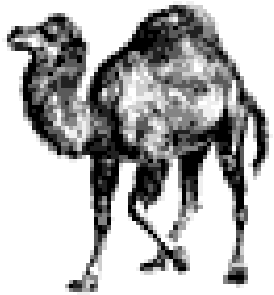


Programming Perl

Perl – podstawy

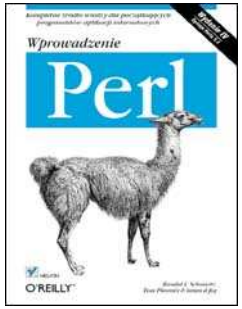
***P**ractical **E**xtraction and **R**eport **L**anguage*

dr inż. Krzysztof Juszkiewicz

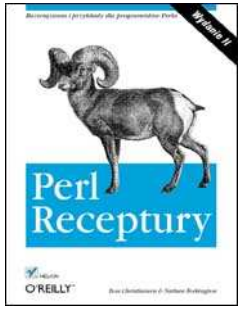
Wiedza i środowisko

- Książki, manuale, FAQ
  - **http://www.kt.agh.edu.pl/perl-faq/Wiedza**
- CPAN – wszystko, co potrzeba
- Standardowy „pakiet” w Linuxie + pakiety rpm z modułami
- ActivePerl dla WIN32
  - http://www.activestate.com/
- Cygwin

Książki

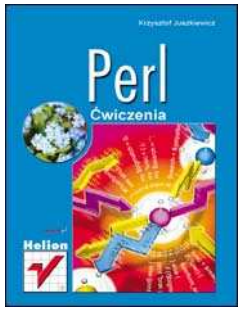


Randal L. Schwartz, ... "Perl. Wprowadzenie.", (*Learning Perl* [*Lama Book*]), Helion 2006,



Tom Christiansen, ... "Perl. Receptury", (*Perl Cookbook),* Helion 2004,

Treść: www.unix.org.ua/orelly/perl/



Krzystof Juszkiewicz, "Perl. Ćwiczenia", Helion 2003,

CPAN

Comprehensive Perl Archive Network

2008-02-13 online since 1995-10-26
4252 MB 236 mirrors
6419 authors 12998 modules

Welcome to CPAN! Here you will find All Things Perl.

Browsing

- Perl modules
- Perl scripts
- Perl binary distributions ("ports")
- Perl source code
- Perl recent arrivals
- recent Perl modules
- CPAN sites list
- CPAN sites map

Searching

- Perl core documentation (perldoc.perl.org; Jon Allen)
- Perl core and CPAN modules documentation (Randy Kobes)
- CPAN modules, distributions, and authors (search.cpan.org)

FAQ etc

- CPAN Frequently Asked Questions
- Perl FAQ
- Perl Mailing Lists
- Perl Bookmarks

Yours Eclectically, The Self-Appointed Master Librarian (OOK!) of the CPAN
Jarkko Hietaniemi cpan@perl.org [Disclaimer] 2001-04-01

Instalacja modułu CPAN

```
perl -MCPAN -e shell
```

- Pierwszy krok to konfiguracja – dobrze jest ustawić szybki serwer CPAN i instalację modułów zależnych

```
prerequisites_policy=
    follow automatically
```

- Kolejny – instalacja modułu (np. URI):

```
install URI
```

CYGWIN

- Cygwin Home
- Cygwin/X Home
- Red Hat Cygwin Product
- Community
  - Reporting Problems
  - Mailing Lists
  - Newsgroups
  - Gold Stars
  - Mirror Sites
  - Donations
- Documentation
  - FAQ
  - User's Guide
  - API Reference
  - Acronyms
- Contributing
  - Snapshots
  - Source in CVS
  - Cygwin Packages

What Is Cygwin?

Install Cygwin now

Cygwin is a Linux-like environment for Windows. It consists of two parts:

- A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality.
- A collection of tools which provide Linux look and feel.

The Cygwin DLL works with all non-beta, non "release candidate", ix86 32 bit versions of Windows since Windows 95, with the exception of Windows CE.

PayPal DONATE

What Isn't Cygwin?

- Cygwin is **not** a way to run native linux apps on Windows. You have to rebuild your application *from source* if you want it to run on Windows.
- Cygwin is **not** a way to magically make native Windows apps aware of UNIX ® functionality, like signals, ptys, etc. Again, you need to build your apps *from source* if you want to take advantage of Cygwin functionality.

Help, contact, web page, other info... Historical cygwin info...

Install or update **now!** (using setup.exe) or get help on using setup.exe. or find where a package or file lives in the cygwin release.

Latest Cygwin DLL release version is 1.5.21-1

Dostęp do manuali

- perldoc.perl.org/,
- `man perl` lub `perldoc perl`,
- `perldoc -f function`,
- `perldoc perlfaq` (części 1-9),
- `perldoc -q word`

Manuale – wybrane przykłady

`perlsyn` – składnia,
`perlop` – operatory,
`perlfunc` – funkcje,
`perlsub` – własne,
`perlrun` – opcje,
`perlvar` – zmienne,
`perlre` – wyr. regul.,
`perltrap` – pułapki,

`perllol` – listy list,
`perlref` – wskaźniki,
`perlipc` – IPC,
`perlmod` – moduły,

`perlintro` – tutorial,
`perl` – spis manuali,

Cechy

- język interpretowany – szybkość nie jest kluczowa,
- przenośność – na wszystkie platformy,
- modułowość i popularność – duża liczba modułów = CPAN,
- łatwość pisania i poprawiania,
- interpreter, kompilator, debugger,
- TMTOWTDI = There's More Than One Way To Do It

Perl

***P**athologically **E**clectic **R**ubbish **L**ister*

Jak się rzuci kota na klawiaturę,

to ...

prawie na pewno napisze ...
poprawny skrypt w Perlu

Edytor Perla

- Vim – podświetlanie składni, debugger,
- Ultraedit (40$), DzSoft Perl Editor (49$),
- Open Perl IDE, Perl Express IDE,
  EPIC Perl IDE,
- Komodo IDE (30-300$),
  Perl Builder IDE (179$),

Komentarze

Znakiem komentarza obowiązującym do końca wiersza jest: #

```
# This is just
# a multiline comment
```

Komentarze wielowierszowe

Alternatywą jest użycie formatu dokumentacji `pod` (*Plain Old Documentation*). Słowo kluczowe (początkowe, tu: **pod**) może być inne.

```
=pod
This is just
a multiline comment
=cut
```

Zmienne liczbowe

- Dynamiczność tworzenia, automatyczny typ i automatyczna konwersja
- Skalary [`$calar`]:
  - wartość liczbowa: `$x = 3;`
  - duża liczba: `$bn = 7_000_000;`
  - naukowa: `$sci = 7e-4;`
  - suma podstaw 8 i 16: `071 + 0x1fa;`
  - liczba binarna: `$bin=0b0110;`

Interpolacja wartości

```
$ame = 7;
```

- wartość tekstowa ze znakiem dolara:

```
$y = 'n$ame';
```

- wartość tekstowa z interpolacją:

```
$y = "n$ame";
$y = "n$ame,rica";
$y = "n${ame}rica";
```

Wartości zmiennych

- zmienna o nazwie w zmiennej `$y`:

```
$var1 = 3;
$y = 'var1';
$new = $$y;
```

- wartość zmiennej niezdefiniowana:

```
$/ = undef;
```

Zmienne (tablice)

Wektory [`@rray`]:

- Pusta lista:

```
@w = ();
```

- Lista wartości:

```
@w = ('Ann',3);
```

- Wartość ósmego elementu:

```
$w[7] = $x;
```

Liczba elementów tablicy

- Ostatni istniejący indeks:

```
$#w = 20;
```

- Liczba elementów (zmiana kontekstu):

```
$num = @w;
```

- Zmiana kontekstu wprost:

```
scalar(@w);
~~@w;
```

Kopiowanie wartości tablic

- Kopia tablicy:

```
@W = @w;
```

- Kopie elementów o indeksie `0` i `2`:

```
($e,undef,$f) = @w;
```

Tablice tylko z wybranymi elem.

```
@tab = (9,8,7,6,5,4,3,2,1);
@numers = (1,4,7,5);
@tab[@numbers];
```

lub

```
@tab[1,4,7,5];
```

Zmiana wielu wartości

```
($e,undef,$f) = (undef,$f,$e);
```

Inaczej:

```
$e = undef;
$f = $e;
```

Jest to sposób na zamianę wartości
dwóch zmiennych bez zmiennej pośredniej.

Kasowanie wartości

Skasowanie jednej wartości:

```
delete $w[0];
```

Usunięcie całej tablicy w dwóch krokach:

```
@w = ();
undef @w;
```

Zmienne (hasze)

- Wektory asocjacyjne [`%hash`], czyli „małe” bazy danych, bez kolejności, szybkie:
  - wartość dla danego klucza:
    ```
    $v{'Kate'} = 5;
    ```
  - podstawienie za hasz:
    ```
    %v = ('Ann' => 3);
    %v = ('Ann' , 3);
    ```

Kasowanie haszy

Skasowanie jednej wartości:

```
delete $v{'Ann'};
```

Usunięcie całego hasza w dwóch krokach:

```
%v = ();
undef %v;
```

Konwersje tablic

- Parzyste = klucze, nieparzyste = wartości

```
@tab = ('Ann',3,'Eve',5);
%v = @tab;
$v{'Ann'} = 3;
```

- Tablica wartości hasza o podanych kluczach

```
@keys = ('Ann');
@hash{@keys};
```

Zmienne wbudowane

- Zmienne wewnętrzne (`man perlvar`):

  `$_`, `@_` (zmienne domyślne),
  `$!` (komunikat błędu),
  `$0` (nazwa skryptu),
  `@ARGV` (argumenty),
  `%ENV` (środowisko),
  `@INC` (ścieżki do modułów),
  `%INC` (dołączone moduły),
  `%SIG` (funkcje obsługi sygnałów), ...

Lokalność zmiennych

Są cztery (5.10) modyfikatory zmiennych:

- `our` – *globalna* (cały program),
- `my` – *lokalna* (tylko w bloku/funkcji),
- `local` – *leksykalna stała,*
- **`state`** – *leksykalna stanu,*

```
local $x=7;  #local value
$::x=10;     #global variable
```

false i defined

Wartość *nieprawda* w Perl:

`undef`, `0`, `""`, `"0"`,

Funkcja `defined` sprawdza wartość `undef` dla zmiennej, tablicy, funkcji, itp.

Wartość `undef` mają elementy, których brak w tablicy.

Konstrukcja for (=foreach)

```
for (my $x=0;$x<=$#nums;$x++) {}

for my $val (@nums) {
  $x++;
  next;      # next value
  last;      # exit loop
}

while() {}          until() {}
```

Konstrukcja warunkowa

```
if () {                    unless () {}
} elsif() {
} else {
}

$x=3 if $var and $val;
$var && $x=3;
($var ? $z1 : $z2) = 7;
```

switch

Od 5.10 jest konstrukcja `switch`:

```
given ($condition) {
   when () { ... }
   when () { ... }
   default { ... }
}
```

Zbędne `break`, a `switch` trzeba włączyć.

Wywołanie skryptu w Perlu

Plik o nazwie `file.pl` uruchamiamy:

```
perl file.pl arg1 arg2
./file.pl arg1 arg2
```

Wartości argumentów `arg1, arg2` są w tablicy `@ARGV`

Uruchamianie jednowierszowe (opcja `-e`):

```
perl -le '$x=1'
```

Opcje wywołania

Manual `perlrun`

- `-e` – wywołanie jednowierszowe,
- `-l` – znak nowego wiersza na końcu
- `-w` – włącz ostrzeżenia,
- `-c` – sprawdź poprawość kodu,
- `-d` – tryb śledzenia,
- `-n`, `-p` – przerób plik z/bez wypisywania,
- `-i` – j.w. z podmianą pliku.

Jednowierszowiec

Zamiana znaku końca wierszy DOS (`\r\n`) na UNIX (`\n`)

```
perl -pi -e 's/\r\n/\n/' plik
perl -pi -e "s/\r\n/\n/" plik
```

Pragma use (`perlmodlib`)

Dodanie nowej semantyki do skryptu:

```
use POSIX;   # new library
use 5.6.1;   # perl version required
use locale;  # localization
use threads; # POSIX threads
use utf8;    # UTF-8 support
use feature; # new adds since 5.10
```

Pierwsze programy

```
#!/usr/bin/perl -w
use warnings;      # the same as -w
use strict;        # no unsafe constr.
use diagnostics;   # verbose warnings
#use sigtrap;      # signal traps
```

- Ostrzeżenia o potencjalnych błędach,
- Sprawdzanie istnienia nazw funkcji, zmiennych, referencji.

Wyłączanie ostrzeżeń

Czasem chcemy wykorzystać **świadomie** cechy Perla normalnie powodujące ostrzeżenia – tylko dla kawałka kodu.

```
no strict 'vars', 'refs', 'subs';
no warnings 'numeric','once', ...;

perldoc perllexwarn
```

Wywołanie funkcji

- Nawiasy są zbędne,
- spacja po nazwie również,
- formalna notacja z użyciem `&`:

```
&subroutine();
```

- z innego pakietu:

```
POSIX::open
```

- archaiczne wywołanie:

```
do subroutine();
```

Funkcja `print`

Funkcja `print` drukuje na ekranie podaną listę lub ciąg znaków (szybsze).

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

my @w=(1,3,7,4);
print 'test:', @w , $w,"\n";
print ('last:' . $w[-1] . ~~@w);
```

say

Zamiast:

```
print "Don't do it\n";
```

można (od 5.10):

```
say "Don't do it";
```

Wbudowane funkcje skalarów

- **substr**(`$var,$off,$len,$new`) – podciąg od `$off` długości `$len` zamień na `$new`,

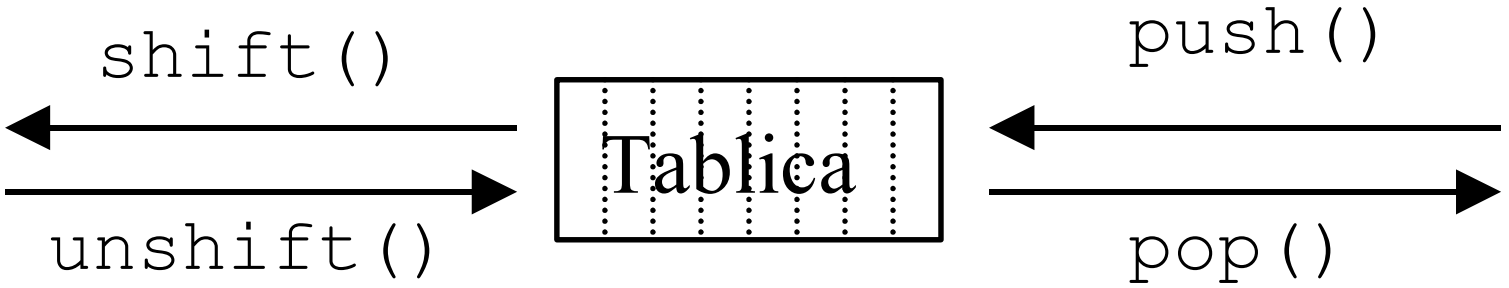
```
my $var = "abcde\n";
my $n = substr($var,2);
$n = "cde\n"; # to the end, no change
```

- **length**(`$var`) – liczba znaków,
- **chomp**(`$var`) – usuń ostatni „biały znak”,
- operatory

Wbudowane funkcje tablic

`join()` (na skalar), `split()` (ze skalara),

`splice()` (edytuj), `grep()` (znajdź),
`map()` (modyfikuj), `sort()` (sortuj)



Wbudowane funkcje haszy

`keys(%v)` – klucze,

`values(%v)` – wartości,

`each(%v)` – pary (klucz,wartość).

```
for $k (keys(%v)) {
    print $k.'->'.$v{$k}."\n";
}
```

Funkcje obsługi błędów

- `exit()` – zakończenie programu,
- `warn($!)` – wyświetleniem tekstu ostatniego błędu (`$!`) na standardowe wyjście błędów,
- `die($!)` – j.w. z zakończeniem programu,