

Introduction to Perl

Gang Chen
chengang@bgitecholutions.com

September 27, 2014

Outline

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
- 4 Examples

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
- 4 Examples

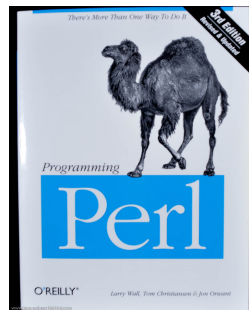
What is Perl?

Perl

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister

Perl: History 1

- 1.0: December 18, 1987, Larry Page
- 2.0: 1988, a better regular expression
- 3.0: 1989, support binary data streams
- 4.0, 1991
- Programming Perl, Camel Book, for Perl 4.0



Perl: History 2

Perl 5

- 5.000: October 17, 1994, rewrite of the interpreter Objects, lexical variables, modules and references are added.
- 5.002: new prototypes feature.
- Comprehensive Perl Archive Network(CPAN), 1995.
- 5.004: May 15, 1997, UNIVERSAL package and CGI.pm module.
- 5.8: July 18, 2002, unicode, a new I/O, thread
- 5.10: December 18, 2007
- 5.20: May 27, 2014, subroutin signature, slice.

Perl: History 3

Perl 6

- Perl 6 design process was first announced on July 19, 2000
- As of 2014, none of Perl 6 implementations are considered "complete".
 - Rakudo Perl: Perl 6 for virtual machines.
 - Pugs: Perl 6 written in Haskell.
 - v6.pm: a pure Perl 5 implementation of Perl 6.
 - Yapsi: a Perl 6 compiler and runtime written in Perl 6 itself.

Applications

Applications

- text processing
- CGI programming: Craigslist, IMDb, Slashdot and so on;
- graphics programming: Perl/Tk, WxPerl
- system administration
- network programming
- bioinformatics

References

Books

- Learning Perl sixth Edition;
- Mastering Perl;
- Advanced Perl;
- Programming Perl;

Official Website

<http://www.perl.org/>

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
- 4 Examples

Install

Download and Install

Download: <http://www.perl.org/get.html>

- Unix/Linux: preinstalled
- Mac OS: preinstalled
- Windows:
 - ActiveState Perl: A binary distribution for Win
 - Strawberry Perl: Open source
 - DWIM Perl: based on Strawberry and include many useful CPAN modules

Install: editor

Editors

- Vim: editor god
- Emacs: god's editor
- Notepad++: fast and easy to use
- Atom, sublime text, textmate ...

Hello Perl!

```
1 #!/usr/bin/perl
2
3 =hello
4 Hello example for GNB5010
5 Author: Gang Chen
6 =cut
7
8 use warnings;
9 use strict;
10
11 print "Hello, Perl!\n";
```

see hello.pl

Input and Run

- ① Input the source codes by using a editor
- ② Save the source codes to a file named hello.pl
- ③ Execute the file:
 - Add execution permission to the file and execute directly
 - Execute the file by using perl interpreter

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax**
 - Basic Syntax
 - Input and Output
 - Regular Expression
- 4 Examples

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
 - Basic Syntax
 - Input and Output
 - Regular Expression
- 4 Examples
 - System Administration
 - String Processing
 - CGI Programming

Scalar Data: Number

```
$a = 1;  
$b = 1.2;  
print $a + $b, "\n";
```

Scalar Data: String

```
# scalar: strings
print "香港, 你好! \n";
print "Hello, Hongkong!\n";
print "中國, 你好! \n";

print "Hello "x10;
print "\n";

my $fname = "Chen";
my $gname = "Gang";
my $name = $gname." ".$fname;
```

Conversion between Numbers and Strings

```
# conversion between numbers and strings
print 1 + 2, "\n";
print "1" + 2, "\n";
print "1" + "2", "\n";
print "1 + 2", "\n";

print "00001" + "002", "\n";

print "one" + 2, "\n";
print "one" + "two", "\n";
```

if Control Structure

```
# if control structure  
my $num1 = 5;  
my $num2 = 3;
```

```
if($num1 > $num2){  
    print "Success\n";  
}else{  
    print "failed\n";  
}
```

```
$num1 > $num2 ? print "Success\n" : print "Failed\n";
```

```
print "Success\n" if ($num1 > $num2);
```

while and for

```
# while and for
my $num = 1;
while($num < 10){
    print $num, "\n";
    $num++;
}
```

```
for($num = 1;$num<10;$num++){
    print $num, "\n";
}
```

List and Array

List and Array

List A list is an ordered collection of scalars.

Array An array is a variable that contains a list.

ELEMENT NUMBERS (INDICES)	VALUES
0	35
1	12.4
2	"hello"
3	1.72e30
4	"bye\n"

List and Array

```
# list and array
my @list1 = (1,2,3,4,5);
my @list2 = ("one", "two", "three");
my @list3 = (1..10);
print $list1[0], "\n";
print $list2[1], "\n";
print $list3[3], "\n";
```

List and Array

- Operate to the start of the array: shift, unshift
- Operate to the end of the array: pop, push
- Any place: splice

foreach

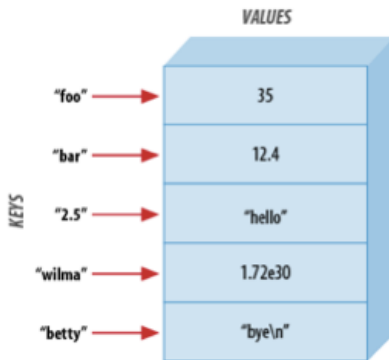
```
foreach (@list1){  
    print $_, "\n";  
}
```

```
for(@list1){  
    print $_, "\n";  
}
```

```
print $_, "\n" for(@list1);
```

```
print $_, "\n" for(1..10);
```

Hash



Hash

```
# hash
my %scores = (
    'Gang' => 60,
    'Chen' => 70,
    'Xu'   => 80,
    'Lu'   => 90,
);

print $scores{'Chen'}, "\n";

for (keys %scores){
    print $_,":",$scores{$_},"\n";
}
```

```
print $_,":",$scores{$_},"\n" for (keys %scores);
```

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
 - Basic Syntax
 - **Input and Output**
 - Regular Expression
- 4 Examples
 - System Administration
 - String Processing
 - CGI Programming

Input from User

```
print "What's your name?\n";  
  
my $name = <STDIN>;  
print "Hello ", $name;  
  
my @names = <STDIN>;  
  
print "Hello ", $_ for(@names);
```

Interact with Filesystem

see io.pl

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
 - Basic Syntax
 - Input and Output
 - Regular Expression
- 4 Examples
 - System Administration
 - String Processing
 - CGI Programming

Regular Expression

Regular Expression

Regular Expression is a template or pattern of strings.

Match

see [regex.pl](#)

References

- Mastering Regular Expressions
- 精通正则表达式
- 正则指引

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
- 4 **Examples**
 - System Administration
 - String Processing
 - CGI Programming

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
 - Basic Syntax
 - Input and Output
 - Regular Expression
- 4 Examples
 - **System Administration**
 - String Processing
 - CGI Programming

Command Options of perl

Options

- -e

- -n

```
while (<>) {  
    # your code goes here  
}
```

- -p

```
while (<>) {  
    # your code goes here  
} continue {  
    print or die "-p destination: $!\n";  
}
```

Process file content

Adding Line Number to file content

```
perl -ne 'print "$. $_"' names.txt  
perl -pe '$_ = "$. $_"' names.txt
```

Next

- 1 Overview
- 2 Quick Get Started
- 3 Syntax
 - Basic Syntax
 - Input and Output
 - Regular Expression
- 4 Examples
 - System Administration
 - **String Processing**
 - CGI Programming

Thanks!