Brief Notes on Learning Perl

jingmi@baidu.com

2009-06-03

What Does

- Perl stand for "Practical Extraction and Report Language“
- Or called "Pathologically Eclectic Rubbish Lister"

Built-in Data Type: Scalar Data

- Scalar Data Type

  (1)Number

  ```
  123
  111.222
  100_000_000
  ```

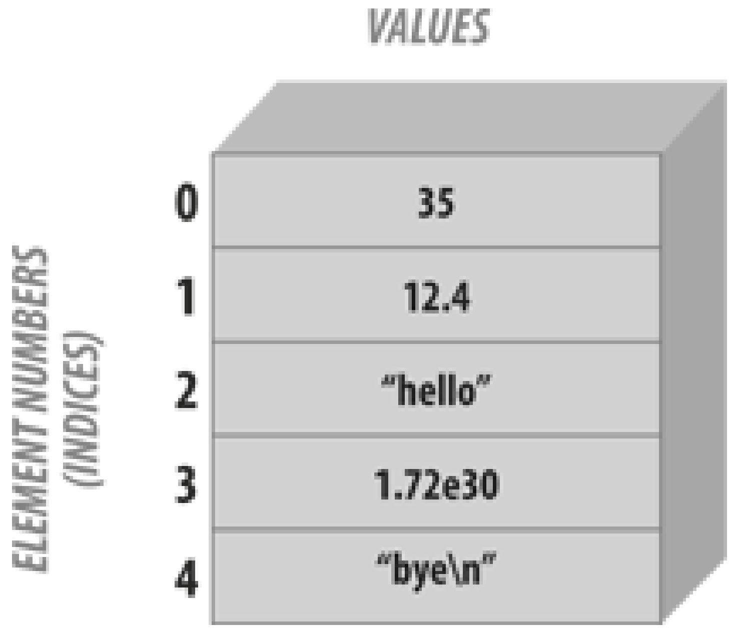
  (2)Strings

  ```
  'hello, world'
  "hello, world\n"

  $num = 100;
  $string = "hehe";
  ```

Built-in Data Type: Lists & Array

- (1, 2, 3)
- (1..100)
- @array = ();
- $array[0] = ‘a’;
- $array[1] = $scalar_data;
- $array[2] = 999;
- $#array == 2
- $array[-1] == 999



Built-in Data Type: Hash

- %birthday = ();

- $birthday{‘jiajia’} = 1990;
- $birthday{‘xiaohe’} = ‘198x’;

Special variables

1. $_
$ARG，常常是一个默认变量

2. @_
@ARG，子例程参数表

3. $0
$PROGRAM_NAME，本程序的名字

4. @ARGV
本程序的命令行参数表

5. $/
$RS，或$INPUT_RECORD_SEPARATOR，输入记录分隔符，改变了readline，
<FH>和chomp对于“行”的看法，默认为换行符

6. $$
$PID，或$PROCESS_ID，本脚本的进程号(PID)

7. $!
$ERRNO，或$OS_ERROR，上一次系统调用错误值

Traverse Data Structure

Traverse an array:

```
foreach (@array)
{
    print $_, “\n”;
}
```

Traverse a hash table:

```
foreach (keys %birthday)
{
   print $birthday{$_}, “\n”;
}
-------------------------------------------------------------
while ( ($key, $value) = each %hash )
{
   print "$key => $value\n";
}
```

Debug: Print Data Structure

```
use Data::Dumper;

print Dumper($scalar_data);
print Dumper(\@array);
print Dumper(\%hash_table);
```

Control Structure(if...elsif...else)

```
if ( ! defined $dino)
{
    print "The value is undef.\n";
}
elsif ($dino =~ /^-?\d+\.?$/)
{
    print "The value is an integer.\n";
}
else
{
    print "The value is the string '$dino'.\n";
}
```

Control Structure(loop)

```
for ($i = 1; $i <= 10; $i++) {  # count from 1 to 10
        print "I can count to $i!\n";
}

foreach (1..10) {  # Really a foreach loop from 1 to 10
        print "I can count to $_!\n";
}

while (1) {
        print "It's another infinite loop!\n";
}
```

Loop Controls

- last
- next
- redo

```
$counter = 0;
for ($cnt=0; $cnt<10; $cnt++, $counter++)
{
    if ($cnt == 9)
    {
        $cnt = 0;
        print "redo now\n";
        redo;
    }
    if ($counter == 20)
    {
        last;
    }
    printf("[%d]\t%d\n", $counter, $cnt);
}
```

Input from Standard Input

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

或者

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

Open File

```
open CONFIG, "dino";
open CONFIG, "<dino";
open BEDROCK, ">fred";
open LOG, ">>logfile";
```

cat.pl

```perl
#!/usr/bin/perl -w

  use strict;


  while (<>)
  {
     print $_;
  }
```

```
jingmi@Odin ~$./cat.pl ./cat.pl
#!/usr/bin/perl -w

use strict;

while (<>)
{
    print $_;
}
```

Read File

```
#!/usr/bin/perl -w
use strict;
open FH, "/etc/passwd" or die "$!";
while (<FH>)
{
    print $_;
}
close FH;
```

Write File

```
open FH, ">output_file";
print FH "hello, world!\n";
close FH;
```

注意 print 里的FH后没有逗号。

File Test

判断一个文件是否已经存在：

```
die "Oops! A file called '$filename' already exists.\n" if -e $filename;
```

-r	File or directory is readable by this (effective) user or group
-w	File or directory is writable by this (effective) user or group
-x	File or directory is executable by this (effective) user or group
-e	File or directory name exists
-z	File exists and has zero size (always false for directories)
-s	File or directory exists and has nonzero size (the value is the size in bytes)
-T	File looks like a "text" file
-B	File looks like a "binary" file

Glue Prgramming Language

```
exec("date");
```

```
system("date");
```

```
use Data::Dumper;

chomp($info = `date`);
print Dumper($info);
```

Regular Expressions

- /fred/
- m/fred/
- m{fred}
- m[fred]
- m,fred,
- m!fred!
- m^fred^

Pattern-matching modifiers and their meanings

Modifier	Meaning
/i	Ignore alphabetic case
/g	Global–match/substitute as often as possible
/s	Let . match newline
/m	Let ^ and $ match next to embedded \n
/o	Compile pattern once only

Split Field

```
use Data::Dumper;

$string = "hehe haha    yes";

@fileds = split(/\s/, $string);
print Dumper(\@fileds);

@fileds = $string =~ /(\S+)/g;
print Dumper(\@fileds);
```

Read Config File

```
my $config_file = "./adif.conf";
open(FH, $config_file) or (log_error($!), exit(-1));
while (<FH>)
{
      next if /^#/;

     ($sf_ftp_main{'addr'})    = $_ =~ m!sf_ftp_main_addr[ \t]*:[ \t]*(.*)!g,
    next   if ($_ =~ /^sf_ftp_main_addr/);

     ($sf_ftp_main{'port'})    = $_ =~ m!sf_ftp_main_port[ \t]*:[ \t]*(.*)!g,
    next   if ($_ =~ /^sf_ftp_main_port/);

     ($sf_ftp_main{'user'})    = $_ =~ m!sf_ftp_main_user[ \t]*:[ \t]*(.*)!g,
    next   if ($_ =~ /^sf_ftp_main_user/);

     ($sf_ftp_main{'passwd'})   =   $_   =~   m!sf_ftp_main_passwd[   \t]*:[
    \t]*(.*)!g, next if ($_ =~ /^sf_ftp_main_passwd/);
}
```

Sort(1)

- sort SUBNAME LIST
- sort BLOCK LIST
- sort LIST

Sort(2)

```
# sort lexically
@articles = sort @files;

# same thing, but with explicit sort routine
@articles = sort {$a cmp $b} @files;

# now case-insensitively
@articles = sort {uc($a) cmp uc($b)} @files;

# same thing in reversed order
@articles = sort {$b cmp $a} @files;

# sort numerically ascending
@articles = sort {$a <=> $b} @files;

# sort numerically descending
@articles = sort {$b <=> $a} @files;
```

sort(3)

```
# this sorts the %age hash by value instead of
  key
# using an in-line function
@eldest = sort { $age{$b} <=> $age{$a} }
  keys %age;

# sort using explicit subroutine name
sub byage {
  $age{$a} <=> $age{$b};
}
@sortedclass = sort byage @class;
```

map(1)

- map BLOCK LIST

- map EXPR,LIST

map(2)

```
@nums = (1, 2, 3, 11, 22, 55);
@chars = map($_+1, @nums);


print    map    {    "$_    =>
  $hash{$_}\n" } keys %hash;
```

map & sort

- `my @file_local_rslt = sort(map { $local_result_file{'path'} . $_ } @{$file_rslt_tobe_download_ref});`
- `my @file_local_rsum = sort(map { $local_result_file{'path'} . $_ } @{$file_rsum_tobe_download_ref});`
- `my @file_backup_rslt = sort(map{ $local_backup_file{'path'} . $_ } @{$file_rslt_tobe_download_ref});`
- `my @file_backup_rsum = sort(map{ $local_backup_file{'path'} . $_ } @{$file_rsum_tobe_download_ref});`

Most Important!!!

- http://perldoc.perl.org/
- http://perldoc.perl.org/perldoc.tar.gz
- http://perldoc.perl.org/perldoc-html.tar.gz

Q&A

THANKS!!!