Adminstrivia

• Homework 3 graded
• Remember:
  – Submit PDF file
  1. Always submit the program
  2. Always submit sample runs
  3. Explanation appreciated!
Adminstrivia

• I'm away next week and the following Monday.

• **Procedure:**
  – Next week there will be lecture slides posted online
  – And one homework
  – **Following Monday class canceled**
Today’s Topics

• File input/output
  – open, <>
• References
• Perl modules
File I/O

• Step 1: call open()

Files and I/O

You can open a file for input or output using the `open()` function. It's documented in extravagant detail in `perlfunc` and `perlopen`/`tut`, but in short:

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
<th>Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>open(my $in, &quot;&lt;&quot;, &quot;input.txt&quot;)</code></td>
<td><code>or die &quot;Can't open input.txt: $!&quot;;</code></td>
</tr>
<tr>
<td>2.</td>
<td><code>open(my $out, &quot;&gt;&quot;, &quot;output.txt&quot;)</code></td>
<td><code>or die &quot;Can't open output.txt: $!&quot;;</code></td>
</tr>
<tr>
<td>3.</td>
<td><code>open(my $log, &quot;&gt;&gt;&quot;, &quot;my.log&quot;)</code></td>
<td><code>or die &quot;Can't open my.log: $!&quot;;</code></td>
</tr>
</tbody>
</table>

**Files**: must be opened for reading “<“ or writing “>” (overwrite or append mode “>>”)

Shell syntax: **I/O redirection “<“ “>”**

Opening a file creates a **file handle** (Perl variable) – not to be confused with filename

Supply the **file handle** for read/write
File I/O

• Step 2: use the <> operator:

You can read from an open filehandle using the <> operator. In scalar context it reads a single line from the filehandle, and in list context it reads the whole file in, assigning each line to an element of the list:

```
1. my $line = <$in>;
2. my @lines = <$in>;
```

Reading in the whole file at one time is called slurping. It can be useful but it may be a memory hog. Most text file processing can be done line at a time with Perl's looping constructs.

$in is the file handle instantiated by the open() call
File I/O

• Line by line:

The `<>` operator is most often seen in a `while` loop:

```perl
while (<$in>) {
  # assigns each line in turn to $_
  print "Just read in this line: $_";
}
```

```perl
open($txtfile, $ARGV[0]) or die "$ARGV[0] not found!\n";
while ($line = <$txtfile>) {
  print "$line"
}
/close($txtfile)
```

the command `$line = <$txtfile>` inside the condition reads in a line from the file referenced by the file handle `$txtfile` and places that line into the variable `$line` *(including the newline at the end of the line)*.

At the end of the file, `$line` is just an empty string (equivalent to false).

the filename is the first parameter to the perl program (arguments go in `@ARGV`).
Useful string functions

Functions for SCALARs or strings

- chomp - remove a trailing record separator from a string
- chop - remove the last character from a string
- chr - get character this number represents
- crypt - one-way passwd-style encryption
- hex - convert a string to a hexadecimal number
- index - find a substring within a string
- lc - return lower-case version of a string
- lcfirst - return a string with just the next letter in lower case
- length - return the number of bytes in a string
- oct - convert a string to an octal number
- ord - find a character's numeric representation
- pack - convert a list into a binary representation
- q/STRING/ - singly quote a string
- qq/STRING/ - doubly quote a string
- reverse - flip a string or a list
- rindex - right-to-left substring search
- sprintf - formatted print into a string
- substr - get or alter a portion of a string
- tr/// - transliterate a string
- uc - return upper-case version of a string
- ucfirst - return a string with just the next letter in upper case
- v/// - transliterate a string

chomp vs. chop

To split a string into words:

```
@a = split " ", $line;
@a = split / /, $line;
```

Note: multiple spaces ok with " " variant
Worked Example

- Read names and frequencies into a hash from a file
More complex data structures

- Arrays and hashes may only contain scalars
- **Question**: How to accomplish nesting, i.e. put non-scalars inside?
- **Answer**: use references (pointers), which happen to be scalars

(Actually a reference is just an unsigned number: computer address)

[http://perldoc.perl.org/perlreftut.html](http://perldoc.perl.org/perlreftut.html)
References

• Two ways to make a

Make Rule 1
If you put a \ in front of a variable, you get a reference to that variable.

1. $aref = \@array;
   # $aref now holds a reference to @array
2. $href = \%hash;
   # $href now holds a reference to %hash
3. $sref = \$scalar;
   # $sref now holds a reference to $scalar

Make Rule 2
[ ITEMS ] makes a new, anonymous array, and returns a reference to that array. { ITEMS }
Anonymous hash, and returns a reference to that hash.

1. $aref = [ 1, "foo", undef, 13 ];
   # $aref now holds a reference to an array
2. $href = { APR => 4, AUG => 8 };
   # $href now holds a reference to a hash

Remember bracketing when initializing:
( ) List – can be used for both arrays and hashes
[] Reference to an array
{} Reference to a hash
References

• Example: array of arrays

```c
#define a = ( [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9] );
```

**Note:** uses Make Rule 2: square brackets

• Let’s figure out what the following mean:

```c
$a[1]$
${$a[1]}[1]$
$a[1] -> [0]$
$a[1][2]$
```

**Arrow Rule**

In between two subscripts, the arrow is optional.

Instead of `$a[1] -> [2]`, we can write `$a[1][2]` ;

dereference
References

• Looping (using **for/foreach** with array/hash references:

```
for my $element (@array) {
  ...
}
```

- replace the array name, `@array`, with the reference:

```
for my $element (@{aref}) {
  ...
}
```

`${aref}[3]` is too hard to read, so you can write `$aref->[3]` instead.

`${shref}{red}` is too hard to read, so you can write `$href->{red}` instead.

Be careful! `$aref->[3]` and `$aref[3]` are different
References

• Code:
  $a = [1, 2, 3, 4, 5];
  print $a+1

• What happens here?
References

- Looping (using `for/foreach`) with array/hash references:

```perl
1. for my $key (keys %hash) {
   2.   print "$key => $hash{$key}\n";
   3. }
```

1. then replace the hash name with the reference:

```perl
1. for my $key (keys %{href}) {
   2.   print "$key => @{$href}{$key}\n";
   3. }
```

- `$href{red}` is too hard to read, so you can write `$href->{red}` instead.

Be careful! `$href->{'red'}` vs. `$href{red}` are different.
Experiment

- Unicode encoding (utf-8)

```perl
use feature 'fc';
use utf8;
binmode STDOUT, "utf8";

my $a = "AbCdéÉ";

print uc($a), "\n";
print lc($a), "\n";
print fc($a), "\n";
```

bash-3.2$ perl case.perl
ABCDÉÉ
abcdéé
Experiment

• Note:

```perl
use feature 'fc';
my $a = "AbCdéÉ";

print uc($a), "\n";
print lc($a), "\n";
print fc($a), "\n";
```

bash-3.2$ perl case2.perl
ABCDÉÉ
abcdéÉ
abcdéÉ

```bash
use feature 'fc';
use utf8;

my $a = "AbCdéÉ";

print uc($a), "\n";
print lc($a), "\n";
print fc($a), "\n";
```

bash-3.2$ perl case2.perl
ABCDÉÉ
abcdéÉ
abcdéÉ

```bash
use feature 'fc';
use utf8;
use open qw(:std :utf8);

my $a = "AbCdéÉ";

print uc($a), "\n";
print lc($a), "\n";
print fc($a), "\n";
```

bash-3.2$ perl case2.perl
ABCDÉÉ
abcdéÉ
abcdéÉ

open pragma
most general solution
Perl Modules

• CPAN: Comprehensive Perl Archive Network

The Comprehensive Perl Archive Network (CPAN) currently has 138,135 Perl modules in 30,330 distributions, written by 11,712 authors, mirrored on 254 servers.

The archive has been online since October 1995 and is constantly growing.
cpanm

- Ubuntu:

```
sandiway@sandiway-VirtualBox: ~
sandiway@sandiway-VirtualBox:~$ which cpanm
The program 'cpanm' is currently not installed. You can install it by typing: sudo apt-get install cpanminus
```

**INSTALLATION**

There are several ways to install cpanminus to your system.

**Package management system**

There are Debian packages, RPMs, FreeBSD ports, and packages for other operation systems available. If you want to use the package management system, search for cpanminus and use the appropriate command to install. This makes it easy to install cpanm to your system without thinking about where to install, and later upgrade.

**Installing to system perl**

You can also use the latest cpanminus to install cpanminus itself:

```
curl -L http://cpanmin.us | perl --sudo App::cpanminus
```

This will install cpanm to your bin directory like /usr/local/bin (unless you configured INSTALL_BASE with local::lib), so you probably need the --sudo option.
cpanm

• OSX: assume command line tools have been installed
  – xcode-select -p
  – /Applications/Xcode.app/Contents/Developer
  – xcode-select --install
  – xcode-select: error: command line tools are already installed, use "Software Update" to install updates

• uses program curl (cURL)
  – a command line tool for getting or sending files using URL syntax

bash-3.2$ which cpanm
bash-3.2$ curl -L http://cpanmin.us | perl --sudo App::cpanminus
--> Working on App::cpanminus
Fetching http://www.cpan.org/authors/id/M/MI/MIYAGAWA/App-cpanminus-1.7000.tar.gz ... OK
Configuring App-cpanminus-1.7000 ... OK
Building and testing App-cpanminus-1.7000 ... Password:
OK
Successfully installed App-cpanminus-1.7000
1 distribution installed
bash-3.2$ which cpanm
/usr/local/bin/cpanm
Example of module use

```
use Date::Calc qw(:all);

if ($#ARGV != 0) {
    die "usage: year\n"
}
$year = $ARGV[0];
$month = 9;
$day = 9;

$dow = Day_of_Week_to_Text(Day_of_Week($year,$month,$day));
print "$month/$day/$year falls on a $dow\n"
```

bash-3.2$ perl dow.perl
9/9/2014 falls on a Tuesday

Example of module use

- Missing Perl Module:

Can't locate Date/Calc.pm in @INC (@INC contains:
/opt/local/lib/perl5/site_perl/5.16.3/darwin-thread-multi-2level
/opt/local/lib/perl5/site_perl/5.16.3
/opt/local/lib/perl5/vendor_perl/5.16.3/darwin-thread-multi-2level
/opt/local/lib/perl5/vendor_perl/5.16.3
/opt/local/lib/perl5/5.16.3/darwin-thread-multi-2level
/opt/local/lib/perl5/5.16.3
/opt/local/lib/perl5/site_perl
/opt/local/lib/perl5/vendor_perl .)
at dow.perl line 1.
BEGIN failed--compilation aborted at dow.perl line 1.
Example of module use

- **Install Perl Module**
- `sudo cpanm Date::Calc`
- --> Working on Date::Calc
- Fetching http://www.cpan.org/authors/id/S/ST/STBEY/Date-Calc-6.4.tar.gz ... OK
- Configuring Date-Calc-6.4 ... OK
- ==> Found dependencies: Bit::Vector
- --> Working on Bit::Vector
- Fetching http://www.cpan.org/authors/id/S/ST/STBEY/Bit-Vector-7.4.tar.gz ... OK
- Configuring Bit-Vector-7.4 ... OK
- Building and testing Bit-Vector-7.4 ... OK
- Successfully installed Bit-Vector-7.4 (upgraded from 7.3)
- Building and testing Date-Calc-6.4 ... OK
- Successfully installed Date-Calc-6.4 (upgraded from 6.3)
- 2 distributions installed
localtime

• Code:

```perl
$now = localtime;
print "$now\n";
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime;
@a = localtime;
print "@a\n";
```

1. $now_string = localtime;  # e.g., "Thu Oct 13 04:54:34 1994"