Perl - sjajno rešenje za analitičke probleme

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Aleksandar Vračarević, Jovan Marić,
Petar Mićić, Tatjana Radovanović
vracarevicaleksandar@gmail.com, maric993@hotmail.com,
micicpetar73@gmail.com, tatjana.tasa.95@gmail.com

6. april 2019

Sažetak

Cilj ovog rada je da čitaocu, kroz primere koda i pregled glavnih namena programskog jezika Perl, izgradi prvi utisak o jeziku koji je zbog svoje jednostavnosti i čiste sintakse, ali i mnogih mehanizama koji su u njega ugrađeni, uticao na razvoj mnogih jezika. U ovom radu biće prikazano kako se Perl razvijao kroz istoriju, kao i zašto se mnogi programeri opredeljuju za njega.

Sadržaj

1 Uvod

Većina programskih jezika je osmišljena sa ciljem da reši određenu grupu problema, tako je i programski jezik Perl nastao da reši neke česte probleme pri razvoju softvera. Kao takav programski jezik Perl je stekao veliku popularnost i ima širok domen primene. Perl spada u jezike koji se lako uče jer imaju jednostavnu sintaksu nalik na prirodne jezike. Uz podršku velikog broja modula, Perl programi ostvaruju obimnu funkcionalnost uz minimalan skup naredbi. Iako su njegovu popularnost preoteli moderniji programski jezici, i dalje predstavlja ozbiljno oružje kod veb programiranja. U ovom radu ćemo napraviti pregled osnovnih mogućnosti Perl-a, njegovih karakteristika i specifičnosti. Navešćemo popularna okruženja koja Perl koristi i uputstvo za instalaciju.

2 Nastanak i razvoj

Priču o Perlu ne možemo a da ne započnemo sa njegovim stvaraocem Lerijem Volom (eng. *Larry Wall*). Kako sam naglašava u tom trenutku postoje i druga rešenja ali nijedno od njih ne rešava ovu vrstu problema na lak i intuitivan način. Zbog prirode problema nameće se jezik koji će pripadati skript paradigmi. Želeo je da naziv bude kratka reč pozitivne konotacije. Opredeljuje se za pearl(srb. *biser*) ali izostavlja slovo a zbog mogućeg podudaranja sa drugim jezikom u izradi.(Larry Wall, the Guru of Perl, Linux Journal 1. maj 1999)

Perl 1.0 postaje dostupan 18. decembra 1987. godine korisnicima Usneta [1], u sekciji posvećenoj računarima. Jedna od ideja vodilja prilikom izrade ovog programa bilo je da, kao i bilo koji drugi jezik, raste i evoluira. Ovakav pristup se ogleda u brojnim izborima koje Vol donosi u fazi projektovanja. Kao primer navodimo da je moguće dotati nove ključne reči u bilo kom trenutku bez narušavanja starih kodova [15]. Još jedna osobina ljudskih jezika je da nije potrebno poznavanje jezika u celini radi efikasnog korisćenja, što važi i za Perl. Ovakve odluke naravno imaju i svoje posledice na performanse koje su za Vola prihvatljive jer on pre svega pokušava da olakša korišćenje programa njegovim korisnicima.

Verzija 4.0 prati izdavanje knjige Programing Perl(Randal Schwartz, Larry Wall, 1991). Ovo izdanje se smatra prvom kompletnom dokumentacijom za Perl. Na svojoj naslovnoj strani ima kamilu koja je postala sinonim za Perl u svetu programiranja[3].

Početak dvadest prvog veka obeležen je nastavkom evoluiranja Perla koji se prilagođava svim potrebama modernog programiranja. Podrška za mrežno programiranje kao i za implementaciju funkcionalne paradigme samo su neki od primera. Sve popularniji pretrazivac DuckDuckGo većinski je implementiran u Perlu[4].

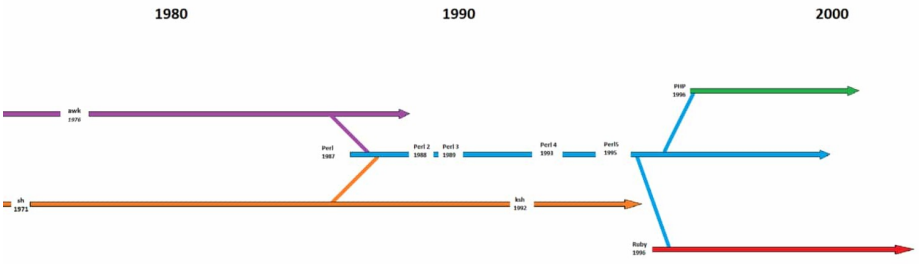
[1] Preteča internta na Unix sistemima za komunikaciju širom sveta
[2] U vidu knjige Programing Perl Randal Schwartz, Larry Wall, 1991

Verzija	Datum	Novine
2.0	05.06.1988	-unapređenje regex funkcija -rekurzivne subrutine -blokovske promenljive -foreach petlja -sort operator
3.0	18.10.1989	-podrška za rad u binarnom zapisu -prosleđivanje promenljivih referencom -nove sistemske funkcije
4.0	21.03.1991	-otklanjanje manjih bagova -objedinjena dokumentacija[2]
5.0	18.10.1994	-novi interpretator -objekti -nove funkcije
5.6.0	22.03.2000	-podrška za utf-8 -podrška za 64-bitni sistem -nadogradnja threadova

Tabela 1: Pregled osnovnih doprinosa različitih verzija Perl-a

2.1 Mesto u razvojnom stablu i uticaji drugih programskih jezika

Ako u obzir uzmemo činjenicu šta je osnovna namena ovog progrmskog jezika neće nas iznenaditi da u Perlu možemo naći uticaj šel skripte(eng. Shell script), AWK, sed alata, kao i programskog jezika C kao glavnog predstavnika proceduralne paradigme tog vremena.[8]



Slika 1: Drvo razvoja

Iako su počeci Perla mnogo prethodili internetu konstantnim evoluiranjem nikada nije prestao da bude relevantan. Verzija 5.004 donosi nam modul koji obezbeđuje API za pisanje web aplikacija. Krajem devedesetih nazivan je lepkom koji održava internet. Neće nas iznenaditi da u razvojnom stablu programskih jezika zauzima mesto pretka PHP-a.
Sredinom poslednje decenije dvadesetog veka Yukihiro Matsumoto želi da

objedini moć tekstualnog procesiranja Perla sa Pythonom. Godine 1993. objavljuje programski jezik Ruby.

3 Osnovna namena, svrhe i mogućnosti

Glavna snaga Perla ogleda se u radu sa regularnim izrazima. Tri glavne operacije su uparivanje (eng. matching), zamena uparene niske drugom i prevođenje koje menja karaktere iz liste sa odgovarajućim karakterima iz liste zamena[15]. Postoje brojne modifikacije regex operatora koje olakšavaju korišćenje i pružaju veliki broj opcija programeru. Na usluzi programeru nalaze se i četiri promenljive u kojima se čuvaju rezulatati poslednjeg uparivanja. Imamo pristup sledećim informacijama: da li je došlo do uparivanja, uparenu nisku, nisku pre uparene i nisku posle[8]. Kada drugi programski jezici predstavljaju svoje mogućnosti vezane za parsiranje teksta i regularne izraze uglavnom to čine poredeći se upravo sa Perlom.
Bezbednost je jedna od najznačajnih komponenti svakog programa. Od naglog razvoja mrežnog programiranja bezbednost eksponencijalno dobija na značaju. Već smo pomenuli da Perl pronalazimo kao bekend(eng *back end*) skripting jezik u mrežnom programiranju. Veliki razlog za to je tejnted mod rada(eng. *Tainted*). Kada Perl operiše u ovom režimu rada svakom podatku koji dolazi od korisnika ili iz okruženja pokretanja, dodaje se malo meta podataka koji ukazuje da ti podaci mogu potencijalno biti problematični. U svakom trenutku u programu se može proveriti poreklo podataka i izdvojiti siguran deo istog[10].
Kao odgovor na sve veću potražnju objektno orijentisanih jezika Perl nudi biblioteku Moose. Moose nudi osnovne mogućnosti kao što su pravljenje klasa, metoda, atributa ali i podržava koncepte kao sto su nasleđivanje i preopterećivanje metoda.

4 Osnovne osobine

Prvobitna zamisao autora Perl-a, Lerija Vola, bila je da napravi alat koji bi na brz i lak način omogućio napredno rukovanje proizvoljnim tekstualnim datotekama. Jezik je osmišljen sa idejom da se veći značaj prida njegovoj praktičnosti(lakoća korišćenja, efikasnost, kompletnost) nego lepoti(minimalnost, elegancija). Autor smatra da je implementacijom Perla uspeo da izdvoji i iskoristi najbolje funkcionalnosti jezika C, awk, sed i sh[13].
Sintaksa je usko povezana sa sintaksom C-a. Umesto da, kao većina Unix alata, ograničava memoriju, Perl je sposoban da čitav fajl učita kao jednu nisku(ukoliko je dovoljno memorije raspoloživo). Pažljivo osmišljeni mehanizmi poklapanja obrazaca (eng. *pattern matching*) programeru pružaju veoma moćan alat za brzo čitanje velikih datoteka. Osim za obradu teksta, Perl je tokom godina postao odličan alat za obavljanje raznih sistemskih zadataka, veb programiranje, grafičko programiranje, pristup bazama podataka, mrežno programiranje i mnoge druge zadatke.

Bitno je još i naglasiti da je Perl interpretirani jezik što omogućava prenosivost jednom napisanih kodova na veliki broj različitih platformi[3]. Za razliku od strogo (eng. *strictly*) interpretiranih jezika, Perl program

[3]Kompletan spisak podržanih platformi može se naći na sledećoj adresi.

se najpre prevodi u sintaksno drvo i nakon toga se izvršava (više reči o korišćenju Perl interpretatora u 9).

4.1 Tipovi podataka

Postoji tri tipa ugrađenih podataka: skalari, nizovi skalara i asocijativni nizovi skalara, poznati i kao heševi (eng. *hashes*). Obični nizovi su uređeni i indeksirani brojevima počevši od nule, dok su heševi neuređene kolekcije skalarnih vrednosti indeksirane dodeljenim niskama (ključevima). Skalarne vrednosti mogu biti brojevi, niske i reference uz mogućnost lake i transparentne konverzije između ove tri varijante[4]. Činjenica da Perl dinamički alocira memoriju omogućava pisanje programa otpornih na greške programera poput prekoračenja bafera ili pristupa nedozvoljenoj memoriji. Interpretacija operatora i vrednosti ponekad zavisi od okolnog konteksta. Postoje dva osnovna konteksta: kontekst liste i kontekst skalara[2].

Primer 4.1 *U narednom primeru videćemo da se ista promenljiva može tumačiti na dva različita načina u zavisnosti od konteksta.*

```
my @niz = ('Zdravo ', 'svete ', ':D!')
# @niz se tumaci u kontekstu liste:
# promenljiva @elementi sadrzi kopiju
    vrednosti promenljive @niz
my @elementi = @niz
# @niz se tumaci u kontekstu skalara:
# promenljva $broj_elemenata sadrzi broj
    elemenata promenljive @niz
my $broj_elemenata = @niz
```

5 Šta je ono što izdvaja Perl?

Velika Perl Arhiva[1] (eng. *Comprenhensive Perl Archive Network(CPAN)*) predstavlja online skladište Perl programa, dokumentacije i modula koji olakšavaju implementaciju i održavanje Perl projekata. Arhiva čuva oko 180 hiljada modula otvorenog koda napisanih od preko 12 hiljada programera.

Glavni zadatak CPAN-a je da omogući programerima da pronađu module koji nisu uključeni u paket osnovnih modula. Paket osnovnih modula se dobija instalacijom Perl kompajlera. Modul pod imenom CPAN.pm omogućava lako preuzimanje i instaliranje drugih Perl modula.

5.1 Parsiranje tekstualnog sadržaja

Prve verzije Perl programskog jezika su u fokus stavljale obradu i manipulaciju tekstualnog sadržaja. Sama skraćenica PERL (Practical Extraction and Reporting Language) prikazuje osnovne namene jezika, a to su izvlačenje nekog sadržaja i generisanje izveštaja na osnovu tog sadržaja.

Naredni primer prikazuje kako na jednostavan način možemo izračunati broj pojavljivanja svake reči u tekstu, pri čemu se za reč smatra niska karaktera koja se sastoji samo od malih i velikih slova abecede.

[4] Iako skalar ne može neposredno sadržati više od jedne vrednosti, moguće je dodeliti mu referencu na niz vrednosti.

```
while (my $line = <$fh>) {
    chomp $line;
    foreach my $str (split /^[A-Za-z]+$/, $line
    ) {
        $count{$str}++;
     }
}
```

Dakle, prvo prolazimo kroz fajl (predstavlja referencu na fajl koji smo prethodno otvorili) liniju po liniju, zatim iz linije uklanjamo znak za novi red, a potom za svaku reč (smeštenu u promenljivoj $str) koja odgovara navedenom regularnom izrazu, ažuriramo strukturu u kojoj čuvamo reč i broj izračunatih pojavljivanja. Rezultat rada navedenog dela koda je smešten u promenljivoj $count [12].

5.2 Jednostavno korišćenje regularnih izraza

Važnu ulogu pri manipulaciji tekstualnog sadržaja igraju regularni izrazi. Regularni izrazi su šabloni koji mogu biti prepoznati u nizovima karaktera. Rezultat rada operacije prepoznavanja šablona jeste tačno ili netačno, odnosno da li niz karaktera odgovara datom šablonu ili ne. Perl ima ugradjene mehanizme koji čine rad sa regularnim izrazima lakšim, nalik na alate grep, sed i awk.[7]

Osnovni operatori primene regularnih izraza, koje Perl nudi, jesu $=\sim$ i $!\sim$. U izrazu $str $=\sim$ m/regex/ se proverava da li se data niska $str podudara sa regularnim izrazom regex. Regularni izrazi se navode između dve kose crte. U slučaju da je došlo do poklapanja, izraz vraća 1, a nula inače. Operator $!\sim$ radi suprotno, u slučaju podudaranja vraća 0, a 1 inače.

Postoje tri osnovne operacije sa regularnim izrazima:

- Provera podudaranja - m/regex/
- Zamena reči - s/regex/word/
- Translacija - tr/regex/word/

Zamena reči funkcioniše tako što se sva podudaranja regularnog izraza u nisci zamene navedenom niskom. Na primer u izrazu "Volim mačke!"= s/mačke/pse, deo niske mačke će biti zamenjen sa pse, i novodobijena niska predstavlja rezultat izraza.

Translacija je slična zameni reči, sa tim što ne koristi regularne izraze prilikom pretrage niske.

6 Podržane paradigme

Izjava Lerija Vola da ”Perl nema nikakvu agendu, osim da bude maksimalno korisan maksimalnom broju ljudi.”[14] oslikava težnju Perla da bude što opštiji jezik. Ta tvrdnja je potkrepljena i time što su podržane proceduralna, funkcionalna i objektno-orijentisana paradigma.

6.1 Proceduralna paradigma

Kao što je već pomenuto u prethodnim poglavljima 3 i 4, uticaj alata awk, sed i programskog jezika C je primetan u Perlu pa se proceduralni način pisanja programa prirodno nameće. Više primera se može naći u poglavlju 8.

6.2 Funkcionalna paradigma

Perl je jezik višeg reda (eng. *higher-order*) koji implementira koncept dinamičkog pravljenja funkcija i prosleđivanja istih drugim funkcijama. Dovoljno je moćan da λ račun izrazi direktno, u samom jeziku, bez potrebe za pisanjem posebnog programa za parsiranje i evaluaciju izraza[9, 5]. Funkciju sa parametrom x i telom B, $\lambda_x.B$ u Perlu možemo predstaviti kao `sub {my $x = shift; B}`. Primena funkcije P na funkciju Q u lambda računu je jednostavno (PQ), dok je ekvivalentan zapis u Perlu `$P->($Q)`.

U sledećem primeru se vidi kako se može napraviti funkcija koja primenjuje prosleđenu funkciju na neki niz element po element.

```
sub mapfunkcija{
        my (@rez_lista);
        my ($funkcija, @lista) = @_;
        for (my $i=0;$i<=$#list;$i++)
        {
                $rez_lista[$i] = $funkcija->(
    $lista[$i]);
        }
        return @rez_lista;
};
my @niz = (65, -42, -92);
my $mapme = sub {
        my $x = $_[0];
        return $x + 42;
};
my @rezultat = mapfun ($mapme, @niz);
```

Nakon izvršavanja ovog kôda, u promenljivoj `@rezultat` se nalaze brojevi 107, 0 i -50.

6.3 Objektno-orijentisana paradigma

Iako Perl nije objektno-orijentisan jezik, može instancirati i manipulisati objektima. Klase su predstavljene paketima, koji moraju posedovati `new` podrutinu da bi se mogli instancirati. Objekat neke klase se može predstaviti uz pomoć niza ili heša. Enkapsulacija se postiže izdvajanjem interfejsa u pakete. Mehanizam za postizanje polimorfizma je korišćenje ključne reči `bless`. Ova ključna reč običnu referencu na strukturu podataka označava kao pripadnika nekom paketu i time proširuje mogućnosti te strukture. Nasleđivanje se ostvaruje pomoću specijalnog niza `@ISA`. Ovaj niz čuva spisak imena modula koji se nasleđuju. U narednom primeru biće prikazani neki od ovih koncepata.[11]

```
package Student;
sub new{            # podrutina za
    instanciranje klase
     my ($ime, $indeks, $prosek) = @_;
     my $ref_stud = {
          "ime"    =>    $ime,
          "indeks" => $indeks,
          "prosek" => $prosek,
     };
     bless $ref_stud, 'Student'; # objekat se
    proglasava instancom klase Student
     return $ref_stud;
}
# doseg prethodne klase je ili do kraja
    datoteke, ili do sledece package kljucne
    reci
```

```
package Profesor;
sub new{            # podrutina za
    instanciranje klase
     my ($ime, $plata, $fakultet) = @_;
     my $ref_prof = {
          "ime"        =>      $ime,
          "plata       =>    $plata,
          "fakultet"   => $fakultet,
     };
     bless $ref_prof, 'Profesor'; # objekat se
    proglasava instancom klase Profesor
     return $ref_prof;
}
# instanciranje objekata klase Student i
    Profesor
$laza_lazic = Student::new('Laza Lazic',
    123456, 9.0);
$mika_mikic = Profesor::new('Mika Mikic',
    100000, 'Matematicki fakultet');
```

7 Najpoznatija okruženja i njihove karakteristike

Perl je stekao veliku popularnost u ranim danima veba i često se opisuje kao lepak koji održava internet. Stoga će u ovom poglavlju biti opisana dva okruženja za razvoj veb aplikacija uz dodatak već pomenutog Moose okruženja namenjenog da olakša upotrebu Perla u objektno orijentisanom stilu.

7.1 Catalyst

Catalyst je jedno od najrasprostranjenijih Perl okruženja za razvoj veb aplikacija zasnovano na MVC (eng. *Model View Controller*) arhitekturi. Ovo okruženje pruža nivo apstrakcije nad uobičajenim ciklusom zahteva i odgovora. Zahtevi omogućavaju lako gledanje argumenata nadolazećih upita, POST podataka, otpremanje datoteka i zaglavlja dok odgovori pružaju mogućnost postavljanja zaglavlja i vraćanja izlaza klijentu.

7.2 Dancer

Dancer je okvir za pisanje veb aplikacija veoma intuitivne i izražajne sintakse. Modelovan je po uzoru na Ruby radni okvir, Sinatra. Aplikacije se grade navođenjem HTTP glagola, URL-ova (putanja) i metoda za obradu saobraćaja ka tim konkretnim URL-ovima.

```
use Dancer2;
# HTTP glagol GET nakon cega sledi koreni URL
    '\',
# i na kraju anonimna podrutina koja vraca
    nisku
get '/' => sub {
  return 'Zdravo svete!';
};
start;
# Ukoliko bi se ovaj primer pokrenuo na
    racunaru, niska 'Zdravo svete!' bi se
    odstampala kada se veb pregledac uputi na
    adresu http://localhost:3000
```

7.3 Moose

Iako Perl ima podršku za objektno orijentisanu paradigmu, Moose pokušava da pisanje OO programa učini još lakšim i da programerima pruži jednostavnu deklarativnu sintaksu koja bi eliminisala potrebu za pisanjem konstruktora, destruktora i pristupnih metoda. U pozadini se i dalje dešava nešto slično onome opisanom u , ali je programer zaštićen od tih detalja implementacije i pruža mu se veći stepen apstrakcije.

```
package Student;
# nakon navodjenja sledece linije, paket
    postaje klasa
use Moose;
has 'ime' => (
    is => 'rw',      # pristupna funkcija koja
    moze da cita i upisuje vrednosti u atribut
    ime
    isa => 'Str',    # naglasavamo da atribut
    ime moze da prima samo niske
);
has 'neki_atribut' => (...)
# instanciranje objekta
use Student;
my student = Student->new(
    ime => 'mika mikic',
    neki_atribut => 'neka vrednost', ...
);
```

8 Primeri koda

Sledeći kod predstavlja primer ispisa Hello world programa. Komentare pišemo iza tarabe. Tekst možemo ispisati tako što ga navedemo iza navodnika ili koristeći funkciju qq.

```
#Komentar
print "Hello, world!\n";
print qq =Did you say "Hello"?\n=;
```

Naredni primer demonstrira program koji računa zbir dva broja. Imena promenljivih počinju znakom $. U komentarima je dato šta koja print naredba ispisuje.

```
$a = 5;
$b = 4;
print qq =a \= $a\n=; # a = 5
print qq =b \= $b\n=; # b = 4
$zbir = $a + $b;
print qq=a + b \= $zbir\n=; #a + b = 9
```

Nazivi nizova u Perlu su u formatu @naziv_niza. Niz se može inicijalizovati tako što elemete navedemo u zagradama, razdvojene zarezom. Ako želimo da ispišemo sve elemete niza, to možemo učiniti samo navođenjem imena niza. Određenom članu niza pristupamo tako što njegov indeks navedemo u uglastim zagradama iza naziva niza. U ovom slučaju ime niza može počinjati i znakom @ i znakom $. Indeks poslednjeg člana niza dobijamo tako što ispred imena niza stavimo $#.

```
@dan = ("Danas", "je", "lep", "dan");
$broj_reci = @dan; #Broje elementa niza
print "broj reci u recenici \"@dan\" je
    $broj_reci.\n";
print "@dan[2], $dan[3]\n";
print "Indeks posednjeg elementa je $#dan\n";
```

Grananje u programu možemo izvršiti koristeći if-else naredbu. Naredni program prvo učitava broj sa standardnog ulaza, za zatim proverava da li je pozitivan ili je negativan. Uslov se navodi u zagradi, a if i else blok se navode između vitičastih zagrada.

```
$broj = <STDIN>;
if($broj >= 0){
    print "Broj je pozitivan\n";
}
else{
    print "Broj je negativan\n";
}
```

Možemo zadavati i argumente komandne linije. Argumentima komandne linije pristupamo isto kao i elementima niza, samo što kao naziv niza navodimo ARGV. Broj argumenata komandne linije možemo dobiti tako što indeks poslednjeg elementa uvećamo za jedan.

```
$br_arg = $#ARGV + 1;
print "Broj argumenata komandne linije je
   $br_arg: @ARGV\n";
```

Sledećim kodom je prikazan primer korišćenja while petlje. Program prihvata argumente sa ulaza i ispisuje ih na ekran. Ključna reč my označava da je domen promenljive samo taj blok. Promenljiva $in u while petlji nema uticaj na promenljivu sa istim nazivom van petlje.

```
$in = 5;
while(my $in = <>) {
    print $in; #Ispisuje element koji smo
   uneli
}
print $in; #Ispisuje broj 5
```

U Perlu možemo koristiti i for petlju na sličan način kao i u programskom jeziku C.

```
for(my $i = 0; $i <= $#ARGV; $i++) {
    print "$ARGV[$i]\n";
}
```

Funkcije započinju ključnom rečju sub iza koje sledi naziv funkcije. Telo funkcije se piše između vitičastih zagrada. Argumenti se navode u telu funkcije na sledeći način my(lista_argumenata) = @_; Poziv funkcije se vrši na standardni način.

```
sub obim_kvadrata {
    my($a) = @_;
    return 4*$a;
}
```

Primer 8.1 *Sledeći kod predstavlja jednostavan način da zamenimo vrednosti dve promenljive.*

```
$aa = 3;
$bb = 2;
($aa, $bb) = ($bb, $aa);
print "$aa $bb\n";
```

9 Instaliranje i pokretanje

Da bi se uspešno pokretao perl skript potrebno je imati kompajler[5]. U ovom poglavlju biće objašnjeno njegovo instaliranje na Linux i Windows operativnim sistemima.

9.1 Instaliranje na Linux operativnom sistemu

Da bismo instalirali Perl kompajler na Linux operativnom sistemu potrebno je da u terminalu unesemo komandu `sudo apt-get install perl`. Od vas će se zatražiti da unesete lozinku. Kada se instalira Perl kompajler u terminalu pokrenite sledeću komandu `curl -L http://xrl.us/installperlnix | bash`. Kada sve bude gotovo, da biste se uverili da je instalacija uspešno izvršena ili ako želite da proverite koja je verzija Perl-a instalirana, to možete uraditi komandom `perl -v`.

9.2 Pokretanje na Linux operativnom sistemu

Pokretanje Perl skripta na Linux operativnom sistemu se radi na jednostavan način. Potrebno je otvoriti terminal, a zatim uneti komandu `perl <putanja_do_fajla>`. Ova komanda će pokrenuti izvršavanje skripta.

9.3 Instairanje na Windows operativnom sistemu

Pre instalacije na Windows operativnom sistemu poželjno je prethodno proveriti da nijedna verzija Perla već nije instalirana. Ako želite da deinstalirate prethodnu verziju idite na Control Panel - Add/Remove programs. Ako i dalje imate C:\Strawberry folder, izbrišite ga ili ga preimenujte. Skinite i instalirajte Padre, the Perl IDE/editor. Strawberry Perl je deo instalacije, ali takođe dobijate i mnoge druge korisne CPAN module. Nakon instalacije, možda ćete morati da restartujete računar. Zatim idite na Start meni/All Programs i pronađite folder sa Perlom. U njemu kliknite na Perl komandnu liniju. Da biste instalirali module iz CPAN-a u komandnoj liniji ukucajte komandu `cpan App::cpanminus`. Ukoliko želite da potvrdite da je instalacija uspešna ukucajte `perl -v`, ovom komandom, takođe, možete proveriti koja je verzija Perl-a instalirana.

9.4 Pokretanje na Windowsu

Ako imamo instaliran Padre, the Perl IDE/editor, komande za pokretanje programa možemo zadati i iz editora. U meniju izaberemo Run, a zatim Run Script, mada možemo da pritisnemo taster F5 i skript će biti pokrenut. Ako želimo, skript možemo pokrenuti i iz komandne linije tako što ukucamo `perl <putanja_do_fajla>`.

[5]Strogo govoreći, perl program nije ni kompajler ni interpretator. Naime, perl program se najpre predstavi sintaksnim drvetom koje nakon toga izvršava perlov izvršni sistem [6]. Imajući ovu činjenicu u vidu, u nastavku rada će biti korišćen naziv kompajler.

10 Zaključak

Ovaj rad prikazuje osnovne osobine programskog jezika Perl, nastanak, istorijski značaj i njegov uticaj. Upoznaje čitaoca sa sintaksom, glavnim elementima, nekim od najčešćih okruženja koje on koristi kao i osnovnim paradigmama koje on podržava. Ono što čini ovaj jezik nezamenljivim oružjem kada je u pitanju tekstualna analiza, jeste ugrađen skup bogatih mehanizama za rad sa regularnim izrazima koji ubrzavaju i olakšavaju razvoj softvera. Oblasti u kojima je Perl pogodan izbor su takođe i mrežno programiranje, programiranje baza podataka, veb programiranje i grafičko programiranje.

Literatura

[1] Comprenhensive Perl Archive Network. on-line at: https://www.cpan.org/.

[2] perldoc. https://perldoc.perl.org/perldata.html#Context. Accessed: 2019-04-03.

[3] Offical site. https://www.perl.org/camel.html. Accessed: 2019-04-03.

[4] Duckduckgo github repo. https://github.com/duckduckgo/duckduckgo. Accessed: 2019-04-03.

[5] Sultan Al-Qahtani, Pawel Pietrzynski, Luis Guzman, Rafik Arif, and Adrien Tevoedjre. Comparing Selected Criteria of Programming Languages Java, PHP, C++, Perl, Haskell, AspectJ, Ruby, COBOL, Bash Scripts and Scheme Revision 1.0. https://arxiv.org/abs/1008.3434. Accessed: 2019-04-03.

[6] Tom Christiansen, brian d foy, Larry Wall, and Jon Orwant. perlglossary. https://perldoc.perl.org/perlglossary.html. Accessed: 2019-04-05.

[7] Free Software Foundation. sed, a stream editor, 2000. on-line at: https://www.gnu.org/software/sed/manual/sed.html.

[8] Jeffrey EF Friedl. *Mastering regular expressions*. "O'Reilly Media, Inc.", 2006.

[9] Chuck Liang. Programming language concepts and perl. *J. Comput. Sci. Coll.*, 19(5):193–204, May 2004.

[10] Michael Saltzman. *Modern Perl Programmingn*. "Onyx Neon Press", 2014.

[11] Sriram Srinivasan. *Advanced PERL Programming*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition, 1997.

[12] Nathan Torkington Tom Christiansen. *Perl Cookbook*. O'Reilly, SAD, 2003.

[13] Larry Wall. The perl programming language.

[14] Larry Wall. Perl, the first postmodern computer language. https://www.perl.com/pub/1999/03/pm.html. Accessed: 2019-04-03.

[15] Tom Christiansen Wall, Larry and Jon Orwant. *Programming Perl, Third Edition*. "O'Reilly Media, Inc.", 2000.