A Quick Guide to Perl

Yubao Liu
liuyb@yahoo-inc.com

Yahoo! Global R & D Center, Beijing

2010-09-13

Outline

Outline

Perl - Practical Extraction and Report Language

- Created by Larry Wall at 1987
- Combines some of best features of C, sed, awk and sh
- The camel book “Programming Perl”
- 18325 modules on `http://www.cpan.org/`
- Perl 5.12.2 released at 7th Sep, 2010
- Rakudo Star released at 29th Jul, 2010

Successful Stories

Bugzilla, Request Tracker, TWiki (Foswiki), MovableType, Webmin, SVK, ack-grep, `http://slashdot.org/`, Majordomo, SpamAssassin, AWStats, MRTG, Perlbal, Mogilefs, BioPerl

Hello World!

```perl
#!/usr/bin/perl
use strict;
use warnings;

greet();
exit(0);

sub greet {
    print "Hello␣world!\n";
}
```

Listing 1: Greeting from Perl

Running: `perl hello.pl`

Recommended Readings

- Learning Perl, 5th Edition
- Advanced Perl Programming, 1st Edition
- `http://www.pgsqldb.org/mwiki/index.php/ProgrammingPerl`
- `http://wiki.perlchina.org`
- `perldoc perl`

Access Perl Documents Easily

- `http://www.perl.org/docs.html`
- `http://search.cpan.org/`
- PodBrowser
- Pod::POM::Web
- Pod::Browser

Outline

Data Types

Scalar	`$var`, number, string, reference
Array	`@var`, any type
Hash	`%var`, map from number or string to scalar
File handle	`FH`, created by open() and sysopen()
Typeglob	`*symbol`, used to manipulate symbol table

Scalar

- Automatically conversion between number and string
- `10/3` gets a float point number, use int() for integer
- Interpolation
- + . x ++, different logic operators for numbers and strings
- $_
- undef(), defined(), `perldoc perlsyn`, /Truth and Falsehood
- substr(), vec()
- octets and strings, length(), use Encode, use encoding, use utf8

Array

- @a, $a[0], $#a, scalar(@a), (elemA, elemB)
- @_, modify parameters
- push(), pop(), shift(), unshift(), splice(),
- ($i, $j) = ($j, $i); ($a, @a, $b) = @b
- Interpolation: `"@a"`
- Sequence generator: 1..100
- Array slice: @a[@b]

Hash

- %a, ${"key"}, ("key", "val"), (key => "val")
- keys(), values(), each(), exists(), delete()
- Hash slice: @h{@k}
- Key *must* be number or string, value *must* be scalar

typeglob

- `perldoc perldata`, `perldoc perlmod`
- To transfer file handles, manipulate symbol table

File handle

- `perldoc -f open`, `perldoc perlopentut`
- File handle, reference to file handle, IO::Handle object

Variable declaration

my	Lexically scoped local variable
state	Lexically scoped static local variable
local	Dynamically scoped local variable
our	Lexically scoped package variable
use vars	Package scoped package variable

Contexts

- Void context, `print "...."`
- Scalar context, `$i < @a; $i < scalar(@a);`
- Array context, `(stat $f)[7]`
- Numeric context, `$a + $b`
- String context, `$a . $b`
- Dual var, `$!, Scalar::Util::dualvar()`
- wantarray()

Statement

- `perldoc perlsyn`
- if elsif else, unless, given when
- while, until, for, foreach
- next, last, redo
- do while, do until (*double scopes for next/last/redo!*)
- goto-LABEL, goto-EXPR, goto-&NAME
- Statement modifiers: `print $i++ while $i < 10;`
- BEGIN ..., END ...

Comment

- Single line comment: #
- Trick for multi line comment: =pod .... =cut

Operators and Special Variables

- `perldoc perlop`
- `perldoc perlvar`

Subroutine

- `perldoc perlsub`
- `perldoc perlfunc`
- Omission of & and parentheses

Reference

- `perldoc perlref,perlreftut,perldsc,perllol`
- Reference: \$a, \@a, \%h, \&name, sub {...}
- Dereference: $$a, @$a, $a->[0], %$h, $h->{"key"}, &$name(), $name->()
- ${$a[0]}, {$a[0]}, %{$a[0]}
- [1, 2, 3], {a => 1, b => 2}
- Autovivification: `my $a; $a->[0][0][1] = 3;`, `\( @h{@k} )`

I/O

- `perldoc -f open`
- binmode()
- `print FH $a, $b; print $fh $a, $b;`
- Buffered I/O: open <> print printf seek tell
- Unbuffered I/O: sysopen, sysread, syswrite, sysseek
- close(), eof()

Regular Expression

"Practical **Extraction** and Report Language"

Regular Expression

- Regexes in Perl aren't strict regular expressions
- Non-greedy quantifiers, possessive quantifiers, (?:...), assertions
- `http://www.regex-engineer.org/slides/perl510_regex.html`
- MUST read: `perldoc perlre`
- Again, MUST read: `perldoc perlre`

Module

```perl
package SomeProduct::SomeModule;
use Exporter 'import';
use strict;
use warnings;

our $VERSION = 0.01;
our @EXPORT_OK = qw(subA subB);

sub subA {
}

sub subB {
}

1;
```

Listing 2: A simple module

Module (cont.)

```
#!/usr/bin/perl
use strict;
use warnings;
use SomeProduct::SomeModule qw(subA);

subA();
SomeProduct::SomeModule::subB();
```

Listing 3: Use module

Module (cont.)

- `perldoc perlmod`
- `perldoc Exporter`
- `perldoc -q "@INC"`
- require, use

Single File, Multiple Packages

- Start with “package main;” if main code isn’t at the beginning
- Use “MODULE->import;” instead of “use” and “require” to use modules in same file
- Enclose whole package with braces if it has package scoped “my” variables
- Take care order of packages if they contain statements outside of subroutines
- Don’t have to append “1;” to the end of packages
- Use __DATA__ , Data::Section, Inline::Files to embed data (Optional)

Processes and Threads

- fork(), exec(), system(), qr(), open(), IPC::Cmd, IPC::Open2, IPC::Open3
- `perldoc threads`, `perldoc perlthrtut`
- `http://migo.sixbit.org/papers/Perl_Threads/slide-index.html`
- `http://search.cpan.org/dist/Coro/`
- `http://search.cpan.org/dist/AnyEvent/`

Unicode Support

- Text strings(character strings), binary strings(byte strings)
- Manuals: perluniintro, perlunitut, perlunicode, perlunifaq
- Modules: Encode, utf8, encoding, bytes
- `/p{Punctuation}/`, `/p{Han}/g`

Perl Formats

“Practical Extraction and **Report** Language”

Perl Formats

- `perldoc perlform`
- `use Perl6::Form`

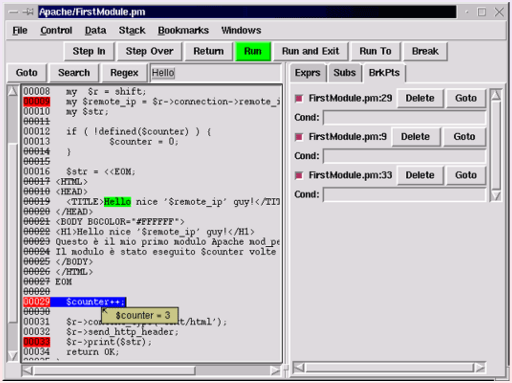
Outline

Logging

- print() is your friend!
- Data::Dumper, Smart::Comments, Carp, CGI::Carp, caller()
- `perl -Mdiagnostics=-t a.pl`
- Log::Any, Log::Dispatch, Log::Log4perl
- Devel::Trace, Devel::LineTrace, Debug::Trace
- Devel::Cover, Devel::NYTProf
- `use re 'debugcolor';`

Debugger

- `perl -d a.pl`; Devel::ptkdb, `perl -d:ptkdb a.pl`
- Emacs GUD, DDD



```
00008   my  $r = shift;
00009   my $remote_ip = $r->connection->remote_i
00010   my $str;
00011
00012   if ( !defined($counter) ) {
00013           $counter = 0;
00014   }
00015
00016   $str = <<EOM;
00017 <HTML>
00018 <HEAD>
00019   <TITLE>Hello nice '$remote_ip' guy!</TI
00020 </HEAD>
00021 <BODY BGCOLOR="#FFFFFF">
00022 <H1>Hello nice '$remote_ip' guy!</H1>
00023 Questo è il mio primo modulo Apache mod_pe
00024 Il modulo è stato eseguito $counter volte
00025 </BODY>
00026 </HTML>
00027 EOM
00028
00029   $counter++;
00030
00031   $r->co[illegible]t/html');
00032   $r->send_http_header;
00033   $r->print($str);
00034   return OK;
```

```
$counter = 3
```

Outline

Vanilla Class Example

```perl
package Person;
use strict;
use warnings;

our $VERSION = 0.01;

sub new {
    my ($class, @args) = @_;
    bless { AGE => 0 }, $class;
}

sub age {
    my $self = shift;

    (defined $_[0]) ? $self->{"AGE"} = $_[0] : $self->{"AGE"};
}

1;
```

Listing 4: A simple class

Vanilla Class Example (cont.)

```perl
#!/usr/bin/perl
use strict;
use warnings;
use Person;

my $p = Person->new();
$p->age(20);
print $p->age, "\n";
```

Listing 5: Use class

RAII - Resource Acquisition Is Initialization

- DESTROY() method
- Guard, Scope::Guard, B::Hooks::EndOfScope, Hook::Scope, Sub::ScopeFinalizer

Advanced OO

- UNIVERSAL
- Class::Struct
- Class::MOP, Moose, Mouse, Any::Moose

Outline

Lazy Perl Programmers

- Default argument $_
- Omission of parentheses
- Omission of “->” in reference of reference:
  `$->[0]{"name"}`
- Don’t forget readability!

Imperious Diamond Operator

```
# right
while (<$fh>) {
    # do something with $_
}

# wrong, see perldoc perlvar, /^\s*\$_
while ($not_found && <$fh>) {
    # do something with $_
}
```

Listing 6: Imperious Diamond Operator

Volatile Capture Buffers

```
# wrong
if ( /(\d+)\./ ) {
     print $1 if $1 =~ /^0/;
}

# right
if ( /(\d+)\./ ) {      # or: my ($s) = /(\d+)\./
    my $s = $1;
     print $s if $s =~ /^0/;
}
```

Listing 7: Volatile Capture Buffers

In-place chomp Function

```
# wrong
$s = chomp $s;

# right
chomp $s;
```

Listing 8: In-place chomp Function

Sticky Iterator

```
# wrong
while (my ($k, $v) = each %h) {
    last if $k < 0;
}

while (my ($k, $v) = each %h) {
}

# right
# reset iterator before next iteration with 'keys %h' or 'values %h'
```

Listing 9: Sticky Iterator

Innocent Falsehood

```
# wrong
/(\d+)/;
print "Got number!\n" if $1;

# right
/(\d+)/;
print "Got number!\n" if defined $1;
```

Listing 10: Innocent Falsehood

Valuable Exception

```
# wrong
sub Object::DESTROY {
    eval {}
}

eval {
    my $obj = Object->new();

    die "foo";
}

if ( $@ ) {
    ...
}
# right
...use Try::Tiny...
```

Listing 11: Valuable Exception

Unexpected Truth

```perl
# wrong
sub foo {
    ...
    return undef if $error;
}

my @a = foo();        # @a can be (undef) which evaluates to true

# right
sub foo {
    ...
    return if $error;   # returns undef for scalar context,
                        # () for array context, nothing for
                        # void context
}
```

Listing 12: Unexpected Truth

Outline

HTTP::Server::Simple

```perl
sub run {
    #....
    if ($server) {
        require join( '/', split /::/, $server ) . '.pm';
        *{"$pkg\::ISA"} = [$server];

        # clear the environment before every request
        require HTTP::Server::Simple::CGI;
        *{"$pkg\::post_accept"} = sub {
            HTTP::Server::Simple::CGI::Environment->setup_environment;
            # $self->SUPER::post_accept uses the wrong super package
            $server->can('post_accept')->(@_);
        };
    #....
}
```

Listing 13: HTTP::Server::Simple

Outline

URL Match

URL = [protocol] host [path [? parameters]]
Task: extract suffix of path or base name if no suffix

```
m{
    ^(?:[^:]+://)?+
    ([^/]++)
    [^?]*
    (?|
        (\.[^?]*)
        |
        /([^?]*)
    )
}x
```

Listing 14: URL match

URL Match

Question:
How to modify the regex to match URLs without path part?

Recursive Match

Task: match AB, AABB, AAABBB, ...

Recursive Match

```
/^( A(?1)?+B)$/
```

Outline

local::lib

- Try interesting modules without affecting host system
- `cpan>o conf build_requires_install_policy yes`
- `cpan>o conf prerequisites_policy follow`
- `cpan>o conf commit`
- `cpan>notest install Some::Module`

Other Good Stuffs

- `http://search.cpan.org/dist/Task-Kensho/`
- `http://search.cpan.org/dist/Inline/`
- `http://par.perl.org`
- `http://search.cpan.org/dist/App-perlbrew/`
- `http://search.cpan.org/dist/Shipwright/`
- `http://search.cpan.org/dist/Emacs-PDE/`
- `http://www.vim.org/scripts/script.php?script_id=556`

__END__