Perl 5 Intro

Tobias Oetiker

OETIKER+PARTNER AG

November 24, 2014

What is Perl?

- Practical Extraction and Report Language
- Many Platforms
- Powerful Text Manipulation
- Web Programming
- *There is more than one way to do it*

Course Documentation

- Booklet: Perl Programming
- 15 Sections
- Most Sections have Exercises
- Bonus Exercises if you have time

A basic program

```perl
#!/usr/bin/env perl
use 5.010;
use strict;
use warnings;
use diagnostics;
say 'Hello world.';
```

- `perldoc`
- `perldoc -f print`
- `perldoc Perl::Module`
- `perl -E 'say "Hello world"'`

Running the program

- `chmod +x progname`
- `./progname` or just `progname` if in your `PATH`
- `perl -e 'program code'`

3 - Scalar variables

```
my $x;
$x = "some text";
$x = 444;
$x = "444"; $x++;

say 'the value of \$x is $x';
say "the value of \$x is $x";
```

Array variables

```
my @rgb = ("red", "green", "blue");
say "First RGB component: $rgb[0]";
say "All components: @rgb";
say "Number of components: ", scalar @rgb;,
```

File handling

```perl
my $file = '/etc/passwd'; # Name the file
open my $info, $file or   # Open the file
    die "Can't open $file: $!\n";
my @lines = <$info>;        # Read it into an array
close $info;              # Close the file
print @lines;             # Print the array
```

File handling (2)

```
open(my $cmd, "who |") or die "Can't run who: $!\n";
while(<$cmd>) { # $_
    print;
}
```

Control structures

- Usual `for`, `while` and `until`
- `foreach` is in reality synonim of `for`
- `for my $a (@list) { block; }`
- `for (my $a=0; $a<10; $a++) { block; }`

Conditionals

```
if (!$a) {
    print "The string is empty\n";
}
elsif (length($a) == 1) {
    say "The string has one character";
}
elsif (length($a) == 2) {
    say "The string has two characters";
}
else {
    say "The string has lots of characters";
}
```

- /etc/mailcap

Hashes

```
my %david = (
     last_name  => "Tobi",
     first_name => "Oetiker",
     email      => 'tobi@oetiker.ch',
);

$david{phone} = '27019';

print "david's email: $david{email}\n";
```

References

```
my @colors      = ( 'red', 'green', 'blue' );
my $colors_ref = \@colors;
$colors_ref = [ 'red', 'green', 'blue' ];

my @colors_copy = @{$colors_ref};       # @colors
my $first_color = ${$colors_ref}[0];    # $colors[0]
$first_color = $colors_ref->[0];
```

String matching

```
my $text = "quick brown fox";

$text =~ /quick/       # quick anywhere in $text ?
$text =~ /^quick/      # quick at beginning of $text
$text =~ /quick.*fox/ # quick - something - fox
$text =~ /quick/i;     # case insensitive

$text =~ /quick(.*)fox/ and print "<$1>\n";

< brown >

$text =~ /\/match\/a\/path/;
$text =~ m|/match/a/path|;
$text =~ m#/match/a/path#;
$text =~ m{/match/a/path};
```

Substitution

```
my $text = "quick brown fox";

$text =~ s/brown/blue/;
$text =~ s#brown#blue#;
$text =~ s{brown}{blue};

$text =~ s/quick(.*)fox/fox$1quick/;
```

Split

```
my $info = "Caine : Michael: Actor :14 - Leafy Drive";

my @personal = split(/\s*:\s*/, $info);

print "<$personal[2]>\n"; # -> <Actor>
```

Subroutines

```perl
sub match {
    my ($a, $b) = @_;
    return $a =~ $b;
}

match('brown fox', 'fox');
```

Fibonacci: $a_0 = 1$, $a_1 = 1$, $a_n = a_{n-2} + a_{n-1}$

14 - Variable scope

```
$x = 1;    # global variable

# $x: 1
sub a {
    # $x: 1
    my $x = 2;
    # $x: 2
    if(sometest) {
         # $x: 2
         my $y = 3;
         # $y: 3
    }
    # $x: 2
    # $y: undefined
}
# $x: 1
```

Installing Perl Modules

```
$ export PERL5LIB=~/perl5/lib/perl5
$ export PATH=~/perl5/bin:$PATH
$ curl -L http://cpanmin.us |\
    perl - -n -l ~/perl5 App::cpanminus
$ cpanm -l ~/perl5 Mojolicious
```