

Februar 2015. Broj 33

# LIBRE!

Časopis o slobodnom softveru



Izgubljeni heroji „Blečli parka”

JOŠ IZDVAJAMO

**bgfx - Cross-platform rendering library**  
**To su samo metapodaci?**



Creative Commons Autorstvo-Nekomercijalno-Deliti pod istim uslovima

## Reč urednika

## Transkripcija

Primitili ste da je većina stranih reči u prethodnom broju časopisa pisana prilagođeno. Odlučili smo da počnemo sa transkripcijom. Odluci je prethodilo razmatranje o tome da li je transkripcija korisna i moguća, i kako bi naši čitaoci reagovali.

U našem jeziku ima mnogo reči stranog porekla. Mi smo ih vremenom prihvatili i prilagodili ih. Kako vreme protiče, stičemo utisak da su pojedine reči oduvek deo našeg jezika. Sledeće generacije ne moraju imati svest o tome da je neka reč stranog porekla. Primećujete da strana reč prihvatanjem postaje reč stranog porekla, a možemo ih sretati jednako često kao domaće reči. Prihvatanje reči - šta se pod tim podrazumeva? Najpre se reč ustalila u narodu i prilagodi se našem izgovoru. Dodeljujemo joj rod, broj i padež. Pišemo je našim pismom. Da li se transkripcijom reč udaljava od izvornog značenja? Odgovor je odričan. Ajnštajn (nem. *Einstein*) je poznati fizičar, Lavoazje (fr. *Lavoisier*) je hemičar, a svima su nam poznate Ezopove basne (grč. *Αἴσωπος* — *Aisōpos*). Sastavljači naših udžbenika oduvek su vršili transkripciju stranih reči. Podrazumeva se da su makar u fusnoti navedeni značenje i izvorna grafija. Transkripcija je sastavni deo našeg jezika.

S druge strane, protivnici transkripcije su mišljenja da je najbitnije optičko prepoznavanje stranih reči, a da bi transkripcija mogla da zbuni čitaoca u smislu da ne zna o čemu se radi, iako se ranije susretao s istim pojmom ali u izvorno napisanom obliku. Da je mogućnost zbunjivanja mala ukazuje višedecenijska praksa u našoj naučnoj literaturi da je strana reč kod prvog spominjanja napisana izvornim pismom. Transkripcija nije puko preslovljavanje s jednog pisma na drugi, već podrazumeva adekvatno



prenošenje zvukova i njihovo zapisivanje.

LiBRE! je strane reči do 32. broja časopisa, čak i u tekstovima na ćirilici, pisao izvornim pismom. Masivno uplitanje latinice u ćirilicu je nepraktično, a takav tekst je nepregledan. Transkribovan oblik reči nam pokazuje kako se one čitaju, te nam otklanja nedoumice. Dobar primer je ime poznatog grafičkog okruženja Mate. Reč potiče iz španskog jezika te se čita Mate, a ne Mejt. Mate je biljka koja raste u Južnoj Americi od koje se spravlja čaj. LiBRE! će slediti Pravopis i preporuke naših poznatih filologa te će većinu ustaljenih reči pisati u transkribovanom obliku. Kod prvog pisanja reči u zagradi će se navesti izvorni oblik da bi se čitaocu omogućila dalja pretraga interneta o pojmu. Kad bi transkripcija bitno otežala prepoznavanje nekog sadržaja (kao, naprimer, kod akronima), takve reči pisaćemo nadalje u izvornom obliku uz povremeno pisanje preporuke kako se reč čita. Maj-es-kju-el bi bilo nepregledno i isuviše dugačko, stoga pišemo *MySQL*. Isto važi za manje poznate nazive programa i za komande unutar programa.

Od izlaska prošlog broja nismo dobili nijednu kritiku na račun transkripcije. To bismo mogli da protumačimo kao znak da čitaoci nemaju zamerku u vezi sa transkripcijom. Dobili smo odgovore na pitanja postavljena u Reči urednika iz prethodnog broja. Ovom prilikom vam se zahvaljujemo na odgovorima. Svi vaši predlozi su dobrodošli i razmatraju se na redovnim sastancima. I dalje očekujemo vaše kritike i predloge, koje nam možete poslati na već poznatu adresu elektronske pošte [libre \[at\] lugons \[dot\] org](mailto:libre[at]lugons[dot]org).

Do sledećeg broja,

LiBRE! tim.

# Sadržaj

## Vesti

str. 6

## Predstavljamo

*bgfx - Cross-platform rendering library*

*rEFInd - Boot management*

str. 9

str. 14

## Kako da...?

Uvod u programski jezik C (9. deo)

str. 18

## Oslobađanje

U potrazi za idealnom distribucijom:

Socijalni kriterijumi za izbor

idealne distribucije (6. deo)

str. 26

## Slobodni profesionalac

*Hibernate ORM* (2. deo)

str. 31

## Internet mreže i komunikacije

To su samo metapodaci?

Izgubljeni heroji „Blečli parka“

str. 36

str. 40

## Sam svoj majstor

*LaTeX* prezentacija: *Beamer* (4. deo)

str. 45

## Hardver

*BagleBone Black Rev C*: Vodič od prvog dana

*BeagleBone Black* kao Tor relej (3. deo)

str. 51

Moć slobodnog  
softvera





## LiBRE! prijatelji



REGIONALNI  
LINUX PORTAL

linuxzasve.com



Grupa korisnika GNU/Linux operativnih sistema u Lovćencu

info i tutorijali na srpskom  
lubunturs.wordpress.com



Broj: 33

Periodika izlaženja: mesečnik

Izvršni urednik: Stefan Nožinić

Glavni lektor:

Aleksandar Božinović

Lektura:

Jelena Munćan Saška Spišjak

Milena Beran Milana Vojnović

Admir Halilkanović

Grafička obrada:

Dejan Maglov

Ivan Radeljić

Dizajn: White Circle Creative Team

Autori u ovom broju:

Nenad Marjanović

Nikola Hardi

Kriptopank

Dejan Čugalj

Branimir Karadžić

Ostali saradnici u ovom broju:

Marko Novaković Mihajlo Bogdanović

Počasni članovi redakcije:

Željko Popivoda

Vladimir Popadić

Aleksandar Stanisavljević

Željko Šarić

Kontakt:

IRC: #floss-magazin na irc.freenode.net

E-pošta: libre@lugons.org

## Vesti

21. januar 2015.

### Kanonikal pokrenuo verziju Ubuntu Kora za Internet stvari

Kanonikal, kompanija zadužena za razvoj Ubuntu Linuks distribucije, pokrenula je svoju verziju Ubuntu Kora (Core) za Internet stvari (*Internet of Things*).



Koristan link: <http://t.co/tN3VmZe5PK>

26. januar 2015.

### Masovno nadgledanje pretiljudskim pravima

U evropskom izveštaju masovno špijuniranje pretildigitalnoj sigurnosti i ljudskim pravima.



Koristan link: <http://t.co/9gcrmXw44q>

27. januar 2015.

### Jutjub koristi HTML5 kao podrazumevani način reprodukcije

Jutjub je zamenio Fleš tehnologiju novijom, otvorenijom i naprednijom tehnologijom - HTML5 koja će se koristiti kao podrazumevani način puštanja video sadržaja.



Koristan link: <http://t.co/zBpjh8Op6x>



28. januar 2015.

## **GHOST - bezbednosni problem**

*GHOST* - novi bezbednosni problem u GNU-ovoj C biblioteci koji dovodi u opasnost sve aplikacije koje koriste funkcije *gethostbyname* i *gethostbyname2* i omogućava udaljenom napadaču da dobije dozvole koje ima korisnik koji izvršava tu aplikaciju.

Koristan link: <http://t.co/GDYKsH2nOo>



1. februar 2015.

## **Pajrat Bej se vratio**

Najpopularniji torrent sajt Pajrat Bej (*The Pirate Bay*) se ponovo vratio posle dvomesečnog perioda kada je bio nedostupan.

Koristan link: <http://t.co/rSAPCVXWji>

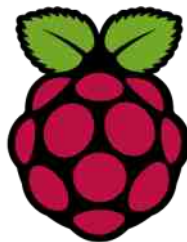


2. februar 2015.

## **Rasberi-Paj 2 je izašao**

Dostupan je u prodaji Rasberi-Paj 2 (*Raspberry PI 2*) sa 4 procesorska jezgra.

Koristan link: <http://t.co/RGI0doeioC>





## Vesti

3. februar 2015.

### Američka vojska otvara svoj program za detekciju napada

Američka vojska je odlučila da svoj program za detekciju napada postavi na Github i time omogućiti celom svetu da koriste softver koji su oni koristili za detekciju napada na mrežu ministarstva odbrane. Oni ovim potezom očekuju povratnu informaciju od ljudi koji ne rade u vladi i nadaju se unapređenju svog programa.



Koristan link: <http://t.co/v3PYIKF2sz>

6. februar 2015.

### GPG u opasnosti

Verner Koh (Werner Koch), programer koji je zadužen za razvoj GPG-a (Dži-Pi-Dži, engl. GNU Privacy Guard), softvera koji svi koristimo za enkripciju kako naše elektronske pošte tako i ostalih bitnih podataka, polako ostaje bez novčanih sredstava. Pozivaju se svi zainteresovani i oni koji su u mogućnosti da pošalju donaciju u vidu novca.



Koristan link: <http://t.co/jGac5emwPX>

9. februar 2015.

### Arduino razvojno okruženje - novo izdanje

Izašlo je novo razvojno okruženje za Arduino u verziji 1.6



Koristan link: <http://t.co/4A0pFcIsVf>





## ***bgfx - Cross-platform rendering library***



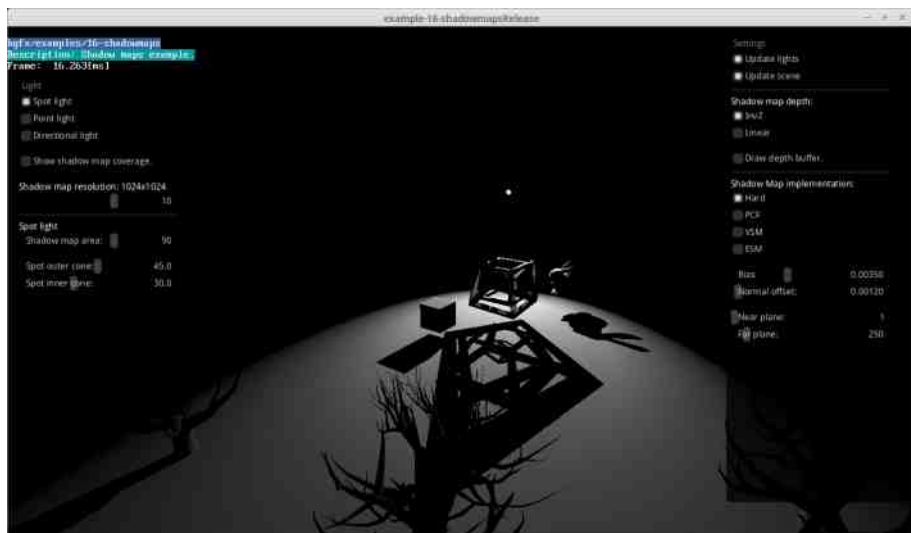
**Autor:** Branimir Karadžić

*bgfx* je rendering biblioteka otvorenog koda sa podrškom za više različitih grafičkih API-ja <sup>1</sup> (eng. *Application Programming Interface*) i operativnih sistema <sup>2</sup>. Biblioteka služi za jednostavniji pristup grafici i razvoj igara na više platformi. Najveći problem sa kompjuterskom grafikom je što postoji više različitih API-ja koji kada se koriste direktno onemogućavaju portovanje igara na druge platforme. Praveći igru koja pristupa samo *bgfx* biblioteci a ne direktno API-u, autor igre omogućava sebi da tu igru na jednostavan način može prebaciti na bilo koji podržan operativni sistem, odnosno platformu.



## Predstavljamo

U početku *bgfx* je bio samo hobi projekat i zato ima nemaštovito ime. Bila je to samo jedna od mnogih biblioteka koje sam napravio za pravljenje igara. Kada sam radio na jednoj igri, postojala je potreba da se igra prebaci sa starog *DX9* (eng. *DirectX 9*) renderinga na *OpenGL* (engl. Open-dži-el), pa sam tom prilikom izvukao deo moje hobi biblioteke za renderovanje i zapakovao je kao celinu. Posle uspešne primene ove biblioteke u mojim projektima, odlučio sam da je objavim kao projekat otvorenog koda na [Githubu](#) (eng. *Github*) pod licencom [BSD 2-clause](#). Po mom mišljenju, ovaj deo renderinga je trebalo već odavno da bude rešen problem, kao i da bude moguće da se igre bez veće muke izdaju na bilo kojoj platformi po želji autora. Namenjena je prevashodno manjim autorima koji žele da imaju potpunu kontrolu nad svojim kodom, bez obzira na platformu na kojoj žele da objave igru. Kod ovakvih komercijalnih proizvoda uvek postoje neka ograničenja, a ako firme izgube interes za proizvodom, mogu ostaviti autore na cedilu, bez podrške i održavanja. Sa druge strane, projekti otvorenog koda omogućuju saradnju autora i unapređenje funkcionalnosti projekta na zadovoljstvo svih.



Upotrebom *bgfx* biblioteke korisnik može da zanemari konkretnu implementaciju različitih tehnologija kao što su *OpenGL*, *OpenGL ES* i *Direct3D*, bez kompromisa sa performansama. Ova biblioteka nije tanka apstrakcija ili emulator nekog postojećeg *API*-ja, tipa Guglovog Engl (eng. *ANGLE*) projekta koji emulira *OpenGL*



ES preko *Direct3D*-a, ili Valvov TuDži-EI (eng. *ToGL*), koji emulira *Direct3D* 9 preko *OpenGL*-a, nego zaseban *API* višeg nivoa. Iako je u prošlosti bilo sličnih projekata, svi oni su uglavnom imali pretenziju da budu endžin (*engine*) ili frejmwork koji bi tvorci igara koristili kao takav. Takav pristup donosi dosta komplikacija i veoma je teško takav projekat integrisati sa postojećim kodom. Iz ovog razloga *bgfx* projekat nema pretenzije da bude ceo endžin ili frejmwork, nego samo biblioteka koju svako može lako da integriše u sopstveni projekat.

*bgfx* biblioteka se trenutno koristi u nekoliko komercijalnih igara, ali ima i dosta interesovanja za projekat kao osnove za pravljenje korisničkih aplikacija, pa čak i alata za programere.



Na projektu je do sada radilo oko deset ljudi. Uglavnom su dodavali podršku za nove platforme, primere, ili su povezivali biblioteku sa drugim jezicima. Što se buduće saradnje na ovom projektu tiče, sve je dobrodošlo, čak i sâmo korišćenje biblioteke, a zahtevi za uvođenje novih mogućnosti unapređuju samu biblioteku.

## Predstavljamo







### <sup>1</sup> Podržava grafičke API-je:

- Direct3D 9
- Direct3D 11
- OpenGL 2.1
- OpenGL 3.1+
- OpenGL ES 2
- OpenGL ES 3.1
- WebGL 1.0

### <sup>2</sup> Podržava platforme:

- Android (14+, ARM, x86, MIPS)
- asm.js/Emscripten (1.25.0)
- iOS
- Linuks (x86, x86\_64)
- Nejtive klajent (Native Client: PPAPI 37+, ARM, x86, x86\_64, PNaCl)
- OS lks (OSX 10.9)
- Rasberi-paj (RaspberryPi - ARM)
- Vindouz (lks-pe, Vista, 7, 8, 10)
- Vin-ar-ti (WinRT, WinPhone 8.0+)



## Predstavljamo



**Autor:** Stefan Nožinić

## Uvod

*rEFInd* je softverska aplikacija za *EFI* (*Extensible Firmware Interface*) sisteme koja omogućava izbor više operativnih sistema na jednostavan način. Ovo je samo upravitelj buta (*boot*) i u ovom broju vam predstavljamo o kakvom projektu je reč i kako se može primeniti.

## Šta je (U)EFI?

Pre nego što predstavimo sam projekat, trebalo bi prvo da skrenemo pažnju i detaljnije da predstavimo problematiku. *EFI* je nastao kao zamena za dosadašnji BIOS (*Basic Input/Output System*) i razvila ga je kompanija Intel. Ovaj pristup je kasnije zamenjen *UEFI* standardom. *EFI* tehnologija ima prednost u slučajevima kao što su:

- Pokretanje sistema na diskovima preko *2TB* memorije korišćenjem *GPT* tehnologije.
- Nezavisnost od specifične arhitekture procesora.
- Modularan dizajn.
- Dobro okruženje pre učitavanja operativnog sistema što podrazumeva i mrežnu podršku.

Tehnologija omogućava upotrebu 32-bitnih i 64-bitnih adresa na procesorima koji to podržavaju. Ranije, BIOS tehnologija to nije omogućila, već se uvek koristila 16-bitna implementacija iz istorijskih razloga.



## Particiona šema

*EFI* tehnologija ima drugačiju šemu particija. Dok je stari pristup podržavao samo *MBR* (*Master Boot Record*) šemu, nova tehnologija podržava *GPT* šemu koja omogućava veći broj primarnih particija. Sa starim pristupom taj broj bio je ograničen na četiri primarne particije, od kojih je svaka morala da ima kapacitet manji od *2TB*.

## Aplikacije

*UEFI* podržava izradu aplikacija koje bi se pokretale nezavisno od operativnog sistema. Ove aplikacije se mogu naknadno instalirati. Primer takvih aplikacija su pokretači (*loader*) operativnih sistema kao i upravnici butovanja.

## EFI systemska particija

*EFI* podržava posebnu particiju za smeštanje raznih potrebnih informacija kao što su potrebni podaci za rukovanje računarom i specifične aplikacije. Ova particija je najčešće *FAT32* sistem datoteka.





## Predstavljamo

### But menadžer i but louder - ima razlike

Od popularizacije *GRUB* upravnika mnogi korisnici su ostali bez objašnjenja kakva je razlika između *but loudera* (*boot loader*) i *but menadžera* (*boot manager*). Ovo se dešava jer *GRUB* pruža obe usluge u sklopu jednog programa, pa se time teže vidi razlika između ova dva pojma. *But menadžer* je upravnik pomoću kojeg je moguće izabrati operativni sistem za pokretanje. On ne zna ništa specijalno o operativnom sistemu, pa time ne zna ni kako da ga pokrene. On samo zna da treba da pokrene *but louder* za taj sistem. Kada je sistem izabran, pokreće se specifičan *but louder* za taj sistem koji je zadužen za dalji proces pokretanja kernela. Za razliku od *GRUB* upravnika, *rEFInd* pruža samo usluge *but menadžera*, što u početku može zvučati poražavajuće za ovaj projekat. Kada uzmemo u obzir da većina sistema danas ima *but louder* u sklopu kernela (uključujući i *Linux* od novijih izdanja) ovaj problem je prevaziđen.





## Instalacija

*rEFInd* postoji kao *.deb* i kao *.rpm* paket, ali postoji i zapakovan kao *.zip* arhiva. Zahvaljujući tome, paket je moguće instalirati upotrebom *dpkg* komande na distribucijama baziranim na Debianu kao i upotrebom *rpm* komande na distribucijama koje koriste taj način pakovanja paketa. Za Ubuntu postoji i *PPA* repozitorijum pa, *rEFInd* može biti instaliran na sledeći način:

```
sudo apt-add-repository ppa:rodsmith/refind
sudo apt-get update
sudo apt-get install refind
```

Nezavisno od toga koji način od gore opisanih koristite, *rEFInd* će automatski biti prebačen na *ESP* i biti postavljen kao podrazumevani upravnik.

## Konfiguracija

*rEFInd* bi trebalo da vam pruži zadovoljavajuću funkcionalnost bez dodatne konfiguracije posle instalacionog procesa. On će automatski detektovati loudere operativnih sistema korišćenjem pretrage po svim particijama koje prepozna. U slučaju da ste napredniji korisnik i da želite da modifikujete ponašanje *rEFInda*, to možete uraditi izmenom njegove konfiguracione datoteke. Postoji globalna konfiguracija i konfiguracija za svaki sistem posebno. Ovaj tekst bi postao previše dugačak kada bismo ulazili u detalje same konfiguracije pa vam zato preporučujemo, ako imate potrebu za dodatnom konfiguracijom, da posetite dokumentaciju projekta ili, ako imate neke nedoumice, da kontaktirate s nama preko naše e-pošte.

## Za kraj

Želeli bismo da ovaj tekst zaključimo ocenom da je ovaj projekat dosta zanimljiv, da pruža zadovoljavajuću funkcionalnost i da omogućava korisnicima koji koriste ovaj sistem pokretanja lako pokretanje svojih operativnih sistema. Još jednom želimo da vam skrenemo pažnju da nam se obratite i ukažete ako smo nešto važno propustili da pomenemo ili ako nam se negde potkrala greška.

Korisni link: <http://www.rodsbooks.com/refind/>

## Kako da...?

# Uvod u programski jezik C

## Rad sa stringovima (9. deo)

**Autor:** Nikola Hardi

Opšte je poznato da su računari dobri u radu sa brojevima. Međutim, vremenom su se i potrebe korisnika i alati za programiranje računara razvijali, i rad sa drugačijim tipovima podataka je postao uobičajen i znatno jednostavniji nego ranije. Svi smo imali priliku da radimo sa programima za obradu fotografija, video materijala, ili zvučnih zapisa. Multimedija je ipak tema koja prevazilazi jedan uvodni kurs o C-u, ali do kraja ovog serijala ćemo se dotaći i tih tema. Za početak, osvrnućemo se na rad sa tekstualnim podacima na C način.

### Stringovi na C način

Definicija tekstualnog podatka u sferi računarstva se uglavnom svodi na to da je tekst niz karaktera, a karakteri su znakovni simboli. Iako imamo mogućnost da radimo sa tekstom ili multimedijom, računari ipak razumeju samo brojeve pa moramo tako da im predstavimo i tekst. Postoje razni standardi kako se pojedini karakteri predstavljaju u memoriji računara, između ostalog najpoznatiji su *ASCII*, *EBCDIC*, *UTF* itd. *EBCDIC* je standard koji je prisutan u Eplovom (eng. *Apple*) svetu i više nema toliko važnu ulogu. *ASCII* je standard koji je vrlo prisutan i danas. *UTF* je nastao kao proširenje *ASCII* standarda, javlja se u više varijanti i uvodi podršku za pisma različita od abecede (kinesko, japansko, ćirilica, arapsko i mnoga druga).

Važno je napomenuti da je originalni *ASCII* standard propisivao 7 bitova za jedan karakter, što je dalo prostora za 128 karaktera. Među tim karakterima se nalaze cifre, mala i velika slova engleskog alfabeta i niz kontrolnih karaktera. Kontrolni karakteri su oznake za novi red, prazno mesto, tabulator, upravljanje štampačima (vraćanje glave štampača na početak), rad sa terminalima (alarm ili obaveštenje



## Uvod u programski jezik C

da se nešto dogodilo). ASCII standard je proširen na 8 bitova, odnosno ceo jedan bajt i sada podržava 256 karaktera. Potražite na internetu termin „ASCII tabela” da biste stekli uvid u to kako izgleda tzv. tabela ASCII karaktera.

Rad sa pojedinačnim znakovima ne obećava mnogo. Postoji realna potreba za radom sa rečima, rečenicama, linijama teksta itd. Podaci koji se sastoje od više od jednog karaktera se u domaćoj literaturi nazivaju znakovnim niskama, mada odomaćen je i naziv „string”, koji je i mnogo češće u upotrebi. U programskom jeziku C se stringovi predstavljaju običnim nizovima karaktera (**char niz[]**), a kraj teksta se označava **NULL** karakterom, odnosno specijalnim znakom **0**, čija brojevnja predstava je, pogađate, nula. Ovakva predstava tekstualnih podataka nosi sa sobom i pojedine komplikacije. Gotovo svi savremeni jezici poznaju tip stringa, dok su u C-u stringovi ipak samo obični nizovi.

Da se podsetimo, nizovi u C-u su zapravo pokazivači na deo u memoriji. Kada pokušamo da pristupimo jednom elementu niza, tada se vrši operacija „dereferenciranja” sa zadatim odstupanjem. Sledi nekoliko primera koji ilustruju rad sa nizovima na primeru nizova karaktera.

```
char niz1[] = "zdravo";  
niz1[0] = 'z';
```

```
char niz2[5];  
niz2[0] = '0';  
niz2[1] = 'k';  
niz2[2] = 0;
```

Pri deklaraciji niza 1 nije zadata dužina zato što je u jednom izrazu zadata i inicijalizacija niza koja glasi „zdravo”. Prevodilac može sam da zaključi koliko memorije je potrebno.

U drugom primeru je kreiran niz karaktera dužine 5. Međutim, popunjena su samo prva tri mesta. Primećujete razliku između prvog i trećeg elementa? Prvi je cifra **'0'** (uokviren je navodnicima), a treći je brojčana vrednost **0** koja označava kraj stringa.

Može se primetiti još jedna razlika u data dva primera. U prvom primeru su korišćeni dvostruki navodnici, dok su u drugom korišćeni jednostruki. Razlog za to

## Kako da...?

je što se znakovni nizovi koji su uokvireni dvostrukim navodnicima smatraju stringovima i podrazumevano im se dodaje „terminator“, odnosno znak **NULL**, koji označava kraj stringa. Jednostruki navodnici se isključivo koriste za predstavljanje pojedinačnih karaktera. Ovo znači da zapisi **“a”** i **'a'** nisu jednaki. Prvi zapis je dužine 2, jer ima i znak **NULL** kojim je završen.

## Ispisivanje i učitavanje stringova

Pre nego što nastavimo dalje, biće objašnjen jednostavan način za ispisivanje i učitavanje stringova pomoću standardnog ulaza/izlaza. Da se podsetimo, biblioteka za standardni ulaz/izlaz (ispis podataka u konzoli odnosno terminalu) je **stdio.h**. Ukoliko se koristi formatirani ispis, potrebno je funkcijama **printf()** i **scanf()** proslediti format string koji sadrži specijalan znak **%s**, čime je označeno da želimo da ispišemo znakovni niz sve do **NULL** karaktera. Drugi način je upotrebom funkcija **puts()**, **gets()** i **getline()** iz biblioteke **string.h**. Sledi primer:

```
#include <stdio.h> include <string.h> int main()
{
    char niz1[] = "Zdravo svete!";

    printf("%s\n", niz1);

    scanf("%s", niz1);
    puts(niz1);

    return 0;
}
```

Može se primetiti da je u slučaju funkcije **printf()** potrebno naglasiti specijalnim kontrolnim karakterom **\n** da nakon ispisa treba preći u novi red. Funkcija **puts()** to podrazumevano radi.

Za ove i sve druge funkcije dodatnu pomoć je moguće pronaći u Linuksovim *man* stranicama u odeljku 3, koji je posvećen standardnim bibliotekama. Recimo, **man** **getline** sadrži detaljno uputstvo o funkciji **getline()**.



## Kopiranje stringova

Kao što je u prethodnom odeljku napomenuto, stringovi su u C-u samo obični nizovi. To znači da sledeći kod verovatno neće uraditi ono šta bi bilo očekivano.

```
niz1 = niz2;
```

Ovime smo pokazivaču **niz1** „rekli“ da pokazuje tamo gde i pokazivač **niz2**. Efektivno, oni sad pokazuju na isti tekst, međutim ako izmenimo **niz1**, to će se odraziti i na **niz2**, a važi i obrnuto. Ovime smo dobili samo dva pokazivača na isti deo memorije, a uglavnom to nije bio cilj.

Drugi način kako bismo sadržaj drugog niza mogli da kopiramo u prvi niz je petlja kojom bismo kopirali element po element. Pošto je operacija kopiranja stringova vrlo često potrebna, postoji standardna biblioteka za rad sa stringovima koja se zove **string.h**, a funkcija za kopiranje stringova je **strcpy()**.

```
strcpy(niz1, niz2);
```

Dati primer kopira sadržaj niza 2 u niz 1. Kopira se element po element sve dok se u nizu 2 ne naiđe na **NULL** karakter. Ovakvo kopiranje (element po element) se inače u stranoj literaturi naziva i „*deep copy*“ i vrlo često se sreće u radu sa strukturama podataka, gde je potrebno obići celu strukturu podataka i svaki element kopirati, a strukturu opet izgraditi u drugom delu memorije.

Zanimljivo je što je kopiranje stringova moguće pojednostaviti ukoliko definišemo novi tip podatka kao strukturu koja sadrži niz karaktera i potom izvršimo kopiranje. Ovo je trik koji se uglavnom ne koristi u praksi i spomenut je samo kao zanimljivost.

```
#include <string.h> struct str {  
    char niz[50];  
};  
  
int main()  
{  
    struct str s1;  
    struct str s2;
```

## Kako da...?

```
strcpy(s1.niz, "zdravo");
s2 = s1;

s2.niz[0] = 'Z';

puts(s1.niz);
puts(s2.niz);

return 0;
}
```

Kreirana je struktura *str* koja sadrži polje niza karaktera pod nazivom *niz*. Pozivom funkcije **strcpy()** kopiramo string “zdravo” u **s1** čime smo „napunili” sadržaj objekta **s1**. Potom je izvršena redovna dodela čime je sadržaj objekta **s1** kopiran u **s2**. Zatim je izmenjen samo prvi karakter objekta **s2** i oba stringa su ispisana. Primećujete da su kopirani svi znakovi, a da je izmenjen samo drugi objekat? Zanimljiv trik.

## Poređenje, spajanje i pretraga stringova

Druga vrlo česta situacija je poređenje dva stringa. Ukoliko bi poređenje bilo izvršeno upotrebom redovnog operatora za poređenje **==**, tada bi u stvari bile poređene samo adrese na koje pokazuju pokazivači tih stringova, jer, da se podsetimo, stringovi su u C-u samo obični nizovi. Ispravan način za poređenje dva stringa je pozivanjem funkcije **strcmp()** čija je povratna vrednost 0 ukoliko su joj prosleđeni stringovi sa jednakim sadržajem.

```
#include <string.h> int main()
{
    char niz1[] = "zdravo";
    char niz2[] = "zdravo";

    if( strcmp(niz1, niz2) == 0)
        puts("Stringovi su jednaki.");
    else
        puts("Stringovi nisu jednaki.");
    return 0;
}
```





Funkcija **strcmp()** poredi svaki element prosleđenih nizova pojedinačno, a dobra je vežba implementiranje nove funkcije **strcmp()** koja će unutar petlje da poredi odgovarajuće elemente prosleđenih nizova, i potom vratiti rezultat **0** ukoliko su svi elementi jednaki. Potrebno je pripaziti na situaciju kada su stringovi različitih dužina.

U svetu računara, spajanje dva sadržaja se zove konkatencija. Odatle i naziv funkcije za spajanje dva stringa, **strcat()**. Važno je uveriti se da je string u koji se dodaje sadržaj dovoljno velik da ga prihvati, inače može da dođe do neželjenog prepisivanja u memoriji, uništavanja drugih podataka i neispravnog ponašanja programa.

Pretraga unutar stringa se može obaviti funkcijom **strstr()**. Ova funkcija kao parametre očekuje pokazivač na niz karaktera (string) u kojem je potrebno pronaći string koji je prosleđen kao drugi parametar. Povratna vrednost je pokazivač na deo memorije gde se željeni string prvi put pojavio.

Sledi malo opširniji primer koji ilustruje spajanje, pretragu i kopiranje stringova.

```
#include <string.h> int main()
{
    char niz_a[50] = "LiBRE! je ";
    char niz_b[] = "najbolji casopis";
    char niz_c[] = "e-magazin";

    strcat(niz_a, niz_b);
    puts(niz_a);

    char *cilj = strstr(niz_a, "casopis");
    strcpy(cilj, niz_c);

    puts(niz_a);

    return 0;
}
```

## Kako da...?

# Pretvaranje tipova podataka u stringove i pretvaranje iz stringova

Pretvaranje podataka iz jednog tipa u drugi ili kastovanje (eng. *type casting*) se za većinu tipova vrši jednostavno pisanjem tipa u zagradama, na primer **(int)**5.23. Stringovi su i po ovom pitanju drugačiji. Postavlja se pitanje, šta pretvoriti u celobrojni tip - da li adresu na koju pokazuje pokazivač niza koji predstavlja string? Prvi element? Šta ako taj string nije broj? Predstavićemo rešenje za konverziju brojeva u stringove i obrnuto.

Ispis podataka na ekranu se takođe može smatrati konverzijom. Broj sa pokretnim zarezom je u memoriji prikazan na pomalo čudan način, a pozivom funkcije **printf()** kada je kao format zadat **%f** na ekranu ćemo dobiti razumljiv niz znakova. Sličan rezultat možemo dobiti funkcijom **sprintf()**, a jedina razlika u odnosu na **printf()** je ta što **sprintf()** kao prvi argument očekuje pokazivač na string, dok su drugi (format string) i treći (argumenti koji se povezuju sa formatnim stringom) slični kao kod funkcije **printf()**.

Kod pretvaranja stringova u brojeve tipove može poslužiti funkcija **sscanf()**, koja se ponaša slično kao funkcija **scanf()**, s tim što je prvi argument adresa (pokazivač na string) koju želimo da skeniramo u nadi da će se poklopiti sa formalnim stringom koji je drugi argument, a rezultati će biti sačuvani u adresama koje su prosleđene kao 3, 4 i ostali parametri. Drugi način za pretvaranje stringova u celobrojne brojeve i brojeve sa pokretnim zarezom su funkcije **atoi()** i **atof()**.

Funkcije koje koriste formatni string su česte u jeziku C i zaslužuju posebnu pažnju, međutim to je priča za poseban članak. U ovu klasu funkcija spadaju, recimo, i funkcije za rad sa datotekama.

## Bezbednost

Rad sa stringovima može da bude izuzetno opasna stvar po pitanju bezbednosti programa. Ukoliko se niz karaktera ne završava **NULL** karakterom, svašta može da pođe po zlu pri pozivu funkcija kao što su **strcpy()**, **strcmp()** i slične. Cela klasa tih funkcija se oslanja na postojanje znaka koji označava kraj stringa. Kada string nije propisno završen, tada te funkcije mogu da zalutaju u deo memorije van stringa i svašta može da pođe po zlu. Ovakav tip napada se inače zove prekoračenje bafera (eng. *buffer overflow*). Preporučeni način kako da izbegnete



## Uvod u programski jezik C

takve neprijatne situacije pri radu sa stringovima je da koristite verzije funkcija koje su ograničene maksimalnom dužinom stringa, kao što su **strncpy()**, **strncat()**, **strncmp()** itd. Sve te funkcije kao treći parametar zahtevaju broj koji određuje maksimalnu očekivanu dužinu stringa. Još jedna vrlo korisna funkcija je **strlen()** čiji zadatak je da odredi dužinu stringa. Budite pažljivi, ovo je prvo mesto gde se traže bezbednosni propusti u programima. Ne verujte podacima koji spolja ulaze u vaš program, a pogotovo kada je reč o stringovima.

# Learn C Programming

Pregled popularnosti *GNU/Linux/BSD* distribucija za mesec februar

## Distrowatch

1	Mint	2919>
2	Ubuntu	1778>
3	Debian	1701>
4	openSUSE	1323>
5	elementary	1204>
6	Mageia	1174>
7	Fedora	1166>
8	Manjaro	1028<
9	CentOS	1020>
10	LXLE	965>
11	Arch	957>
12	Bodhi	907>
13	Android x86	900<
14	Kali	760>
15	PCLinuxOS	742<
16	Puppy	732>
17	Robolinux	695>
18	Lubuntu	670>
19	Q4OS	663>
20	Black Lab	655>
21	Netrunner	652>
22	Simplicity	629>
23	Evolve	607>
24	Zorin	583>
25	4MLinux	517=

Pad <  
 Porast >  
 Isti rejting =  
 (Korišćeni podaci sa Distrovoča)

**Oslobađanje**

## U potrazi za idealnom distribucijom:

# Socijalni kriterijumi za izbor idealne distribucije (6. deo)

**Autor:** Dejan Maglov

Projekti slobodnog softvera obično počinju kao projekti pojedinca ili male grupe programera, a koji rade na razvoju softvera čiji je cilj da zadovolji neke od njihovih ličnih potreba. Projekat će se smatrati uspešnim ukoliko se šira zajednica zainteresuje za njega nakon što prepozna korisnost softvera ne samo za pojedince koji su započeli projekat, već i za širu zajednicu. Kada je u pitanju slobodni softver, vremenom je postala stalna praksa da se zajednica umeša u razvoj projekata. Dobri projekti sami privlače zajednicu ljudi koji su zainteresovani za njih i ona, u većini slučajeva, preuzima na sebe testiranje, poboljšanje funkcionalnosti, održavanje i tehničku podršku projekta.

Zajednice slobodnog softvera mogu da se okupe: oko projekta, slobodnog softvera, po teritorijalnom principu ili da nastanu kombinacijom ovih principa.

Primeri:

- LiBRE! projekat okuplja zajednicu oko projekta bez teritorijalnog ograničenja.
- LUGoNS (*Linux User Group of Novi Sad*) je zajednica korisnika Linuksa po teritorijalnom principu bez obzira na konkretan projekat.
- Ubuntu-rs zajednica koja je kombinacija okupljanja zajednice oko konkretnog projekta (Ubuntu) i teritorijalnog principa koji se ograničava na teritoriju Srbije.

Ova sinergija projekata i korisnika slobodnog softvera ima više prednosti i za korisnike i za projekat. Projekat se na ovaj način brže razvija, dobija na



## U potrazi za idealnom distribucijom

funkcionalnosti, greške se brzo ispravljaju, itd. Korisnici dobijaju osećaj da pripadaju grupi, druže se sa istomišljenicima, učestvuju u promociji kvalitetnih pojedinaca itd.

Dve najvažnije uloge lokalnih zajednica za GNU/Linuks distribucije su:

1. pružanje tehničke podrške i edukacije na maternjem jeziku;
2. rad na lokalizaciji i prilagođavanje distribucije lokalnim potrebama.

Informacija da postoje lokalne zajednice slobodnog softvera je od naročitog značaja za početnike i korisnike koji nedovoljno dobro znaju strane jezike (naročito engleski jezik). Ako izuzmemo komercijalne projekte otvorenog koda (Red Hat, SUSE), ne postoje zvanične škole i kursevi za slobodni softver. Za edukaciju početnika je uglavnom zadužena zajednica, a jasno je da je učenje na maternjem jeziku uvek lakše nego na bilo kom drugom stranom, ma koliko dobro ga poznajete.

Forumi i viki stranice lokalnih zajednica su prave biblioteke u kojima korisnici mogu da nađu većinu odgovora na svoja pitanja, a za sva ostala tu su forumi i IRC kanali zajednica na kojima uvek ima nekoga i koji imaju već gotova rešenja za veliki broj problema.

## Oslobađanje

### Lokalne zajednice kao kriterijum za izbor idealne distribucije

Početnicima i korisnicima koji ne vladaju dobro engleskim jezikom, toplo preporučujemo da biraju GNU/Linux distribucije koje imaju aktivnu lokalnu zajednicu. Ovladavanje novim operativnim sistemom je mnogo lakše ukoliko vam „leđa čuva“ lokalna zajednica na koju se uvek možete osloniti.

Već smo spomenuli da su popularnije GNU/Linux distribucije namenjene početnicima. Slučaj je isti i sa lokalnim zajednicama okupljenim oko njih. Ovo međutim ne znači da je kvalitet usluga koje nude manje popularne lokalne zajednice, okupljene oko distribucija namenjenih naprednijim korisnicima, lošiji. Manje popularne distribucije okupljaju manje ljudi, ali zato oni poseduju veće znanje. Iskustvo sa zajednicama okupljenim oko slobodnog softvera je uglavnom pozitivno i ukazuje na to da kvalitet usluga koje one nude korisniku nije uslovljen popularnošću i masovnošću zajednice nego isključivo njenom (ne)aktivnošću.

U svakom slučaju, dosadašnji saveti za izbor idealne distribucije važe i dalje. Socijalni aspekt slobodnog softvera je samo dodatni faktor koji vam može pomoći pri donošenju odluke kada se dvoumite između nekoliko mogućih rešenja.

Još uvek ne postoji na jednom mestu potpuni i ažurirani spisak svih aktivnih srpskih zajednica. Nešto najbliže tome može da se nađe na adresi <https://pulsslobode.wordpress.com/>. Za informacije o nekim drugim lokalnim zajednicama koje možda postoje ali nisu na ovom spisku moraćete sami da pretražite internet.

### Lokalizacija kao kriterijum za izbor

Za lokalizaciju GNU/Linux distribucija i GNU programa odgovorni su pojedinci i lokalne zajednice. Sasvim solidnu srpsku lokalizaciju mogu imati i projekti koji uopšte nemaju svoju lokalnu zajednicu, a isto tako, lošu lokalizaciju mogu imati i projekti sa dobrom lokalnom zajednicom. Sve je stvar prioriteta zajednica i aktivnosti pojedinaca.

Pravu i kontrolisanu lokalizaciju može imati samo lokalni projekat. Ovakvih projekata u Srbiji ima malo i teško se probijaju, čak i u lokalu u konkurenciji svetskih projekata. Istina je da mnoge veće GNU/Linux distribucije imaju srpsku lokalizaciju



## U potrazi za idealnom distribucijom

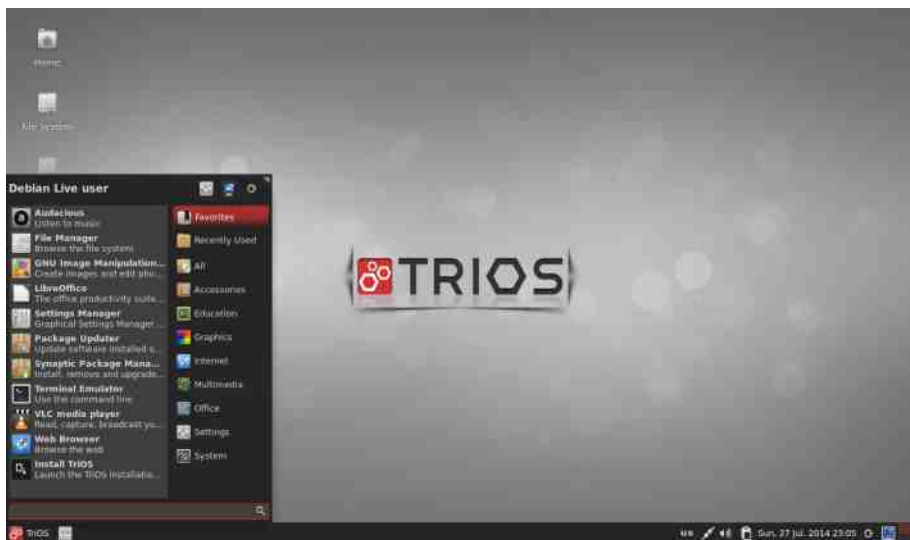
ju u ponudi, pa se možda slični lokalni projekti koji su usmereni samo na striktnu lokalizaciju smatraju nepotrebnima. Uzvodna lokalizacija u slučaju Srbije vrlo često bude veoma loše, čak nakaradno, sprovedena. Jedna polovina programa je lokalizovana srpskom ćirilicom, a druga latinicom. Ako postoji još delova sistema koji uopšte nije lokalizovan, onda se pojavljuju i delovi programa koji su na engleskom. Zato ćemo se vratiti na prvu rečenicu ovog pasusa i ponovo utvrditi da samo lokalni projekat može potpuno kontrolisati pravu srpsku lokalizaciju.

U Srbiji za sada imamo dva aktivna projekta GNU/Linux distribucije - **TriOS** (<http://trios.rs/>) i **Serbian** (<http://www.debian-srbija.iz.rs/p/serbian.html>). Oba projekta su mlada i imaju svojih prednosti i mana. Kome treba lokalizovan operativni sistem može da proba jedan od ova dva projekta.





## Oslobađanje



Iako ove dve domaće distribucije nisu baš najidealnije, interesantno je probati ih i uporediti sa drugim distribucijama.

## Za kraj serijala

Namera nam je bila da kroz ovaj serijal o kriterijumima pri izboru idealne distribucije definišemo većinu kriterijuma pri izboru i pokažemo zašto je teško odgovoriti na jednostavno pitanje: „Koju mi distribuciju preporučujete?”

Ni na samom kraju nismo dali konkretan odgovor na pitanje koja je idealna GNU/Linux distribucija. Suviše je subjektivnih ocena da bi se mogao dati konkretan odgovor. Smatramo da je jako teško napisati neki algoritam izbora po kriterijumima koje smo identifikovali i definisali, a možda je čak i nemoguće, pošto se izlazi stalno menjaju. Izlazi iz tog algoritma su GNU/Linux distribucije kojih iz dana u dan ima sve više a i one stare se stalno menjaju.

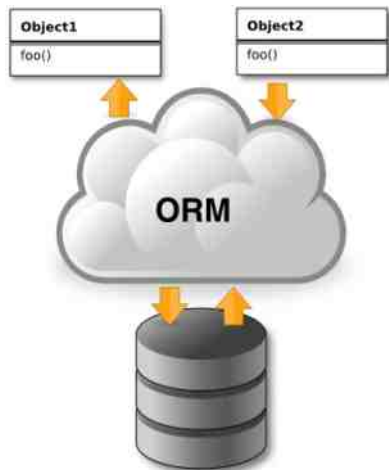
Sam serijal „U potrazi za idealnom distribucijom” neće biti završen samo zato što se više neće baviti kriterijumima pri izboru, nego će pokušati da predstavlja GNU/Linux distribucije koje su po nečemu „idealne”.



## HIBERNATE ORM (2. deo)

**Autor:** Dejan Čugalj

### Object relation mapping (ORM)



ORM (eng. *Object Relation Mapping*) je način postizanja trajnosti (eng. *persistence*) objekata unutar relacionih baza podataka. Model deluje kao posrednik na logičkom delu aplikacije. Automatski preslikava podatke iz objekata u bazu i nazad, na zahtev aplikacije. Da bismo objasnili kako ovo sve radi, moramo da vam predstavimo ORM frejmwork (eng. *framework*) programa otvorenog koda - Hajbernejt (eng. *Hibernate* - <http://hibernate.org/orm/>).

### Šta je ORM?

Ukratko, ORM je automatizovano, transparentno preslikavanje objekata korišćenjem metapodataka koji opisuju mapiranje veze između objekta i baze

## Slobodni profesionalac

podataka u aplikacijama koje su napisane u objektno orijentisanim programskim jezicima. Aplikacija komunicira sa *ORM*-om preko njegovog vlastitog *API*-ja (eng. *application programming interface*) i određenih objekata. Ova komunikacija je potpuno odvojena od komunikacije koja se odvija između *ORM*-a i baze podataka preko *SQL*-a i *DBC*-a.

Da bi se postigla i dostigla potpuna snaga *ORM*-a u aplikaciji koju projektujemo, nikako se ne sme isključiti znanje relacionih modela, *SQL* jezika, te poznavanje *DBMS*-a u kome se radi. *ORM* nije ništa drugo do srednji sloj između gore pomenutih tehnologija i objektno orijentisanog programiranja, kao što smo to do sada već više puta naglasili.

## Implementacija *ORM*-a

Postoje različiti načini implementacije *ORM*-a. Mark Fasl (*Mark Fussel*) je 1997. definisao četiri nivoa:

1. „Čist” relacioni način
2. Slabo mapiranje objekata
3. Umereno mapiranje objekata
4. Potpuno mapiranje objekata

### „Čist” relacioni način

Cela aplikacija, uključujući i korisnički interfejs, dizajnirana je relacionim modelom i *SQL* relacionim operatorima. Ovakav pristup je odličan za male projekte. Direktni pristup *SQL*-u omogućava fino podešavanje svakog aspekta upita koji se postavlja bazi podataka. Osnovne mane su portabilnost (prenosivost) i održavanje ovakvih sistema, pogotovo ako se planira korišćenje aplikacije na duži rok. Ovakvi sistemi masovno koriste sačuvane procedure (eng. *stored procedures*), što znači da je poslovna logika (eng. *business logic*) prebačena na samu bazu podataka.

### Slabo mapiranje objekata

Entiteti su predstavljeni kao klase i mapiranje se radi eksplicitno ka relacionim tabelama. *SQL/JDBC* je sakriven od poslovne logike korišćenjem poznatih šablona (*patterns*).

### Umereno mapiranje objekata



Dizajn aplikacija se svodi na objektno modeliranje. SQL se generiše u vreme kompajliranja (eng. *build-time*) ili u vreme izvršavanja (eng. *run-time*). Specifikacija upita nad bazama podataka se radi na objektno orijentisanom jeziku. Ovakvi sistemi se koriste u aplikacijama srednjih veličina kod kojih je bitna portabilnost između različitih sistema za upravljanje bazama podataka (DBMS).

### Potpuno mapiranje objekata

Podržava sofisticirano objektno modeliranje, nasleđivanje i polimorfizam. Efikasnost dobijanja podataka je na visokom nivou i koriste se metode kao što su lenjo učitavanje (eng. *lazy loading*) i strategija keširanja (eng. *caching*). Implementacija je transparentna u aplikacijama. Ovaj nivo je jako teško dostići. Pojedincu su potrebne godine razvoja da postigne ovakvu fleksibilnost pristupa i manipulacija u relacionim bazama podataka. Upravo ovaj nivo se postiže ako se koriste rešenja ORM-a kao što je Hajbernejt.

### Zašto ORM?



Implementacija ORM rešenja, pogotovo ako se prvi put susrećete sa ovim modelom, može da bude veoma frustrirajuća i malo teža za implementaciju. Kada ovo kažemo, mislimo da programer koji implementira ORM i hoće da isti bude kvalitetno urađen, mora da poseduje predznanje objektno orijentisanog programiranja i naprednih tehnika kao što su nasleđivanje i polimorfizam.

Ovo inicira fundamentalna pitanja:

- Pa zašto bi iko onda implementirao ovako nešto kompleksno?
- Vredi li truda upuštati se uopšte u ORM?

Odgovor nije jednostavan. Dosadašnja iskustva u korišćenju ORM-a i njegova popularnost već daju odgovor na postavljeno pitanje. Očigledno je da su benefiti koje donosi ovaj model veći od kompleksnosti njegove implementacije, a to su:

## Slobodni profesionalac

### Veća produktivnost

*ORM* smanjuje količinu koda potrebnog za ostvarivanje trajnosti podataka. Ovo omogućava programeru da vreme koje bi utrošio na ručno mapiranje podataka, utroši na druge delove aplikacije.

### Lakše održavanje

Manje linija koda osigurava bolju razumljivost (i možda bolji kvalitet koda) jer naglašava aplikacijsku logiku. *ORM* takođe ublažava uticaj nastalih promena kao što su promene u bazi, strukturi tabela, itd.

### Prenosivost

Većina *ORM* rešenja podržava sisteme za upravljanje bazama podataka raznih proizvođača. *ORM* u potpunosti odvaja sloj aplikacije od stvarne *SQL* komunikacije koja se odvija u pozadini. Time omogućava da se pišu aplikacije koje će moći da se izvršavaju na različitim bazama podataka bez potrebe za dodatnim kodiranjem. Ova komunikacija se odvija sa programskim interfejsima *ORM*-a nezavisno o *DBMS*-u sa kojim je u bilo kom momentu povezan. *ORM* takođe omogućava lakši prelaz sa manjih na veće baze podataka tokom razvoja aplikacije.

\* \* \*

Uvideli smo da trenutno postoje dve paradigme koje se „sudaraju” i koje su podjednako dobre same za sebe. Relaciona paradigma potvrđuje da sve što je izgrađeno na matematičkim osnovama je i trajno, u smislu kvaliteta koje poseduje. U domenu u kom relaciona paradigma egzistira, ona je, ako je to u ovo vreme uopšte moguće tvrditi, nezamenjiva.

Ono što otvara pitanja i daje razlog za sumnju je, koja je krajnja granica upotrebljivosti relacione paradigme, tj. koliki su kapaciteti složenosti zahteva aplikacija sa kojom može da se nosi. Praksa je pokazala da je sasvim moguće „izvesti” složene projekte - ali održavanje, skalabilnost, razumljivost, prenosivost - pitanja su sa kojim se relaciona paradigma teško može izboriti.

Sa druge strane, svoj habitat u informacionim tehnologijama je našla i objektna



paradigma. Iako na prvi pogled ova dva modela nemaju ništa zajedničko - što je u neku ruku i istinito gledajući prirodu podataka kojim manipulišu - praksa je pokazala da su usko povezane i skoro da imaju sinaptičku vezu.

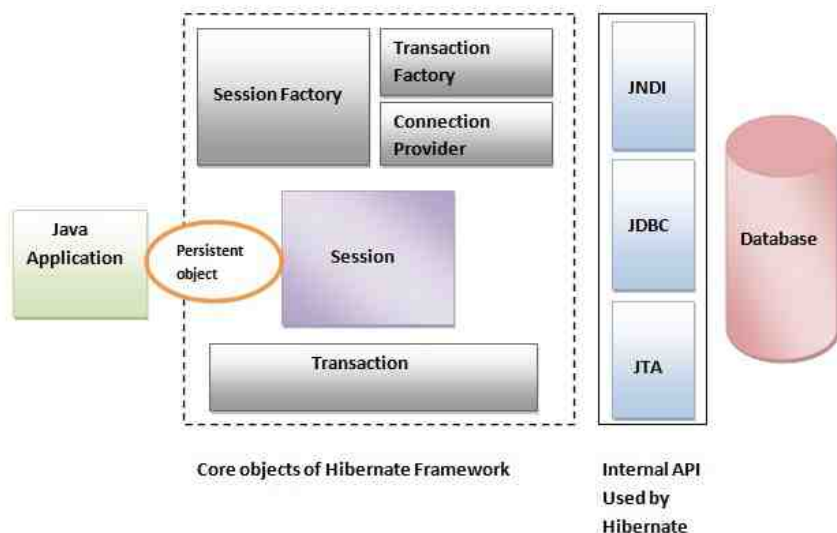
Pitanje „kako spojiti te sinapse” dovelo je do žučnih rasprava u uskim naučnim, programersko-administratorskim krugovima. Rodili su se razni modeli, ali jedan je iskočio i polako ulazi u široku primenu - a to je *ORM*. Svojim suptilnim pristupom problemu nekompatibilnih paradigmi nekako je uspeo da stvori taj nedostajući „električni signal” koji ih je povezao. Ovo je omogućilo da dve strane, koje su svaka za sebe u svom domenu ostavile i ostavljaju duboki trag u informacionim tehnologijama, sarađuju i donesu novi kvalitet.

Korisni linkovi:

[1] Hibernejt: <http://hibernate.org/>

[2] Viki: <http://goo.gl/LIZulw>

[3] Tutorijali: <http://goo.gl/CGzvnc>



## Internet mreže i komunikacije

# To su samo metapodaci?

**Autor:** Kriptopank

Šta su, zapravo, metapodaci? Najkraće rečeno, to su podaci o podacima ili informacije koje opisuju podatke.

Najpre da napravimo razliku između podataka i metapodataka. Podaci su, naprimer, snimak (ili transkript) vaših telefonskih razgovora, sadržaj poruka koje pišete, primete ili šaljete, podaci koje kačite na internet ili preuzimate, pretrage Guglom, postovi na forumima, dopisivanja na IRC kanalima ili drugim četovima, audio i video snimci Skajp razgovora, i tako dalje.



Hajde da vidimo šta bi bili odgovarajući metapodaci za ove primere:

- Za telefonske razgovore metapodaci mogu biti lokacija učesnika, brojevi telefona, trajanje razgovora, *IP* i *MAC* adresa.
- Za poruke, četove i Skajp, ili druge video-audio komunikacije, slično je kao kod telefonskih razgovora.





## To su samo metapodaci?

- Kod foruma i pretraga Guglom ima malo manje metapodataka, ali Gugl i razni sajtovi i forumi ionako čuvaju logove o korisnicima što su takođe metapodaci.

Ako do sada, kojim slučajem, niste uvideli opasnost po privatnost u ovim metapodacima, koje svako od nas smrtnika svakodnevno ostavlja za sobom u svetu interneta i ostalih digitalnih komunikacija, nadamo se da će vas ovaj primer makar podstaći na razmišljanje, a možda i na nešto više.

Zamislite da neko (zvaćemo ga posmatrač) stalno sakuplja metapodatke koje za sobom ostavljate kao mrvice hleba. Posmatrač, recimo, zna da se nalazite na autoputu nedaleko od famoznog Feketića, da se ne krećete, da zovete AMSS, zatim šlep službu i možda taksi, ali ne zna šta ste razgovarali. Ili posmatrač vidi da ste prvo zvali aerodrom Nikola Tesla, pa odmah potom taksi, ali ne zna šta ste razgovarali. Ili posmatrač vidi da ste zvali prvo šest brojeva čije identitete takođe zna, pa vidi da se oni zatim kreću ka vama - *GPS* (eng. *Global Positioning System*), pa zatim pica servis, Mekdonalds, i neku poznatu prodavnicu kineske hrane, ali ne zna šta ste razgovarali.

Posmatrač ne mora da bude mnogo pametan niti da prisluškuje vaše razgovore da bi shvatio šta se dešava, ili šta će se desiti. Sasvim mu je jasno da vam je u prvom primeru auto stao na autoputu, zatim da ste se u drugom slučaju uputili na aerodrom i da možda putujete van zemlje, ili ste poslali taksi po nekoga sa aerodroma, a da u trećem slučaju okupljate žurku kod vas. Više možete pogledati na: <http://goo.gl/7vUjP>.

Iako su to „samo“ metapodaci, treba biti svestan da ti metapodaci o vama stvaraju jednu digitalnu sliku, ili bolje reći digitalni mozaik, i na neki način vas jedinstveno određuju. Pogotovo kada se planski prikupljaju na više nivoa (kompjuter, pametni telefon, pametni automobil i kreditna kartica), metapodaci čine vaš digitalni identitet. Ali, da stvar bude još gora, ta slika o vama ne mora nužno biti istinita, jer se lako može lažirati. Recimo, možete zameniti telefone ili automobile sa nekim namerno ili slučajno, ili vam neko može jednostavno ukrasti telefon, laptop, ili kreditnu karticu. Zanimljiv video: <http://goo.gl/KO1Kf>.

Evo jednog primera koji vas može uvući u velike nevolje. Recimo da prolazite nekom zabačenom ulicom i da vam neprimetno ispadne telefon, ali vi to ne primetite i nastavite dalje svojim putem. U toj ulici se samo nekoliko minuta kasnije desi neko ubistvo, i vas pozovu u policiju kao osumnjičenog. Ako kojim





## To su samo metapodaci?

Ljudi se konstantno prate, metapodaci se sakupljaju i to nije sada već ništa novo, ali možemo da se trudimo da iza sebe ostavljamo malo manje neželjenih metapodataka, i da ih takoreći čistimo iz podataka koje razmenjujemo. Na sreću postoje programi koji ovo dobro rade.

Najpre je potrebno da vidimo koje sve metapodatke neki fajl sadrži, kako bismo se uverili da ovi metapodaci stvarno postoje i da je fajl bezbedan nakon njihovog brisanja.

Ako koristite operativne sisteme GNU/Linuksa, treba da iz terminala preuzmete i instalirate program pod nazivom **exiftool**:

```
sudo apt-get install exiftool
```

Ovaj program će nam pružiti uvid u sve metapodatke koje fajl sadrži. Korišćenje je iz terminala, ali je zato veoma jednostavno, samo ga pokrenite na sledeći način:

```
exiftool File
```

ili

```
exiftool /putanja/do/fajla/počev/od/korenog/direktorijuma
```

Dobićete prikaz metapodataka koje je program pronašao. Sada je potrebno da vidimo da li je podatak „prljav“, tj. da li ima u njemu nešto da se briše, i da, ako ima, iz njega sve nepotrebno obrišemo. Tu će nam pomoći drugi program zvan **MAT** (eng. *Metadata Anonymisation Toolkit*) koga možete naći na sajtu: <https://mat.boum.org/> ili jednostavno za korisnike Ubuntua i Debijana u Softverskom centru, ili ga možete instalirati iz terminala:

```
sudo apt-get install mat
```

**MAT** će proveriti da li je podatak „prljav“ ili ne, i za vas ga očistiti od neželjenih metapodataka.

Kada očistite fajl, on je spreman za deljenje sa drugima, a vi ste bezbedni jer niko ne može reći da je fajl potekao sa vašeg računara.

## Internet mreže i komunikacije

# Izgubljeni heroji „Blečli parka”

**Autor:** Dejan Čugalj

**G.H. Hardi:** „Prava matematika nema uticaj na rat.” - odbrana matematičara, 1940. godina.



Bil Tat - britanski matematičar, verovatno niste čuli za njega, 1942. godine je izveo ogroman poduhvat koji je skratio rat dve godine (barem po rečima Ajzenhauera) i spasio milione života. Nažalost, njegova veličina je preminula 2002. godine obavijena velom tajne i nikada zvanično nije priznato njegovo dostignuće.



Tomi Flauers - bivši inženjer poštanskog saobraćaja, pretvorio je matematičke ideje Tata u prvi računar, a to nije bio „ENIAC”<sup>1</sup>. Preminuo je 1998. godine, a pretpostavljamo da nikada niste čuli ni za njegovu veličinu.

---

<sup>1</sup> engl. **Electronic Numerical Integrator And Computer** - prvi digitalni elektronski računar

Ovi umni giganti „Blečli parka” omogućili su Britaniji da dešifruje strogo poverljivu mašinu koju je koristio Hitler da diriguje drugim svetskim ratom, a to nije bila „ENIGMA”, nego nešto mnogo tajnije i značajnije. Velika je verovatnoća da niste čuli ni za ovo. Neki kažu da je to bio Hitlerov Blekberi (*BlackBerry*).



## Izgubljeni heroji „Blečli parka”

\* \* \*



1939. godine „Blečli park” je postao ratni štab *M16* (britanske vojne obaveštajne službe). Do danas je procurelo u javnost i dokumentovano brojnim filmskim dokumentarcima samo da je tu Alan Tjuring uspeo da dešifruje nemačku mornaričku šifru poznatu kao „ENIGMA” i uveliko doprineo pobedi saveznika, a to je samo deo priče koji je ispričan.

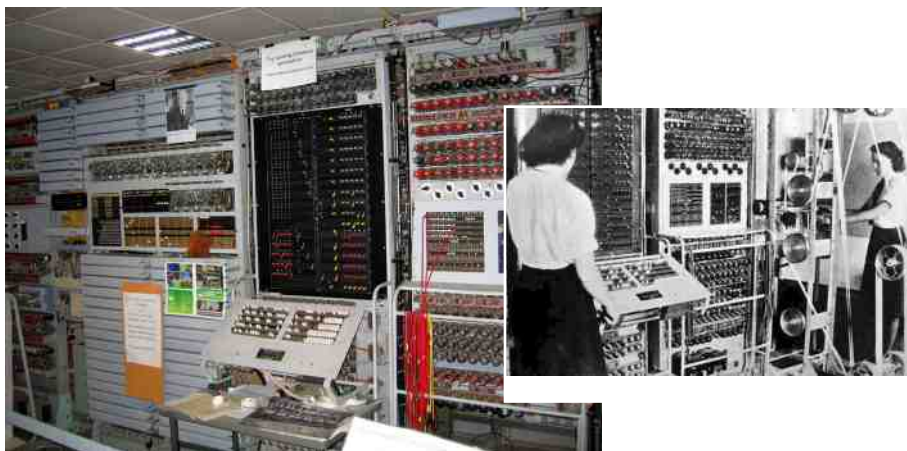


U Blečli parku su bila tri heroja: Alan Tjuring, Bil Tat i Tomi Flavers. Od njih trojice jedino Alanu Tjuringu je priznat deo zasluga za pobedu saveznika u Drugom svetskom ratu. Pregledom otkrivenih podataka, danas možemo da tvrdimo da je „ENIGMA” već sa početka drugog svetskog rata bila zastarela tehnologija. Zbog toga je bilo lako skinuti veo tajne sa

ovog uspeha heroja Blečli parka.

## Internet mreže i komunikacije

Bil Tat, koji je dešifrovaio „TANI sistem“, (TANI) i Tomi Flauers, koji je samo svojom intuicijom i znanjem sklopio prvi računar na svetu („KOLOS“) u svrhu bržeg dolaženja do informacija razbijenog koda „TANI“ tako da zastarelost šifrata bude svedena na minimum - radili su na novoj generaciji mašina za šifrovanje. Ova tehnologija je bila prepoznata kao tehnologija budućnosti pa i danas iznad informacija o tim tehnologijama stoji natpis „vrhunska tajna“ (eng. *Top Secret*). Upravo ovaj veo tajni oko šifrata „TANI“ i prvog računara je rezultirao iskrivljenju istorije informatike.



Za uspehe u prvim godinama rata je zaista zaslužno razbijanje „ENIGMA“ šifrata i to nije sporno, ali krajem 1941. godine u etru planete se začuo novi zvuk. To je bila nova mašina za šifrovanje; nije bila zasnovana na „Morzeu“, već je radila na principu teleprinteru.

Jedna od ključnih ideja Hitlerovog ratovanja je bila u velikoj mobilnosti i brzom delovanju trupa na terenu, a da bi se to izvelo, komunikacija je isto tako morala da bude u najmanju ruku putem radio signala.

Razlika je ogromna. Razlika je u količini podataka koji može da se prenese ovim „tajnim“ putem. Prvi svetski rat je bio zanimljiv jer su se šifrovane poruke bazirale na rečima, dok je Drugi u potpunosti otvorio vrata današnjoj kriptografiji, a to je matematika.

Originalan nemački kôdni naziv za ovaj šifrat bio je „LORENC“. Britanska





## Izgubljeni heroji „Blečli parka”

obaveštajna služba mu je dala nadimak „TANI”. Čak ni dan danas nije baš poznato kako je sve nastalo, ali pouzdano se zna da je tajnu mašinu Hitler nazvao „Gehajmsrajber” (nem. *Geheimschreiber*) - mašina tajni.



U srž i tehnologiju šifrovanja „LORENC” šifrata nećemo ulaziti u ovom članku, ali verujte nam na reč - nije bila jednostavna. Može se pretpostaviti da je inspiracija Šenonovog <sup>2</sup> rada slučajnosti bila upravo ova tehnologija iz Drugog svetskog rata.

Ono čime se Treći rajh vodio je „sigurna” šifrovana veza, koju niko nije smeo da dešifruje. Prenosnje šifrovanih poruka pomoću radio signala je dalo obaveštajnim službama mogućnost presretanja poruka poslatih radio talasima. Da li se ovo i danas dešava? Imamo li sigurnost u našim šifrovanim podacima današnjice?

Ono što je usledilo nakon proboja ovog „neuhvatljivog” šifrata je upravo ono što je princip današnjice, kao neka vrsta protokola koji mora da se ispoštuje u perfektnom šifarskom Šenonovom sistemu:

1. NIKADA, NIKADA, NE KORISTITI DVA PUTA ISTI KLJUČ ZA ŠIFROVANJE.
2. NIKADA, NIKADA, KLJUČ NE SME DA SE ŠALJE ISTIM KOMUNIKACIONIM KANALOM KAO PORUKA.
3. NIKADA, NIKADA, KLJUČ NE SME DA BUDE MANJI NEGOTI ŠTO JE PORUKA (engl.

---

<sup>2</sup> **Klod Elvud Šenon** (eng. *Claude Elwood Shannon*) (1916-2001) - Američki naučnik i inženjer



## Internet mreže i komunikacije

*one time pad*).

4. NIKADA, NIKADA, NE SMEMO DA SAKRIVAMO SISTEM ŠIFROVANJA, SVE MORA DA JE OTVORENOG KÔDA.

Najveći propusti kriptologije, koji su uočeni baš u Drugom svetskom ratu, jesu pokušaj sakrivanja sredstava kojim se generiše tajna poruka i nepoznavanje Šenonovih krajnjih granica informacija. Kada kažemo „sredstvo i sakrivanje“, mislimo na algoritme. Postavlja se pitanje zašto je danas kriptologija sigurnija nego četrdesetih i pedesetih godina prošlog veka?

Danas, uprkos tome što su algoritmi za kriptografiju otvorenog kôda, tajnost informacija je na višem nivou nego što je to bilo dok je algoritam bio strogo poverljiv. Tajnost tajnih podataka se ne zasniva na tajnosti algoritama koji generišu taj isti „tajni izlaz“ (eng. *cipher text*) nego u nemogućnosti matematičke inverzne funkcije koja bi šifrovan tekst otvorila. Ovde se vidi da je svrha strukture otvorenog kôda kao jednog od značajnih delova današnje kriptologije ona koja ga čini sigurnim i svima dostupnim.

Sada, kada se sa ove distance osvrnemo na Drugi svetski rat, možemo videti koliko smo bili blizu apokalipse. Naime, Albert Ajnštajn je već 1944. godine imao u rukama recept za nuklearnu bombu, Šenon je mogao već u to vreme da nam predoči 100% siguran šifrat (tzv. Hitlerovu super šifru), ali ipak, sreća nas je poslužila da čovečanstvo tada nije bilo spremno da upotrebi svu raspoloživu tehnologiju.

| **Albert Ajnštajn**, „Četvrti svetski rat će se voditi toljagama.”

\* \* \*

Zaboravljeni heroji Blečli parka, iako su odigrali važnu ulogu u pobedi saveznika u svetskom ratu, postali su žrtve „top secret” doktrine, potpuno su zaboravljeni i nestali su iz istorije.

Zaključak ovog članka je da je ideologija otvorenog kôda sasvim sigurno tu da nam pomogne i da otvori vrata ka većoj sigurnosti, novim idejama i kvalitetu same infrastrukture sistema koju svi zajedno razvijamo.



L<sup>A</sup>T<sub>E</sub>X prezentacija:

## Beamer (4. deo)

**Autor:** Nikola Hardi

### Nabrajanja

#### Bez brojeva

Za nabranje pojmov u Bimeru (eng. *Beamer*) koristi se okruženje **itemize**, a unutar tog okruženja možemo da dodajemo nove stavke pomoću **\item**. Primer izgleda ovako:

```
\begin{itemize}
  \item Stavka A
  \item Stavka B
  \item Stavka C
\end{itemize}
```

- ▶ Stavka A
- ▶ Stavka B
- ▶ Stavka C

#### Sa brojevima

Postoje situacije kada uz nabranje treba da stoje i redni brojevi - za to se koristi **enumerate** okruženje, a nove stavke se takođe dodaju pomoću **\item**.

```
\begin{enumerate}
  \item Redni broj 1
  \item Redni broj 2
  \item Redni broj 3
\end{enumerate}
```

1. Redni broj 1
2. Redni broj 2
3. Redni broj 3

## Sam svoj majstor

### Definicije

Treća situacija je nabranje parova termin - definicija. U tom slučaju se koristi okruženje **description**. Već možete da pretpostavite kako se dodaju nove stavke.

Primer:

```
\begin{description}
  \item[Pojam A] Definicija izmišljenog pojma A
  \item[Pojam B] Definicija izmišljenog pojma B
  \item[Pojam C] Definicija izmišljenog pojma C
\end{description}
```

Pojam A Definicija izmišljenog pojma A  
 Pojam B Definicija izmišljenog pojma B  
 Pojam C Definicija izmišljenog pojma C

### Ostala podešavanja i zanimljivosti

Sva prethodna okruženja mogu da budu ugnježena jedno unutar drugog, do trećeg nivoa. Treba biti pažljiv jer slajdovi vrlo lako mogu da postanu nečitljivi. Osim toga, moguće je izabrati i format oznake ili rednih brojeva - na primer ovako:

```
\begin{enumerate} [I]
\begin{enumerate} [a]
```

Takođe, moguće je upravljati pojavljivanjem pojedinih stavki na slajdovima. Drugim rečima, možemo da odredimo da se nabranja postepeno otkrivaju tako što ćemo uz stavku dodati **<1->**, odnosno broj od kojeg slajda želimo da se ta stavka pojavljuje. Tada će jedno nabranje biti „razvučeno” na više slajdova. Jednostavnija mogućnost je da u zaglavlje okruženja dodamo **{+-}**, što znači da želimo da se svakim slajdom otkriva po jedna nova stavka iz nabranja. Znak „-” u oba primera znači da želimo da se stavka pojavi na sledećem slajdu i da ostane



prisutna. Možemo da izostavimo znak „-“ i tada će stavka biti prisutna samo na jednom slajdu, ili da napišemo raspon slajdova u kojima želimo da ta stavka bude prisutna.

- I Redni broj 1
- II Redni broj 2
- III Redni broj 3
- IV Redni broj 4

## Rad sa kolonama i blokovima

Do sada su svi primeri podrazumevali da je sadržaj napisan preko cele širine slajda, ali Bimer (eng. *Beamer*) ima i jednostavne mehanizme za razmeštanje sadržaja po kolonama. Za rad sa kolonama se koristi okruženje **columns**, unutar kojeg možemo da dodajemo nove kolone pomoću **column{širina}**. Sadržaj je moguće potom pisati i van okruženja sa kolonama.

U paru sa kolonama, često se sreće i okruženje **block**, koje se parametrizuje naslovom. Primerom su ilustrovani i okruženje **column** i okruženje **block**.

```
\begin{frame}
\begin{columns}
\column{0.5\textwidth}
\begin{block}{Naslov levog bloka}
Sadržaj u prvoj (levoj) koloni \\
\end{block}

\column{0.5\textwidth}
\begin{block}{Naslov desnog bloka}
Sadržaj u drugoj (desnoj) koloni \\
\end{block}
\end{columns}

\begin{block}{Van kolona}
```

## Sam svoj majstor

Tekst koji se nalazi ispod kolona.

```
\end{block}
\end{frame}
```

Naslov levog bloka

Sadržaj u prvoj (levoj) koloni

Van kolona

Tekst koji se nalazi ispod kolona.

Naslov desnog bloka

Sadržaj u drugoj (desnoj) koloni

## Tabele

Iako se u radu sa prezentacijama sadržaj najčešće može uredno rasporediti pomoću kolona i nabravanja, nekada to nije dovoljno jer su tabele pravi format. Bimer podržava standardno Lateh (LaTeX) okruženje za rad sa tabelama. Okruženje za rad sa tabelama se zove tabular. Pri kreiranju nove tabele moguće je podesiti koje kolone će biti razdvojene linijom i da li će njihov sadržaj biti centriran, ili uz levu ili uz desnu ivicu. Dodavanje novog reda se jednostavno radi dodavanjem **\hline**. Lateh pruža zaista jako fleksibilan rad sa tabelama i sigurno će doskočiti svim mogućim i nemogućim zamislima. Potražite bilo koje Lateh uputstvo za rad sa tabelama i radiće i sa Bimerom.

```
\begin{frame}
  \begin{tabular}{| c | c | c | c |}
    \hline
      & Ponedeljak & & Utorak & & Sreda \\ \hline
      1 & 16 & & 32 & & 64 \\ \hline
      2 & 16 & & 128 & & 1024 \\ \hline
      3 & 16 & & 512 & & 4096 \\ \hline
    \end{tabular}
  \end{frame}
```



	Ponedjeljak	Utorak	Sreda
1	16	32	64
2	16	128	1024
3	16	512	4096

## Teoreme i istaknut tekst

U nastavi su česta izlaganja teorema, njihovih dokaza i primera. U Bimer je ugrađena podrška za elegantno predstavljanje ove trojke. Mehanizam takođe podržava postepeno otkrivanje sadržaja slajda, tako da slušaocima ne bude odmah otkriven dokaz teoreme. Mehanizam za postepeno otkrivanje se, kao i kod nabiranja, postiže pomoću „<1->” konstrukcije. Evo kako to izgleda:

```
\begin{frame}
\frametitle{Primer teoreme}

\begin{theorem}<1->    Teorema kaže da su beamer prezentacije
zanimljive.
\end{theorem}

\begin{example}<2->    Ovaj kod kreira slajd beamer prezentacije.
\end{example}

\begin{proof}<3->    Čestitamo, stigli ste do kraja serijala o \\\
    pravljenju beamer prezentacija.
\end{proof}
\end{frame}
```

## Sam svoj majstor

### Theorem

*Teorema kaže da su beamer prezentacije zanimljive.*

### Example

Ovaj kod kreira slajd beamer prezentacije.

### Proof.

Čestitamo, stigli ste do kraja serijala o  
pravljenju beamer prezentacija.



## Za kraj

Ovim člankom završavamo serijal o Bimeru. Nadamo se da smo uspjeli da vas zainteresujemo za ovu vrlo korisnu klasu Lateha. Trudili smo se da vam prezentujemo najvažnije delove Bimera koje ćete najčešće koristiti prilikom kreiranja svojih prezentacija.

Napominjemo još jednom da je Bimer samo jedna od klasa Lateha što znači da i za Bimer važi sve ono što ste već mogli da pročitate u ovom časopisu kad smo pisali serijal „Uvod u *LaTeX*” (LiBRE! brojevi 17 do 21). Štaviše, ovaj serijal o Bimeru je bio nastavak upravo tog serijala. U ovom serijalu nismo pisali o samom formatiranju teksta u prezentacijama jer se podrazumeva da važe ista pravila kao i u Latehu, a o tome smo pisali u prethodnom serijalu.

Hvala vam što ste pratili ovaj serijal i želimo vam puno uspeha u kreiranju Bimer prezentacija.

# L<sup>A</sup>T<sub>E</sub>X





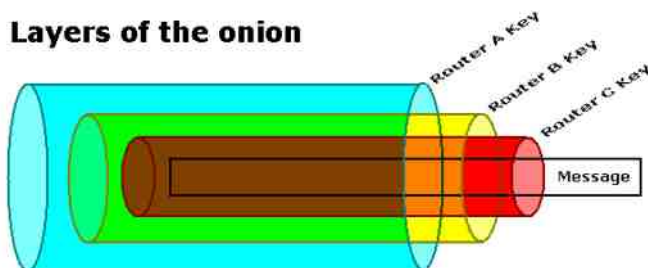
## BeagleBone Black Rev C

# Vodič od prvog dana (3. deo) - BeagleBone Black kao Tor relej

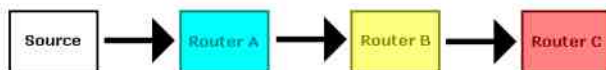
**Autor:** Nenad Marjanović

Kada smo se upoznali sa podešavanjima sistema i umrežavanjem, vreme je za naš prvi Biglbon Blek (BBB - eng. BeagleBone Black) projekat. Svakom od nas će ovaj izazov dati različite ideje, a jedna od njih može biti i zaštita privatnosti internet korisnika. Nažalost, u Srbiji ne postoji nijedan izlazni tor server (eng. *node*) i to je pomalo tužna slika s obzirom da govorimo o populaciji od sedam miliona stanovnika. Ali ovu sliku možemo promeniti udruženim snagama i, za početak, kreiranjem Tor transfer i izlaznog servera. U ovde prikazanom grafikonu fokusiraćemo se na „Router B” tačku, odnosno transfer server konfiguraciju.

### Layers of the onion



### Routing path



## Hardver

Tor predstavlja mrežu servera koji se povezuju radi bolje enkripcije podataka i zaštite IP adrese korisnika. Naš cilj je da danas postanemo deo ovog predivnog projekta i da se nađemo na svetskoj karti Tor servera.

## Brzina interneta i zakonske regulative

Potrebno je imati na umu nekoliko bitnih stvari:

1. Naš server = naša odgovornost
2. Obavezno naznačite da ste vi administrator a ne korisnik istog servera
3. Da posedujete internet brzinu od 2 Mb/s

Ako ispunjavamo tri navedena pravila i odgovornosti, možemo se posvetiti administrativnim poslovima.

## Instalacija i podešavanja

U ovom primeru koristimo Debijan Vizi (eng. Wheezy), odnosno Debijan 7 verziju na našem BBB uređaju. Vi možete testirati i druge Linuks platforme. Prvo dodajemo izvornu biblioteku u **/etc/apt/sources.list** fajl pokretanjem sledeće komande u terminalu:

```
echo "deb http://deb.torproject.org/torproject.org wheezy main" > \
/etc/apt/sources.list.d/tor.list
```

Zatim dodajemo izvorni ključ, vršimo ažuriranje sistema i instaliramo **tor** i **tor-arm** (Tor monitoring sistem).

```
gpg --keyserver keys.gnupg.net --recv 886DDD89
gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | sudo apt-key
add -
apt-get update
apt-get install deb.torproject.org-keyring -y
apt-get install tor tor-arm egrep screen -y
```

Ovim smo uspešno završili instalaciju Tor servera i sistema za monitoring, odnosno nadgledanja našeg Tor servera. Sledećom komandom radimo na podešavanjima Tor servera koja se nalaze u **/etc/tor/torrc** fajlu. Obratite pažnju i izvršite potrebne



izmenu u poljima *IP* adresa (eng. *Address*), nadimka (eng. *Nickname*) i kontakt podataka (eng. *ContactInfo*) pre pokretanja ove komande. Kopiramo i pokrećemo komandu u celosti u terminalu.

```
egrep -v "^(#|$)" /etc/tor/torrc
ORPort 443
Address 192.168.1.1
Nickname LibreTorRelay
RelayBandwidthRate 100 KB
RelayBandwidthBurst 120 KB
ContactInfo <vi at gmail dot com> irPort 8080
DirPortFrontPage /etc/tor/tor.html
ExitPolicy reject *:*
```

Detalje o svim vrednostima nećemo iznositi u ovom broju LiBRE! časopisa, ali ćemo o njima pisati u sledećem, u kome ćemo pričati o zaštiti Tor servera i mogućnosti podešavanja određenih izlaznih pravila (eng. *ExitPolicy*).

Nakon ovoga možemo primetiti da se u folderu **/etc/tor/** ne nalazi **tor.html** fajl. Ovaj fajl nije od velikog značaja i možete ga lično napraviti, ili preuzeti jednu kopiju - naprimer, sa sledeće lokacije:

```
cp /usr/share/doc/tor/tor.html /etc/tor/tor.html
```

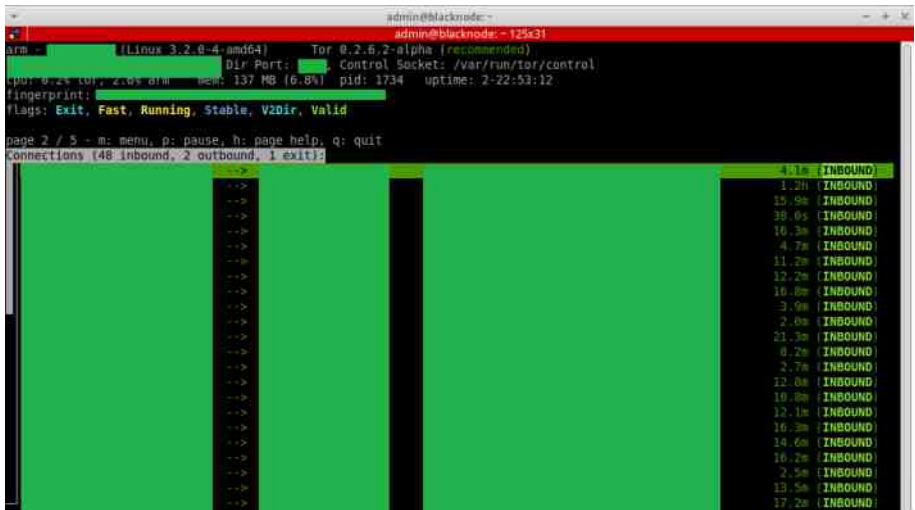
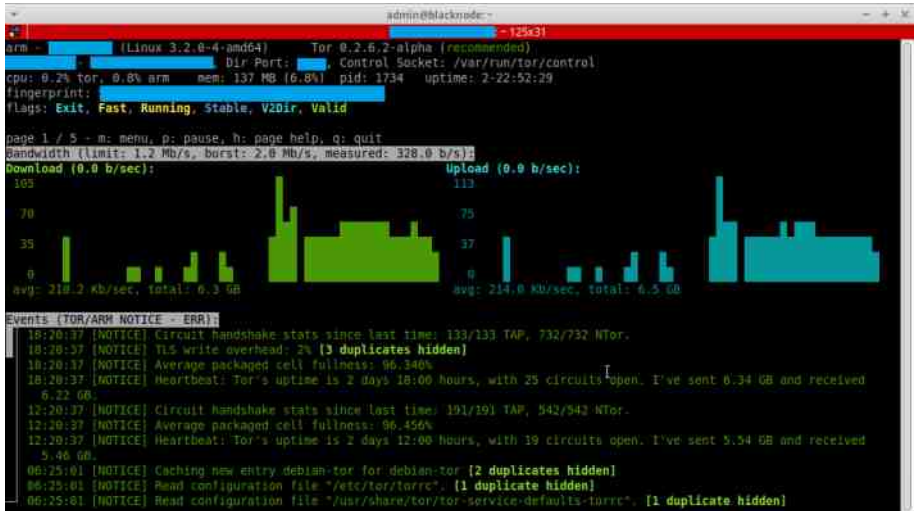
## Opcije i Tor Arm monitoring

Da bismo ispravno koristili Tor Arm, potrebno je urediti još jednom **/etc/tor/torrc** fajl i restartovati servis.

```
echo -e "DisableDebuggerAttachment 0\n AvoidDiskWrites" >> /etc/tor/torrc
sudo service tor reload
```

Ukoliko želite koristiti Tor Arm za nadgledanje aktivnosti na serveru pokrenite program **screen**, kliknite na „Enter” i kucajte **sudo -u debian-tor arm**. Na ekranu ćete imati sličan prikaz:

# Hardver





```

admin@blacknode:~$
admin@blacknode:~$ ./tor
(Linux 3.2.0-4-amd64) Tor 0.2.6.2-alpha (recommended)
Dir Port: 8080 Control Socket: /var/run/tor/control
cpu: 0.2% tor, 1.2% arm mem: 137 MB (6.8%) pid: 1734 uptime: 2-22:53:45
Fingerprint: [redacted]
Flags: Exit, Fast, Running, Stable, V2Dir, Valid

page 3 / 5 - m: menu, p: pause, h: page help, q: quit
Tor Configuration (press 'a' to show all options):
BandwidthRate (General Option)
Value: 1 GB (default, DataSize, usage: N bytes|KBytes|MBytes|GBytes|KBits|MBits|GBits)
Description: A token bucket limits the average incoming bandwidth usage on this node to the specified number of bytes per
second, and the average outgoing bandwidth usage to that same value. If you want to run a relay in the public network,
this needs to be at the very least 30 KBytes (that is, 30720 bytes). (Default: 1 GByte)

BandwidthRate 1 GB Average bandwidth usage limit
BandwidthBurst 1 GB Maximum bandwidth usage limit
RelayBandwidthRate 160 KB Average bandwidth usage limit for relaying
RelayBandwidthBurst 260 KB Maximum bandwidth usage limit for relaying
ControlPort <none> Port providing access to tor controllers (arm, vidalia, etc)
WashedControlPassword <none> Hash of the password for authenticating to the control port
CookieAuthentication True If set, authenticates controllers via a cookie
DataDirectory /var/lib/tor Location for storing runtime data (state, keys, etc)
Log notice file... Runlevels and location for tor logging
RunAsDaemon True Toggles if tor runs as a daemon process
User debian-tor UID for the process when started
Bridge <none> Available bridges
ExcludeNodes <none> Relays or locales never to be used in circuits
MaxCircuitDirtyTime 10 minutes Duration for routing constructed circuits
SocksPort <none> Port for using tor as a Socks proxy
UseBridges False Make use of configured bridges

```

Da bismo napustili **screen** program, koristimo **Ctrl + a + d** i, na taj način, kada se sledeći put prijavimo na server, kucamo samo **screen -r** u terminalu - i naći ćemo se ponovo na čuvenom Tor Arm monitoring ekranu.

Nakon svake promene konfiguracionih fajlova, ako slučajno nismo nešto podesili kako treba, restartujemo service.

U sledećem broju časopisa govorićemo o minimalnim podešavanjima i konfigurisanju izlaznog Tor servera (eng. *Tor Exit Node*), opcijama, sigurnosti, i sličnim detaljima u vezi sa našim prvim projektom. Do čitanja u sledećem broju!



beaglebone

# LUGoNS BarCamp №4

subota 7. mart 2015. godine



**Četvrti LUGoNS-ov Barkemp (BarCamp) održaće se u Novom Sadu u subotu 7. marta 2015. godine na Fakultetu tehničkih nauka sa početkom u 12 časova.**

**Ulaz je slobodan!**

## Poziv za predaju radova

LUGoNS poziva sve zainteresovane da pošalju svoje radove (predavanja, prezentacije, radionice i diskusije) do **28. februara 2015. godine.**

Oblasti iz kojih možete da prijavite vaše radove su:

- Slobodan softver
- Zanimljiv hardver
- Hakerski zakoni
- Sigurnosne noćne more
- Anonimnost i privatnost na internetu
- Duboki veb (eng. *Deep Web*)
- Socijalne mreže
- Socijalni inženjering
- Hakovanje mobilnih uređaja
- Internet u oblaku (eng. *Clouds*) – hakovanje, razbijanje i neočekivano korišćenje
- *DPI* (eng. *deep packet inspection*)
- Mreža i mrežna neutralnost – vlasništvo, cenzura, nadmudrivanje i politika „de fakto” standarda
- Programiranje i programski jezici

Nemojte se ustručavati i slobodno nam se javite ako imate rad na gore navedene ili slične teme koji smatrate interesantnim.

Stranica za predaju radova nalazi se na: <https://events.lugons.org/?p=1658>

