Lekcija 2 - Uvod u Perl jezik

Contents

LearningObject

L2: Uvod u Perl jezik

Uvod *
Ovo poglavlje predstavlja uvod u Perl jezik.

Skriptnig jezici

Cilj

- Upoznavanje sa skriptnig jezicima

Skriptni jezici

Skriptni jezici

Definicija

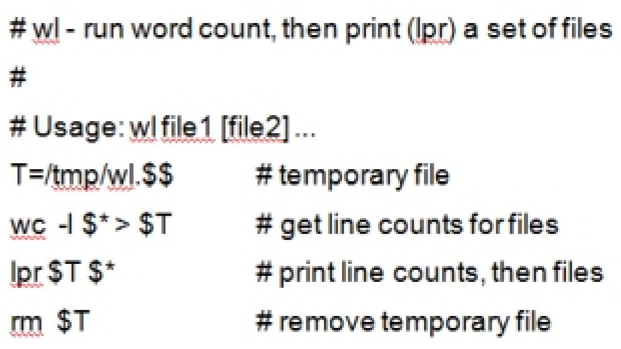
Skriptni jezik je programski jezik koji podržava pisanje skripti, programa napisanih za softverska okruženja koja automatizuju izvršavanje zadataka, a koje bi operatorr inače morao da izvršava korak po korak.

Okruženja koje se mogu automatizovati uz pomoć skripti uključuju softverske aplikacije, web stranice unutar web pretraživača, ljuske operacionih sistema i nekoliko programskih jezika opšte namjene.

Skriptni jezici se razlikuju od programskim jezika po tome [to su napravljeni za "lijepljenje" aplikacija zajedno. Oni koriste “typeless” pristup kako bi se postigao veći nivo programiranja i brži razvoj aplikacija u odnosu na programske jezike.

Primjer lepljenja

Slika 1 primer lepljenja.



```
# wl - run word count, then print (lpr) a set of files
#
# Usage: wl file1 [file2] ...
T=/tmp/wl.$$          # temporary file
wc  -l $* > $T        # get line counts for files
lpr $T $*             # print line counts, then files
rm  $T                # remove temporary file
```

Slika 1 Primer lepljenja

U ovom prmjeru wl script „lijepi“ zajedo tri Unix alata wc, lpr, and rm. Svaki od njih može pojedinačno biti još jedan skript. Na ovaj način je kreiran koristan alat .

Skripta se može biti napisati i izvršiti "on-the-fly", bez eksplicitnih koraka prevođenje (engl. compile) i povezivanja (engl. link).

Pojam skripta obično je rezervisan za male programe (do nekoliko hiljada linija koda).

Karakteristike skripting jezika uključuju:

- Koriste ze obično za administraciju sistema i brzu izradu prototipova (engl. rapid prototyping)
- Skripting dozvoljava grupisanje tipično korišćenih komandi u batch datoteka za procesiranje
- Kreiranje novih fleksibilnih i konfigurabilnih alata koji „razumiju“ skripte i alate drugih korisnika
- Neformalan što se tiče tipiziranja varijabli,tj. nema razlike između tipova kao što su integer, floating-point ili string varijabla
- Funkcije mogu vratiti nonskalare , kao to nizovi (engl. array), neskalari se mogu koristiti kao ideksi petlji
- Mnogo operacija visikog nivoa ugrađene u dati jezik, na pr. push/pop.instrukcije
- Više interpretirani jezici nego kompilirani u transformaciji u skup mašinskih instrukcija na host mašini
- Skriptovi izbjegavaju manje greške nedosljednosti grešaka u procesiranju
- Tipični skripting jezici ukjlučuju ljuske skripta (sh, bash, csh, tcsh) i druge skripte (TCL, Perl, Python
- Ponovljeni zadatke može obaviti puno brže

Glavni nedostatak skriptnih jezika je da se izvršni kod skripta može i slučajno preuzeti s udaljenog servera na mašinu web pretraživača, instalirati i pokrenuti pomoću interpretera lokalnog pretraživača . To je lako učiniti tako da posjetite sumnjive web stranice ili preuzimanjem programa bez valjane autentičnosti. Korisnik vjerojatno nije svjestan svega što može dogoditi.

Ime jezika Perl

Cilj

- Upoznavanje sa istorijom imena jezika Perl

Ime jezika Perl

Ime jezika Perl

Perl ponekad se naziva i "" iako je također bio nazvan "Pathologically Eclectic Rubbish Lister". Practical Extraction and Report Language

U nekom pisanom obliku može se vidjeti i “perl” s malim slovom P. U cjelini, "Perl" s velikim slovom P se odnosi na jezik, "Perl" s malim slovom p odnosi se na stvarni interpreter koji prevodi i izvršava programe.

Perl filozofija

Moto Perl jezika se može izraziti kao “Postoji više načina da se nešto učini” (engl. "There’s more than one way to do it"’ abbreviated TMTOWTDI).

To znači da za svaki problem postoji više načina da mu se pristupi koristeći Perl jezik. Neki će biti brži, neki elegantiji, neki čitljiviji od drugih, ali to ne čini njihova rješenja pogrešnim.

Perl programski jezik opšte namjene koji je razvio razvijen Larry Wall 1987 godine. Perl je postao važan jezik za WWW razvoj, obradu teksta, internet usluge, mail filtriranje, grafičko programiranje, i svaki drugi zadatak koji zahtijeva portabilna i lako rješiva rješenja.

Perl je interpretirani jezik. To znači da čim korisnik napiše program, može ga odmah pokrenuti. Ne postoji obvezna faza prevođenja ili kompilacije. Perl programi se mogu izvoditi na Unix, Windows NT, MacOS, DOS, OS / 2, VMS i Amiga OS.

Perl je kolaborativni jezik. CPAN softverska arhiva sadrži besplatne Perl alate napisne zajednički čime se štedi vrijeme.

Perl je besplatan, izvorni kod i prevodilac su dostuoni bez naknade.

Perl je brz. Perl interpreter je pisan u C jeziju i nakon mnogih optimizacija je postao vrlo brz.

Perl je kompletan i pruža podršku za regularne izraze (engl.regular expressions), internu podršku heš tabele (engl. hash table), ugrađeni program za otklanjanje grešaka (engl. debugger), alat za generaciju izvještaja, mrežne funkcije, alate za CGI skripte, interfejse za baze podataka itd.

Perl is bezbjedan. Perl je napravljen, pored ostalog da onemogući zlonamjerne korisnike da izvršavaju komande na računaru korisnika (engl. "taint checking")..

Perl je jednostavan za učenje.

Perl je objektno orentisan. “Inheritance”, “polymorphism”, i “encapsulation” su dio OO karakteristika Perl jezika.

Perl je fleksibilan sa motom “Postoji više načina da se nešto učini”.

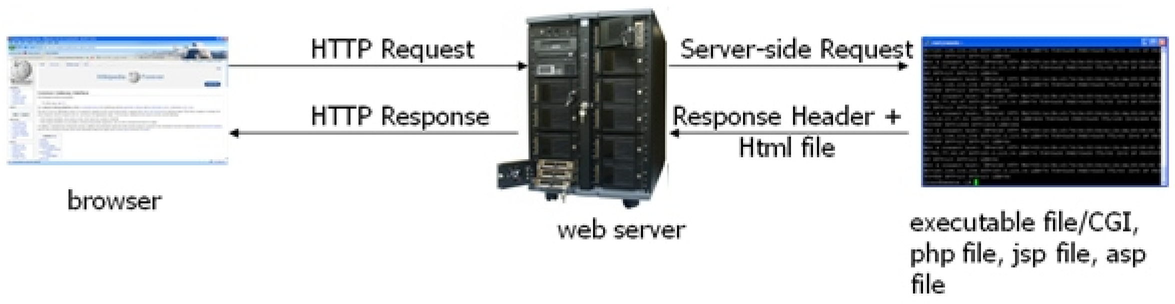
Definicije

Practical Extraction and Report Language-Perl

Perl je programski jezik koji je razvio Larry Wall, posebno dizajniran za obradu teksta. Zbog sposobnosti kvalitetne obrade teksta, Perl je postao jedan od najpopularnijih jezika za pisanje CGI skripti. Perl je Interpretirani jezik, što ga čini lako za izgradnju i testiranje jednostavne programe.

Common Gateway Interface-CGI

CGI je standardni način da web server prenese zahtjev web korisnika do aplikacionog programa i primi podatke natrag i proslijedi ih korisniku, Slika 1. Kada korisnik zatraži web-stranicu (na primjer, klikom na izabranu riječ ili unosom adrese web stranice), server šalje natrag traženu stranicu. Međutim, kada korisnik ispuni obrazac na web-stranici i pošalje ga servru, potrebna je obrada zahtjeva od strane aplikacijskog programa.Web server obično [alje obrazac sa informacijana aplikacionom programu koji obrađuje podatke i [alje natrag poruku s potvrdom. Ovaj metod ili konvencija za preno[enje podataka natrag i naprijed između servera i aplikacija se zove CGI i dio je Interneta Hypertext Transfer Protocol-HTTP protokola.


HTTP Request
HTTP Response
Server-side Request
Response Header + Html file
browser
web server
executable file/CGI, php file, jsp file, asp file

Slika 1 CGI protokol

CGI defineše kako se prenose informacije od web servera do izvršnog programa i kako se informacije prenose natrag do servera.

Interpretirani jezik

Interpretirani jezik je programski jezik u kojem se programi “nezavisno izvršavaju” ("interpretiraju") pomoću programa koji se zove interpreter.

Ovaj programski jezik se razlikuje od prevođenog programa (engl. compiled program) kojeg program zvani prevodilac (engl. compiler) prevodi u mašinski jezik (engl. machine code), zatim CPU “direktno” izvršava kod. Teoretski, svaki jezik može biti prevođen ili interpretiran.

Comprehensive Perl Archive Network-CPAN

Comprehensive Perl Archive Network-CPAN je skup Internet arhiva koja sadrži materijale koji se odnose Perl jezik (Slika 2). CPAN uključuje:

Listu Perl modula i njihovih izvornih kodava

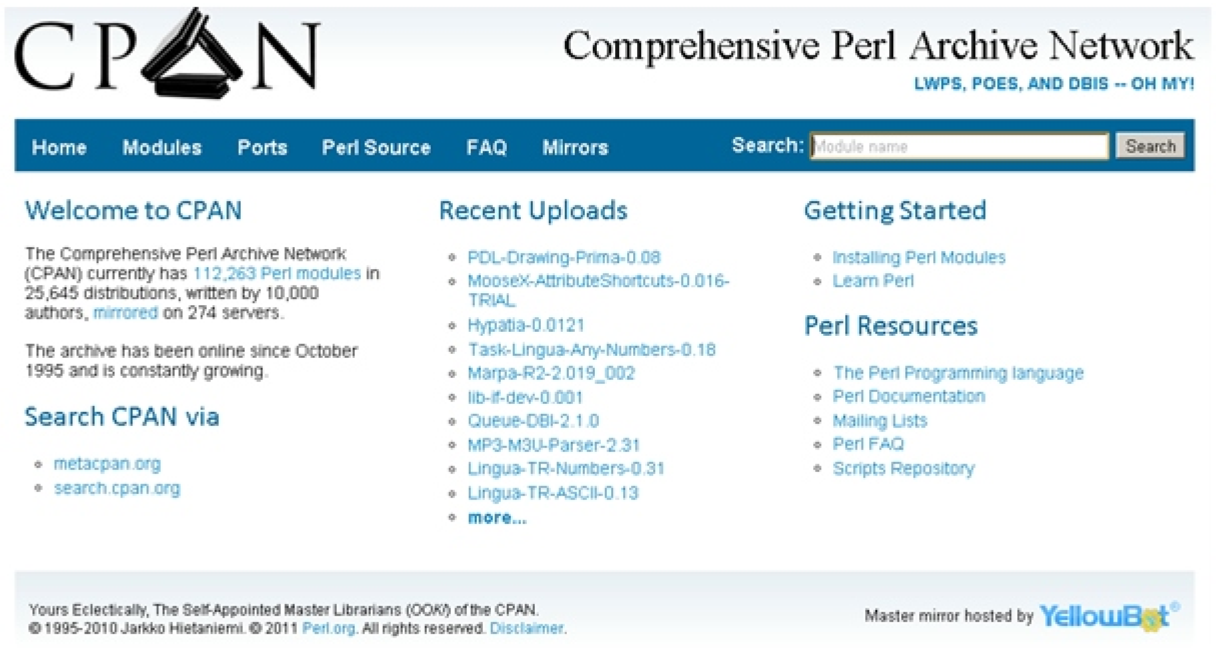
Dokumentaciju za Perl i Perl module

Distribucije Perl jezika

Informacije o Perl modulima i autorima modula

Savjete za programiranje u Perlu

Uputstva i smjernice za podnošenje materijala CPAN-u.



Welcome to CPAN

The Comprehensive Perl Archive Network (CPAN) currently has 112,263 Perl modules in 25,645 distributions, written by 10,000 authors, mirrored on 274 servers.

The archive has been online since October 1995 and is constantly growing.

Search CPAN via

- metacpan.org
- search.cpan.org

Recent Uploads

- PDL-Drawing-Prima-0.08
- MooseX-AttributeShortcuts-0.016-TRIAL
- Hypatia-0.0121
- Task-Lingua-Any-Numbers-0.18
- Marpa-R2-2.019_002
- lib-if-dev-0.001
- Queue-DBI-2.1.0
- MP3-M3U-Parser-2.31
- Lingua-TR-Numbers-0.31
- Lingua-TR-ASCII-0.13
- more...

Getting Started

- Installing Perl Modules
- Learn Perl

Perl Resources

- The Perl Programming language
- Perl Documentation
- Mailing Lists
- Perl FAQ
- Scripts Repository

Yours Eclectically, The Self-Appointed Master Librarians (OOK!) of the CPAN.
© 1995-2010 Jarkko Hietaniemi. © 2011 Perl.org. All rights reserved. Disclaimer.

Master mirror hosted by YellowBot

Slika 2 Comprehensive Perl Archive Network-CPAN

Osnovi programskog jezika Perl

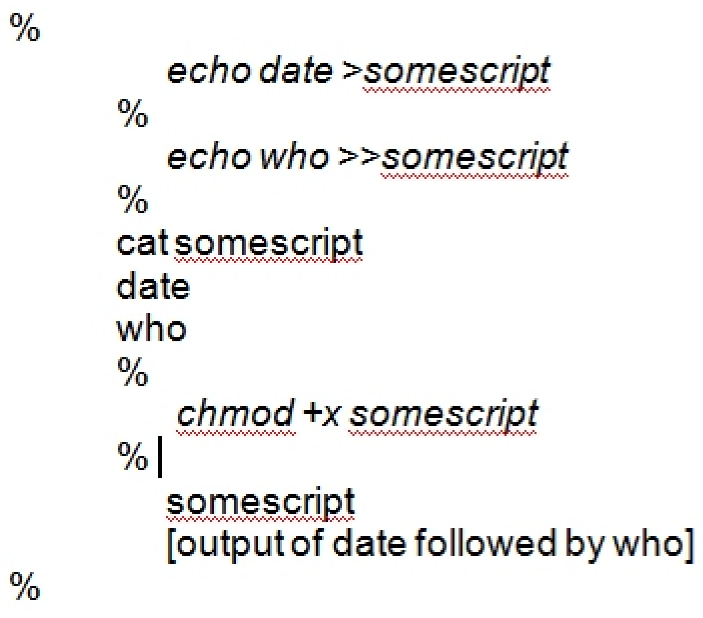
Cilj

- Upoznavanje sa osnovama programskog jezika Perl

Osnovi programskog jezika Perl

Osnovi programskog jezika Perl

Skripta ljuske (engl. shell) nije ništa više od niza naredbe ljuske u u tekstualnos datoteci. Datoteka onda postaje izvršna (engl. executable) komandom ***chmod +x datoteka**. Onda* se ukucava ime datoteke na komandnom nivou ljuske. Na primjer, skripta koja izvršava komandu date, zatim komandu who se može kreirati i izvršiti na slijedeći način, Slika 1.



```
%
        echo date >somescript
    %
        echo who >>somescript
    %
    cat somescript
    date
    who
    %
         chmod +x somescript
    %
        somescript
        [output of date followed by who]
%
```

Slika 1 Primer skripte

Slika 1 Primer skripte

Isto tako, Perl program je skup Perl izjava i definicija sačuvan u datoteci. Zatim se aktivira izvršni bit (engl. the execute bit), ukuca ime datoteke na komandnoj liniji ljuske. Međutim, datoteka mora takođe da naznači da je to Perl program i ne program ljuske, tako da će trebati još jedan dodatni korak.

Perl je uglavnom jezik sa slobodnim formatom (eng. free-format) poput C jezika - razmak između tokena (elemenata programa, poput štampanja ili +) je opcija, osim ako se dva tokena ne slože zajedno da se pogrešno mogu protumačiti kao drugi token, tako da je razmak (engl. whitespace) obavezan (razmak čine blank, “newline”, “returns” ili “form feeds”).

Iako gotovo svaki Perl program može biti napisan na jednoj liniji, tipični Perl program se razvija poput C programu, više s ugniježđenim instrukcijama nego sa potprogramskim segementima.

Baš poput skripte ljuske, Perl program se sastoji od Perl instrukcija u datoteci koje se mogu uzeti kao jedna velika rutina koja se izvršava. Nema koncepta "glavne" rutine kao u C jeziku..

Perl komentari su slični komentarima skripte ljuske. Nema komentara sa više linija sličnim komentarima C jezika.

Za razliku od većine ljuski (ali kao awk i sed), Perl interpreter u potpunosti analizira sintaksu (engl. parse) i prevodi program u interni format prije izvršenja programa. To znači da se nikada ne može desitii sintaktička greška programa kada se program pokrenete, razmaci i komentari jednostavno nestaju i ne usporavaju program.

Ova faza prevođenja osigurava brzo izvršavanje Perl instrukcija nakon što se program pokrene i daje dodatnu motivaciju za izbacivanje C jezika na temelju činjenice da se C program kompilira.

Ova kompilacija traži vrijeme, nije efikasno imati veliki Perl program koji izvršava jedan mali zadatak (od mnogih potencijalnih zadataka), a zatim izlazi, jer je vrijeme izvršavanja tog program suviše dugo.

Dakle, Perl je i prevodilac i interpreter. To je prevodilac, jer se program čita i sintaktički analizira prije izvršavanja prve instrukcije programa. To je i interpreter, jer ne postoji objektni kod koji bi zauzimao prostor na disku. Na neki način, uzete se najbolje karakteristike oba metoda.

Perl je posebno popularan kod sistem i web programera,a i kod šite publike. Napravljen za obradu teksta, izrastao u sofisticirani programski jezik opšte namjene s moćnim sistemom za razvoj softvera To uljučuje program za otklanjanje grešaka (engl. debugger), prevodioce, interpretere, biblioteke, editore koji su orijetisani sinaksom..

Perl je programski jezik opšte namjene idealan za procesiranje reči i teksta. Perl je cijenjen što je ostao jednostavan kada treba činiti jednostavne stvari, ali ima sposobnosti da ispuni svaki zahtjev koji system administrator može zahtijevati.

Još jedna prednost Perl jezika je mogućnost ostvarivanja datog zadatka programiranja na više načina. Ova snaga nadahnula je Perl programerima da koriste moto "TIMTOWTDI" (izgovara se "timtoady"), odnosno akronim "There Is More Than One Way To Do It." ili "Postoji više od jednog način da nešto uradi " Ova raznolikost dovodi do različitih ličnih stilova i preferenci.

Dodatak

Perl is a family of high-level, general-purpose, interpreted, dynamic programming languages. The languages in this family include Perl 5 and Perl 6. Though Perl is not officially an acronym, there are various backronyms in use, such as: Practical Extraction and Reporting Language. Perl was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier.

Since then, it has undergone many changes and revisions. The latest major stable revision of Perl 5 is 5.18, released in May 2013. Perl 6, which began as a redesign of Perl 5 in 2000, eventually evolved into a separate language. Both languages continue to be developed independently by different development teams and liberally borrow ideas from one another.

The Perl languages borrow features from other programming languages including C, shell scripting (sh), AWK, and sed. They provide powerful text processing facilities without the arbitrary data-length limits of many contemporary Unix command line tools, facilitating easy manipulation of text files. Perl 5 gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its parsing abilities.

In addition to CGI, Perl 5 is used for graphics programming, system administration, network programming, finance, bioinformatics, and other applications. It is nicknamed "the Swiss Army chainsaw of scripting languages" because of its flexibility and power, and possibly also because of its "ugliness". In 1998, it was also referred to as the "duct tape“ that holds the Internet together", in reference to both its ubiquitous use as aglue languageand its inelegance

Duct tapeorduck tapeis cloth- orscrim-backedpressure-sensitive tapeoften coated withpolyethylene. There are a variety of constructions using different backings and adhesives.

Linkovi i termini

Linkovi

Primer 1 - http://www.perl.com/pub/2000/10/begperl1.html

Primer 2 - http://www.activestate.com/activeperl

Primer 3 - http://www.perl.org/

Termini

free-format - jezik sa slobodnim formatom

ljuska - shell

Razvoj i zadaci Perl jezika

Cilj

- Upoznavanje sa razvojem i zadacima Perl jezika

Razvoj i zadaci Perl jezika

Example section one bodyRazvoj i zadaci Perl jezika

Perl je besplatan, jer je rezultat rada nekoliko nekoliko pojedinaca koji su svoje slobodno vreme iskoristili za razvile velike komade svoje slobodno vrijeme za razvoj, održavanje i evangelizaciju od Perl.

Perl jezik je stvorio Larry Wall u nastojanju da generiše izvještaje za sistem za prijavu grešaka (engl. bug-reporting system). Larry je napravio novi skriptni jezik za tu svrhu, a zatim objavio na Internet, misleći da bi taj jezik mogao biti nekom koristan. U duhu beslatnog softvera (engl. freeware), drugi su predložili neka poboljšanja, pa čak i načine kako ih provesti i tako se Perl pretvorio od malog skriptnog jezika u robusni programski jezik.

Danas, Larry se malo bavi razvojem Perl jezika, ali je on vođa tima jezgre za razvoj Perl jezika koji je poznat pod nazivom Perl Porters. Ovaj tim odlučuje koje se nove karakteristike mogu dodati Perl jeziku i koje greške treba ispraviti. Da bi se Per držao da bude slobodan-za-sve, obično postoji jedna osoba koja odgovorna za isporuku nove verzije Perl jezika, s nekoliko "razvojnih izdanja" u međuvremenu.

Perl je posebno dobro prilagođen za prikupljanje i manipulaciju teksta. To je razlog zašto je omiljen među administratorima sistema kao i HTML programerima.

Perl podrška

Cilj

- Upoznavanje sa metodama Perl podrške

Perl podrška

Perl podrška

Isporuka Web stranica preko CGI protokola

Server zna da se određene određene URL adrese koriste za pokretanje Perl skripti i kad nađe takve URL adrese, pokreće te Perl skripte automatski. Perl program generiše izlaz u obliku web stranice koja se šalje pretraživaču. Korisnik vidi stranicu stvorena samo za taj zahtjev.

Kontakt sa web sajtovima i izvještavanje o traženim informacijama

Perl može otići i posjetiti web stranicu za korisina bez upotrebe pretraživača.Takođe može “dovući cijelu stranicu i naći traženu informaciju na njoj. Ova informacija onda može biti prepakovana u bilo kojem traženom obliku: web-stranica, štampani izvještaj ili e-pošta.

Takođe, može podnijeti obrazac korisniku uz interakciju s Web serverom na drugom kraju. Primjer za to je e-commerce transakcija. Perl može poslati određene informacije serveru za plaćanje i onda moeža čekati da se vidi da lei je plačanje prošlo uspješno.

Izgradnja Web stranice

Perl se može koristiti za izgradnju web stranice prema vlastito definisanim pravilima.

Editovanje Web stranica

Perl skripte mogu vršitii globalne promjene na web stranicama.

Upravljanje bazama podataka

Perl jezik ima sposobnost za interakciju s online "pozadinskim” bazama podataka (engl. "backend" databases), spremištima velike količine koje se nalaze “izvan scene” na velikim web stranicama. Može da pretažuje zapise baza podataka koji sadrže određene podatke i može takođe dodavati nove zapise ili ažurirati postojeće zapise. Uz sposobnost za interakciju s korisnikom putem web pretraživača, Perl može djelovati kao interfejs između korisnika i baze podataka pokrenute na serveru

Podrška različitim platformama

Iako je Perl razvijen za UNIX OS i usko je isprepleten s UNIX kulturom, ona također ima jake veze sa Windows i Macintosh platformama. Perl daje Windows 95, Windows NT, Macintosh, pa čak i VMS korisnicima priliku da koriste snagu skriptnog jezima što Unix korisnici uzimaju zdravo za gotovo.

Za većina UNIX mašina, Perl je već instaliran, jer to je jedna od prvih stvari koje će administrator Unix sistema kreirati na novoj mašini (a zapravo se distribuira sa operativnim sistemom na nekim verzijama Unix, kao što je Linux i FreeBSD). Za Windows NT, Windows 95 i Macintosh, postoje binarne distribucije jezika Perl koje se mogu preuzeti besplatno.

Regardless of the program you choose to use, a PERL file must be saved with a .pl (.PL) file extension in order to be recognized as a functioning PERL script. File names can contain numbers, symbols, and letters but must not contain a space. Use an underscore (_) in places of spaces.

Linkovi i termini

Linkovi

Primer 1 - http://www.perl.com/CPAN-local/doc/FAQs/cgi/perl-cgi-faq.html

Primer 2 - http://www.perl.com

Termini

CPAN - Comprehensive Perl Archive Network

backend databases - pozadinske baze podataka

FAQ - Frequently Asked Questions FAQ

Perl FAQ

Cilj

- Upoznavanje sa Perl FAQ pitanjima

Perl FAQ

Linkovi i termini

Linkovi

Primer 1 - http://www.perl.com/CPAN-local/doc/FAQs/cgi/perl-cgi-faq.html

Primer 2 - http://www.perl.com

Termini

FAQ - Frequently Asked Questions FAQ

CPAN - Comprehensive Perl Archive Network

Perl editor

Cilj

- Upoznavanje sa Perl editorima

Perl editor

Perl editor

Perl skripta može biti kreiran unutar svakog jednostavnog tekst editor programa. Postoji nekoliko programi dostupnih za svaku vrstu platforme.

Bez obzira na izbor editora, Perl datoteka se mora pohraniti sa ekstenzijom .pl (.PL) kako bi se prepoznala kao Perl skripta. Imena datoteka može sadržavati brojeve, simbole i slova, ali ne smije sadržavati razmak. Koristiti podvučenu linu (_) u mjestima tih razmak.

Perl Editors available on different platforms

To choose an editor from Perl Editors list available on different platforms, you must consider that Perl scripts are just plain text files so you can use any plain text editor in order to create them. Your operating system, doesn’t matter if it is Windows, UNIX or another one, offers you the possibility to select between them your favorite one.

Keep in mind, though, that because you must indent or unindent some blocks of code in order to make your script more readable, the most likely way is to use a programmers’ text editor, available on any operating system and which offer you the possibility to highlight the text, expand or collapse subroutines, menus and so on.

It is not recommendable at all to use a word processor for your code, but if you do this, be sure to save your script files as "text only", otherwise you’ll be unpleasantly surprised to see some awkward characters inserted into your code that you don’t want to be there.

And don’t forget that Perl is deeply portable, so you have the possibility to transfer the program code from one machine to another, but if you do this, do not forget to use "text" or "ASCII" mode instead of "binary" mode.

Padre, the Perl IDE (http://padre.perlide.org/)

Padre is a Perl IDE, an integrated development environment, or in other words a text editor that is simple to use for new Perl programmers but also supports large multi-lingual and multi-technology projects.

Dodatak

Padre, the Perl IDE

http://padre.perlide.org/

Padre is a Perl IDE, an integrated development environment, or in other words a text editor that is simple to use for new Perl programmers but also supports large multi-lingual and multi-technology projects.

Our primary focus is to create a peerless environment for learning Perl and creating Perl scripts, modules and distributions, with an extensible plug-in system to support the addition of related functionality and languages and to support advanced developers taking the editor anywhere they want it to go.

Features

Customizable syntax highlighting for many languages and visual editor effects

Syntax checking for Perl 5 and Perl 6

Refactoring tools for Perl 5 and Perl 6

Context sensitive help and code completion

Beginner-friendly

Extra features for advanced programmers

Multi-platform: Runs on Windows, Linux, Mac OS X

Free and Open Source under the "Perl license"

Written in Perl 5

Learning Perl

Once you have Perl and Padre installed, you could check out the Perl Tutorial written by Gabor Szabo, the original creator of Padre.

http://perlmaven.com/perl-tutorial

Make the Switch to Git

https://www.atlassian.com/git/tutorial/git-basics

http://padre.perlide.org/trac/wiki/PadrePluginGit

Perl moduli

Cilj

- Upoznavanje sa Perl modulima

Perl moduli

Perl moduli

Perl modul je pojedinačna komponenta softvera za Perl programski jezik. Tehnički, to je određeni skup pravila za korištenje mehanizma Perl pakovanja (engl. package) koji je postao generalno prihvaćen .

Perl modul definiše svoj izvorni koji treba da se nalazi u pakovanju (kao što je Java pakovanje).

Java pakovanje (engl. Java package) je mehanizam organizovanja Java klasa u prostore imena (engl. namespaces). Java pakovanja se čuvaju u kopresovanim JAR datotekama.

Perl mehanizam za definisanje prostora imena, na primer, CGI ili Net :: FTP ili XML :: Parser.

Perl modul je ekvivalentan klasama u OO programiranju.

Skup modula ima i prateću dokumentaciju, "build" skriptove koji čine distribuciju. Perl zajednica ima prilično veliku biblioteku distribucija koja je dostupna preko CPAN (engl. Comprehensive Perl Archive Network**)** sajta (engl. http://en.wikipedia.org/wiki/CPAN).

Dodatak

Perl is a language allowing many different styles of programming. You're as likely to find a module written in aproceduralstyle (for example,Test::Simple) asobject-oriented(e.g. XML::Parser), both are considered equally valid according to what the module needs to do. Modules can even be used to alter the syntax of the language. The effect of Perl modules are usually limited to the currentscopein which it was loaded.

It is common for Perl modules to have embedded documentation in Perl'sPlain Old Documentationformat. POD imposes little structure on the author. It is flexible enough to be used to write articles, web pages and even entire books such asProgramming Perl. Contrast with javadoc jwhich is specialized to documentingJavaclasses. By convention, module documentation typically follows the structure of aUnix man page. (http://en.wikipedia.org/wiki/Manual_page_(Unix)).

The language of Perl is defined by the single implementation (referred to as "perl") and is added to (and in rare occasions taken away from) each new release. For this reason it is important for a module author to be aware what features they're making use of and what the minimum required version of Perl is. The code on this page requires perl 5.6.0 which is considered rather old by now.

Perl also comes with a collection of modules. These are Perl libraries which carry out certain common tasks, and can be included as common libraries in any Perl script. Less commonly usedmodules aren't included with the distribution, but can be downloaded from CPAN and installed separately (see the site http://www.cpan.org/modules/index.html).

Sajt http://www.cpan.org/modules/INSTALL.html daje uputstva za instalaciju Perl modula

Perl compiled or interpreted

Cilj

- Upoznavanje sa pitanjem da li se Perl program kompilira ili interpretira

Da li se Perl program kompilira ili interpretira?

Da li se Perl program kompilira ili interpretira?

Odgovor može biti da je interpretiran u obliku u kojem je (izvorni kod) bez prevođenja ili kompilacije.

Definicije

Prevodilac (engl. compiler)

Prevodilac (engl. compiler) je računarski program (ili skup programa) koji pretvaraju izvorni kod napisan u programskom jeziku (engl. source language) u drugi računarski jezik ili ciljani jezik the (engl. target language) koji često ima binarni oblik poznat i poznat je kao kao objektni kod (engl. object code),

Interpreter

Interpreter eksplicitno izvršava pohrarnjeni “precompiled” kod kojeg je prevodilac generisao (prevodilac je dio interpreter sistema).

Tradionalni prevodioci prevode programe u mašinski jezik. Kada se pokrene Perl program, on se prvo prevodi u bytecode, koji se zatim prevodi (dok se program izvršava) u mašinske instsukcije. Dakle Perl nije kao ljuske ili Tcl (http://en.wikipedia.org/wiki/Tcl) koji se interpretiraju bez prevođenja u među kod. Perl nije ni kao većina verzija C ili C++, koji se direktno prevode ili kompiliraju u oblik koji zavisi od formata mašine.

Perl je negdje u sredini između dva pomenuta rešenja, kao i Python, awk i emacs .elc datoteke

Po definiciji, Perl5 prevodilac uzima Perl5 izvorni kod i generiše graf **OPCODE** objekata. Međutim. Perl5 prevodilac ne generiše mašinski kod. Svaki opcode ima “*next* / *other* and *first* / *sibling” pointer* tako da se može generisati struktura u obliku stabla opkodova kao OP stablo koje počinje od od korjena (engl. root).

Perl **interpreter** je “šetač” po stablu (engl. tree walker) koji koji putuje po stablu opkodova stabla u poretku izvršavanja od početnog čvora slijedeći next ili druge pointere.

Ovo znači da se OPCODE interpretira.

Faza prevođenja je sakrivena od krajnjeg korisnika

Perl virtualna mašina

Perl virtualna mašina je virtulana mašina bazirana na stek mehanizmu implementirana kao interpreter opkodova koja izvršava predhodno kompilirani program. Opcode interpreter je dio Perl intrpretera koji takođe sadrži prevodilac (lexer, parser and optimizer) u jednoj izvršnoj datoteci, obično /usr/bin/perl na raznim UNIX-baziranim sistemima ili perl.exe na Microsoft Windows sistemima.

Perl resursi

Cilj

- Upoznavanje sa Perl resursima

Perl resursi

Perl resursi

comp.lang.perl.* Newsgroups

Središnje mjesto okupljanja Perl poklonika je Usenet. Perl programeri treba razmotre pretpaltu na na sljedeće grupe:

comp.lang.perl.announce

Grupa s moderatorom sa najavama o novim servisima ili proizvodima vezanih za Perl.

comp.lang.perl.misc

Newsgroup opšte namjene posvećena non-CGI pitanjima o Perl programiranju.

comp.lang.perl.moderated

Grupa s moderatorom kao forum za diskusije o Perl jeziku.

comp.lang.perl.modules

Newsgroup posvećena korišćenju i razvoju Perl modula.

comp.lang.perl.tk

Newsgroup fokusirana na Perl/Tk, Perl grafičku ekstenziju.

comp.infosystems.www.authoring.cgi

Newsgroup for CGI pitanja opšte namjene.

Perl compiled or interpreted

Cilj

- Upoznavanje sa pitanjem da li se Perl program kompilira ili interpretira

Da li se Perl program kompilira ili interpretira?

Da li se Perl program kompilira ili interpretira?

Odgovor može biti da je interpretiran u obliku u kojem je (izvorni kod) bez prevođenja ili kompilacije.

Definicije

Prevodilac (engl. compiler)

Prevodilac (engl. compiler) je računarski program (ili skup programa) koji pretvaraju izvorni kod napisan u programskom jeziku (engl. source language) u drugi računarski jezik ili ciljani jezik the (engl. target language) koji često ima binarni oblik poznat i poznat je kao kao objektni kod (engl. object code),

Interpreter

Interpreter eksplicitno izvršava pohrarnjeni "precompiled" kod kojeg je prevodilac generisao (prevodilac je dio interpreter sistema).

Tradionalni prevodioci prevode programe u mašinski jezik. Kada se pokrene Perl program, on se prvo prevodi u bytecode, koji se zatim prevodi (dok se program izvršava) u mašinske instsukcije. Dakle Perl nije kao ljuske ili Tcl (http://en.wikipedia.org/wiki/Tcl) koji se interpretiraju bez prevođenja u među kod. Perl nije ni kao većina verzija C ili C++, koji se direktno prevode ili kompiliraju u oblik koji zavisi od formata mašine.

Perl je negdje u sredini između dva pomenuta rešenja, kao i Python, awk i emacs .elc datoteke

Po definiciji, Perl5 prevodilac uzima Perl5 izvorni kod i generiše graf **OPCODE** objekata. Međutim. Perl5 prevodilac ne generiše mašinski kod. Svaki opcode ima "*next* / *other* and *first* / *sibling" pointer* tako da se može generisati struktura u obliku stabla opkodova kao OP stablo koje počinje od od korjena (engl. root).

Perl **interpreter** je “šetač” po stablu (engl. tree walker) koji koji putuje po stablu opkodova stabla u poretku izvršavanja od početnog čvora slijedeći next ili druge pointere.

Ovo znači da se OPCODE interpretira.

Faza prevođenja je sakrivena od krajnjeg korisnika

Perl virtualna mašina

Perl virtualna mašina je virtulana mašina bazirana na stek mehanizmu implementirana kao interpreter opkodova koja izvršava predhodno kompilirani program. Opcode interpreter je dio Perl intrpretera koji takođe sadrži prevodilac (lexer, parser and optimizer) u jednoj izvršnoj datoteci, obično /usr/bin/perl na raznim UNIX-baziranim sistemima ili perl.exe na Microsoft Windows sistemima.

Detekcija Perla

Cilj

- Upoznavanje sa načinom provere da li je Perl jezik već instaliran

Detekcija Perla

Detekcija Perla

Da li je Perl instalitan na sistemu?

Prije nego se pokrene Perl skripta, na računa treba instalirati Perl program. No, najprije ipak treba ustanoviti da li je Perl već instaliran na računaru.

Ako se radi o UNIX ili Linux sistemu, postoji velika vjerovatnoća da Perl je već instaliran. U tom slučaju, treba preskočiti ovaj korak.

If you have UNIX or Linux, simple log in and you’ll have access to the command prompt. If you have a graphical interface instead, after the log in you need to open a terminal window in order to have access to the command prompt.

On a Windows system, click Start-->Run and in the Open field type cmd and than confirm by clicking the OK button. It will open a cmd window where you can type the commands you want to be executed.

At the command prompt, please type the following:

perl –v

and you’ll receive either the message "command not found" which means perhaps you must install Perl, or Perl’s version number.

C:\>perl -v

This is perl 5, version 14, subversion 2 (v5.14.2) built for MSWin32-x86- (with 1 registered patch, see perl -V for more detail)

Copyright 1987-2011, Larry Wall

Binary build 1402 [295342] provided by ActiveState http://www.ActiveState Built Oct 7 2011 15:49:44

Perl may be copied only under the terms of either the Artistic License or GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on this system using "man perl" or "perldoc perl". If you have access to the Internet, point your browser at http://www.perl.org/, the Perl Home Page

Instalacija Perla

Cilj

- Upoznavanje sa načinom instalacije jezika Perl

Instalacija Perla

Instalacija Perla

Install 'make' through your package manager. You can then use all of the tools mentioned above.

Install Perl on Linux

First, you need a copy of Perl source bundle and you can download a stable version from the site "http://www.perl.com". At the moment I am writing these words, I found available the archive perl-5.8.8.tar.gz. You also need an ANSI C compiler, but don’t worry about this, Perl’s configuration program will check for one. If it will not find it, you can install a prebuilt version or install the C compiler supplied by your OS vendor, or for a free C compiler try "http://gcc.gnu.org/".Now, get a command prompt and enter the following commands:

```
gunzip perl-5.8.8.tar.gz
```

```
tar xf perl-5.8.8.tar
```

and wait the end of decompression. If you don’t have the gunzip decompression program, you can download gzip bundle from http://www.gnu.org. The next step is to launch the configure program of the Perl bundle. The installation shell script will examine your system and ask your questions to determine how perl5 package should be installed. If you want to bypass all the questions and use the computed defaults or the previous answers if there was already a config.sh file, you can run "Configure –d":

```
cd perl-5.8.8
```

```
sh Configure –d
```

After the Perl configuration step, you must build Perl by typing in the following command:

```
make
```

After this step, you can verify Perl’s installation, running again the following command at the command prompt:

```
perl –v
```

Install Perl on Windows (32 and 64 bit)

Strawberry Perl is an open source binary distribution of Perl for the Windows operating system. It includes a compiler and pre-installed modules that offer the ability to install XS CPAN modules directly from CPAN *(cpanm can be installed by running cpan App::cpanminus).

ActiveState provide a binary distribution of Perl (for many platforms), as well as their own the Perl Package Manager (PPM). Some modules are not available as ppm's or have reported errors on the ppm build system, this does not mean they do not work. You to use the cpan script to build modules from CPAN against ActiveState Perl

The free ActivePerl binary distribution includes core Perl, popular modules, the Perl Package Manager (PPM), and complete documentation. You can download it from "http://www.activestate.com" and install it on your computer – the distribution is self-installing and the installation process is just the same as the one you used for other similar windows applications.

ActivePerl

Glavni izvor ActivePerl distribucije je sajt http://www.activestate.com/ koji uključuje:

Perl for Win32

Binary for the core Perl distribution

Perl for ISAPI

IIS plug-in for use with ISAPI-compliant web server

PerlScript

ActiveX scripting engine

Perl Package Manager

Manager for Perl modules and extension

Perl on Mac OSX

OSX comes with Perl pre-installed, in order to build and install your own modules you will need to install the 'developer' package which can be found on your OSX install DVD (you only need the 'unix tools'). Once you have done this you can use all of the tools mentioned above

Perl interpreter

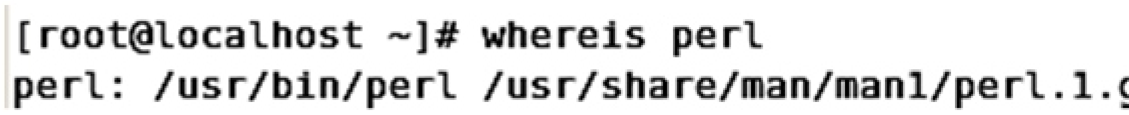
Cilj

- Upoznavanje sa Perl interpreterom

Perl interpreter

Perl interpreter

Perl izvršna datoteka, obično instalirana u **/usr/bin** ili **/usr/local/bin** ili **/usr/share/man/man1** na računaru, naziva se Perl interpreter.



```
[root@localhost ~]# whereis perl
perl: /usr/bin/perl /usr/share/man/man1/perl.1.g
```

.

Slika 1 Perl interpreter

Svaki Perl program mora proći kroz Perl interpreter kako bi se mogao izvršiti. Prva linija u mnogim Perl programa je nešto poput:

#!/usr/bin/perl

Za UNIX sistem, ovaj #! (hash-bang ili shebang) linija kaže ljusci da traži /usr/bin/ perl program i onda mu predaje ostatak programa za izvršenje. Ponekad se može vidjeti put (engl. pathname) za Perl kao /usr/local/bin/perl. Može se vidjeti perl5 umjesto Perl na mjestima koja još uvijek ovise o starijim verzijama Perl.

Dakle, šta Perl interpreter radi? Ona interno prevodi program, a zatim ga odmah izvršava. Perl je obično poznat kao interpretirani jezik, ali to nije strogo istinito.

Budući da interpreter zapravo prevodi program u bytecode kod prije njegovog izvršenja, često se naziva interpreter /prevodilac. Iako se prevedeni program ne pohranjuje kao datoteka, Perl verzija 5.,005 uključuje radni verziju samostalnog Perl prevodioca.

Kada se napiše Perl program, treba dati točnu #! liniju na vrhu scipta, čime postaje izvršni program pomoću chmod +x naredbe i onda ga pokrećete. Za 95% Perl programera u ovom svijetu, to je sve što je potrebno.

Linkovi i termini

Linkovi

Primer 1 - http://www.perl.org/

Primer 2 - http://docs.activestate.com/activeperl/5.8/lib/pods/perlrun.html

Termini

compiler - prevodilac koda

interpreter - interpreter koda

path - putanja

Perl jezik

Cilj

- Upoznavanje sa programskim jezikom Perl

Perl jezik

Perl jezik (engl. Perl language)

Program Structure

Perl is a particularly forgiving language, as far as program layout goes. There are no rules about indentation, newlines, etc. Most lines end with semicolons, but not everything has to. Most things don't have to be declared, except for a couple of things that do. Here are the bare essentials:

A Perl script or program consists of one or more statements. These statements are simply written in the script in a straightforward fashion. There is no need to have a main() function or anything of that kind.

Perl statements end in a semi-colon:

print "Hello, world"**;**

Comments start with a hash symbol and run to the end of the line:

This is a comment

Whitespace

Prazan proctor je potreban između dva elementa, jer bi zabunom oni mogli biti smatrani jednim elementom

Whitespace is required only between items that would otherwise be confused as a single term. All types of whitespace - spaces, tabs, newlines, etc. - are equivalent in this context. A comment counts as whitespace. Different types of whitespace are distinguishable within quoted strings, formats, and certain line-oriented forms of quoting. For example, in a quoted string, a newline, a space, and a tab are interpreted as unique characters.

Semicolons

***S*vaka izjava mora imati na kraju tačka-zapeta (;) završetak.**

Every simple statement must end with a semicolon. Compound statements contain brace-delimited blocks of other statements and do not require terminating semicolons after the ending brace. A final simple statement in a block also does not require a semicolon.

Declarations

Samo podprogrami i formati izveštaja trebaju biti eksplicitno definisani.

Only subroutines and report formats need to be explicitly declared. All other user-created objects are automatically created with a null or 0 value unless they are defined by some explicit operation such as assignment. The-wcommand-line switch will warn you about using undefined values.

Comments and documentation

Komentari u program se označavaju simbolom (#).

Comments within a program are indicated by a pound sign (#). Everything following a pound sign to the end of the line is interpreted as a comment. Lines starting with=are interpreted as the startof a section of embedded documentation (pod), and all subsequent lines until the next=cutare ignored by the compiler .

Perl - case sensitivity

Perl je osetljiv na velika i mala slova. Ako ime variable kod definicije počinje sa velikim slovom, u tom format se mora i dalje koristiti.

File names, variables, and arrays are all case sensitive. If you capitalize a variable name when you define it, you must capitalize it to call it.

PERL File Extension (.pl ili .PL)

Perl skripta se može kreirati pomoću normalnog tekst editora.

A PERL script can be created inside of any normal simple-text editor program. There are several programs available for every type of platform. There are many programs designed for programmers available for download on the web.

Regardless of the program you choose to use, a PERL file must be saved with a .pl (.PL) file extension in order to be recognized as a functioning PERL script. File names can contain numbers

Programiranje u Perlu

Cilj

- Upoznavanje sa metodama programiranja u Perlu

Programiranje u Perlu

Programiranje u Perlu

Perl kod je prenosiv (engl. code is portable). Većina skiptova su pisana u verzijama 5.8 ili višim. Kada se počinje programiranje u Perlu, prvo treba odrediti Perl binary putanju.

Na većini UNIX/Linux sistema to se postiže whereis komandom:

whereis perl

Kao rezultat se može dobiti: /usr/bin/perl.

Zato prva linija skripta može biti:

#!/usr/bin/perl

# rest of code # will be used for comments.

Ova prva linija izvornog koda pomaže ljusci da nađe gdje je izvšni kod (engl. binary) kada se izvršava script.

Da bi se korektno izvršavao script, izvorni kod mora imati izvršnu zastavicu (engl. executable flag) za korisnika koji izvršava skriptu

Za slučaj datoteke prog1.pl to se postiže dodavanjem izvršne zastave na sledeći način:

chown u+x progr1.pl

So, to start your Perl script you will follow:

a) Create an empty file:

touch program.pl

b) Add executable flag to that file:

chown u+x program.pl

c) Find where your Perl binary is:

whereis perl (something like **/usr/bin/perl**).

d) Edit that file with your text editor, and add perl path with this syntax: #!/usr/bin/perl (not that # on first line of your code will not be seen as comment) edit program.pl and put there:

#!/usr/bin/perl

Output with print

It’s generally a good idea to have your program produce some output; otherwise, someone may think it didn’t do anything. The print operator makes this possible: it takes a scalar argument and puts it out without any embellishment onto standard output. Unless you've done something odd, this will be your terminal display. For example:

print "hello world\n"; # say hello world, followed by a newline

print "The answer is ";

print 6 * 7;

print ".\n";

You can give print a series of values, separated by commas:

print "The answer is ", 6 * 7, ".\n";

Pragmas

use strict;

A pragma is a hint to a compiler, telling it something about the code. In this case, the use strict pragma tells

Perl's internal compiler that it should enforce some good programming rules for the rest of this block or source file. Many programming languages allow you to give hints to the compiler. In Perl, these hints are conveyed to the compiler with the use declaration.

use warnings;

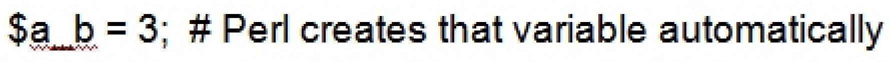
use strict;

use integer;

use bytes;

The use strict Pragma

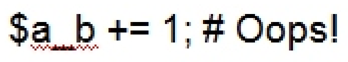
Asume the following line:



```
$a_b = 3;  # Perl creates that variable automatically
```

Slika 1 Primer 1

Now, you keep typing for a while. After that line has scrolled off the top of the screen, you type this line to increment the variable:



```
$a_b += 1; # Oops!
```

Slika 2 Primer 2

Since Perl sees a new variable name (the underscore *is* significant in a variable name), it creates a new variable and increments that one. If a user has turned on warnings, and Perl can tell you that you used one or both of those global variable names only a single time in your program.

But if you're merely smart, you used each name more than once, and Perl won't be able to warn you. To tell Perl that you're ready to be more restrictive, put the use strict pragma at the top of your program (or in any block or file where you want to enforce these rules):

use strict; # Enforce some good programming rules

Starting with Perl 5.12, you implicitly use this pragma when you declare a minimum Perl version:

use 5.012; # loads strict for you

Controlling Warnings

Perl pragma for control optional warnings

use warnings;

say (Perl 5.10.0+)

Perl 5.10.0 and later versions brought in a number of new features into the language. One of the most handy of these is **say**; print with a newline. say works exactly the same as print but it adds a newline as the last character printed. Thus the output of the following code is two identical lines.

use v5.10.0;

say "Hello World!";

print "Hello World!\n";

Example

use v5.10.0;

use constant SECONDS_PER_DAY => 86400;

say SECONDS_PER_DAY;

Create the first Perl program

We’re about to create our first, simple Perl script: a "hello world" program. There are a couple of things one should know in advance:

Perl programs (or scripts --- the words are interchangeable) consist of a series of statements

When you run the program, each statement is executed in turn, from the top of your script to the bottom. (There are two special cases where this doesn’t occur, one of which is subroutine declaration)

Each statement ends in a semi-colon

Statements can flow over several lines

Whitespace (spaces, tabs and newlines) is ignored in most places in a Perl script.

Now, just for practice, open a file called **hello.pl in** your editor. Type in the following one-line Perl program:

print "Hello world!\n";

This one-line program calls the print function with a single parameter, the string literal "Hello world!" followed by a newline character.

Save it and exit.

Linkovi i termini

Linkovi

Primer 1 - http://padre.perlide.org/trac/wiki/Features/Autocomplete

Primer 2 - http://www.cpan.org/

Termini

portable code - prenosiv kod

executable flag - zastavica

Komandna linija

Cilj

- Upoznavanje sa izvršavajem Perl programa sa komandne linije

Izvršavanje Perl programa sa komandne linije

Izvršavanje Perl programa sa komandne linije

Dat je Perl program hello.pl.

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

Perl program se se može pokrenuti sa komande linije:

```
% perl hello.pl
```

Rezultat:

```
Hello world!
```

Ovdje \n ("backslash N") definieše novu liniju. Perl koristi istu istu notaciju za predstavljnanje karaktera (newlines, tabs, bells) kao i jezik C.

Izvršavanje koda

There are a lot of ways of running Perl, it depends on what you intend to do and the platform on which Perl is installed. Before playing with Perl, you must check if you have a proper Perl version installed.

Komanda:

```
perl –v
```

ispituje da li je Perl instaliran i daje veriju Perla ako je instaliran

Najjednostavniji način pokretanja Perla sa komandne linije je na primder:

```
perl -e "print qq [Hello world!\n]"
```

Ova jednostavna komanda štampa na ekranu klasičnu poruku *Hello world!". Čitav programski kod je sadržan između navodnih znaka.

Osim definisanja #! linije, možete odrediti kratku skriptu direktno na komandnoj liniji. Ovdje su neki od mogućih načina da se pokretanja Perla:

Izdati naredbu perl, napisati skripte liniju po liniju putem e-prekidača na komandnoj liniji:

```
perl -e 'print "Hello, world\n"' #Unix
perl -e "print \"Hello, world\n\"" #Win32
```

Izdati naredbu perl, zatim ime svoje Perl skripte kao prvi parametar (nakon svih prekidača):

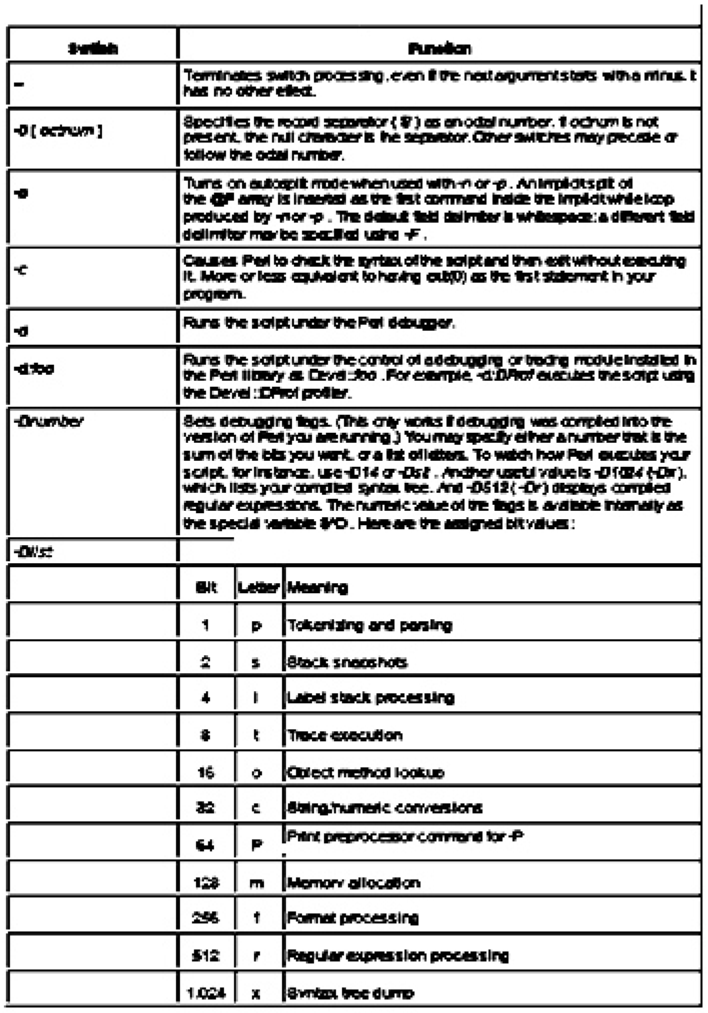
```
perl testpgm
```

Na UNIX sistemima koji podržavaju #! notaciju, odrediti Perl komandu na #! liniji, napraviti kod izvršnim (engl. executable script) i pokrenuti ga sa komadne linije (engl. shell).

Predati skriptu Perlu preko standardnog ulaza. Na primjer, za UNIX OS:

```
echo "print 'Hello, world'" | perl –
```

Drugi način je pokretanje Perla korišćenje standardnog ulaznok toka (engl. input stream) za prijem koda kao na primer: like in the example below:



Switch	Function		
--	Terminates switch processing, even if the next argument starts with a minus. It has no other effect.		
-0 [octnum]	Specifies the record separator ($/) as an octal number. If octnum is not present, the null character is the separator. Other switches may precede or follow the octal number.		
-a	Turns on autosplit mode when used with -n or -p . An implicit split of the @F array is inserted as the first command inside the implicit while loop produced by -n or -p . The default field delimiter is whitespace; a different field delimiter may be specified using -F .		
-c	Causes Perl to check the syntax of the script and then exit without executing it. More or less equivalent to having exit(0) as the first statement in your program.		
-d	Runs the script under the Perl debugger.		
-d:foo	Runs the script under the control of a debugging or tracing module installed in the Perl library as Devel::foo . For example, -d:DProf executes the script using the Devel::DProf profiler.		
-Dnumber	Sets debugging flags. (This only works if debugging was compiled into the version of Perl you are running.) You may specify either a number that is the sum of the bits you want, or a list of letters. To watch how Perl executes your script, for instance, use -D14 or -Dslt . Another useful value is -D1024 (-Dx), which lists your compiled syntax tree. And -D512 (-Dr) displays compiled regular expressions. The numeric value of the flags is available internally as the special variable $^D . Here are the assigned bit values:		
-Dlist			
	Bit	Letter	Meaning
	1	p	Tokenizing and parsing
	2	s	Stack snapshots
	4	l	Label stack processing
	8	t	Trace execution
	16	o	Object method lookup
	32	c	String/numeric conversions
	64	P	Print preprocessor command for -P
	128	m	Memory allocation
	256	f	Format processing
	512	r	Regular expression processing
	1,024	x	Syntax tree dump

Slika 1 Primer

Zadnji karakter programa je ovdje Ctrl/Z koji znači kraj datoteke (engl. End of File). Ovaj primer ima problem u tome da je program sačuvan samo privremeno i biće izgubljen poslije izvršavanja. Ovaj metode se može koristiti samo za proveru koda ili u fazi učenja.

Opcije komandne linije

Perl očekuje bilo koju opciju komandne linije, takođe poznatu kao prekidača ili zastavica, na bude na prvom mjestu na komandnoj liniji.

Sledeći dolazi obično ime skripte, pa bilo koji dodatni argumenti (često imena datoteka) koji se prenose u skript).

Prekidač sa jednim karakterom bez argumenta može biti u kombinaciji (u paketu) s prekidačem koji ga prati, ako ih ima.

Na primjer:

```
#! / usr/bin/perl -spi.bak
```

ili

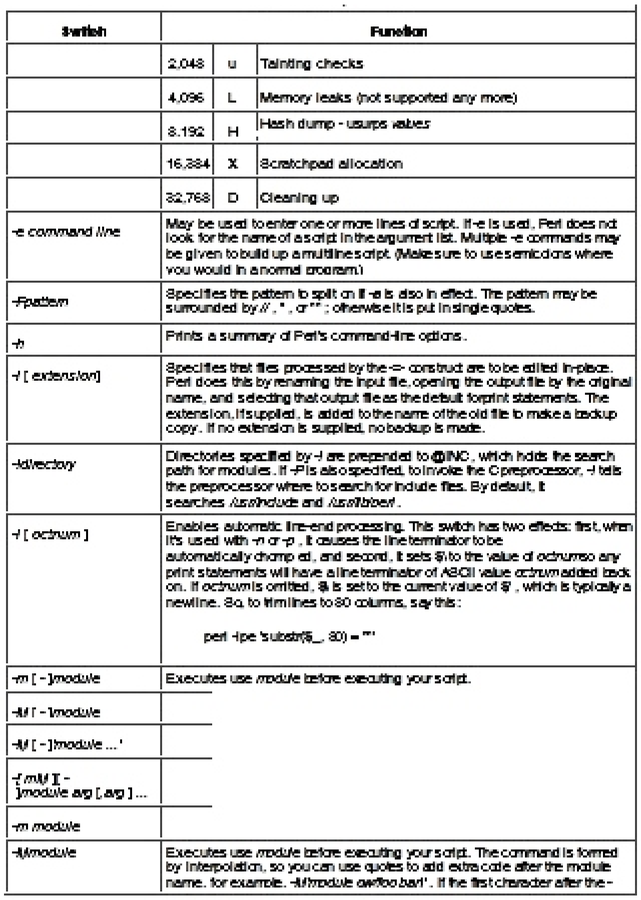
```
#! /usr/bin/perl -s -p -i.bak
```

Perl prepoznaje prekidače navedene na Slici 1, Slici 2, Slici 3 i Slici 4.

Windows- which provides WordPad and Notepad, but as we mentioned before, they lack many features for editing code so you can use nedit,UltraEdit or PFE (Programmer’s Favorite Editor) instead;

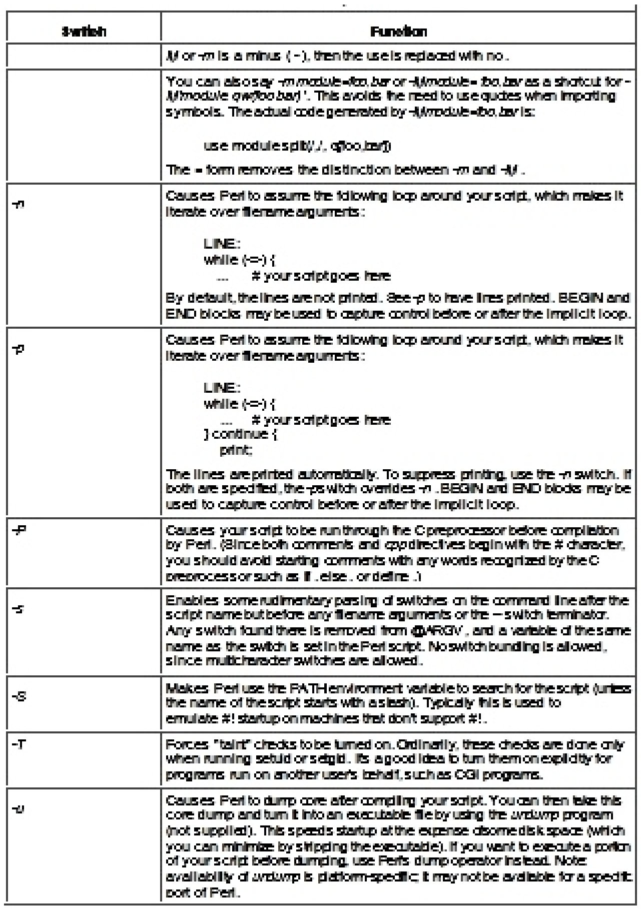
UNIX- the most popular programmers’s editors are emacs and vi (with their several successors like vim or vile) which, in some versions, offer a graphical interface;

Mac OS X - with the BBEdit (this one is a popular commercial text editor considered by many developers as best available) and Alpha;



Switch	Function		
	2,048	u	Tainting checks
	4,096	L	Memory leaks (not supported any more)
	8,192	H	Hash dump - usurps *values*
	16,384	X	Scratchpad allocation
	32,768	D	Cleaning up
-e *command line*	May be used to enter one or more lines of script. If -e is used, Perl does not look for the name of a script in the argument list. Multiple -e commands may be given to build up a multiline script. (Make sure to use semicolons where you would in a normal program.)		
-F*pattern*	Specifies the pattern to split on if -a is also in effect. The pattern may be surrounded by // , " " , or '' ; otherwise it is put in single quotes.		
-h	Prints a summary of Perl's command-line options.		
-i [*extension*]	Specifies that files processed by the <> construct are to be edited in-place. Perl does this by renaming the input file, opening the output file by the original name, and selecting that output file as the default for print statements. The extension, if supplied, is added to the name of the old file to make a backup copy. If no extension is supplied, no backup is made.		
-I*directory*	Directories specified by -I are prepended to @INC , which holds the search path for modules. If -P is also specified, to invoke the C preprocessor, -I tells the preprocessor where to search for include files. By default, it searches */usr/include* and */usr/lib/perl* .		
-l [*octnum*]	Enables automatic line-end processing. This switch has two effects: first, when it's used with -n or -p , it causes the line terminator to be automatically chomp ed, and second, it sets $\ to the value of *octnum* so any print statements will have a line terminator of ASCII value *octnum* added back on. If *octnum* is omitted, $\ is set to the current value of $/ , which is typically a newline. So, to trim lines to 80 columns, say this: `perl -lpe 'substr($_, 80) = ""'`		
-m [-]*module*	Executes use *module* before executing your script.		
-M [-]*module*			
-M [-]'*module* ...'			
-[*mM*][-]*module=arg* [,*arg*] ...			
-m *module*			
-M*module*	Executes use *module* before executing your script. The command is formed by interpolation, so you can use quotes to add extra code after the module name, for example, -M'*module qw(foo bar)*' . If the first character after the -		

Slika 2 Perl prekidači 1

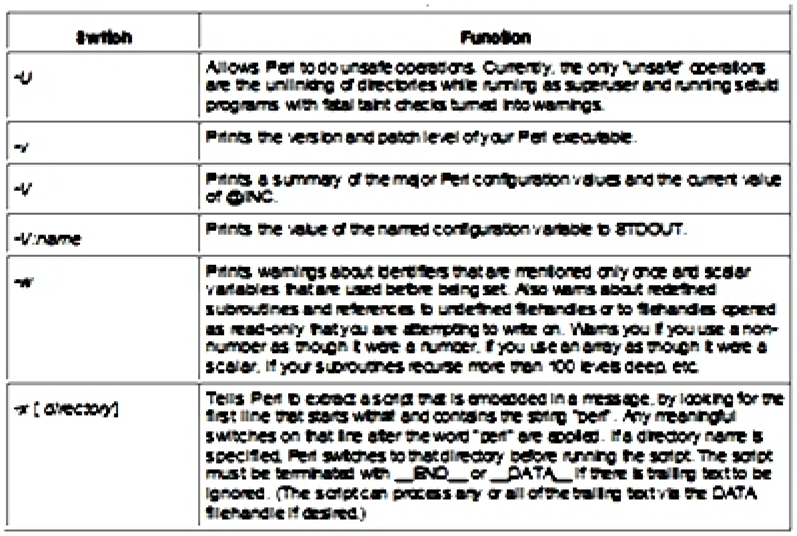


Switch	Function
	-M or *-m* is a minus (-), then the use is replaced with no .
	You can also say *-m module=foo,bar* or *-M module= foo,bar* as a shortcut for *-M module qw(foo bar)* '. This avoids the need to use quotes when importing symbols. The actual code generated by *-M module=foo,bar* is: use module split(/,/, q{foo,bar}) The = form removes the distinction between *-m* and *-M* .
-n	Causes Perl to assume the following loop around your script, which makes it iterate over filename arguments : LINE: while (<>) { ... # your script goes here By default, the lines are not printed . See *-p* to have lines printed . BEGIN and END blocks may be used to capture control before or after the implicit loop.

-p	Causes Perl to assume the following loop around your script, which makes it iterate over filename arguments :
	
	LINE:
	while (<>) {
	... # your script goes here
	} continue {
	print;
	
	The lines are printed automatically. To suppress printing, use the *-n* switch. If both are specified, the *-p* switch overrides *-n* . BEGIN and END blocks may be used to capture control before or after the implicit loop.

-P	Causes your script to be run through the C preprocessor before compilation by Perl. (Since both comments and *cpp* directives begin with the # character, you should avoid starting comments with any words recognized by the C preprocessor such as if , else , or define .)
-s	Enables some rudimentary parsing of switches on the command line after the script name but before any filename arguments or the -- switch terminator. Any switch found there is removed from @ARGV , and a variable of the same name as the switch is set in the Perl script. No switch bundling is allowed, since multicharacter switches are allowed.
-S	Makes Perl use the PATH environment variable to search for the script (unless the name of the script starts with a slash). Typically this is used to emulate #! startup on machines that don't support #!.
-T	Forces "taint" checks to be turned on. Ordinarily, these checks are done only when running setuid or setgid. It's a good idea to turn them on explicitly for programs run on another user's behalf, such as CGI programs.
-u	Causes Perl to dump core after compiling your script. You can then take this core dump and turn it into an executable file by using the *undump* program (not supplied). This speeds startup at the expense of some disk space (which you can minimize by stripping the executable). If you want to execute a portion of your script before dumping, use Perl's dump operator instead. Note: availability of *undump* is platform-specific; it may not be available for a specific port of Perl.

Slika 3 Perl prekidači 2



Switch	Function
-U	Allows Perl to do unsafe operations. Currently, the only "unsafe" operations are the unlinking of directories while running as superuser and running setuid programs with fatal taint checks turned into warnings.
-v	Prints the version and patch level of your Perl executable.
-V	Prints a summary of the major Perl configuration values and the current value of @INC.
-V:name	Prints the value of the named configuration variable to STDOUT.
-w	Prints warnings about identifiers that are mentioned only once and scalar variables that are used before being set. Also warns about redefined subroutines and references to undefined filehandles or to filehandles opened as read-only that you are attempting to write on. Warns you if you use a non-number as though it were a number, if you use an array as though it were a scalar, if your subroutines recurse more than 100 levels deep, etc.
-x [directory]	Tells Perl to extract a script that is embedded in a message, by looking for the first line that starts with #! and contains the string "perl" . Any meaningful switches on that line after the word "perl" are applied. If a directory name is specified, Perl switches to that directory before running the script. The script must be terminated with __END__ or __DATA__ if there is trailing text to be ignored . (The script can process any or all of the trailing text via the DATA filehandle if desired.)

Slika 4 Perl prekidači 3

Slika 5 Perl prekidači 4

Example

Create a very simple example of a Perl script, Hello.pl.

There are a few elements every PERL script must contain in order to function.

Open up your favorite simple text editor, the file extension for PERL scripts is .pl.

Save your files with this extension (Hello.pl).

```
#!/usr/bin/perl
print "content-type: text/html \n\n";
print "Hello, Perl!";
```

Display:

```
Hello, Perl!
```

The first line of every PERL script is a commented line directed toward the PERL interpreter, usually in Linux /usr/bin/perl.

```
#!/usr/bin/perl
```

We have to introduce some HTTP headers so that Perl understands we are working with a web browser. To do this we have to run another line of strange code called an HTTP header:

```
print "content-type: text/html \n\n";
```

Now that we have located the interpreter and told PERL we are working with the web, we can print text to the browser with the print function.

```
print "Hello, Perl!";
```

You should see "Hello, Perl!" in the top left corner of your browser. If you have a problem, ask about it on the Perl Forum

Opcije i komentari

Cilj

- Upoznavanje sa Perl opcijama komandne linije i komentarima

Opcije i komentari

Opcije i komentari

Command line options and warnings

Perl has a number of command line options, which you can specify on the command line by typing Perl options hello.pl or which you can include in the shebang line. The most commonly used option is -w to turn on warnings:

```
#!/usr/bin/perl –w
```

It’s always a good idea to turn on warnings while you’re developing code, and often once your code has gone into production, too.

Lexical warnings

In Perl versions 5.6 and above you can use Perl’s warnings pragma rather than the command lineswitch if you prefer. This also gives you the option to specify which warnings you wish to receive,and to upgrade those warnings to exceptions if necessary.

```
#!/usr/bin/perl
```

```
use warnings;
```

Specifying your version

To find the version of Perl available on your machine you can use the perl -v command. This will

print something like:

```
# perl –v
```

This is perl, v5.10.1 (*) built for i486-linux-gnu-thread-multi (with 56 registered patches, see perl -V for more detail) If you wish to take advantage of the new features that your Perl version provides you can tell Perl to do so by specifying the minimum version of Perl your code is targeting:

```
#!/usr/bin/perl

use v5.10.1;

use strict;

use warnings;
```

In versions 5.12 and above, this will also turn on strictures for us automatically.

Comments

Comments in Perl start with a hash sign (#), either on a line on their own or after a statement.

Anything after a hash is a comment up to the end of the line.

```
#!/usr/bin/perl -w

# This is a hello world program

print "Hello world!\n"; # print the message
```

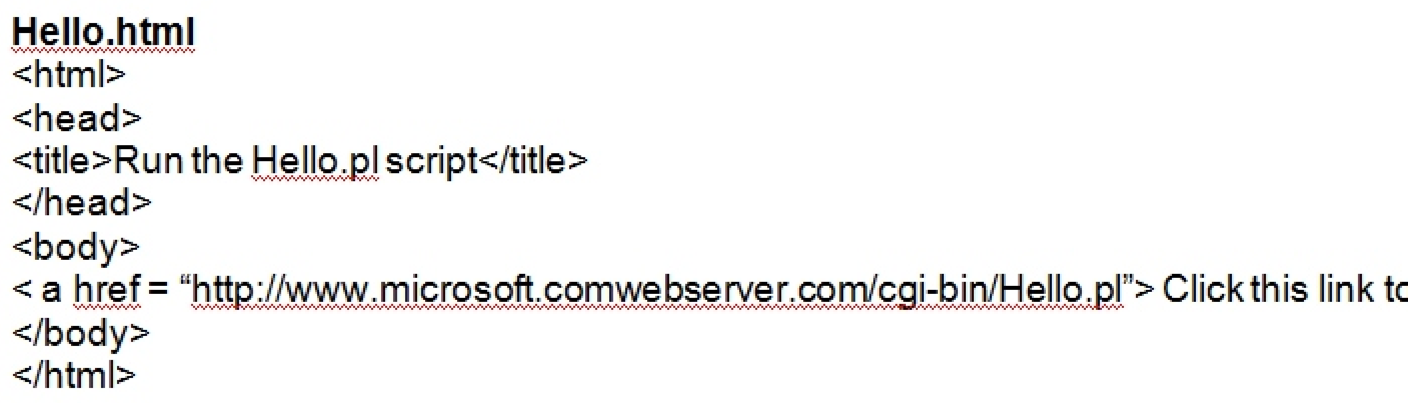
Block comments

To comment a block of text (or code) you can use Perl’s Plain Old Documentation-POD tags. You

can read more about POD in perldoc perlpod.

```
=begin comment

This content is commented out.

It may span many lines.

print "This statement won’t be executed by Perl\n";

=end comment

=cut

print "Hello world!\n"; # print the message
```

The next steps include the uploading (generally through a FTP client) of your script on your web server in the cgi-bin directory and the setting up of the script permissions. After that, you can create a simple web page to run the script, by creating the file Hello.html and running it from your favorite browser.



```
Hello.html
<html>
<head>
<title>Run the Hello.pl script</title>
</head>
<body>
< a href = “http://www.microsoft.comwebserver.com/cgi-bin/Hello.pl”> Click this link to
</body>
</html>
```

Slika 1 Hello program

Shebang

Cilj

- Upoznavanje sa pojmom Shebang

Shebang

Shebang

The "shebang" line for Unix

Unix and Unix-like operating systems do not automatically realize that a Perl script (which is just a text file after all) should be executable. As a result, we have ask the operating system to change the file’s permissions to allow execution:

% chmod +x hello.pl

Once the file is executable we also need to tell Unix how to execute the program. This allows the operating system to have many executable programs written in different scripting languages. We tell the operating system to use the Perl interpreter by adding a "shebang" line (called such because the # is a "hash" sign, and the ! is referred to as a "bang", hence "hashbang" or "shebang").

#!usr/bin/perl

Of course, if the Perl interpreter were somewhere else on our system, we’d have to change the shebang line to reflect that . This allows us to run our scripts just by typing:

% ./hello.pl

For security purposes, many UNIX and UNIX-like systems do not include your current directory in those which are searched for commands, by default. This means that if you try to invoke your script by typing:

% hello.pl # this doesn’t work

you’ll get the error: bash: hello.pl: command not found. This is why we prepend our command with the current working directory (./hello.pl).

The "shebang" line for non-Unixes

It’s always considered a good idea for all Perl programs to contain a shebang line. This is helpful because it allows us to include command line options, which we’ll cover shortly.

If your program will only ever be run on your single operating system then you can use the line:

```
#!perl
```

However it is considered good practice to use the traditional:

```
#!/usr/bin/perl
```

as this assists with cross-platform portability.

Dodatak

*Incomputing, a**shebang**(also called a**sha-bang**,**hashbang**, **pound-bang**,**hash-exclam**, or**hash-pling**), is the character sequence consisting of the charactersnumbersignandexclamation mark(that is, "`#!`") at the beginning of ascript.*

Under UNIX-like operating systems, when a script with a shebang is run as a program, theprogram loaderparses the rest of the script's initial line as aninterpreter directive; the specified interpreter program is run instead, passing to it as an argument the path that was initially used when attempting to run the script.For example, if a script is named with the path "`path/to/script`", and it starts with the following line:

```
#!/bin/sh
```

CrunchBang Linux

CrunchBang Linux (abbreviated #!) is a Linux distribution derived from Debian by Philip Newborough.

CrunchBang is designed to use comparatively few system resources. Instead of a desktop environment it employs a customized implementation of the Openbox window manager. Many of its preinstalled applications use the GTK+ widget toolkit.

CrunchBang has its own software repository but draws the vast majority of packages from Debian's repositories.

Linkovi i termini

Linkovi

Primer 1 - http://en.wikipedia.org/wiki/Shebang_(Unix)

Primer 2 - http://en.wikipedia.org/wiki/CrunchBang_Linux Termini

cross-platform portability - prenosivost na razne platforme

shebang - sekvenca karaktera koja se sastoji od # karaktera i ! znaka

Starting Perl program

Cilj

- Startovanje Perl programa

Starting Perl program

Starting Perl program

To summarize, Perl program should always start with:

A shebang line

A comment (what your program does)

The strict pragma

The warnings pragma

For example:

```
#!/usr/bin/perl
# This program ......
use strict;
use warnings;
```

You may wish to add use diagnostics; while your program is in development.

Na Win32 sistemu, povezati ekstenziju (npr. .plx s vrstom datoteke i dvaputa liknite na ikonu za Perl skript s tom vrstom datoteke. Ako koristite ActiveState verziju Win32 Perl, instalacini skript normalno odgovara i od traži da stvoriti povezivanje.

Na Win32 sistemima, ako dvaput kliknite na ikonu za Perl izvršnu verziju, naći ćete se u prozoru komandne linije s trepćućim pokazivačem.

Možete unijeti svoje Perl naredbe, ukazati na kraj vašeg ulaza sa CTRL-Z, i Perl će prevesti program i izvršiti skriptu. Perl analizira (engl. parse) ulaznu datoteku od početka, osim ako ste naveli -x prekidač. Ako postoji #! linija, uvijek se ispituje za prekidače u toku analiziranja linije. Dakle, prekidači se ponašaju dosljedno, bez obzira koliko se Perl poziva.

Nakon lociranja skripte, Perl prevodi cijeli skript u unutarnju formu. Ako postoje bilo kakve greške kompilacije, neće se izvršiti skripta. Ako je skripta sintaktički ispravna, ona se izvršava. Ako se skripta izvršava do kraja bez nalaženja exit ili die operatora, implicit exit (0) se generiše pod uslovom da ukazuju na uspješan završetak izvršavanja.

Linkovi i termini

Linkovi

Primer 1 - http://perlmaven.com/perl-tutorial

Primer 2 - http://perldoc.perl.org/perlpragma.html

Termini

parsing - gramatičko raščlanjivanje

Ulaz sa tastature

Cilj

- Upoznavanje sa načinom unošenja ulaza sa tastature

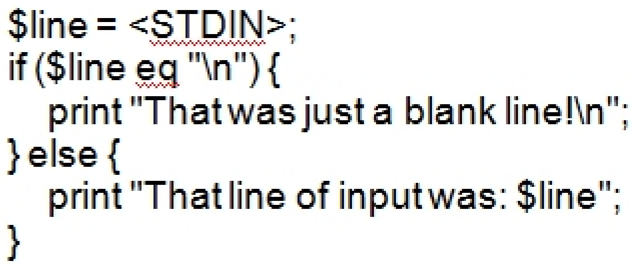
Ulaz sa tastature

Unošenje ulaza sa tastature

Za unošenje vrijednosti sa tastature u Perl program može se koristiti operator <STDIN>.

Svaki puta kada se koristi <STDIN>, Perl očekuje skalarnu vrijednost, Perl čita cijelu liniju sa standardnog ulaza (sve do "newline" karaktera) i koristi string kao učitanu vrijednost sa <STDIN>. String vrednost <STDIN> obično ima newline karakter na kraju.

Primjer



```
$line = <STDIN>;
if ($line eq "\n") {
   print "That was just a blank line!\n";
} else {
   print "That line of input was: $line";
}
```

Slika 1 Primer 1

Rezultat

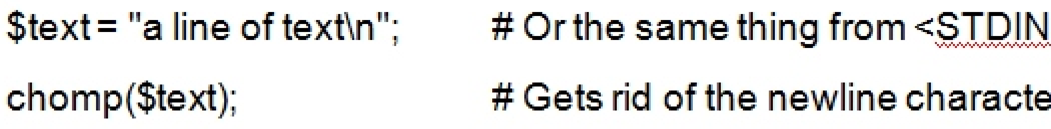
C:\>perl s.pl

That was just a blank line!

chomp Operator

It works on a variable The variable has to hold a string, and if the string ends in a newline character, chomp() removes the newline.

Example

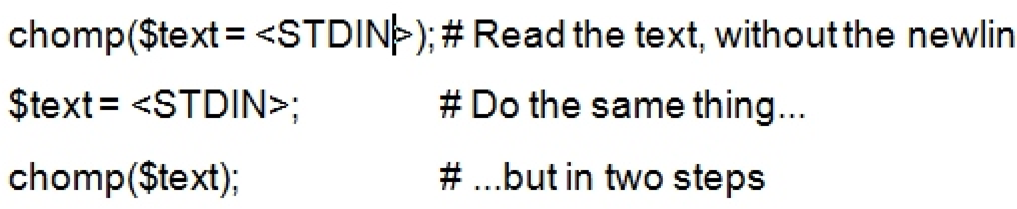


```
$text = "a line of text\n";        # Or the same thing from <STDIN
chomp($text);                      # Gets rid of the newline characte
```

Slika 2 Primer 2

It's a way to remove a trailing newline from a string in a variable. There's an easier way to use chomp() because of a simple rule: any time that you need a variable in Perl, you can use an assignment instead. First, Perl does the assignment. Then it uses the variable in whatever way you requested.

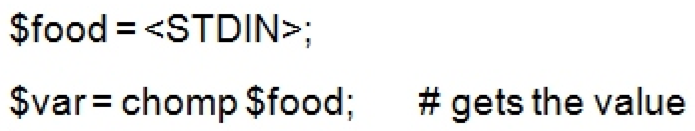
Example



```
chomp($text = <STDIN>); # Read the text, without the newlin
$text = <STDIN>;        # Do the same thing...
chomp($text);           # ...but in two steps
```

Slika 3 Primer 3

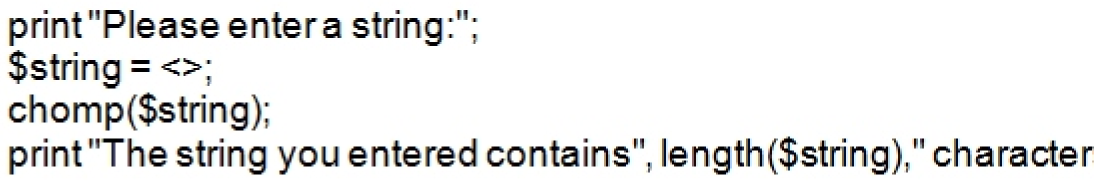
chomp() is actually a function. As a function, it has a return value, which is the number of characters removed. This number is hardly ever useful:



```
$food = <STDIN>;
$var = chomp $food;    # gets the value
```

Slika 4 Primer 4

Example



```
print "Please enter a string:";
$string = <>;
chomp($string);
print "The string you entered contains", length($string)," character
```

Slika 5 Primer 5

Linkovi i termini

Linkovi

Primer 1 - http://padre.perlide.org/trac/wiki/Features/Autocomplete

Primer 2 - http://perldoc.perl.org/functions/chomp.html

Primer 3 - http://www.tutorialspoint.com/perl/perl_chomp.htm

Termini

newline karakter, karakter nove linije

Perl varijable

Cilj

- Upoznavanje sa pojmom Perl varijable

Perl varijable

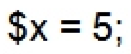
Perl variable

A variable is a place where we can store data. Think of it like a pigeonhole with a name on it indicating what data is stored in it.

The Perl language is very much like human languages in many ways, so you can think of variables as being the "nouns" of Perl. For instance, you might have a variable called "total" or "employee".

Declaration of Variables

A variable need needs not be explicitly declared; its “declaration” consists of its first usage. For example, if the statement:



```
$x = 5;
```

Slika 1 Primer 1

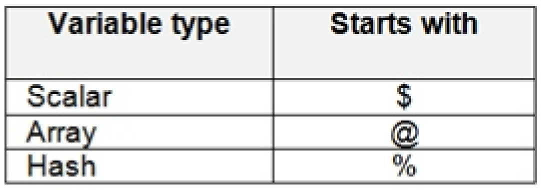
If you wish to have protection against accidentally using a variable which has not been previously defined, say due to a misspelling, include a line use strict; at the top of your source code.

Variable names

Variable names in Perl may contain alphanumeric characters in upper or lower case, and underscores. A variable name may not start with a number, as that means something special, which we’ll encounter later. Likewise, variables that start with anything non-alphanumeric are also special, and we’ll discuss that later, too.

It’s standard Perl style to name variables in lower case, with underscores separating words in the name. For instance, the employee_number. Upper case is usually used for constants, for instance LIGHT_SPEED or PI. Following these conventions will help make your Perl more maintainable and more easily understood by others.

Lastly, variable names all start with a punctuation sign (correctly known as a sigil) depending on what sort of variable they are (Slika 2):



Variable type	Starts with
Scalar	$
Array	@
Hash	%

Slika 2 Variable types

Definition

my variable

A **my** declares the listed variables to be local (lexically) to the enclosing block, file, or eval. If more than one value is listed, the list must be placed in parentheses.

Example: My variable

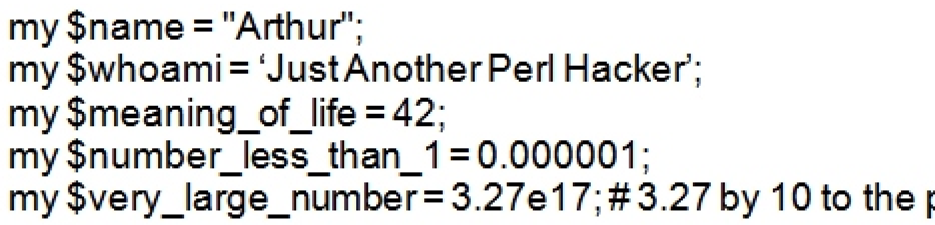
my EXPR

my TYPE EXPR

my EXPR : ATTRS

my TYPE EXPR : ATTRS

Example: Assign values to scalar variables



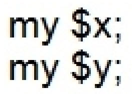
```
my $name = "Arthur";
my $whoami = ‘Just Another Perl Hacker’;
my $meaning_of_life = 42;
my $number_less_than_1 = 0.000001;
my $very_large_number = 3.27e17; # 3.27 by 10 to the
```

Slika 3 Primer 1

Scope of Variables

Variables in Perl are <u>global by default</u>. To make a variable <u>local</u> to subroutine or block, the **my** construct is used.

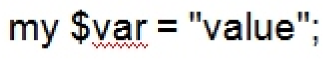
For instance,



```
my $x;
my $y;
```

Slika 4 Primer 2

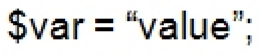
The previous examples used the syntax



```
my $var = "value";
```

Slika 5 Primer 3

The **my** is actually not required; one could just use:

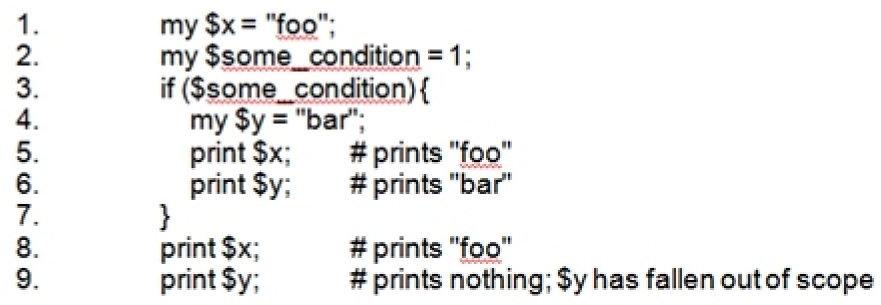


```
$var = “value”;
```

Slika 6 Primer 4

However, the above usage will create global variables throughout your program, which is bad programming practice, my creates lexically scoped variables instead. The variables are scoped to the block (i.e. a bunch of statements surrounded by curly-braces) in which they are defined.

Example



```
my $x = "foo";
my $some_condition = 1;
if ($some_condition) {
    my $y = "bar";
    print $x;       # prints "foo"
    print $y;       # prints "bar"
}
print $x;           # prints "foo"
print $y;           # prints nothing; $y has fallen out of scope
```

Slika 7 Primer 5

Using my in combination with a use strict; at the top of your Perl scripts means that the interpreter will pick up certain common programming errors. For instance, in the example above, the final print would cause a compile-time error and prevent you from running the program. Using strict is highly recommended.

Variable scoping and the strict pragma

Many programming languages require you to "pre-declare" variables --- that is, say that you're going to use them before you do so. Variables can either be declared as global (that is, they can be used anywhere in the program) or local (they can only be used in the same part of the program in which they were declared).

In Perl, it is not necessary to declare your variables before you begin. You can summon a variable into existence simply by using it, and it will be globally available to any routine in your program. If you're used to programming in C or any of a number of other languages, this may seem odd and even dangerous to you. This is indeed the case. That's why you want to use the **strict pragma**.

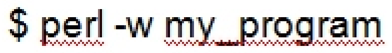
Strictures are turned on automatically for you if your program specifies a version of Perl greater than or equal to 5.12.0.

use v5.12.0; # no need to write use strict!

Perl can be told to warn you when it sees something suspicious going on in your program. With Perl 5.6 and later, you can turn on warnings with a pragma (but be careful because it won't work for people with earlier versions of Perl):

use warnings; # still need to turn warnings on though

One can use the -w option on the command line, which turns on warnings everywhere in your program:



```
$ perl -w my_program
```

Slika 8 Primer 6

You can also specify the command-line switches on the shebang line:

#!/usr/bin/perl -w

The ***perldiag****documentation* has both the short warning and the longer diagnostic description, and is the source of diagnostics' helpfulness:

#!/usr/bin/perl

use diagnostics;

Variable types

Cilj

- Upoznavanje sa tipovima Perl varijabli

Variable types

Variable types

Perl has three main types of variables:

Scalars

Arrays

Hashes

Scalars are essentially simple variables. They are preceded by a dollar sign. A scalar is a number, a string, or a reference. (A reference is a scalar that points to another piece of data. References are discussed later in this chapter.) If you provide a string where a number is expected or vice versa, Perl automatically converts the operand using fairly intuitive rules.

Arrays are ordered lists of scalars that you access with a numeric subscript (subscripts start at 0). They are preceded by an "at" sign (and).

A hash is a set of key/value pairs. Hashes are preceded by a percent (%) sign. To refer to a single element of a hash, you use the hash variable name followed by the "key" associated with the value in curly brackets. For example, the hash:

%fruit = ('apples', 3, 'oranges', 6);

has two values (in key/value pairs). If you want to get the value associated with the key apples, you use$fruit{'apples'}. It is often more readable to use the => operator in defining key/value pairs. The => operator is similar to a comma, but it's more visually distinctive, and it also quotes any bare identifiers to the left of it:

%fruit = (

apples => 3,

oranges => 6

);

Example: Hash

#!/usr/bin/perl

print "content-type: text/html \n\n";

# DEFINE A HASH

%coins = ("Quarter", 25, "Dime", 10, "Nickel", 5);

# PRINT THE HASH

print %coins;

Display:

Nickel5Dime10Quarter25

Scalar variables

Cilj

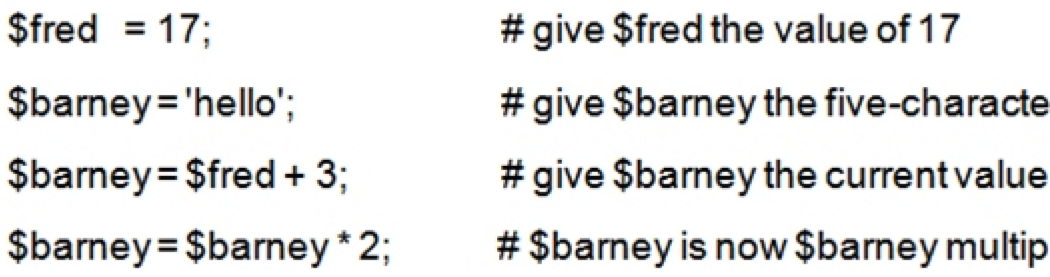
- Upoznavanje sa skalarnim varijablama

Scalar variables

Scalar variables

The most common operation on a scalar variable is *assignment,* which is the way to give a value to a variable. The Perl assignment operator is the equals sign (much like other languages), which takes a variable name on the left side, and gives it the value of the expression on the right.

For example:

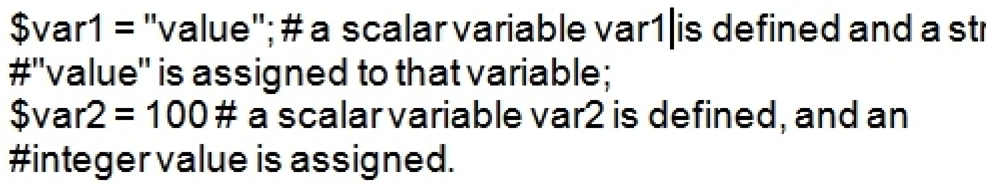


```
$fred   = 17;                 # give $fred the value of 17
$barney = 'hello';            # give $barney the five-characte
$barney = $fred + 3;          # give $barney the current value
$barney = $barney * 2;        # $barney is now $barney multip
```

Slika 1 Primer 1

Notice that last line uses the $barney variable twice: once to get its value (on the right side of the equals sign), and once to define where to put the computed expression (on the left side of the equals sign). This is legal, safe, and rather common. In fact, it’s so common that you can write it using convenient shorthand, as you’ll see in the next section.

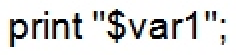
Example : Scalar variable in Perl



```
$var1 = "value"; # a scalar variable var1 is defined and a str
#"value" is assigned to that variable;
$var2 = 100 # a scalar variable var2 is defined, and an
#integer value is assigned.
```

Slika 2 Primer 2

Example: Print a scalar values



```
print "$var1";
```

Slika 3 Primer 3

You can find a selection of general-utility scalar subroutines in the Scalar::Util module. The simplest form of variable in Perl is the scalar.

A scalar is a single item of data such as:

Arthur

Just Another Perl Hacker

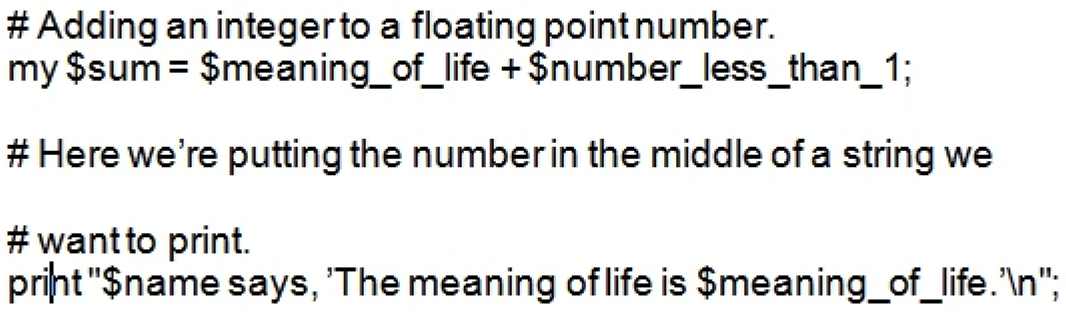
42

0.000001

3.27e^{17}

A scalar can be text of any length, and numbers of any precision (machine dependent, of course).

Perl doesn’t need us to tell it what type of data we’re going to put into the scalar. In fact, Perl doesn’t care if the type of data in the scalar changes throughout your program. Perl magically converts between them when it needs to. For instance, it’s quite legal to say:



```
# Adding an integer to a floating point number.
my $sum = $meaning_of_life + $number_less_than_1;

# Here we’re putting the number in the middle of a string we

# want to print.
print "$name says, ’The meaning of life is $meaning_of_life.’\n";
```

Slika 4 Primer 4

This may seem extraordinarily alien to those used to strictly typed languages; but believe it or not, the ability to transparently convert between variable types is one of the great strengths of Perl.

Perl liste

Cilj

- Upoznavanje sa Perl listama

Perl liste

Linkovi i termini

Perl brojevi

Cilj

- Upoznavanje sa Perl formatom brojeva

Perl brojevi

Perl brojevi

Perl čuva interno brojeve ili kao označene cele brojeve (engl. signed integers) ili kao vrednosti sa pomičnom tačkom dvostruke preciznosti (engl. double-precision floating-point values). Numerički literalie su definisani u jednom od sledećih formata:

12345 # integer

-54321 # negative integer

12345.67 # floating point

6.02E23 # scientific notation

0xffff # hexadecimal

0377 # octal

4_294_967_296 # underline for legibility

Since Perl uses the comma as a list separator, you cannot use a comma for improving legibility of a large number. To improve legibility, Perl allows you to use an underscore character instead. The underscore only works within literal numbers specified in your program, not in strings functioning as numbers or in data read from somewhere else. Similarly, the leading 0x for hex and 0 for octal work only for literals. The automatic conversion of a string to a number does not recognize these prefixes - you must do an explicit conversion

Example

print "5 + 6 = ", 5+6, "\n";

print "(2 + 3) * 6 = ", (2+3)*6, "\n";

print "2 + 3 * 6 = ", 2+3*6, "\n";

print "2 raised to the power of 8 is ", 2**8, "\n";

print "10-5 = ", 10-5, ". 5-10 = ", 5-10, "\n";

print "2/3 = ", 2/3, "\n";

Floating-point literals

A literal is how you represent a value in your Perl source code. A literal is not the result of a calculation or an I/O operation; it’s data that you type directly into your program. Perl’s floating-point literals should look familiar to you.

Numbers with and without decimal points are allowed (including an optional plus or minus prefix), as well as tacking on a power-of-10 indicator (exponential notation) with E notation. For example:

1.25

255.000

255.0

7.25e45 # 7.25 times 10 to the 45th power (a big number)

–6.5e24 # negative 6.5 times 10 to the 24^{th} (a big negative number)

–12e–24 # negative 12 times 10 to the $–24^{th}$ (a very small negative number)

–1.2E–23 # another way to say that the E may be uppercase

Integer literals

Integer literals are also straightforward, as in:

0

2001

–40

255

61298040283768

That last one is a little hard to read. Perl allows you to add underscores for clarity within integer literals, so you can also write that number with embedded underscores to make it easier to read:

61_298_040_283_768

It’s the same value; it merely looks different to us human beings. You might have thought that commas should be used for this purpose, but commas are already used for a more-important purpose in Perl

Non-decimal integer literals

Like many other programming languages, Perl allows you to specify numbers in other ways than base 10 (decimal). Octal (base 8) literals start with a leading 0, hexadecimal (base 16) literals start with a leading 0x, and binary (base 2) literals start with a leading 0b.

The hex digits A through F (or a through f) represent the conventional digit values of 10 through 15. For example:

0377 # 377 octal, same as 255 decimal

0xff # FF hex, also 255 decimal

0b11111111 # also 255 decimal

Although these values look different to us humans, they’re all three the same number to Perl. It makes no difference to Perl whether you write 0xFF or 255.000, so choose the representation that makes the most sense to you and your maintenance programmer (by which we mean the poor chap who gets stuck trying to figure out what you meant when you wrote your code. Most often, this poor chap is you, and you can’t recall why you did what you did three months ago).

When a nondecimal literal is more than about four characters long, it may be hard to read, so underscores are handy:

0x1377_0B77

0x50_65_72_7C

Numeric operators

Perl provides the typical ordinary addition, subtraction, multiplication, and division operators, and so on. For example:

2 + 3 # 2 plus 3, or 5

5.1 – 2.4 # 5.1 minus 2.4, or 2.7

3 * 12 # 3 times 12 = 36

14 / 2 # 14 divided by 2, or 7

10.2 / 0.3 # 10.2 divided by 0.3, or 34

10 / 3 # always floating-point divide, so 3.3333333...

Perl also supports a modulusoperator (%). The value of the expression 10 % 3 is the remainder when 0 is divided by 3, which is 1. Both values are first reduced to their integer values, so 10.5 % 3.2 is computed as 10 % 3.

Linkovi i termini

Linkovi

Primer 1 - http://www.tizag.com/perlT/perlnumbers.php

Primer 2 - http://perldoc.perl.org/perlnumber.html

Termini

označeni brojevi

brojevi sa pomičnom tačkom dvostruke preciznosti

Strings

Cilj

- Upoznavanje sa stringovima

Strings

Strings

Strings are sequences of characters. Strings may contain any combination of any characters. The shortest possible string has no characters, and is called the *empty string*. The longest string fills all of your available memory (although you wouldn’t be able to do much with that). This is in accordance with the principle of “no built-in limits” that Perl follows at every opportunity. Typical strings are printable sequences of letters and digits and punctuation. However, the ability to have any character in a string means you can create, scan, and manipulate raw binary data as strings—something with which many other utilities would have great difficulty. For example, you could update a graphical image or compiled program by reading it into a Perl string, making the change, and writing the result back out.

Perl has full support for Unicode, and your string can contain any of the valid Unicode characters. However, because of Perl’s history, it doesn’t automatically interpret your source code as Unicode. If you want to use Unicode literally in your program, you need to add the utf8 pragma:

use utf8;

For the rest of this book, we assume that you’re using that pragma. In some cases it won’t matter, but if you see characters outside the ASCII range in the source, you’ll need it. Also, you should ensure that you save your files with the UTF-8 encoding.

String Operators

You can concatenate, or join, string values with the . operator. (Yes, that’s a single period.) This does not alter either string, any more than 2+3 alters either 2 or 3. The resulting (longer) string is then available for further computation or assignment to a variable. For example:

"hello" . "world" # same as "helloworld"

"hello" . ' ' . "world" # same as 'hello world' 'hello world' . "\n" # same as "hello world\n"

Note that you must explicitly use the concatenation operator, unlike in some other languages where you merely have to stick the two values next to each other.

A special string operator is *thestring repetitionoperator*, consisting of the single lowercase letter x. This operator takes its left operand (a string) and makes as many concatenated copies of that string as indicated by its right operand (a number). For example:

"fred" x 3 # is "fredfredfred"

"barney" x (4+1) # is "barney" x 5, or "barneybarneybarneybarneybarney"

5 x 4.8 # is really "5" x 4, which is "5555"

That last example is worth noting carefully. The string repetition operator wants a string for a left operand, so the number 5 is converted to the string "5" (using rules described in detail later), giving a one-character string.

Error message

Unrecognized character \x92

Chances are that the system's substitutes some quotes with a "fancy" quote that now causes trouble.

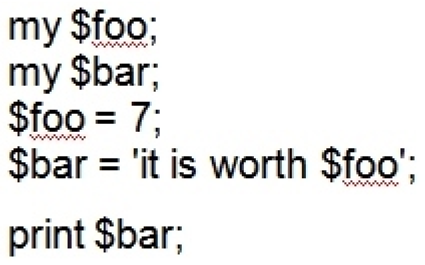
' ’

Exercise

The above example is available in your directory as exercises/interpolate.pl. Run the script to see what happens. Try changing the type of quotes for each string. What happens?

Single quotation marks are used to enclose data you want taken literally. Just as the <code></code> tags here at the Monastery make all text they enclose literally rendered, whitespace and all, so too does a set of single-quotes in Perl ensure that what they enclose is used literally:

Example

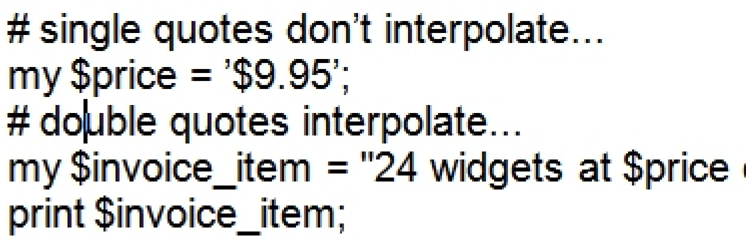


```
my $foo;
my $bar;
$foo = 7;
$bar = 'it is worth $foo';

print $bar;
```

Slika 1 Primer 1

Double and single quotes

In Perl, quotes are required to distinguish strings from the language’s reserved words or other expressions. Either type of quote can be used, but there is one important difference: double quotes can include other variable names inside them, and those variables will then be interpolated --- as in the print example above --- while single quotes do not interpolate.



```
# single quotes don't interpolate...
my $price = '$9.95';
# double quotes interpolate...
my $invoice_item = "24 widgets at $price
print $invoice_item;
```

Slika 2 Primer 2

Exercise

Run the script to see what happens. Try changing the type of quotes for each string. What happens?

Linkovi i termini

Linkovi

Primer 1 - http://perl.about.com/od/perltutorials/a/Perl-String-Manipulation.htm

Primer 2 - http://www.comp.leeds.ac.uk/Perl/matching.html

Termini

interpolation	umetanje
concatenation	spajanje

Special characters

Cilj

- Upoznavanje sa specijalnim karakterima

Special characters

Special characters

Special characters, such as the \n newline character, are only available within double quotes. Single quotes will fail to expand these special characters just as they fail to expand variable names. The only exceptions are that you can quote a single quote or backslash with a backslash.

print ’Here\’s an example’;

When using either type of quotes, you must have a matching pair of opening and closing quotes. If you want to include a quote mark in the actual quoted text, you can escape it by preceding it with a backslash (as shown in next figure)

When using either type of quotes, you must have a matching pair of opening and closing quotes. If you want to include a quote mark in the actual quoted text, you can escape it by preceding it with a back slash

print "He said, \"Hello!\"\n";

```
print ’It was Jamie\’s birthday.’;
```

You can also use a backslash to escape other special characters such as dollar signs within double quotes:

```
print "The price is \$300\n";
```

To include a literal backslash in a double-quoted string, use two backslashes: \\

Be careful, there’s a common syntax error when the last character of a string is a backslash:

```
print ’This is a backslash: \’;
```

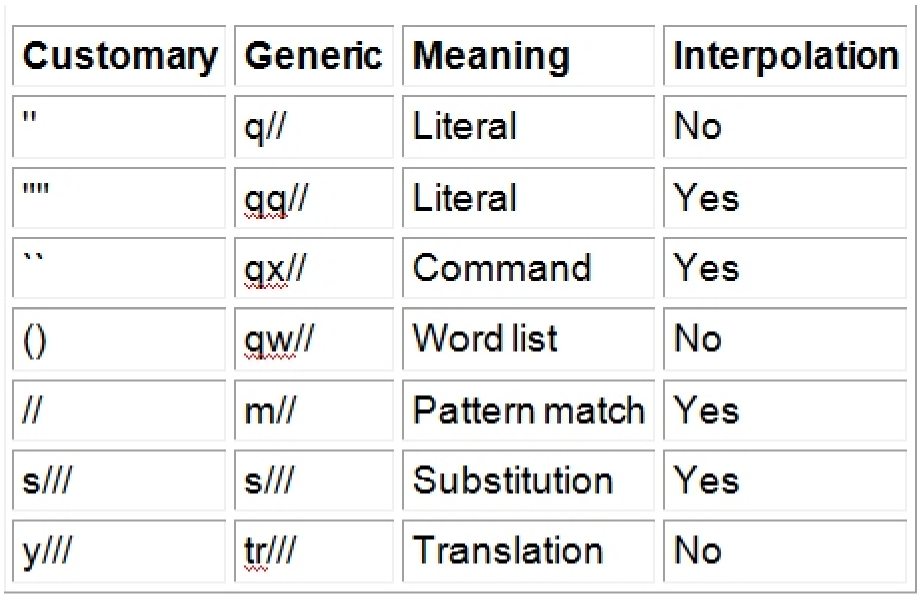
In this case Perl reads the backslash as an escape for the quote character, and thus our string does not terminate. In this case you must tell Perl that you don’t want this effect by escaping the backslash:

```
print ’This is a backslash: \\’;
```

The backslash can precede many different characters to mean different things (generally called a *backslash escape*). The nearly complete list of doule-quoted string escapes.

Slika 1 Double-quoted string backslash escapes

Another feature of double-quoted strings is that they are *variable **interpolated***, meaning that some variable names within the string are replaced with their current values when the strings are used. You haven’t formally been introduced to what a variable looks like yet, so we’ll get back to that later in this chapter. Next figure lists alternative quoting schemes that can be used in Perl. They are useful in diminishing the number of commas and quotes you may have to type, and also allow you to not worry about escaping characters such as backslashes when there are many instances in your data. The generic forms allow you to use any non-alphanumeric, non-whitespace characters as delimiters in place of the slash (/). If the delimiters are single quotes, no variable interpolation is done on the pattern. Parentheses, brackets, braces, and angle brackets can be used as delimiters in their standard opening and closing pairs.



Customary	Generic	Meaning	Interpolation
''	q//	Literal	No
""	qq//	Literal	Yes
``	qx//	Command	Yes
()	qw//	Word list	No
//	m//	Pattern match	Yes
s///	s///	Substitution	Yes
y///	tr///	Translation	No

Slika 2 Quoting Syntax in Perl

The undef Value

Variables have the special **undef** value before they are first assigned. string. But undef is neither a number nor a string; it’s an entirely separate kind of scalar value.

Example

```
# Add up some odd numbers
$n = 1;
while ($n < 10) {
$sum += $n;
$n += 2; # On to the next odd number
}
print "The total was $sum.\n";
```

Napredna interpolacija varijabli

Cilj

- Upoznavanje sa naprednom interpolacijom varijabli

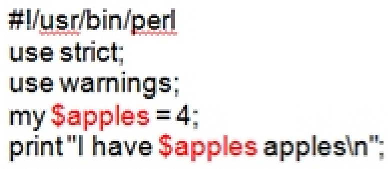
Napredna interpolacija varijabli

Napredna interpolacija varijabli

Interpolacija, što znači "uvođenje ili ubacivanje nečega , je ime metode zamene varijable njenom vrednošću.

Interpolacija stringova

U Perl, bilo koji string sa navodnicima (ili nešto što označava navodnike) će se interpolirati. Bilo koja varijabla ili escape char koji se pojavljuje u nizu će biti zamenjena sa vrijednošću te varijable. Na Slici 1 je dat mali primjer:



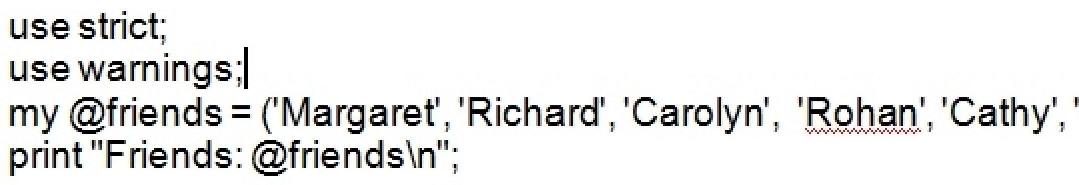
```
#!/usr/bin/perl
use strict;
use warnings;
my $apples = 4;
print "I have $apples apples\n";
```

Slika1 Primer 1

Rezultat:

I have 4 apples

Kada se štampa Perl niz u okviru navodnika, elementi niza se štampaju sa razmacima (engl. spaces) između njih, Slika 2:



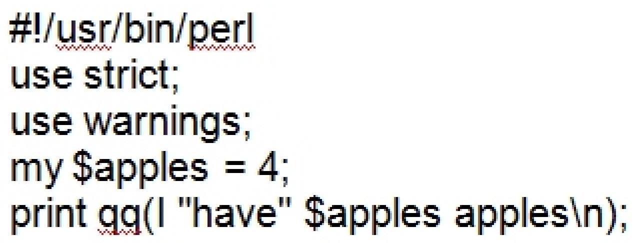
```
use strict;
use warnings;
my @friends = ('Margaret', 'Richard', 'Carolyn',  'Rohan', 'Cathy', '
print "Friends: @friends\n";
```

Rezultat

Friends: Margaret Richard Carolyn Rohan Cathy Yukiko

Ovo je jako korisno kod ispravljanje grešaka u programu (engl. debugging).

Funkcija qq() ima istu funkciju kao navodnici samo je lakše raditi s njom, Slika 3.



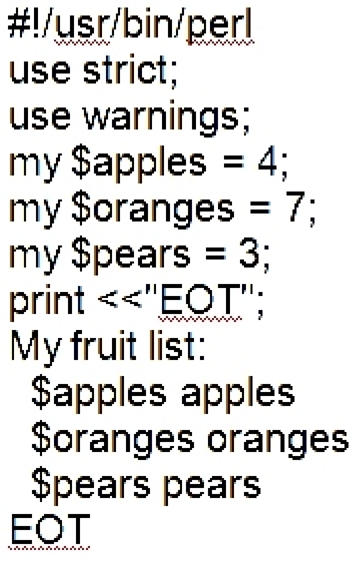
```
#!/usr/bin/perl
use strict;
use warnings;
my $apples  = 4;
print qq(I "have" $apples apples\n);
```

Slika 3 Primer 3

Rezultat

I "have" 4 apples

Ako se end token ovog dokumenta (na pr. <<"EOT") stavi pod navodnke, onda će varijable u ovom dokumentu biti interpolirane, Slika 4:



```
#!/usr/bin/perl
use strict;
use warnings;
my $apples = 4;
my $oranges = 7;
my $pears = 3;
print <<"EOT";
My fruit list:
  $apples apples
  $oranges oranges
  $pears pears
EOT
```

Slika 4 Primer 4

Rezultat:

My fruit list:

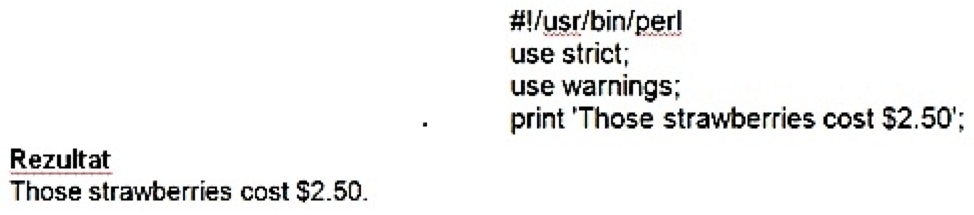
4 apples

7 oranges

3 pears

Ne interpolirani stringovi

Jednostruki navodni znaci (‘) (engl. single quotation) u predstavljanju stringova ne generišu interpolaciju varijabli. Jednostruki navodni znaci se koriste kada se treba uključiti posebne znake u rezultatu, Slika 5.

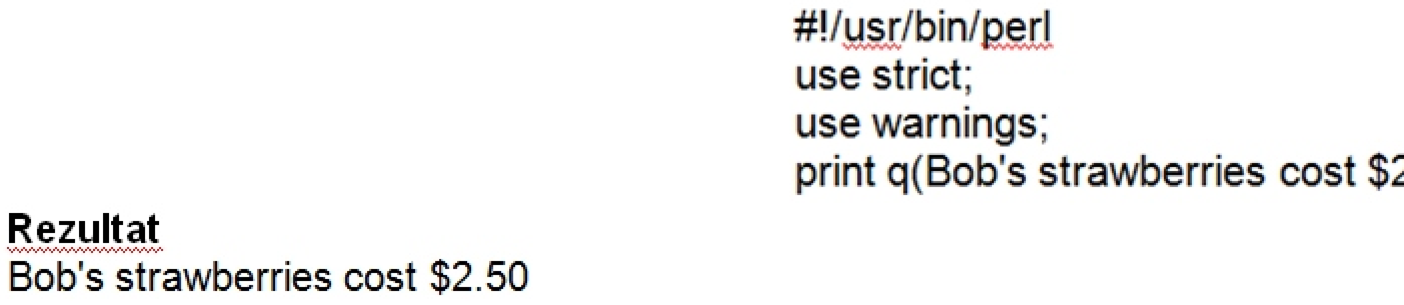


```
#!/usr/bin/perl
use strict;
use warnings;
print 'Those strawberries cost $2.50';
```

```
Rezultat
Those strawberries cost $2.50.
```

Slika 5 Primer 5

Funkcija q() ima istu funkciju kao jednostruki navodni znaci samo je lakše raditi s njom, Slika 6.



```
#!/usr/bin/perl
use strict;
use warnings;
print q(Bob's strawberries cost $2
```

Rezultat

```
Bob's strawberries cost $2.50
```

Slika 6 Primer 6

Dodatak

1. Write a script which sets some variables:

a. your name

b. your street number

c. your favorite color

2. Print out the values of these variables using double quotes for variable interpolation.

3. Change the quotes to single quotes. What happens?

4. Print out the string C:\WINDOWS\SYSTEM\ twice -- once using double quotes, once using single quotes. How do you have to escape the backslashes in each case?

Linkovi

- Primer 1 - http://perlmeme.org/howtos/using_perl/interpolation.html
- Primer 2 - http://en.wikipedia.org/wiki/Variable_interpolation

Rečnik

- debugging - ispravljanje grešaka u programu
- interpolation - uvođenje ili ubacivanje nečega

L2: Uvod u Perl jezik

Zaključak*
Nakon studiranja sadržaja ovog poglavlja, studenti će steći početna znanja iz oblasti programiranja u Perl jeziku.

LearningObject

Uvod u Perl jezik

Uvod u Perl jezik

Napišite Perl program koji traži od korisnika da unese cjelobrojnu vrijednost, a onda daje informaciju da li je broj paran ili ne. Uključite upozorenja (engl. warnings) i isprobajte ovaj program sa ne-numeričkim ulazom. Koje upozorenje dobijate?

Koja se operacija izvršava prvo u datoj sekvenci Perl operacija? Da li je to važno za krajnji rezultat?

2 ** 3 ** 4

2 / 3 * 4

2 + 3 * 4 ** 5 – 6

Napisati Perl potprogram **show_args** koji štampa svoje argumente. Na primjer, ako su argumenti “fred”, “barney”, i “betty”, potprogram se poziva na slijedeći način:

show_args('fred', 'barney', 'betty');

Rezultat ovog poziva:

The arguments are fred barney betty

Napisati drugi Perl potprogram koji se zove **show_args_again** koji radi istu stvar kao i potprogram show_args u predhodnom primjeru. Iz potprograma **show_args_again**, pozvati potprogram show_args, koristeći symbol "AND" (i, eng. ampersand) ispred imena potprogarma. Pokušajte i bez simbola "AND". Objasnite razliku između ova dva poziva.

Napišite program koji traži od korisnika da unese dva broja i izvjesti koji od njih je veći. Što će se dogoditi ako su brojevi isti? Što će se dogoditi ako se ne unesu brojevi već nešto poput broja trideset sedam?

Zadaću poslati na adresu: nemanja.stojanovic.799@metropolitan.ac.rs

Naslov mejla treba da bude:

IT2008-IPT-SCRP-STRL-DZ-DZ02-Ime-Prezime-BrojIndeksa.docx

Perl programski jezik

Uvod

Perl programski jezik je jezik opšte namjene koji je razvio Larry Wall 1987 godine. Perl vuče svoje korijene iz drugih jezika kao što su C, sed, awk i Unix shell.

Perl je postao važan jezik za WWW razvoj, obradu teksta, Internet usluge, mail filtriranje, programiranje, i svaki drugi zadatak koji zahtijeva portabilna i lako rješiva rješenja.

Karaktetistike

Perl je interpretirani jezik. To znači da čim korisnik napiše program, može ga odmah pokrenuti. Ne postoji obavezna faza prevođenja ili kompilacije. Perl programi se mogu izvoditi na Unix, Windows NT, MacOS, DOS, OS / 2, VMS i Amiga OS.

Perl je kolaborativni jezik. CPAN (engl. Comprehensive Perl Archive Network) softverska arhiva sadrži besplatne Perl alate napisne zajednički čime se štedi vrijeme.

Perl je besplatan, izvorni kod i prevodilac su dostupnii bez naknade.

Perl je brz. Perl interpreter je pisan u C jeziju i nakon mnogih optimizacija je postao vrlo brz.

Perl je danas ne samo programski jezik već i vrlo aktivna zajednica programera i korisnika. Odlikuje ga kvalitetan repozitorij gotovih programskih rješenja.

Perl je prema svojim karakteristikama objektno orijentisani programski jezik opšte namjene s naglaskom na funkcionalnost, proširivost te relativno lako učenje jezika. Perl je bio jezik izbora za razvoj WWW aplikacija sredinom 90-ih godina. Od samih svojih početaka to je jezik UNIX i Linux sistemskih administratora koji ga koriste u svakodnevnom radu prvenstveno za automatizaciju procesa. Danas postoji i čitav niz korisničkih komercijalnih aplikacija pisanih u Perlu.

Perl prevodilac ima dvostruku namjenu: prevođenje izvornog koda u međuoblik pogodan za neposredno izvođenje (engl. code compilation) te samo izvođenje koda (engl. code execution).

Obje funkcije dostupne su u svakom trenutku, odnosno Perl omogućava kreiranje novih Perl programa odnosno Perl funkcija u toku samog izvođenja. Time je Perl blizak i funkcionalnim programskim jezicima kao što su Smalltalk i Haskell.

Trenutna verzija Perla koja je danas u širokoj upotrebi (Perl 5) ipak sadrži previše već ponešto zastarjelih programerskih tehnika, a zbog željene potpune kompatibilnosti sa starijim inačicama iz 80-ih godina ima izvjesnih nedostataka. Ovdje je potrebno napomenuti da je današnji stil programiranja u Perlu daleko od stila iz sredine 90-ih. Naglasak je prije svega na korištenju gotovih dobro testiranih korisničkih biblioteka dostupnih preko već spomenute CPAN arhive. Premda je Perl na glasu kao "kriptičan" jezik "hackera", uz nešto profesionalne discipline te pridržavanja određenih pravila programiranja, moguće je pisati vrlo pregledne i uredne programe

U gornjem primjeru vidimo da je Perl kod bitno kraći, nema deklaracije varijabli, a tip varijabli se određuje dinamički (ako se koristi operator **+**, a sadržaj varijable je numerički, interpreter izvršava operaciju sumiranja bez obzira što je varijabla $s inicijalno definirana kao**string**.

Installation of Perl language on different OSs

You must install Perl on your computer in order to run a Perl script. But before going into the hassle to install it, you need to check up if Perl is already installed on your computer. If your operating system is ***Unix*** or ***Linux***, there is a big chance that Perl is already installed. In this case, all you have to do is to skip this section.

To see if you have Perl installed on your computer you need to get to a command prompt. To get this, it depends on your operating system.

If you have ***Unix*** or ***Linux***, simple log in and you'll have access to the command prompt. If you have a graphical interface instead, after the log in you need to open a terminal window in order to have access to the command prompt.

On a Windows system, click Start-->Run and in the Open field type cmd and than confirm by clicking the OK button. It will open a cmd window where you can type the commands you want to be executed. At the command prompt, please type the following:

perl –v

and you'll receive either the message "command not found" which means perhaps you must install Perl, or Perl's version number.

C:\>perl –v

This is perl 5, version 14, subversion 2 (v5.14.2) built for MSWin32-x86-

(with 1 registered patch, see perl -V for more detail)

Copyright 1987-2011, Larry Wall

Binary build 1402 [295342] provided by ActiveState http://www.ActiveState

Built Oct 7 2011 15:49:44

Perl may be copied only under the terms of either the Artistic License or

GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on

this system using "man perl" or "perldoc perl". If you have access to the

Internet, point your browser at http://www.perl.org/, the Perl Home Page

Perl on Unix like OSs

Install 'make' through your package manager. You can then use all of the tools mentioned above.

Install Perl on Linux

First, you need a copy of Perl source bundle and you can download a stable version from the site "http://www.perl.com". At the moment I am writing these words, I found available the archive perl-5.8.8.tar.gz. You also need an ANSI C compiler, but don't worry about this, Perl's configuration program will check for one. If it will not find it, you can install a prebuilt version or install the C compiler supplied by your OS vendor, or for a free C compiler try "http://gcc.gnu.org/".Now, get a command prompt and enter the following commands:

gunzip perl-5.8.8.tar.gz

tar xf perl-5.8.8.tar

and wait the end of decompression. If you don't have the gunzip decompression program, you can download gzip bundle from http://www.gnu.org.The next step is to launch the configure program of the Perl bundle. The installation shell script will examine your system and ask you questions to determine how perl5 package should be installed. If you want to bypass all the questions and use the computed defaults or the previous answers if there was already a config.sh file, you can run "Configure –d":

cd perl-5.8.8

sh Configure –d

After the Perl configuration step, you must build Perl by typing in the following command:

make

After this step, you can verify Perl's installation, running again the following command at the command prompt:

perl –v

Install Perl on Windows (32 and 64 bit)

Perl on Windows (Win32 and Win64)

Strawberry Perl is an open source binary distribution of Perl for the Windows operating system. It includes a compiler and pre-installed modules that offer the ability to install XS CPAN modules directly from CPAN. cpanm can be installed by running cpan App::cpanminus.

ActiveState provide a binary distribution of Perl (for many platforms), as well as their own the Perl Package Manager (PPM). Some modules are not available as ppm's or have reported errors on the ppm build system, this does not mean they do not work. You to use the cpan script to build modules from CPAN against ActiveState Perl

The free ActivePerl binary distribution includes core Perl, popular modules, the Perl Package Manager (PPM), and complete documentation. You can download it from "http://www.activestate.com" and install it on your computer – the distribution is self-installing and the installation process is just the same as the one you used for other similar windows applications.

ActivePerl

Glavni izvor ActivePerl distribucije je sajt http://www.activestate.com/ koji uključuje:

Perl for Win32

Binary for the core Perl distribution

Perl for ISAPI

IIS plug-in for use with ISAPI-compliant web server

PerlScript

ActiveX scripting engine

Perl Package Manager

Manager for Perl modules and extension

Perl interpreter

Perl izvršna datoteka, obično instalirana u **/usr/bin** ili **/usr/local/bin** ili **/usr/share/man/man1** na računaru, naziva se Perl interpreter.

```
[root@localhost~] # whereis perl
```

```
perl: /usr/bin/perl /usr/share/man/man1/perl.1.gz
```

Svaki Perl program mora proći kroz Perl interpreter kako bi se mogao izvršiti.Prva linija u mnogim Perl programa je nešto poput:

```
#!/usr/bin/perl
```

Za Unix sistem, ovaj #! (hash-bang ili shebang) linija kaže ljusci da traži /usr/bin/ perl program i onda mu predaje ostatak programa za izvršenje. Ponekad se može vidjeti put (engl. pathname) za Perl kao /usr/local/bin/perl. Može se vidjeti perl5 umjesto Perl na mjestima koja još uvijek ovise o starijim verzijama Perl.

Dakle, šta Perl interpreter radi? Ona interno prevodi program, a zatim ga odmah izvršava. Perl je obično poznat kao interpretirani jezik, ali to nije strogo istinito.

Budući da interpreter zapravo prevodi program u bytecode kod prije njegovog izvršenja, često se naziva interpreter /prevodilac. Iako se prevedeni program ne pohranjuje kao datoteka, Perl verzija 5.,005 uključuje radni verziju samostalnog Perl prevodioca.

Kada se napiše Perl program, treba dati točnu #! liniju na vrhu scipta, čime postaje izvršni program pomoću chmod +x naredbe i onda ga pokrećete. Za 95% Perl programera u ovom svijetu, to je sve što je potrebno.

Perl Editors available on different platforms

To choose an editor from **Perl Editors** list available on different platforms, you must consider that Perl scripts are just plain text files so you can use any plain text editor in order to create them. Your operating system, doesn't matter if it is Windows, Unix or another one, offers you the possibility to select between them your favorite one.

Keep in mind, though, that because you must indent or unindent some blocks of code in order to make your script more readable, the most likely way is to use a **programmers' text**

editor, available on any operating system and which offer you the possibility to highlight the text, expand or collapse subroutines, menus and so on.

It is not recommendable at all to use a word processor for your code, but if you do this, be sure to save your script files as "text only", otherwise you’ll be unpleasantly surprised to see some awkward characters inserted into your code that you don’t want to be there.

And don’t forget that Perl is deeply portable, so you have the possibility to transfer the program code from one machine to another, but if you do this, do not forget to use "text" or "ASCII" mode instead of "binary" mode.

Windows – which provides WordPad and Notepad, but as we mentioned before, they lack many features for editing code so you can use nedit,UltraEdit or PFE (Programmer’s Favorite Editor) instead;

Unix – the most popular programmers’s editors are emacs and vi (with their several successors like vim or vile) which, in some versions, offer a graphical interface;

Mac OS X – with the BBEdit (this one is a popular commercial text editor considered by many developers as best available) and Alpha;

Padre, the Perl IDE (http://padre.perlide.org/)

Padre is a Perl IDE, an integrated development environment, or in other words a text editor that is simple to use for new Perl programmers but also supports large multi-lingual and multi-technology projects.

The first Perl program

We’re about to create our first, simple Perl script: a "hello world" program. There are a couple of things one should know in advance:

Perl programs (or scripts --- the words are interchangeable) consist of a series of statements

When you run the program, each statement is executed in turn, from the top of your script to the bottom. (There are two special cases where this doesn’t occur, one of which is subroutine declaration)

Each statement ends in a semi-colon

Statements can flow over several lines

Whitespace (spaces, tabs and newlines) is ignored in most places in a Perl script.

Now, just for practice, open a file called **hello.pl in** your editor. Type in the following one-line Perl **program:**

```
print "Hello world!\n";
```

This one-line program calls the print function with a single parameter, the string literal "Hello world!" followed by a newline character.

Save it and exit.

Running a Perl program from the command line

We can run the program from the command line by typing in:

```
% perl hello.pl
```

You should see this output:

```
Hello world!
```

This program should, of course, be entirely self-explanatory. The only thing you really need to note is the \n ("backslash N") which denotes a new line. If you are familiar with the C programming language, you’ll be pleased to know that Perl uses the same notation to represent characters such as newlines, tabs, and bells as does C.

Executing code

There are a lot of ways of running Perl, it depends on what you intend to do and the platform on which Perl is installed. Before playing with Perl, you must check if you have a proper Perl version installed.

For this, you may run the following command from a command prompt:

perl –v which will tell you if Perl is installed and if so, what version it is.

Now let’s see a few ways of running Perl and what possibilities we have in view. The simplest way for running Perl is from a command line as in the following example:

```
perl -e "print qq [Hello world!\n]"
```

This simple command will print on the screen the classic welcome message "Hello world!". As you see in our example, the entire program code was included between double quotes. We can do this only with very small programs which we can include into a command line. It is very useful if we want to verify how a short sample of Perl code works.

Osim definisanja #! linije, možete odrediti kratku skriptu direktno na komandnoj liniji. Ovdje su neki od mogućih načina da se pokretanja Perla:

Izdati naredbu perl, napisati skripte liniju po liniju putem e-prekidača na komandnoj liniji:

```
perl -e 'print "Hello, world\n"' #Unix
```

```
perl -e "print \"Hello, world\n\"" #Win32
```

Izdati naredbu perl, zatim ime svoje Perl skripte kao prvi parametar (nakon svih prekidača):

```
perl testpgm
```

On Unix systems that support the #! notation, specify the Perl command on the #! line, make your script executable, and invoke it from the shell (as described above).

Pass your script to Perl via standard input. For example, under Unix:

```
echo "print 'Hello, world'" | perl –
```

Another way for running Perl is to use for the program line of code the standard input stream like in the example below:

```
>perl

print "Hello World!\n";

$pi=3.14159;

print $pi;
```

^Z Note that the last character of the program is Ctrl/Z, which is the indicator for the End of File. This example has the inconvenience that your program code is saved only temporally and will be lost after the execution. You can use this method if you want to verify some code samples or play a bit with Perl.

Example

Create a very simple example of a Perl script, Hello.pl.

There are a few elements every PERL script must contain in order to function.

Open up your favorite simple text editor, the file extension for PERL scripts is .pl.

Save your files with this extension (Hello.pl).

```
#!/usr/bin/perl

print "content-type: text/html \n\n";

print "Hello, Perl!";
```

Display:

```
Hello, Perl!
```

The first line of every PERL script is a commented line directed toward the PERL interpreter, usually in Linux **/usr/bin/perl**.

```
#!/usr/bin/perl
```

We have to introduce some HTTP headers so that Perl understands we are working with a web browser. To do this we have to run another line of strange code called an HTTP header:

```
print "content-type: text/html \n\n";
```

Now that we have located the interpreter and told PERL we are working with the web, we can print text to the browser with the print function.

```
print "Hello, Perl!";
```

You should see "Hello, Perl!" in the top left corner of your browser. If you have a problem, ask about it on the Perl Forum

The "shebang" line for Unix

Unix and Unix-like operating systems do not automatically realise that a Perl script (which is just atext file after all) should be executable. As a result, we have ask the operating system to change the file’s permissions to allow execution:

```
% chmod +x hello.pl
```

Once the file is executable we also need to tell Unix how to execute the program. This allows the operating system to have many executable programs written in different scripting languages.

We tell the operating system to use the Perl interpreter by adding a "shebang" line (called such because the # is a "hash" sign, and the ! is referred to as a "bang", hence "hashbang" or "shebang").

```
#!usr/bin/perl
```

Of course, if the Perl interpreter were somewhere else on our system, we’d have to change the shebang line to reflect that This allows us to run our scripts just by typing:

```
% ./hello.pl
```

For security purposes, many Unix and Unix-like systems do not include your current directory in those which are searched for commands, by default. This means that if you try to invoke your script by typing:

```
% hello.pl # this doesn’t work
```

you’ll get the error: bash: hello.pl: command not found. This is why we prepend our command with the current working directory (./hello.pl).

The "shebang" line for non-Unixes

It’s always considered a good idea for all Perl programs to contain a shebang line. This is helpful because it allows us to include command line options, which we’ll cover shortly.

If your program will only ever be run on your single operating system then you can use the line:

```
#!perl
```

However it is considered good practice to use the traditional:

```
#!/usr/bin/perl
```

as this assists with cross-platform portability.

Command line options and warnings

Perl has a number of command line options, which you can specify on the command line by typing Perl options hello.pl or which you can include in the shebang line. The most commonly used option is -w to turn on warnings:

```
#!/usr/bin/perl -w
```

It’s always a good idea to turn on warnings while you’re developing code, and often once your code has gone into production, too.

Lexical warnings

In Perl versions 5.6 and above you can use Perl’s warnings pragma rather than the command lineswitch if you prefer. This also gives you the option to specify which warnings you wish to receive,and to upgrade those warnings to exceptions if necessary.

```
#!/usr/bin/perl

use warnings;
```

Specifying your version

To find the version of Perl available on your machine you can use the perl -v command. This will

print something like:

$ perl -v

This is perl, v5.10.1 (*) built for i486-linux-gnu-thread-multi (with 56 registered patches, see perl -V for more detail) If you wish to take advantage of the new features that your Perl version provides you can tell Perl to do so by specifying the minimum version of Perl your code is targeting:

```
#!/usr/bin/perl

use v5.10.1;

use strict;

use warnings;
```

In versions 5.12 and above, this will also turn on strictures for us automatically.

Comments

Comments in Perl start with a hash sign (#), either on a line on their own or after a statement.

Anything after a hash is a comment up to the end of the line.

```
#!/usr/bin/perl -w

# This is a hello world program

print "Hello world!\n"; # print the message
```

Block comments

To comment a block of text (or code) you can use Perl’s Plain Old Documentation tags (POD). You can read more about POD in perldoc perlpod.

=begin comment

This content is commented out.

It may span many lines.

print "This statement won’t be executed by Perl\n";

=end comment

=cut

print "Hello world!\n"; # print the message

The next steps include the uploading (generally through a FTP client) of your script on your web server in the cgi-bin directory and the setting up of the script permissions. After that, you can create a simple web page to run the script, by creating the file Hello.html and running it from your favorite browser.

Hello.html

<html>

<head>

<title>Run the Hello.pl script</title>

</head>

<body>

< a href = “http://www.microsoft.comwebserver.com/cgi-bin/Hello.pl”> Click this link to run the script </a>

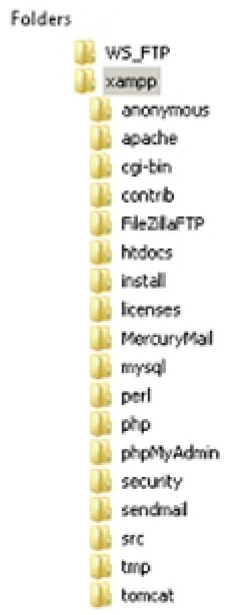
</body>

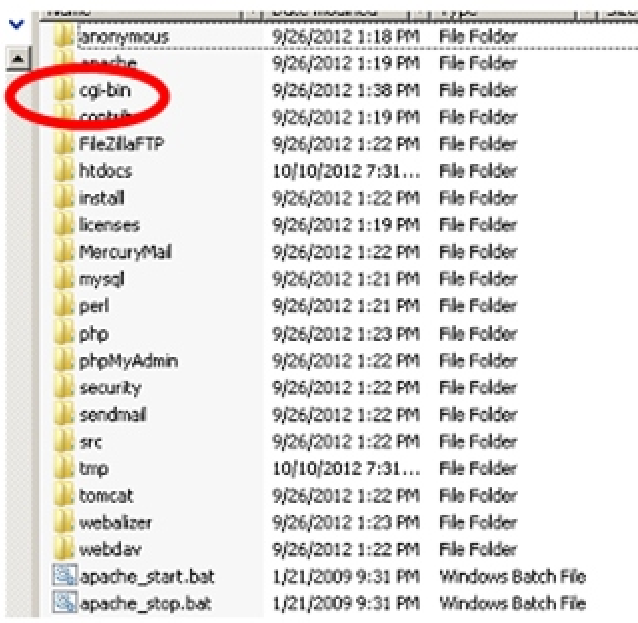
</html>

Xamp distribucija

XAMPP jel Apache distribucija koja uključuje MySQL, PHP and Perl.

http://www.apachefriends.org/en/xampp.html

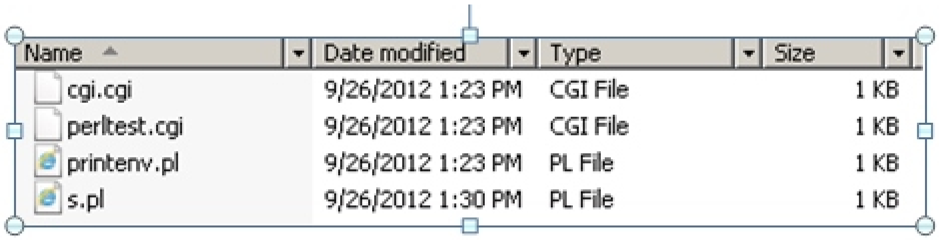




```
anonymous          9/26/2012 1:18 PM    File Folder
apache             9/26/2012 1:19 PM    File Folder
cgi-bin            9/26/2012 1:38 PM    File Folder
contrib            9/26/2012 1:19 PM    File Folder
FileZillaFTP       9/26/2012 1:22 PM    File Folder
htdocs             10/10/2012 7:31...   File Folder
install            9/26/2012 1:22 PM    File Folder
licenses           9/26/2012 1:19 PM    File Folder
MercuryMail        9/26/2012 1:22 PM    File Folder
mysql              9/26/2012 1:21 PM    File Folder
perl               9/26/2012 1:21 PM    File Folder
php                9/26/2012 1:23 PM    File Folder
phpMyAdmin         9/26/2012 1:22 PM    File Folder
security           9/26/2012 1:22 PM    File Folder
sendmail           9/26/2012 1:22 PM    File Folder
src                9/26/2012 1:22 PM    File Folder
tmp                10/10/2012 7:31...   File Folder
tomcat             9/26/2012 1:22 PM    File Folder
webalizer          9/26/2012 1:23 PM    File Folder
webdav             9/26/2012 1:22 PM    File Folder
apache_start.bat   1/21/2009 9:31 PM    Windows Batch File
apache_stop.bat    1/21/2009 9:31 PM    Windows Batch File
```

Slika 1 Datoteka xampp\cgi-bin

Datoteka **xampp\cgi-bin** i **s.pl** program.



Name	Date modified	Type	Size
cgi.cgi	9/26/2012 1:23 PM	CGI File	1 KB
perltest.cgi	9/26/2012 1:23 PM	CGI File	1 KB
printenv.pl	9/26/2012 1:23 PM	PL File	1 KB
s.pl	9/26/2012 1:30 PM	PL File	1 KB

Slika 2 Perl program s.pl

Example

Program s.pl

```
#!c://perl_active/bin/perl
print "Content-type: text/plain;
print "Hello, world";
http://localhost/cgi-bin/s.pl
```