Perl

CS174

Chris Pollett

Oct 23, 2006.

Outline

- Introduction to Perl
- Perl Basics

History of Perl

- Perl was invented in 1987 as a small language to expand upon the string processing abilities of the Unix commands awk and sed.
- It has since grown into a quite extensive language with a large library of useful modules: CPAN.
- Since it is very good for doing string processing, when the web came along it was ideal for CGI programming.
- Like Java and Pascal and CLR languages, Perl programs are compiled to a byte-code which is then interpreted during execution.
- Versions of Perl run on every common platform available today: Windows, Linux, Mac, Amiga…

Variables, Numeric and String literals

- Like Javascript, Perl is not strongly typed.
- Perl has three categories of variables: scalars (begin with $), arrays (begin with @), and hashes (begin with %):

  $var, @arr, %hash
- Numbers stored in scalar variables are represented in double-precision floating point form. Number literals are similar to Javascript:

  72, 7.2, .72, 72., 7E2, 7e2, .7e2, 7.2E-2, etc
- Character strings are treated as scalar units in Perl. String literals can be formed using either ‘ or “; however, they have a slightly different meaning. Namely, single quotes do not evaluate escape sequences (except \’) or variables; whereas, double quoted sequence do.

```
$a =5;
echo “I have $a dollars\n”;  --> I have 5 dollars  (newline)
echo ‘I have $a dollars’; --> I have $a dollars\n
 q$ a single string with a different delimiter$
qq@ a double quote string with a different delimiter@
 ‘’ “” -- both are the empty string
```

Scalar Variables

- As we said before scalar variables are always preceded by a $ sign.
- Variables are case sensitive. So the variables below are different:

  $test $Test $teSt$
- Like Javascript variables, Perl variables do not need to be declared before they are used.
- A scalar variable that has not been assigned a value has value undef. The numeric value of undef is 0 and its string value is “”.
- In addition to the variables you define, Perl has a large number of implicit variables which after the $ begin with _, or ^, or \.
- More information about Perl variables can be found through the Perl documentation software by typing:

  perldoc perlvar

Numeric Operators

- Perl has all the familiar numeric operators:
  - +, -, *, /, ** (exponentiation), %.
- In most circumstances arithmetic is floating point. So 5/2 will evaluate to 2.5

String Operators

- To concatenate strings in Perl one uses the period operator:
  - “hi”.“ there” to get the string “hi there”
- Perl also supports a repetition operator x.
  - “More ” x 3 gives the string “More More More ”

String Functions

- Perl functions and operators are closely related and can often be used interchangeably.
- For example, if there were a predefined unary operator blah, then it could be called using either:

  blah x

  or

  blah(x).
- A function with no parameters can be called with or without empty parentheses.
- The most commonly used string functions are:
  - chomp -- removes terminating newline char’s and returns the number of removed characters
  - length -- returns the length of a string
  - lc -- converts string to lower case
  - uc -- converts string to upper case
  - hex -- return the decimal value for a hex string
  - join(“c”, @list_of_strings) -- makes a single string with delimiter c

Assignment Statements

- Assignment statement are like in most languages descended from C:

$a = *value*;

- We can also use binary and unary assignment operators:

$a++;

$a += 5;

$str .=“hello”;

- Comments in Perl can be started with a # sign. The remainder of the line is then a comment.

Keyboard Input

- All input and output in Perl is thought of as file input and output.
- Files have external names but are referenced in programs through internal names called *filehandles*.
- There are three predefined filehandles: STDIN (usually keyboard), STDOUT (usually screen), STDERR (usually screen).
- These are the standard input streams, output, and error streams.
- The line input operator <> acts on input file handles. So

  ```
  chomp($in_data = <STDIN>);
  ```

  gets a line from standard in and chops off the newline characters.

Screen Output

- The commands echo and print can be used to write to a filehandle.
- If no filehandle is specified then we default to standard out.

```
print “Enter a number to square\n”;
$x = <STDIN>;
print “The square of $x is”;
$x *= $x;
print “$x \n”;
```

Running a Perl Program

- From a command prompt one can run a perl program with a command like:

```
perl filename.pl
```

- To compile to bytecode without interpreting one can write:

```
perl -c filename.pl
```

- To get diagnostic warning statements one can use the -w flag.

```
perl -w filename.pl
```

Control Expressions

- Perl has a two distinct kinds of relational expressions: those for numeric operands and those for strings.
- The numeric operands will look familiar:

==, !=, <, >, <=, >=, <=> (returns -1, 0, 1 depending on which argument was bigger).

- The corresponding string literals are:

eq, ne, lt, gt, le, ge, cmp

Roughly, the Fortran name corresponding to the C expression.

- Perl has two sets of operators for AND, OR, NOT: &&, ||, ! as well as *and*, *or*, and *not*. The former have higher precedence then the relational expressions; the latter have lower precedence.

Selection Statements

- Perl’s if statement is similar to C’s except that the “then” clause must have curly braces:

```
if($a >5) $b++; #is not legal
if($a >5 ){$b++;} #is legal
```

- You can also do more complicated constructs:

```
if( $a >1) {print “hi\n”;}
elsif($a <1) {print “bye\n”;} #notice spelling of elsif
else {print “I’m in doubt.\n”;}
```

- Perl also has a construct for the negation of an if:

```
unless ($a>1) {print “\$a is too small”;}
```