`
- `rewinddir $dh;`
 - Equivalent of `seek()` for directories
- `my @all_files = readdir $dh;`

Change, Create, and Delete

- `chdir` → change working directory.
 - `mkdir` → create directory
 - `rmdir` → remove directory
 - works if and only if directory is empty
- ```
chdir 'public_html' or die "...";
mkdir 'images' or die "...";
rmdir 'temp' or die "...";
```

---

---

---

---

---

---

---

---

## Don't do useless work

- There's no reason to change to a directory simply to open a file in that directory
- You can specify the path of the file in the file open statement:
- `open my $html, '<', 'web/index.htm' or die "Can't open index.htm: $!";`
- This opens the file, without going through the bother of changing your working directory twice.

---

---

---

---

---

---

---

---

## Watch for legacy code

- Bareword Global Filehandles
  - `open FILE, '<', 'file.txt';`
  - `@lines = <FILE>;`
  - don't automatically close
  - not subject to `use strict`
  - global only
- 2-argument form of `open()`
  - `open my $file, $filename;`
  - `open my $of, ">$filename";`
  - what if `$filename` contains leading spaces?
  - what if a malicious user puts shell characters in `$filename`?

---

---

---

---

---

---

---

---