PERL – zarządzanie złożonością

Pavel Pachkouski

Plan prezentacji

- Introduction
- Hello world
- Podstawy programowania w Perl
- Wyrażenia regularne
- Pewne ogólne Perl funkcje
- Zaawansowane funkcje
- Wnioski

Co to jest PERL ?

- Praktyczny język ekstrakcji i raportowania
- Stworzony w 1987 przez Larriego Walla i teraz powszechnie wykorzystywany
- Skryptowy język wysokiego poziomu, w którym są łączane cechy C, Sed, Awk, Unix shells i wiele innych
- Pierwotnie stworzony do przetwarzania tekstu
- „Perl is designed to make the easy jobs easy and hard jobs possible.”
- Motto Perla: „There’s more than one way to do it.”

INTRODUCTION : Cechy PERLa

- **Cechy języka**
  - Łatwy dla początkujących
  - Zawiera funkcje, ma OO rozszerzenia (obiektowość)
  - C-style struktury programu. Free-style składni poleceń
  - Wbudowane wyrażenia regularne
  - Wbudowany dostęp do baz danych
  - zgodny z POSIX
  - Rozszerzenia w postaci możliwości korzystania z różnego rodzaju modułów

INTRODUCTION : Cechy PERLa

- **Inne cechy:**
  - **Kompilowanie w czasie rzeczywistym**
  - **Może optymatyzować C kod**
  - **Może być bezpośrednio integrowany z C/C++ kodem**
  - **Wbudowany debugger**

Powszechna praktyka

- Zawsze dodawać #!/usr/bin/perl –w lub use warnings;
- Zastanowić się nad wykorzystaniem use strict dla skryptów większych niż 10 linijek
- Nie porządane jest mieć zbyt dużo komentarzy
  - #
  - =head

    =cut
  - perldoc

HELLO WORLD!

- **Przykład „hello.pl”:**

```
#!/usr/bin/perl
print „Hello world!”
```

- **Wiersz poleceń**

```
> perl hello.pl
Hello world!
>
```

- **Musimy zmienić uprawnienia najpierw:**

```
> chmod +x hello.pl
> ./hello.pl
```

Podstawy programowania w PERL: Typy danych

- **Scalary**

$var

- **Listy (Arrays)**

@var $var[1..n]

- **Hash'y (tablice asocjacyjne)**

%var $var['key']

- **Złożone struktury danych**

Podstawy programowania w PERL: Zmienne specjalne

- Default argument: $_
- Input record separator: $/
- Output field separator: $,
- List separator: $”
- Process Id: $$
- Program name: $0
- Command-line arguments: @ARGV
- Subroutine arguments: @_
- Environment variables: %ENV

Podstawy programowania w PERL: Konstrukcje sterujące

```
if ( expression ) { block }
if ( expression ) { block } else { block }
if ( expression ) { block }
   elsif ( expression ) { block }
   elsif ( expression ) { block }
   ...
   else { block }
unless ( expression ) { block }
unless (expression ) { block } else { block }
```

Podstawy programowania w PERL: Konstrukcje sterujące

- while (expression) { block }
- while (expression) { block } continue { block }
- do { block } while (expression);
- until (expression) { block }
- for (statement ; expression; statement) { block }
- foreach variable (list) { block }
- label :

goto label;

Podstawy programowania w PERL: Subroutines

```
sub Add {
        local ($x, $sum);
        $sum = 0;
        foreach $x (@_){
                $sum +=$x;
                }
        $sum
        }
$test = &Add (2, $number, @lisofnumbers);
```

Podstawy programowania w PERL: Wyrażenia regularne

- **Specyficzne nie tylko dla Perl**
- **Bardzo wygodne**
- **Co mogą robić:**
  - porównywać linijki (strings)
  - zastępować linijki (strings)
  - wybierać linijki (strings)

Podstawy programowania w PERL: Regex ?

$string = ~/find/
$string =~/^find/

$string =~/find$/
$string=~/^find$/

. Match any character

\w Match „word” character (alphanumeric plus ”_”)

\W Match non-word character

\s Match whitespace character

\S Match non-whitespace character

\d Match digit character

\D Match non-digit character

\t Match tab

\n Match newline

\r Match return

Podstawy programowania w PERL: Znowu wyrażenia regularne

Podstawowe elementy:

|	alternacja
()	grupowanie
[]	klasa znaków
[^]	nie klasa znaków
*	pasuje 0 lub więcej
+	pasuje 1 lub więcej
?	pasuje 1 lub 0
{n}	pasuje dokładnie n razy
{n,}	pasuje n lub więcej
{n,m}	pasuje n lub więcej ale mniej niż m

Pewne ogólne Perl funkcje: Strings

- length
  ```
  $l = length $string
  $l = length; #uses $_
  ```
- split
  ```
  @list = split /[,\s]/, $string, 10;
  ($name, $value) = split /=/;
  ```
- substr
  ```
  $piece = substr $string, 2, 10;
  ```
- chop & chomp
  ```
  $c = chop $string;
  chomp @lines;
  $/= ”; chomp;
  ```
- pack & unpack

Pewne ogólne Perl funkcje: Lists

- **push & pop**

```
push @lists, $item;
$num = push @list, @items;
$item = pop @list;
```

- **shift & unshift**

```
$item = shift @list;
unshift @list, $item;
```

- **sort**

```
@sorted = sort @list;
print sort {$a ⇔ $b} @list;
```

- **splice**

```
@sublist = splice @list, 2, 5;
splice @list, $off, $len, @newitems;
```

Pewne ogólne Perl funkcje: Inne

- **time & localtime**

```
($sec,$min,$hour,$mday,$mon,$year,$wday,
$yday,$isdst) = localtime time;
$now = localtime
# ”Thu Oct 13 04:54:34 1994”
```

- **rand & srand**

```
srand time;
$x = rand 10;
```

Kiedy wykorzystujemy PERL

- Zaawansowane ”shell scripts”.
- Zarządzanie procesami
- Szybkie procedury do wsadowego przetwarzania plików lub baz danych
- Cokolwiek co potrzebuje obsługi dużej liczby linijek
- CGI i inne(web-programming).

Kiedy NIE wykorzystujemy PERL

- Obliczenia o wysokim stopniu obciążenia
- Duże aplikacje

Przekazywanie wartości do programu lub podprogramu

- Przekazywanie jest przez wartość
  - scalar może mieć jako wartość „wskaźnik” na array, hash, function etc.
- Parametry do programu lub funkcji przychodzą w wartości @_
- my $first_value = shift @_;
- my first_value = $_[1];
- my first_value = shift;

Co to jest moduł?

- Dwa typy
  - Obiektowo-zorientowany typ
    - Dostarcza coś podobnego do definicji klasy
  - Zdalne wywołanie funkcji
    - Dostarcza metodę importu subroutines lub zmiennych do użytku w main programu

Perl DBI

- Metoda perla łączenia z bazą danych (wirtualnie z dowolną bazą danych), czytania i modyfikacji bazy danych.
- Składnia jest bardzo podobna do składni SQL. Znajomość SQL jest wymagana!

Instrukcje DBI

- **Connect**
  - wykorzystywana do tworzenia połączenia z bazą danych
- **Prepare**
  - wykorzystywana do przygotowania instrukcji do wykonania
- **Execute**
  - wykorzystywana do wykonania instrukcji
- **Do**
  - przygotowanie instrukcji która nic nie zwraca i wykonać ją

Instrukcje DBI

- **Fetch**
  - Kilka typów wykorzystane żeby zwrócić dane
- **Disconnect**
  - Rozłączyć z serwerem (bazą danych)

Typy fetch

- „fetchrow_arrow”
  - Wykorzystana żeby pobrać tablicę skalarów (array of scalars) za każdym razem
  - jako alternatywa może być wykorzystana „fetchrow_arrayref”
- „fetchrow_hash”
  - Wykorzystana by pobrać hash indeksowany nazwą kolumny
  - Wolniej, ale kod bardziej przejrzysty
  - jako alternatywa może być wykorzystana „fetchrow_hashref”.

Bardziej zaawansowane instrukcje

- **Quote**
  - Wykorzystane żeby właściwe zacytować dane do wykorzystania z „prepare” instrukcją
    - „$value = $dbh->quote($blast_result);”
- **Placeholders**
  - Szybkość wykonania znacznie wzrasta
    - my $prep = $dbh -> prepare („select x from y where z = ?”);
    - loop_start
    - $prep -> bind_param(1,$z);
    - $prep -> execute();
    - loop_end

Gdzie można znaleźć więcej informacji?

- **Books**
  - „Programming Perl” 2/e, Larry Wall, Tom Christiansen & Randal Schwartz (The camel book).
  - „Learning Perl” 2/e, Randal Schwartz, Tom Christiansen & Larry Wall. (The llama book.)
  - „Perl in a Nutshell”, Ellen Siever, Stephen Spainhour & Nathan Ptwardhan.
  - „Advanced Perl Programming”, Sriram Srinivasan.

More INFO

- **Web-sites:**
  - Perl Core Documentation
    - http://www.perldoc.com/
  - Nik Silver's Perl Tutorial:
    - http://www.comp.leeds.ac.uk/nik/start.html

Dziękuję za uwagę !!!