Systemy operacyjne 07

Wstęp do języka Perl

1. Czym jest język Perl

Jest to bowiem połączenie elementów języków takich jak C, awk, sed, grep i Bourne shell. Perl jest doskonałym narzędziem do obróbki tekstu i plików binarnych, pobierania z nich informacji i generowania komunikatów na ich podstawie Linux wszystko traktuje jako plik, niezależnie od tego czy to jest plik zwykły, katalog, urządzenie blokowe (klawiatura, ekran) itd. Nie inaczej jest ze strumieniami, BASH identyfikuje je za pomocą przyporządkowanych im liczb całkowitych (od 0 do 2) tak zwanych deskryptorów plików. Stąd też wzięła się jego nazwa, będąca skrótem od Practical Extraction and Report Language(Praktyczny Język Pobierania Danych i Raportowania). Najczęściej stosowany do pisania skryptów CGI (Common Gateway Interface).

Perl należy do tych języków programowania, których można się nauczyć szybko. Perl nie wymaga deklarowania typów zmiennych przed ich użyciem. Wystarczy „po prostu" napisać, co ma być zrobione. Warto więc już na początku zapamiętać, ze Perl nie jest najlepszy do wszystkiego – w szczególności nie należy rozwiązywać przy jego pomocy skomplikowanych problemów wymagających użycia złożonych struktur danych. Łatwo natomiast przetwarzać dane binarne.

2. Cechy Perla

- nazywany językiem zarządzania systemem, gdyż może zastąpić dotychczasowe skrypty shella,
- w języku angielskim istnieje inne złośliwe rozwinięcie skrótu Perl: "Pathologically Eclectic Rubbish Lister". Wynika to z użycia wielu znaków przestankowych w składni języka,
- kompilatory Perla są bezpłatne i dostępne dla wielu systemów operacyjnych. Większość skryptów jest przenośna. Sam Perl to wolne oprogramowanie,
- stosunkowo łatwy w użyciu oraz wydajny, przypłacając to może nieco elegancją i czytelnością,
- Perl nie jest językiem kompilowanym, jednak jest szybszy od większości języków interpretowanych. Perl jest językiem skryptowym, tzn. tworzone w nim programy są po prostu plikami tekstowymi, które następnie są wykonywane przez interpreter Perla,
- oprócz programów wykonywanych "wiersz po wierszu" pozwala on konstruować skomplikowane struktury danych i programować w stylu obiektowym,
- hasło przewodnie „Każde zadanie można wykonać na więcej niż jeden sposób". Powoduje to, iż programista może opracować swój własny koncept myślenia i własny styl programowania.
- w celu rozróżnienia język Perl pisze się dużą, zaś nazwę programu, małą litera,

- w zastosowaniach www Perl wykorzystywany jest po stronie serwera**.**

3. Edytor

Do pisania w Perlu wystarczy zwykły edytor tekstowy (np. notatnik), ale dużo łatwiej pisze się w edytorze zpodświetlaną składnią.

PERL Code Editor ma status freeware. Jego rola ogranicza się wyłącznie do edytora plików Perla – prawie zerowe możliwości konfiguracji. http://www.perlvision.com/pce/

Context to darmowy, obsługujący wiele języków edytor. Obsługuje makra, eksportuje do formatu rtf, oraz html, bardzo szybki. http://www.fixedsys.com/contex

CodeWhiz. (wersja trial) http://www.incatec.com/

Visual Perl Editor. http://www.xarka.com/vpe.html

Ultra Edit. http://www.idmcomp.com

DzSoft Perl Editor. http://www.dzsoft.com/dzperl.htm

Zabobon Edytor. Podświetla składnię Perla, php, javy, pascala.
http://www.zabobonedytor.prv.pl/

Visual Studio Code https://code.visualstudio.com/

4. Pisanie i uruchamianie skryptu

Perl jest, podobnie jak C, językiem bez ścisłego formatu. Nie występuje w nim struktura linii używana w Fortranie. W zależności od tego gdzie znajduje się interpreter. **Każda komenda musi kończyć się średnikiem (;).** Tekst rozpoczynający się od znaku hash (#) jest traktowany jako komentarz. Bloki kodu, które obejmują ciało warunkowe lub pętle są ograniczane nawiasami podobnie jak w C ({...}). Po napisaniu odpowiedniego kodu, zapisujemy go pod wybraną nazwą, nadając rozszerzenie ".pl".

Najprostszy program:

```
#!/usr/bin/perl

print "Hello, World!\n";
```

Pierwszą linią każdego programu jest "shebang". Linijka ta informuje nas, pod jaką lokalizacją znaleźć można interpreter Perla. Funkcja "print" powoduje wysłanie na ekran łańcucha "Hello, World!". Po niej występuje średnik, który musi pojawić się na końcu każdej instrukcji.

Aby uruchomić skrypt należy w wierszu

poleceń wpisać "**perl**" i **nazwę skrytpu**:

```
perl hello.pl
```

5. Opcje wywoływania programu

Wymienionych poniżej opcji można używać podczas wywołania programu z linii poleceń, lub (systemy UNIX) w linii, gdzie podajemy ścieżkę do interpretatora:

-h # wyświetla pomoc dotyczącą parametrów uruchamiania programu,

-c # kompilacja programu w celu wykrycia błędów, bez jego wykonania,

-d[:debugger] # uruchamia skrypt pod danym debuggerem,

-T # uniemożliwia wykonanie operacji związanych z systemem plików lub systemem operacyjnym, co powoduje # zwiększenie bezpieczeństwa programów CGI,

-v # wyświetla aktualnie używaną wersję interpretatora Perl,

-w # powoduje wyświetlenie dodatkowych ostrzeżeń przy debugowaniu programu. Może powodować problemy #w skryptach wykorzystujących interfejs CGI (przez serwer WWW).

6. Typy danych

1. Zmienne

Zmienna jest podstawowym typem w Perlu. Zmienna może mieć wartość całkowitą, zmiennoprzecinkową, lub znakową. Perl ustala typ zmiennej z kontekstu. Zmienne zawsze posiadają prefiks $. Np.:

```
$str = "Hello world!";
```

Zmienne w Perlu nie musza być deklarowane. Są alokowane dynamicznie. Domyślną wartością zmiennej jest, w zależności od kontekstu, 0 lub ciąg pusty. Zmienne znakowe użyte w kontekście liczbowym są interpolowane do ich wartości. Np.:

```
$x = 8; # zmienna całkowita
$y = "15"; # ciąg znaków
$z = $x+$y; # zmienna z jest typu całkowitego i równa 23
```

Konwersja może zachodzić również w drugą stronę np.:

```
$answer = 42;
print "the answer is $answer"; #na ekranie:"the answer is 42
```

2. Wektory zmiennych

Perl zawiera wektory (lub listy) zmiennych. Bieżącą wartość udostępnia prefiks at (@). Można również przypisać elementy wektora przez nazwę. Oto kilka sposobów:

```
@numbers = (3,1,4,1,5,9);
@letters = ("this","is","a","test");
($word,$another_word) = ("one","two");
```

Można się odwoływać do poszczególnych elementów, przy czym pierwszy element ma indeks 0:

```
$cos[2] = 2.718281828;
$message[12] = "wiadomosc\n";
```

Ciąg $# używa się do znalezienia ostatniego ważnego indeksu wektora, a nie jego rozmiaru. Zaś zmienna $[oznacza numer pierwszego elementu w każdym wektorze.

Domyślną jego wartością jest 0. Przykład programu informującego o ilości elementów w wektorze @a :

```
$n = $#a - $[ + 1;
print "ilosc elementow w wektorze: $n \n";
```

Wektory są rozwijane dynamicznie. Wystarczy przypisać wartość zmiennej $#, a dany wektor zostanie alokowany.

```
$#months = 11; # elementy 0 - 11
```

3. Skojarzone wektory zmiennych

Jest to najbardziej użyteczna cecha Perla. Można dzięki niej tworzyć tablice użytkowników według "login name" i tablice nazw plików. Prefiksem dla tego rodzaju zmiennych jest znak procenta (%). Kluczem dla tych wektorów są zmienne znakowe (numeryczne ulegają konwersji do znakowych). Np.

```
%quota = ("root",100000,
"pat",256,
"fiona",4000);
```

odwoływanie się do elementów ma następującą postać:

```
$quota{dave} = 3000; # dave - klucz, 3000 - wartość
```

Należy zwrócić uwagę na wyłączanie się nazw. W Perlu zmienne, wektory, wektory skojarzone, funkcje i pakiety mogą mieć ta sama nazwę i nie będzie to rodzić konfliktów.

7. Operatory i znaki porównania

Zestaw operatorów i znaków porównania w Perlu jest bardzo zbliżony do C. Wszystkie operacje arytmetyczne z C są przeniesione do Perla. Poniższe są ważne tylko dla Perla:

** Operator wykładniczy

() Zerowa lista wektora

.= Przypisanie połączenia

**= Przypisanie wykładnicze

. Połączenie dwóch ciągów znakowych

eq Równość ciągów znakowych (odpowiednik ==)

x Operator powtórzenia

.. Operator zakresu

8. Zmienne predefiniowane

Perl ma pewien zbiór zmiennych predefiniowanych. O wszystkich można przeczytać w manualu. Oto niektóre z nich:

$_ Domyślny argument funkcji i struktur.

$liczba Kolejne dopasowane podciągi z wyrażenia regularnego.

$. Numer linii w ostatnio czytanym pliku.

@ARGV Lista argumentów skryptu. $ARGV[0] jest pierwszym argumentem, nie jak w

C, nazwą programu. Nazwa kryje się pod $0

%ENV Wykaz zmiennych środowiska

9. Instrukcja warunkowa if

W Perlu jest podobnie jak w C. Używa się tu tych samych operatorów: "&&" to "i", "||" - lub, "!" zaś to negacja. Jedyną różnicą jest brak jednolinijkowego wykonania warunku. To znaczy zamiast:

```
if ($error < 0)
fprintf(stderr,"Blad o kodzie %d\n",error);
```

należy w Perlu zapisać:

```
if ($error < 0)
{ print STDERR "Błąd o kodzie $error\n"; }
```

Natomiast odpowiednikiem dwóch słów w C else if jest w Perlu słowo elsif, poza tym istnieje słowo przeciwstawne unless. Na przykład:

```
unless ($#ARGV > 0) # czy są jakieś argumenty
{ print "Błąd, brak argumentow\n"; exit 1; }
```

Idea wartości logicznych jest identyczna do C. Zero to fałsz, nie zero prawda. Pusty ciąg znaków - fałsz, odługości 1 lub więcej - prawda. Wektory i ich skojarzone odpowiedniki o ilości elementów 0 - fałsz, więcej -prawda. Nieistniejące zmienne, mają z definicji wartość zero, czyli fałszu.

10. Przykłady prostych programów

1. Proste obliczenia matematyczne

```
#!/usr/bin/perl
# Operands
$a = 10;
$b = 4;

# using arithmetic operators
print "Addition is: ", $a + $b, "\n";
print "Subtraction is: ", $a - $b, "\n" ;
print "Multiplication is: ", $a * $b, "\n";
print "Division is: ", $a / $b, "\n";
print "Modulus is: ", $a % $b, "\n";
print "Exponent is: ", $a ** $b, "\n";
```

2. Działanie operatorów logicznych

```
#!/usr/bin/perl
# Operands
$a = true;
$b = false;

# AND operator
$result = $a && $b;
print "AND Operator: ", $result,"\n";

# OR operator
$result = $a || $b;
print "OR Operator: ", $result,"\n";

# NOT operator
$d = 0;
$result = not($d);
print "NOT Operator: ", $result;
```

3. Instrukcja warunkowa if

```
# Perl Program to illustrate the Operators

# Operands
$a = 10;
$b = 4;
$c = true;
$d = false;

# using arithmetic operators
print "Addition is: ", $a + $b, "\n";
print "Subtraction is: ", $a - $b, "\n" ;

# using Relational Operators
if ($a == $b)
{
   print "Equal To Operator is True\n";
}
else
{
   print "Equal To Operator is False\n";
}

# using Logical Operator 'AND'
$result = $a && $b;
print "AND Operator: ", $result, "\n";

# using Bitwise AND Operator
$result = $a & $b;
print "Bitwise AND: ", $result, "\n";

# using Assignment Operators
print "Addition Assignment Operator: ", $a += $b, "\n";
```

Zadania do wykonania

1. Napisać skrypt pytający się czy jest wieczór. Dla odpowiedzi 'tak' powinien wypisać 'Dobry wieczór', dla odpowiedzi 'nie' - 'Dzień dobry', dla pozostałych odpowiedzi 'Nie rozpoznana odpowiedz: ' i przytoczyć treść odpowiedzi. (Użyj instrukcji warunkowej if.

2. Napisać skrypt pobierający numer dnia tygodnia i wypisujący jego nazwę lub informację "Nic nie wybrałeś”.