Introduction to Perl

Bartosz Chodorowski

<chomzee@ethernet.pl>

Selected choice from computer science
November 23, 2009

General information

- Perl is a high-level programming language with an eclectic heritage written by Larry Wall

General information

- Perl is a high-level programming language with an eclectic heritage written by Larry Wall
- Derives from C, sed, awk, Unix shell, dozens of other languages

General information

- Perl is a high-level programming language with an eclectic heritage written by Larry Wall
- Derives from C, sed, awk, Unix shell, dozens of other languages
- Specializes in process, file and text manipulation...

General information

- Perl is a high-level programming language with an eclectic heritage written by Larry Wall
- Derives from C, sed, awk, Unix shell, dozens of other languages
- Specializes in process, file and text manipulation...
- ...what makes it particularly well-suited for tasks involving quick prototyping, system utilities, software tools, system management tasks, database access, graphical programming, networking, and world wide web programming

General information

- Perl is a high-level programming language with an eclectic heritage written by Larry Wall
- Derives from C, sed, awk, Unix shell, dozens of other languages
- Specializes in process, file and text manipulation...
- ...what makes it particularly well-suited for tasks involving quick prototyping, system utilities, software tools, system management tasks, database access, graphical programming, networking, and world wide web programming
- **Multi-task language**

PERL as an acronym

- Practical Extraction and Report Language

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister
- Programmers Everywhere Relish Leisure

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister
- Programmers Everywhere Relish Leisure
- Perfectly Easy, Really. Look!

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister
- Programmers Everywhere Relish Leisure
- Perfectly Easy, Really. Look!
- Perl's an Extremely Reliable Language

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister
- Programmers Everywhere Relish Leisure
- Perfectly Easy, Really. Look!
- Perl's an Extremely Reliable Language
- Programmers Escaping Real Life

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister
- Programmers Everywhere Relish Leisure
- Perfectly Easy, Really. Look!
- Perl's an Extremely Reliable Language
- Programmers Escaping Real Life
- Programmers Enter Riding Llamas

PERL as an acronym

- Practical Extraction and Report Language
- Pathologically Eclectic Rubbish Lister
- Programmers Everywhere Relish Leisure
- Perfectly Easy, Really. Look!
- Perl's an Extremely Reliable Language
- Programmers Escaping Real Life
- Programmers Enter Riding Llamas **(WTF?)**

PERL, Perl or perl?

- Perl was originally named „Pearl”, after the Parable of the Pearl from the Gospel of Matthew

PERL, Perl or perl?

- Perl was originally named „Pearl”, after the Parable of the Pearl from the Gospel of Matthew
- Perl is **not** an acronym and should be written with capitalized „P” (like „Python”)

PERL, Perl or perl?

- Perl was originally named „Pearl”, after the Parable of the Pearl from the Gospel of Matthew
- Perl is **not** an acronym and should be written with capitalized „P” (like „Python”)
- `perl` as an interpreter of Perl should be written lowercase (like `awk`, `sed`)

PERL, Perl or perl?

- Perl was originally named „Pearl", after the Parable of the Pearl from the Gospel of Matthew
- Perl is **not** an acronym and should be written with capitalized „P" (like „Python")
- `perl` as an interpreter of Perl should be written lowercase (like `awk`, `sed`)
- *„Nothing but perl can parse Perl" – Tom Christiansen*

Perl's history

- Larry Wall began work on Perl in 1987

Perl's history

- Larry Wall began work on Perl in 1987
- version 1.0 was released to comp.sources.misc newsgroup on December 18, 1987

Perl's history

- Larry Wall began work on Perl in 1987
- version 1.0 was released to comp.sources.misc newsgroup on December 18, 1987
- The language expanded rapidly over the next few years

Perl's history

- Larry Wall began work on Perl in 1987
- version 1.0 was released to comp.sources.misc newsgroup on December 18, 1987
- The language expanded rapidly over the next few years
- Perl 2, released in 1988, featured a better regular expression engine

Perl's history

- Larry Wall began work on Perl in 1987
- version 1.0 was released to comp.sources.misc newsgroup on December 18, 1987
- The language expanded rapidly over the next few years
- Perl 2, released in 1988, featured a better regular expression engine
- Perl 3, released in 1989, added support for binary data streams.

Perl's history

- Larry Wall began work on Perl in 1987
- version 1.0 was released to comp.sources.misc newsgroup on December 18, 1987
- The language expanded rapidly over the next few years
- Perl 2, released in 1988, featured a better regular expression engine
- Perl 3, released in 1989, added support for binary data streams.
- Perl 4.036, released in 1993

Perl's history

- Larry Wall began work on Perl in 1987
- version 1.0 was released to comp.sources.misc newsgroup on December 18, 1987
- The language expanded rapidly over the next few years
- Perl 2, released in 1988, featured a better regular expression engine
- Perl 3, released in 1989, added support for binary data streams.
- Perl 4.036, released in 1993
- Perl 5, released on October 17, 1994.

The present and the future

- Perl 5.8, released on July 18th, 2002

The present and the future

- Perl 5.8, released on July 18th, 2002
- Perl 5.10, released on December 18th, 2007 (20th anniversary)

The present and the future

- Perl 5.8, released on July 18th, 2002
- Perl 5.10, released on December 18th, 2007 (20th anniversary)
- Perl 6 – design process began in 2000

Licence

- Perl is a free software

Licence

- Perl is a free software
- Artistic License

Licence

- Perl is a free software
- Artistic License
- GNU General Public License

What a mess!

- *Perl is a mess and that's good because the problem space is also a mess. – Larry Wall*

What a mess!

- *Perl is a mess and that's good because the problem space is also a mess. – Larry Wall*
- Syntax full of special characters

What a mess!

- *Perl is a mess and that's good because the problem space is also a mess. – Larry Wall*
- Syntax full of special characters
- Easy to write...

What a mess!

- *Perl is a mess and that's good because the problem space is also a mess. – Larry Wall*
- Syntax full of special characters
- Easy to write...
- ...but and provokes lazy programmers to write messy code

What a mess!

- *Perl is a mess and that's good because the problem space is also a mess. – Larry Wall*
- Syntax full of special characters
- Easy to write...
- ...but and provokes lazy programmers to write messy code
- **Making Easy Things Easy and Hard Things Possible**

What a mess!

- *Perl is a mess and that's good because the problem space is also a mess. – Larry Wall*
- Syntax full of special characters
- Easy to write...
- ...but and provokes lazy programmers to write messy code
- **Making Easy Things Easy and Hard Things Possible**
- *„We will encourage you to develop the three great virtues of a programmer: laziness, impatience, and hubris" – Larry Wall*

There's more than one way to do it

- **There's more than one way to do it** – well known Perl motto

There's more than one way to do it

- **There's more than one way to do it** – well known Perl motto
- Often abbreviated to TIMTOWTDI

There's more than one way to do it

- **There's more than one way to do it** – well known Perl motto
- Often abbreviated to TIMTOWTDI
- Perl is multi-paradigm: functional, imperative, object-oriented

There's more than one way to do it

- **There's more than one way to do it** – well known Perl motto
- Often abbreviated to TIMTOWTDI
- Perl is multi-paradigm: functional, imperative, object-oriented
- Rich syntax, sometimes similar to natural language

There's more than one way to do it

- **There's more than one way to do it** – well known Perl motto
- Often abbreviated to TIMTOWTDI
- Perl is multi-paradigm: functional, imperative, object-oriented
- Rich syntax, sometimes similar to natural language
- Freedom is good

There's more than one way to do it

- **There's more than one way to do it** – well known Perl motto
- Often abbreviated to TIMTOWTDI
- Perl is multi-paradigm: functional, imperative, object-oriented
- Rich syntax, sometimes similar to natural language
- Freedom is good
- Perl's philosophy differs from „Zen of Python" where „There should be one – and preferably only one – obvious way to do it"

Pandora's box

- CPAN – Comprehensive Perl Archive Network

Pandora's box

- CPAN – Comprehensive Perl Archive Network
- `http://www.cpan.org/`

Pandora's box

- CPAN – Comprehensive Perl Archive Network
- `http://www.cpan.org/`
- **Lots** of documentation, libraries (modules)

Pandora's box

- CPAN – Comprehensive Perl Archive Network
- `http://www.cpan.org/`
- **Lots** of documentation, libraries (modules)
- Today it has:
  - 6362 MB
  - 243 mirrors
  - 7794 authors
  - 16942 modules

hello1.pl

```
#!/usr/bin/perl
print "Hello World!\n"
```

hello2.pl

```
#!/usr/bin/perl
print "What is your name? ";
my $name = <STDIN>;
chomp $name;
print "Hello, $name!\n";
if ($name eq 'Bartosz')
{
        print "I'm here to serve, my Master!\n"
}
```

hello3.pl

```
#!/usr/bin/perl
print "What is your name? ";
$_ = <>;
chomp;
print "Hello, $_!\n";
print "I'm here to serve, my Master!\n" if /^Bartosz$/;
```

quotation.pl

```
#!/usr/bin/perl
$str1 = "foo\n"; $str2 = "bar$str1";
$str3 = 'foo\n'; $str4 = 'bar$str3\n';
print $str1, $str2, $str3, "\n", $str4, "\n";

$str5 = <<END
That's a multiline method to
input strings with "quotation"
and variable substitution: $str3
END
;
print $str5;
```

operators.pl

```
#!/usr/bin/perl
$apples = "10 apples";
$oranges = "10 oranges";

print $apples . $oranges, "\n";
print $apples + $oranges, "\n";
print "equal (==)\n" if $apples == $oranges;
print "equal (eq)\n" if $apples eq $oranges
```

array1.pl

```
#!/usr/bin/perl
@array = (31337, 'a', 'b', 'c', 10.66);
$count = @array;
print $_,', ' foreach @array;
print "array has $count elements\n";
```

array2.pl

```
#!/usr/bin/perl
@array = qw/Practical Extraction and Report Language/;
print "array has ".@array." elements\n";
print $e,"\n" while ($e = shift @array);
print "array has ".@array." elements\n";
```

context.pl

```
#!/usr/bin/perl
$var1 = localtime;
@var2 = localtime;

print $var1,"\n";
print $_,"\n" foreach @var2;
```

hash.pl

```
#!/usr/bin/perl
%countries = (
        'us' => 'United States of America',
        'pl' => 'Poland',
        'de' => 'Germany',
        'fi' => 'Finland'
);

print $countries{'pl'},"\n";
delete $countries{'pl'};
$countries{'se'} = 'Sweden';

while (($key, $value) = each(%countries))
{
        print "$key is $value\n"
}
```

regexp.pl

```
#!/usr/bin/perl
while (<STDIN>)
{
        print "Hooray!\n" if (/Perl/);
        print "Number: $1\n" if (/(\d+)/);
        if (/Windows/)
        {
                s/Windows/Linux/g;
                print;
        }
}
```

open1.pl

```
#!/usr/bin/perl
open (FILE, "> /tmp/perltest") or die "can't open file: $!";
print FILE "Hello World!\n";
close FILE
```

open2.pl

```
#!/usr/bin/perl
open (PROGRAM, "uname |") or die "can't open file: $!";
$_ = <PROGRAM>;
chomp;
print "Our system is $_\n";
close PROGRAM
```

backquote.pl

```
#!/usr/bin/perl
$_=`uname`;
chomp;
print "Our system is $_\n";
```

socket.pl

```
#!/usr/bin/perl
use Socket;
$|=1;

socket(SOCK, AF_INET, SOCK_STREAM, getprotobyname('tcp'))
        or die "socket() failed: $!";
setsockopt(SOCK, SOL_SOCKET, SO_REUSEADDR, 1)
        or die "Can't set SO_REUSEADDR: $!";
bind(SOCK, sockaddr_in(12345, INADDR_ANY))
        or die "bind() failed: $!";
listen(SOCK, SOMAXCONN)
        or die "listen() failed: $!";
accept(SESSION, SOCK);
print while(<SESSION>);
close SESSION
```

rhtml1.pl

```
#!/usr/bin/perl
while (<STDIN>)
{
        s/<.*?>//g;
        print
}
```

rhtml2

```
$ perl -ne 's/<.*?>//g;print'
```

rhtml3

```
$ perl -pe 's/<.*?>//g'
```

rhtml4

```
$ perl -i.bak -pe 's/<.*?>//g' *.html
```

perl.cgi

```
#!/usr/bin/perl -w
use CGI;

print "Content-Type: text/html\n\n";

my $ua = $ENV{'HTTP_USER_AGENT'};

print <<EOF
<body>
Your browser is $ua.
</body>
EOF
```

Simple JAPH

```
print "just another perl hacker\n"
```

japh1.pl

```
#!/usr/bin/perl

undef$/;$_=<DATA>;y/ODA\n / /ds;@yoda=map{length}split;print chr
oct join('',splice(@yoda,0,3))-111 while@yoda;
__DATA__
          OOOOOOOOOOOOOOOO       OOOO          DDOOOOOOOO
        ODOOOOOOOOOOOOOOOO      OOOOOO         OOOOOOOOOO
        OOOO       OOO           OOO   OOO     OOD      ODO
         OOOO      OOO          OOD     OOO    DOODOOOOOD
          OOOO     DOD          OOOOOOOOOOO    OOO   OOOO
 DDOOOODOOOOO      OOO        OOOOOOOOODOODO    OOD    DOOOOOODOOOO
 OOOOODOOOOO       OOO      OOOO          OODO   OOD     OOOOOOOOODOO

 OOOOO    ODOD    OOOOO    OOOO          DDDOOOOOOO       OOOOOOOOO
  OOOOO  OOODDO  OOOOO    OOODOO         OOOOOOOOOOO     OOOOOOOOOO
   OOOOOOODOOOOODDD    OOO  ODO         DDD      DOO     OOOOD
    OOOOOOOOOOOOOO    OOO     DOO       DOODOOOOOO       OODOO
     DOOOO   OOOOO    OOOOOOOOOODO      OOO   OOOO         OODOO
      OOO     OOO    OOOOODOOOOOOO     OOO    ODOOOOOOOOOOOOOO
       O       O    OOOD         OOOO   OOO     OOOOOOOOOOOOOO
```

japh2.pl

```
      $LOVE=                    AMOUR.
    true.cards.             ecstacy.crush
  .hon.promise.de      .votion.partners.
 tender.truelovers.  treasure.affection.
devotion.care.woo.baby.ardor.romancing.
enthusiasm.fealty.fondness.turtledoves.
lovers.sentiment.worship.sweetling.pure
.attachment.flowers.roses.promise.poem;
 $LOVE=~ s/AMOUR/adore/g; @a=split(//,
  $LOVE); $o.= chr (ord($a[1])+6). chr
   (ord($a[3])+3). $a[16]. $a[5]. chr
    (32). $a[0]. $a[(26+2)]. $a[27].
      $a[5].$a[25]. $a[8].$a[3].chr
        (32).$a[29]. $a[8].$a[3].
          $a[62].chr(32).$a[62].
           $a[2].$a[38].$a[4].
               $a[3].'.';
                  print
                   $o;
```

japh3.pl

```
not exp log srand xor s qq qx xor
s x x length uc ord and print chr
ord for qw q join use sub tied qx
xor eval xor print qq q q xor int
eval lc q m cos and print chr ord
for qw y abs ne open tied hex exp
ref y m xor scalar srand print qq
q q xor int eval lc qq y sqrt cos
and print chr ord for qw x printf
each return local x y or print qq
s s and eval q s undef or oct xor
time xor ref print chr int ord lc
foreach qw y hex alarm chdir kill
exec return y s gt sin sort split
```

Summary

- No time to tell about:
  - GUI
  - Advanced CGI
  - Object Oriented programming
  - Perl 6 and its innovations
  - ...

Summary

- No time to tell about:
  - GUI
  - Advanced CGI
  - Object Oriented programming
  - Perl 6 and its innovations
  - ...
- Perl is a multi-task programming language

Summary

- No time to tell about:
  - GUI
  - Advanced CGI
  - Object Oriented programming
  - Perl 6 and its innovations
  - ...
- Perl is a multi-task programming language
- Combines advantages of many other languages

Summary

- No time to tell about:
  - GUI
  - Advanced CGI
  - Object Oriented programming
  - Perl 6 and its innovations
  - ...
- Perl is a multi-task programming language
- Combines advantages of many other languages
- *I don't know if it's what you want, but it's what you get. :-) – Larry Wall*

Questions?

Bibliography

- `man perl`

Bibliography

- `man perl`
- `http://perlmonks.org/`

Bibliography

- man perl
- http://perlmonks.org/
- http://en.wikipedia.org/

Thank you for your attention.