

Perl Scripting: Course Organization

- Lectures . . .
- Handouts
- Examination Rules
- Optional Project

Perl Design

*Perl is designed to make the easy jobs easy,
without making the hard jobs impossible.*

There Is More Than One Way To Do It.

- Scripted language with compilation/syntax checking.
- Easy to “glue” things together: run programs, extract data.

Typical Perl Tasks

- ① Lot of connections to simpler tools: shell, sed, awk.
- ② (Text) file processing/generation.
- ③ Pattern matching: lines containing AAA sequences followed by BBB but containing one CCC in between.
- ④ Convenient data structures: word statistics!

General Script Structure

- No variable declaration necessary – they ‘spring into existence’.
- Adjectives: `my $counter`, `local $variable`.
- Verbs: `print "String"`.
- Nouns: `variables print $i`.
- Comments: `#` and Plain Old Text (POD).

What Perl is About

Be merry with Perl: warn, die.

```
open FILE, "data.txt" or die;
```

One-liners with Perl:

```
ls -l | perl -n -e '@a=split;print if @a[2] eq qq(pawsa)'  
perl -pi -e 's/ugly hack/beautiful code/g' report.txt
```

More Details

- `-n` executes given code for each line read from input:

```
perl -n -e 'print "if(/Final energy/) {print}" input_file
```
- Data types: scalars, arrays, hashes, subroutines.
- `split` is same as `split ' ', $_`
- Many string quoting methods available.
- inline file modification (create a file and overwrite the input).
- multi-file programs with separate name spaces `Dog:: ...`

Even More Details

- Ordered Lists: `@a=('cat', 'dog', 'horse');`
- Unordered Hashes:
`%b=('Sun' => 'Sunday', 'Mon' => 'Monday');`
- Hash of Lists:
`%students = ('kth' => ['Name1', 'Name2'],
 'su' => ['Name1']);`

Basic IO

- You can read files open FL, "file"
- write files: open FL, ">file"
- append to files: open FL, ">>file"
- read program output: open FL, "program|"
- feed programs with input: open FL, "|program"

Get line with <FL>. close with close FL.

Regular Expressions

Matching:

```
print "Right choice!\n" if $usedOS =~ /Linux/;
```

Search and Replace:

```
$os =~ s/Windows/Linux/gi;
```

- Regular expression are much more powerful than this.

Example: Produce Average Grades

Noel 25
Ben 76
Clementine 49
Norm 66
Chris 92
Doug 42
Carol 25
Ben 12
Clementine 0
Norm 66
...

Example Code

```
1 #!/usr/bin/perl
2
3 open GRADES, "grades" or die "Can't open grades: $!\n";
4 while ($line = <GRADES>) {
5     ($student, $grade) = split(" ", $line);
6     $grades{$student} .= $grade . " ";
7 }
8
9 foreach $student (sort keys %grades) {
10     $scores = 0;
11     $total = 0;
12     @grades = split(" ", $grades{$student});
13     foreach $grade (@grades) {
14         $total += $grade;
15         $scores++;
16     }
17     $average = $total / $scores;
18     print "$student: $grades{$student}\tAverage: $average\n";
19 }
```

Ending Comments

- Reading after: Chapter 1.
- Hand in for the next session:
 - Print all the files as that were modified on the first day of any month (1 Jan, 1 Feb, 1 Mar, . . .).
 - Modify the grading example to print gender-specific averages.