Maj 2014.

LiBRE!

Časopis o slobodnom softveru

broj 25

7. maj 2014.
Prvo javno izdanje *LXQt*-a dostupno je za preuzimanje.

31. maj, 2014.
Objavljen je *Linux Mint 17 „Qiana“*.

Reč urednika

Mediji, *FLOSS*, LiBRE! i prirodna katastrofa

Priroda, s vremena na vreme, podseti čoveka koliko je mali, koliko su njegove rukotvorine beznačajne i podseća čoveka da nije gospodar sveta, nego smo jedan njegov mali deo. Suočen sa snagom prirode, čovek zaboravlja stečene navike i oslanja se na bazične instinkte. Kada se nešto ovako katastrofalno desi, „resetovani" na osnovne instinkte, pokazujemo ko smo zapravo, koliko znamo tj. ne znamo, koliko smo spretni, organizovani, gde smo ranjivi, kako smo pripremljeni, šta nismo na vreme uradili, a trebalo je, šta smo uradili, a nismo smeli pa sad trpimo posledice i tako dalje.

Glavna opasnost od prirodne nepogode je prošla. Preživelima je sada ostalo da analiziraju šta se desilo, da popišu gubitke i štetu i da svi zajedno počnemo da se obnavljamo. U ovu analizu dešavanja treba uključiti i upotrebu medija s ciljem kriznog informisanja. Pravovremena, tačna informacija je bila od ključnog značaja za ispravno reagovanje s ciljem preživljavanja.

Ovo nije prva vanredna situacija u kojoj je najstarija komunikaciona tehnologija bila od ključne važnosti. Radio tehnika i radio amateri još jednom su pokazali da su nezaobilazan faktor dobre civilne zaštite. Prosto je neverovatno da niko iz Ministarstva odbrane nikad nije uključio te ljude u institucionalni sistem odbrane, iako su se u mnogo prilika do sad pokazali kao vrlo pouzdani sistem informisanja u kriznim situacijama. Oni ne traže mnogo, nikad i nisu tražili gotovo ništa, a uvek su na prvim linijama odbrane.

Radio i televizija su se uglavnom dobro pokazali u ovim vremenima. Radio a naročito televizija pokazali su još jednom da su najmoćniji mediji na našim prostorima. Radio i televizija bili su glavni informativni, motivacioni i mobilizacioni faktor u kriznoj situaciji.

Najnovije tehnologije su podbacile. Ni internet ni mobilna telefonija nisu bili iskorišćeni u meri koliko je to moguće u širenju prave informacije. Moglo bi se reći da su te tehnologije bile čak zloupotrebljene za širenje lažnih i „destabilišućih" informacija. Ova kritika ide direktno na račun Vlade Srbije i državnih organa. Državni organi još nisu shvatili moć *ICT*-a. Prepuštanje *ICT*-a stihiji i nepostojanje zvaničnih informacija mogu samo da dovedu do širenja dezinformacija na internetu, a internet je komunikacioni medij broj jedan u svetu. Dezinformacija na internetu će uvek biti. Setimo se vesti o pijanom Srbinu koji je ubio ajkulu slučajnim skokom na njenu glavu, čime je spasio kupače u Egiptu. Međutim, postoje pouzdani izvori na koje se obraća pažnja i ostali izvori koji se zanemaruju. Pouzdani izvori moraju biti stranice državnih organa i lokalnih samouprava koje su ovoga puta kasno reagovale, ili nisu reagovale uopšte, a to je otvorilo prostor nezvaničnim izvorima kao što su društvene mreže gde nisu svi dobronamerni.

Jedan od svetlih primera privatne inicijative je lokacija http://poplave.rs. Ovo je primer kako privatna inicijativa, dobra namera, odlična ideja, *FLOSS* i znanje mogu zajedno da iskoriste pravu snagu interneta. Stranica je bila toliko dobro odrađena i pregledna da smo svi u početku pomislili da je to zvanična stranica državnih organa. Stranica je bila važna za koordinaciju akcija dobrovoljaca. Jedini problem sa tom stranicom bio je preveliki promet koji je povremeno rušio server.

Gde je tu mesto *FLOSS*-u? Barem što se tiče interneta, *FLOSS* predstavlja najbolji alat za brze intervencije. *Wordpress*, besplatni *CMS* (eng. *Content Manager System* – sistem za upravljanje sadržajem) otvorenog koda sa svojim dodacima idealni su alati za brzo reagovanje i uspostavljanje informacionog sistema u najkraćem vremenu. U rukama dobrih *web* dizajnera predstavljaju moćan alat. To su momci i devojke iz građanskog pokreta *poplave.rs* iskoristili da u najkraćem roku, za manje od dvadeset i četiri sata, uspostave informacioni centar sa proverenim informacijama.

Na kraju, pohvale idu svim dobrovoljcima, radio amaterima, korisnicima *twittera*, građanskim inicijativama, savezima, državnim organima i svima drugima na dobrim akcijama koje su predupredile još gore posledice prirodne stihije.

Za LiBRE! je u ovom trenutku najbitnije da je preživeo i ovu prirodnu nepogodu. S obzirom na relativno niski prioritet rada u ovom projektu, dozvolili smo sebi kašnjenje od sedam dana da bismo uspeli da se pregrupišemo. Još nam se nisu javili svi saradnici. Nadamo se da su svi dobro. Čim se stanje smiri, moći ćemo ponovo da radimo punim kapacitetom. Pozivamo lektore, grafičare, autore i sve druge koji bi želeli da nam se priključe, da se jave na našu već poznatu adresu libre[et]lugons [dot]org.

Do čitanja

LiBRE! Tim

Moć slobodnog softvera

Broj: 25
Periodika izlaženja: mesečnik

Izvršni urednik: Stefan Nožinić

Glavni lektor: Aleksandar Božinović

Lektura:
Milena Beran
Jelena Munćan
Maja Panajotović
Aleksandra Ristović

Redakcija:
Aleksandar Stanisavljević
Dejan Čugalj **Sandrina Dimitrijević**
Goran Mekić **Aleksandar Todorović**
Marko Kažić **Nedeljko Stefanović**
Veljko Simić **Mihajlo Bogdanović**
Nikola Hardi **Gavrilo Prodanović**
Željko Šarić **Aleksandar Brković**
Danilo Đokić **Vladimir Cicović**
Petar Simović **Joakim Janjatović**
Romeo Mlinar **Stefan Stojanović**
Dejan Petrović **Marko Novaković**
Zlatan Vasović **Željko Popivoda**

Saradnici:
Velimir Baksa **Milovan Krivokapić**
Ivan Bulatović **Ladislav Urošević**
Tamara Đorđević **Bojan Bogdanović**
Vladimir Popadić

Grafička obrada:
Dejan Maglov **Ivan Radeljić**

Dizajn:
Mladen Šćekić **Zoran Lojpur**

Kontakt:
IRC: #floss-magazin na irc.freenode.net
E-pošta: libre@lugons.org

http://libre.lugons.org

LiBRE! sadržaj

LiBRE! prijatelji

ubuntu SRBIJA

Slackware Srbija

webserveri.info
info@webserveri:~$ Sve o serverima i programiranju

ubuntu LoCo Montenegro

LinuX User Group of Novi Sad
LUGoNS
Grupa korisnika GNU/Linux operativnog sistema u Novom Sadu

ЛИНог
Линукс блог | вести и мишљења

REGIONALNI
LINUX PORTAL
linuxzasve.com

archlinux.rs

LXQt - okruženje radne površi sledeće generacije
7. maj, 2014.

Prvo javno izdanje *LXQt*-a, sledeća generacija popularnog lakog (eng. *lightweight*) *Linux* okruženja radne površi *LXDE*, dostupno je za preuzimanje.

Koristan link: http://j.mp/1kn3lUO

Oracle dobio spor protiv Googla u vezi sa korišćenjem Jave na Androidu
12. maj, 2014.

Američki sud za žalbe je sada preinačio odluku iz 2012. godine, po kojoj struktura *Java API*-ja ne može biti zaštićena autorskim pravom. *Oracle* sada može da nastavi da od *Googla* traži otštetu u visini od milijardu dolara.

Koristan link: http://j.mp/1oLaLV2

Sledeća tri Linux Mint izdanja baziraće se na Ubuntu 14.04 LTS
14. maj, 2014.

Linux Mint 17, *17.1*, *17.2* i *17.3* koristiće *Ubuntu 14.04 LTS* kao bazu umesto da budu bazirani na novijim *Ubuntu* izdanjima.

Koristan link: http://j.mp/1kn3Iio

Plasma Next
14. maj, 2014.

Objavljeno je beta izdanje *Plasma Nexta*. *Plasma Next* se razvija sa planom da zameni aktuelnu *KDE Plasma* platformu.

Koristan link: http://j.mp/1gUU3jB

Objavljen je Deepin 2014 Beta
15. maj, 2014.

Deepin je *Linux* distribucija bazirana na *Ubuntuu*. *Deepin 2014* donosi novo grafičko okruženje pod nazivom *DDE* (*Deepin Desktop Environment*) koje je bazirano na *HTML5*.

Koristan link: http://j.mp/1n1l7Nz

Open source Novena laptop stiže na zimu
19. maj, 2014.

Novena laptop uspeo je da prikupi sredstva koja će mu omogućiti proizvodnju. Isporuke se očekuju za početak sledeće godine.

Koristan link: http://j.mp/1ky9xEY

Telenav prebacuje Scaut navigacionu aplikaciju na OpenStreetMap
20. maj, 2014.

Kompanija za navigaciju *Telenav* najavila je da će u svojim aplikacijama namenjenim američkom tržištu od sada koristiti *OpenStreatMap* umesto *TomToma*.

Koristan link: http://j.mp/1u61A1g

RawTherapee 4.1
22. maj, 2014.

Objavljena je nova 4.1 verzija *RawTherapeea*. *RawTherapee* je program za obradu *RAW* slika koji dolazi sa velikim brojem mogućnosti.

Koristan link: http://j.mp/1jKLIcW

Kina je zabranila Windows 8
20. maj, 2014.

Kineska Vlada je zabranila korišćenje *Windows 8* na zvaničnim Vladinim računarima. Kao razlog se navodi bezbednost računara. Pretpostavlja se da će to biti dobra prilika da *Ubuntu Kylin* zauzme mesto sad već nepodržanog *Windows XP*-a.

Koristan link: http://j.mp/1oLb9Ty

The Incredible Adventures of Van Helsing
23. maj, 2014.

Autori akcione *RPG* igre *The Incredible Adventures of Van Helsing* najavili su verziju za *Linux*.

Koristan link: http://j.mp/1rzFGpp

Zvanično je objavljen Git 2.0
29. maj, 2014.

Nova verzija donosi nekoliko suptilnih novosti koje stariji korisnici možda neće ni primetiti. Nove *default* funkcije su sada više „prijateljski" nastrojene prema *Git* početnicima.

Koristan link: http://j.mp/1oLbohm

Objavljen je Linux Mint 17 Qiana
31. maj, 2014.

Objavljen je *Linux Mint 17 LTS* koji će biti podržan do 2019. godine. Nova verzija donosi značajna unapređenja korisničkog interfejsa i menadžera ažuriranja.

Koristan link: http://j.mp/1kwYZei

Ugovor Republike Srbije sa *Microsoftom*

(7. deo)

FLOSS u preduzećima

Autor: Dejan Maglov

Do sada u ovom serijalu smo analizirali stanje primene *FLOSS*-a u Srbiji i preporučili smo šta bi trebalo uraditi da bi *FLOSS* imao značajnije mesto u korist svih. Smatramo da su naši predlozi dobri i da su realno ispunjivi ako bi se stekli određeni uslovi. Uslovi za ostvarivanje naših preporuka jesu politička volja, bolje informisanje, veći uticaj, bolja organizovanost i veća aktivnost *FLOSS* zajednica Srbije. Apeli nisu dovoljni za popularizaciju *FLOSS*-a. Svet pokreću interesi i to uglavnom ekonomski.

Kako to izgleda u svetu vlasničkog softvera?

Interes vlasnika softvera jeste da napravi dobar softver i da ga reklamiraju kao najboljeg za predviđenu funkciju. Cilj kompanija koje stoje iza vlasničkog softvera, jeste da zauzmu monopolistički položaj kako bi mogli svoj proizvod da iznajmljuju, a ne da ga prodaju. S tim ciljem prave se karteli kompanija koji na bazi međusobnog dogovora obezbeđuju sebi monopolsko mesto na tržištu. Postoji dogovor kompanija vlasničkog softvera kao i kompanija koji proizvode hardver o zajedničkom nastupanju na tržištu. Kompanije koje zajedno nastupaju, međusobno ne konkurišu jedna drugoj. Korisnik je tako prinuđen da, ukoliko želi da koristi određeni proizvod, kupi ili iznajmi proizvode još pet-šest kompanija koje su spregnute u tom lancu. Kartelsko povezivanje je ilegalno u zapadnom kapitalizmu, ali je teško je dokazati da je ovo sprezanje namerno. Praćenje dešavanja u *IT* industriji jasno navodi na sumnju da kartel postoji.

Sledeći u nizu interesa jesu dileri. Njihovo zaduženje je da kontrolisano distribuiraju hardver i softver na određenoj teritoriji. Oni se trude da klijentima uvek, kada je to moguće, ponude paket hardvera i vlasničkog softvera. Njihov interes je da prodaju što više ovih proizvoda i da uzmu što veću maržu. I oni, takođe, imaju interes da iznajmljuju vlasnički softver umesto da ga trajno prodaju.

Krajnji korisnik je poslednji u tom nizu interesa koji plaća sve. Ovde moramo da stanemo i razdvojimo pravna od fizičkih lica. I jedni i drugi imaju interes da koriste računare kako bi povećali svoju produktivnost i olakšali sebi rad.

Razlika između pravnih i fizičkih lica je u tome što fizička lica niko ne proverava šta dalje rade sa svojim hardverom i softverom dok se ne pojave na tržištu sa nekim proizvodom koji je produkt tog hardvera i softvera. Ovo znači da fizička lica mogu nekažnjeno da koriste piratske kopije softvera dok ih koriste u lične svrhe ili da koriste legalni vlasnički softver i nakon isteka perioda iznajmljivanja. Fizičkim licima se nude i jeftinije licence po principu „ako prođe, prošlo je" (APP). Kompanijama je jasno da je insistiranje da fizička lica poštuju licencu kontraproduktivno. Za njih je bolje da fizička lica koriste piratsku kopiju njihovog proizvoda, nego da ga uopšte ne koriste. Iako na pirateriji gube veliki prihod, na ovaj način se stvara zavisnost od njihovih proizvoda tako da na strani pravnih lica imaju veću zastupljenost i veliki procenat naplate.

Pravna lica nemaju izbor ukoliko koriste vlasnički softver. Oni moraju da budu legalni jer ih na to teraju državni organi koji su poslednja ključna karika u lancu interesa. Ne samo da pravna lica moraju da plate sve licence, nego su čak te licence skuplje za njih u odnosu na fizička lica. Koji su onda interesi pravnih lica da koriste vlasničke softvere? Tu postoje objektivni i subjektivni razlozi ali i razlozi koji su plod zabluda i loše informisanosti.

Objektivni razlozi za korišćenje vlasničkog softvera su:

- Nepostojanje *FLOSS* alternativa za specifični softver potreban za obavljanje delatnosti.
- Ne postoje *FLOSS* upravljački programi (eng. *drivers*) za specifični hardver potreban za obavljanje delatnosti.
- Potreba za komunikacijom sa saradnicima, klijentima, bankom i državnim organima koji koriste vlasnički softver i formate koji su vezani samo za vlasnički softver.

Subjektivni razlozi su:

- Stvorena navika da se koristi isključivo vlasnički softver.
- Nepoznavanje *FLOSS*-a i njegovih mogućnosti odbija eksperimentisanje u uslovima poslovnog ambijenta gde je vreme jednako novac.
- Nedovoljna ponuda obučenih radnika koja bi radila na *FLOSS* rešenjima.

FLOSS još uvek prate gomile zabluda koje odbijaju korisnike:

- *FLOSS* je pretežak za korišćenje i održavanje.
- Prelazak na *FLOSS* rešenja smanjuje produktivnost.
- *FLOSS* nije bezbedan jer iza njega ne stoji firma koja ga održava.
- Model razvoja *FLOSS*-a ne pruža nikakve garancije da će određeni proizvod nastaviti da se razvija.

Ovakvo stanje „cementira" država svojom pravnom regulativom, svojim informacionim sistemom i inspekcijskim nadzorom. Jedini interes države jeste da obezbedi sebi porez od prodaje vlasničkog softvera i profita koji se na taj način ostvaruje. Ovde se krije privid da država zarađuje na vlasničkom softveru. Sve dok Srbija koristi vlasnički softver stranih kompanija, država i njen budžet neće biti na dobitku nego na čistom gubitku. Svi ti prihodi koji se ostvaruju na osnovu naplate poreza na vlasnički softver, ne podmiruju troškove licenci vlasničkog softvera za računare u državnim organima ni troškove inspekcijskog nadzora.

Analiza interesnog lanca vlasničkog softvera

Analizom interesnog lanca vlasničkog softvera može se zaključiti da stvarni ekonomski interes za korišćenje vlasničkog softvera imaju samo kompanije koje su vlasnici tog softvera i njihovi dileri. Krajnji korisnici imaju delimični interes jer obavljaju uspešno svoju delatnost, ali svoj profit umanjuju plaćanjem iznajmljivanja „alata" za rad.

U uslovima kada kompanije vlasničkog softvera nisu na teritoriji države korisnika tih softvera, država može samo da gubi jer ne uzima porez na ukupnu dobit te kompanije. Porez od distribucije nije dovoljan, jer on samo smanjuje gubitke u budžetu. U ovom lancu interesa nešto nije u redu. Kako je moguće da ovaj interesni lanac opstaje, a da je jedna karika (država) na gubitku, a druga (krajnji korisnik), vrlo važna, duplo plaća (kroz direktno plaćanje licenci i posredno kao budžetski obveznik) i time značajno umanjuje svoj profit?

Lanac interesa za korišćenje *FLOSS*-a

Mi, kao bivša socijalistička država, potpuno smo se pogubili u tranziciji. Malo smo pozaboravljali definicije društvenih uređenja. Jureći za boljim životom, izgubili smo i ono što smo imali. Sredstva za rad su iz ruku radnika uz njihovo prećutno odobrenje prešla u ruke države da bi ih i država za male pare predala u ruke kapitaliste

(vlasnika preduzeća). Definicija kapitalizma glasi da je to takav tip društvenog uređenja u kojem je kapitalista vlasnik sredstava za proizvodnju i da na osnovu tog vlasništva ostvaruje pravo na dobit. Sad dolazimo do apsurda. Modernizacijom sredstava za proizvodnju naši kapitalisti su uveli kompjutere u sve segmente poslovanja. Kompjuteri pak, koriste vlasnički softver koji su po zakonu o intelektualnoj svojini uvek u vlasništvu vlasnika softvera, prema tome, naš kapitalista nije u potpunosti vlasnik sredstava za proizvodnju i zato svoju dobit mora da deli sa pravim vlasnikom.

Ako naš kapitalista želi da zadrži svu dobit od svog posla, mora da povrati vlasništvo nad sredstvima za proizvodnju. Trebalo bi da to bude jedan od osnovnih interesa za korišćenje *FLOSS*-a u poslovne svrhe. Nekad zbog specifičnosti posla i okruženja nije moguće koristiti *FLOSS* rešenja, ali u principu u interesu vlasnika jeste da koristi, kad god je to moguće, softver koji je pod javnom licencom i omogućava mu da povrati vlasništvo nad sredstvima za proizvodnju.

Iznajmljivanje sredstava za proizvodnju (vlasničkog softvera) bi trebalo da znači da je održavanje istih odgovornost pravog vlasnika (vlasnika softvera). Međutim, u praksi to nije baš tako. *Enterprise* održavanje vlasničkog softvera se dodatno naplaćuje, a licence pokrivaju samo pravo na korišćenje tj. čisti namet na vlasničko pravo.

Održavanje *FLOSS* rešenja je odgovornost korisnika. Postoje gomile besplatne dokumentacije o načinu održavanja *FLOSS* rešenja tako da korisnik može sam da ga održava, ali isto tako može i da pita za savet *FLOSS* zajednicu koja uglavnom brzo reaguje i može da reši većinu bitnih problema. U interesu *FLOSS* zajednica je da se popularizuje upotreba *FLOSS*-a u poslovne svrhe. *FLOSS* zajednice su fleksibilne i verujemo da bi brzo mogle da reaguju na izmene klime (veći interes za održavanje poslovnih sistema) i da obezbede legalnu *enterprise* podršku.

Srpske *FLOSS* zajednice već pokazuju znanje i umeće u razvijanju srpske *Linux* distribucije i u slučaju povećanog interesovanja barem jedna može da bude ponuđena sa *enterprise* podrškom.

Programeri bi mogli da se ne slože sa nama i da kažu da im koncepcija vlasničkog softvera više odgovara. Mi mislimo da nisu u pravu i da će prelaskom na *FLOSS* imati više posla. Koncept komercijalnog *OSS*-a pa čak i vlasničkog softvera koji može da se portuje na *OS* otvorenog kôda, neće im smanjiti prihode, imaće manju konkurenciju velikih firmi koje proizvode softver, radiće za poznatog naručioca. Koncept vlasničkog softvera nije izmišljen za malog programera, nego da zaštiti interese velikih kompanija. Velike kompanije nemaju problem da za male pare otkupe dobru ideju od malog programera, ili mu je čak ukradu, ukoliko ideju nije adekvatno zaštitio. *Copyright* nije prilagođen malom programeru, previše je skup i komplikovan za njega. Treći interes programera da pređu na *OSS*, jeste taj da alat za programera aplikacija za vlasnički softver je takođe vlasnički softver pa prema tome i tu mali programer gubi deo svoje dobiti.

Masovnije prihvatanje *FLOSS*-a, naročito u poslovnom okruženju, otvara mogućnost daljeg razvoja *IT* industrije koji je ovog puta baziran na *OSS*-u. Svi školovani programeri ne mogu da se zaposle u velikim firmama vlasničkog softvera. Ovim se otvara prostor za manje softverske firme koje će razvijati *OSS* rešenje ili nuditi usluge vezane za *OSS*. Država koja je bila gubitnik u lancu interesa oko vlasničkog softvera, delimičnim prelaskom firmi na *FLOSS* će izgubiti nešto prihoda od poreza na promet vlasničkog softvera. To može da

nadoknadi iz poreza na dobit novih *OSS* firmi. Sa druge strane, država takođe mora da barem delimično pređe na *FLOSS*. To rade zemlje Evropske unije pa prema tome to je budućnost i srpske administracije. Ove tri godine, koliko važe licence za *Microsoftove* proizvode po ugovoru koji je i povod za ovaj serijal, treba iskoristiti za početak promene kursa i preusmeravanje administracije na *FLOSS*. Na ruku *FLOSS*-a bi trebalo da ide i činjenica da *Microsoft* više ne pruža podršku za *Windows XP*. Naš predlog je da se prelazak na *FLOSS* upravo pokrene prelaskom tih računara na neku laganiju verziju *Linuxa* i proveri koliko takvo rešenje može da zadovolji potrebe korisnika tih računara. Ispravni računari svakako nisu za reciklažu. U najgorem slučaju, mogu biti donirani nekoj školi koja nema računare. Mi verujemo da će nastavnici znati šta će sa njima.

Interes odgovorne države jeste da ima balansirani odnos između prikupljenog poreza na promet vlasničkog softvera i troškova licenci za vlasnički softver u svojoj administraciji. Zatim da svojim delovanjem ne nameće nepotrebni trošak firmama u vidu obaveze da koriste vlasnički softver da bi komunicirale sa državom. Na kraju, ali možda i najvažnije, da svojim delovanjem omogući nesmetani razvoj svih grana *IT* industrije pa i one bazirane na *OSS*-u.

Za kraj

Ovaj nastavak serijala „Ugovor Republike Srbije sa *Microsoftom*" je prirodno došao do teme korišćenja *FLOSS*-a u poslovnom okruženju, ali se to odlično uklapa u novi koncept LiBRE! časopisa koji ima plan da se više obraća poslovnim korisnicima. Ovog puta smo akcenat stavili na ekonomske interese korišćenja *OSS*-a (komercijalnog i besplatnog softvera otvorenog kôda). Pored ekonomskog interesa, tu postoji interes zaštite privatnosti, interes veće kontrole sistema, sigurnosti, nezavisnosti od vendora (vlasnika softvera), razvoja iz sopstvenih resursa i drugi.

O ovim i drugim interesima neće biti reči unutar ovog serijala, ali će biti u ostalim člancima ovog časopisa. U ovom serijalu je ostala još jedna važna tema – *FLOSS* zajednice, analiza stanja i preporuke.

Nastaviće se.

Temelj sazdan od *FLOSS*-a

Autor: Marko Kažić

Zašto *Linux* i otvoreni kôd moraju u učionice?

Kineska poslovica kaže: Kada vetar promene duva, neki dižu zidove, a neki vetrenjače. Razmišljajući o svom školovanju i o *Linuxu* kojeg koristimo svakodnevno, autor ovog teksta ne uspeva da spoji ta dva pojma u jednu skladnu sliku. Računari u školi su vazda bili stari, ispisani, zagađeni *bloatware.om*, sa trideset ikonica za *Minesweeper* koji je užasno izgledao na izmučenom *Windows 98 Second Editionu*. Zbog toga su naši počeci sa *Linuxom* bili nepotrebno mučni i zakasneli (prim. aut.).

Kada pričamo o edukaciji i korišćenju otvorenog koda u školama, daleko smo iza aktuelnih tokova, poslovično tvrdoglavi, ne prihvatamo *Linux* i *FLOSS* kao platformu i filozofiju dobru za našu decu i razvoj naše zajednice. Ako *NASA*, *Google*, Međunarodna svemirska stanica, *IBM*, administracija Ruske Federacije i mnogi drugi[1] koriste *Linux*, zašto nije dobar za nas? Možda jer od njega ne profitira niko osim običnog korisnika.

O *Linuxu* se često priča kao o sistemu koji koriste samo *geekovi*, oni koji „mrze" *Microsoft* i *Apple* ili oni koji imaju stari računar za reciklažu. Možemo to tako da gledamo, ali nećemo biti u pravu. *Ubuntu*, kao samo jedna od bezbroj *Linux* distribucija, u martu 2014. godine ima oko dvadeset i dva miliona korisnika. To nisu razlozi zbog kojih je *Linux* platforma budućnosti a slobodan softver filozofija buduće informatike. Razlozi su mnogo bitniji i mogu se sadržati u ovome – Škola i deca bi profitirali jer bi u *Linuxu* imali jeftinu (besplatnu), moćnu, pouzdanu i bezbednu platformu koja podstiče lični razvoj i ideje deljenja i jačanja zajednice. Kako?

[1] *List of Linux adopters* – http://en.wikipedia.org/wiki/List_of_Linux_adopters

Sloboda, sigurnost i izbor

Pogotovo danas kada je privatnost podataka i njihova zaštita aktuelna, treba da razmišljamo o svojim slobodama na internetu i u softveru koji koristimo svakodnevno. U školama ćemo često naletati na računare pune

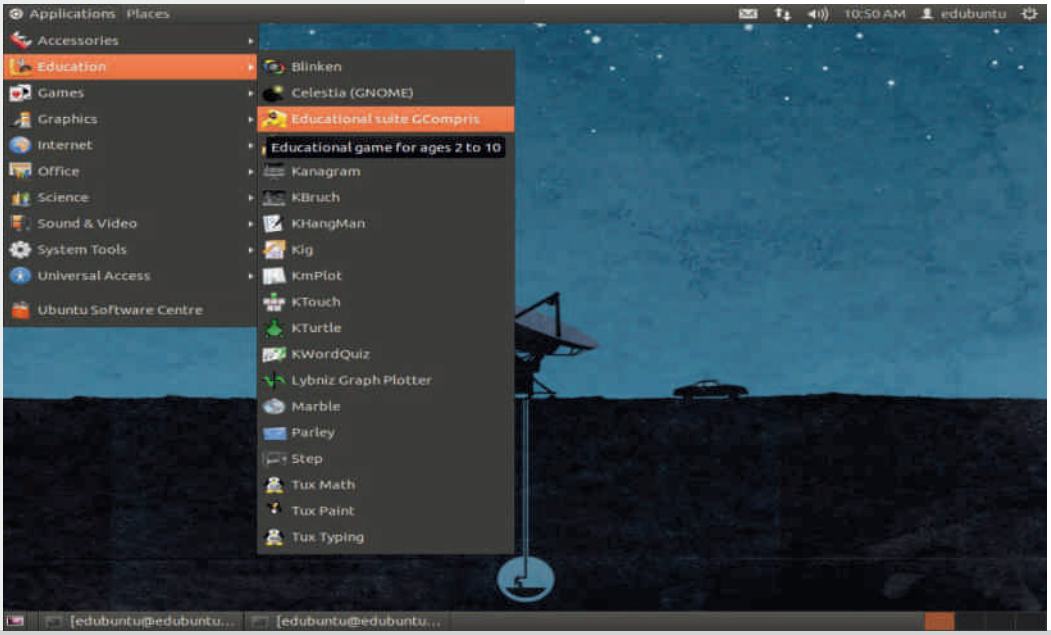
spyware-a i *malware*-a, a to jednostavno nisu računari na kojima je bezbedno raditi. Umesto što zabranjujemo deci da instaliraju aplikacije na te iste računare i ograničavamo im prava, trebalo bi da koristimo sistem koji podrazumeva bezbednost, koji je iole otporan na takvu vrstu zloupotrebe korisničkog poverenja.

Deca treba da nauče da preispituju kome ostavljaju svoje podatke i kako se ti podaci koriste. U softveru otvorenog koda, situacija je mnogo bolja nego kod vlasničkog softvera. Ako se naviknemo da sistem koji možemo besplatno da instaliramo, promenimo radnu okolinu, distribuciju, softver koji koristimo, postajemo vlasnici svog izbora i svojih podataka, a tako nešto ne učimo decu u školama.

Sloboda od vendora, sloboda od zatvorenih platformi

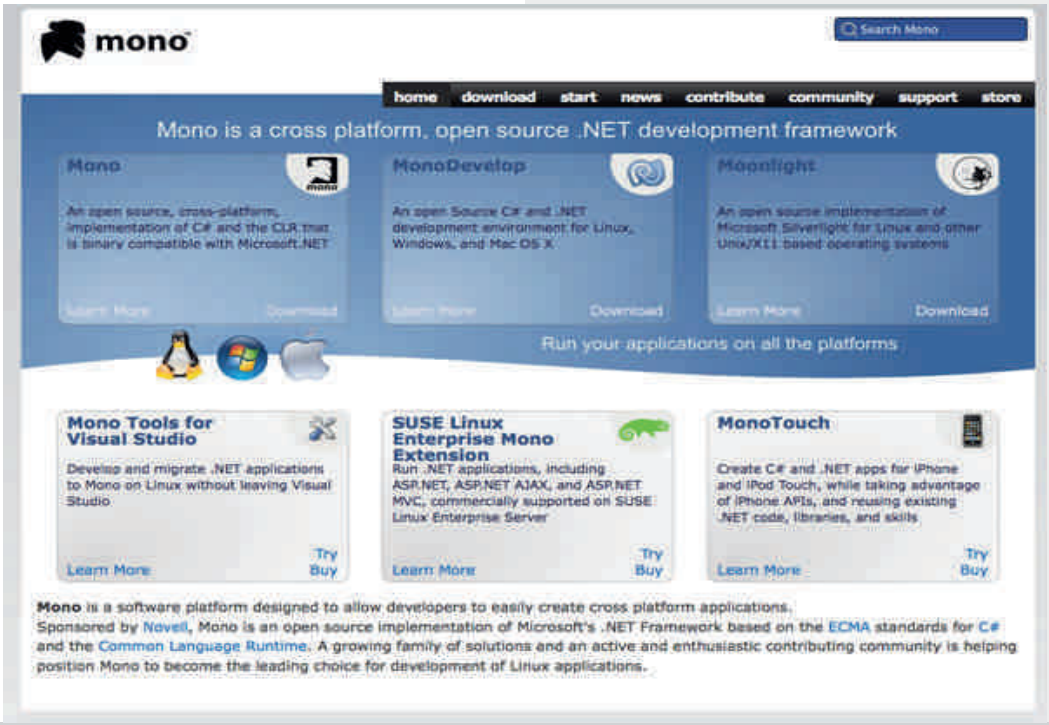
Svi mi koji smo imali kakav-takav kontakt sa informatikom u školama, znamo čemu smo naučeni - od *Borlanda*, do *managed* okruženja kao što su *C#* i *.NET*. Vlasničke platforme su neminovno zlo današnjeg *IT* sveta, ali one nisu jedine koje postoje. Platforma koja prvenstveno služi da unapredi rad i poboljša razvoj, od korisnika i programera vrlo brzo pravi taoce koji postaju zavisni od platforme koju plaćaju. Ovo je i rekurzivna poenta ovog članka. Ako pitamo nekoga zašto koristi *Windows*, uglavnom ćete čuti da aplikacije koje koristi, nisu dostupne na drugim platformama i sistemima, što ne bi bio slučaj da su te aplikacije otvorenog koda.

Deca u Srbiji danas koriste vlasničke



programe često ne znajući da postoji alternativa. *Office* je odličan primer. Pored zatvorene platforme koja podrazumeva i zatvorene formate za čuvanje dokumenata, u poslednje vreme imamo i *OneDrive cloud* gde, gle čuda, na dva klika možete da sačuvate vaše podatke na nekom serveru hiljadama kilometara od vas i da ga koristite na drugom računaru bez mnogo muke. Mi više nismo vlasnici svojih podataka. Paradoksalno je ali istinito. Vlasničke platforme ulažu mnogo u korisničko iskustvo i sve je dostupno na klik ili dva. Naš je zadatak da edukujemo korisnike, da svaku platformu, bez obzira na njeno sučelje ili temu, preispitaju. Oni koji to ne mogu, moraju sistemski biti zaštićeni i mora im se omogućiti slobodna i sigurna platforma.

Korisnici su lišeni slobode. Školski program je napisan da podrži vlasničke platforme i licence koje nisu jeftine. Kažu da bi promena programa koštala mnogo. Dobar primer da ta tvrdnja nije tačna, jesu upravo *C#* i *.NET*. Platforme koje su zamišljene i realizovane tako da budu vrhunac vlasničkog koda, portovane su na *Linux*. Korisnici (ne kompanije), napravili su Mono[2]. Desilo se nešto nezamislivo, dešava se da sve ono što je urađeno u *C#* i *.NET*-u, radi na otvorenoj platformi. Dešava se da sve što mladi uče o *C#* i *.NET*-u, mogu da primene na otvorenoj platformi. *Mono* je velika pobeda *open-sourcea*, projekat na koji se treba ugledati tražeći primer, prvenstveno jer ne kvari



tako skup i delikatno probran plan i program za informatiku koji propisuje Ministarstvo, uglavnom uz pomoć korporativnih sponzora.

[2] *Mono Project* – http://www.mono-project.com

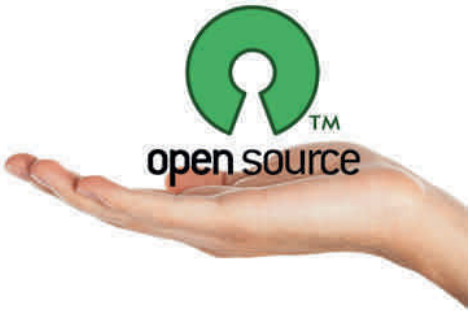
Računari traju duže i dostupni su svima

Sa dozom skepticizma ćemo uzeti sopstvene reči, ali u praksi računari u školama su stari, „slabi" i teško se obnavljaju. S druge strane, čak i kada se neka škola ponovi, uobičajena je praksa da se stari računari bace, jer ne podržavaju najnoviju verziju *Windowsa*. To je klasična priča jednog konzumenta. Ni na trenutak ne stanemo da razmislimo, koliko škola u Srbiji nema računare, koliko dece iz najugroženijih slojeva nema računare kod kuće, koliko njih bi profitiralo znanjem i napretkom kada bi samo postojao neki način da te stare računare ne bacimo, nego da ih ponovo iskoristimo u neku drugu svrhu.

Sloboda – to je ono što krasi *Linux*, a jedna od ključnih prednosti te slobode je da možete naći dobre distribucije, koje sasvim komotno rade na starijim računarima. Iako nova tehnologija ima svoje prednosti i to je neosporno, ono što mi shvatamo kao smeće, postaje savršeno upotrebljiv uređaj onog trenutka kada mu se promeni softver koji ga pokreće. Rashodovan računar, koji bi koristio *Linux*, mogao bi da služi u nekoj drugoj školi koja nije imala sreće da dobije sredstva za nabavku nove računarske opreme. Besplatan hardver i softver, a zajednica sve jača - To je prosto nedopustivo.

Znanje, znanje, znanje

Uticaj *Linuxa* na razvoj korisnika i razvoj dece kroz edukaciju ima toliko podstavki da ih je gotovo nemoguće skoro sve navesti. Ranije smo napomenuli da se izbegavanjem vlasničkih platformi dobija sloboda. Za početak, slobodni smo da softver redistribuišemo i delimo. Mlade treba učiti da dele i da to nije ništa loše, iako su oni godinama uspešno delili instalacione diskove *Windowsa*, sada to mogu da rade kao deo jedne potkulture, legalno i bez razmišljanja o prirodi softvera.



Mogućnost menjanja softvera je još jedna sloboda koju nemamo sa vlasničkim softverom. Korisnici (oni koji žele) treba da znaju kako njihov softver funkcioniše i treba da imaju pravo da učestvuju u njegovom razvoju ukoliko to žele. *FLOSS* je neposredna demokratija za softver. Iako mladi danas širom Srbije u svom džepu nose *Android* telefon ne znajući da je *Android*

u suštini platforma zasnovana delom na *Linuxu*, taj isti *Android* je manje-više proizvod gore pomenute slobode. Neke nove platforme koje u budućnosti mogu stvarati ljudi iz Srbije, neće nastati od vlasničkih platformi i tehnologija, već će nastati od slobodnog softvera jer se slobode potrebne za dobar razvoj pružaju samo njegovim korisnicima. Korisnici moraju da koriste softver po svojoj meri. Kod kuće, na poslu, u učionici, gde god da ste, vaš softver može biti potpuno prilagođen vama. Mnogi su primeri uspešne migracije na slobodan softver, koji je pružio osnovu za pravljenje otvorene edukativne platforme, u kojoj svi učestvuju. Te platforme su potpuno prilagođene potrebama škola i učenika ili studenata, kako praktično po mogućnostima, tako i kulturološki i jezički. Znanje i zajednica su na prvom mestu.

Razloga je mnogo i nema mesta za sve, ali se nadamo da su najbitniji ovde. Neki su praktični, neki etički i filozofski, ali su svakako neosporni i tu su da ostanu. *Linux* je platforma koju deca danas moraju da upoznaju, jer su oni budućnost, a našu informatičku budućnost ne smemo graditi ni na kojim temeljima osim na onim sazdanim od *FLOSS*-a.

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

Distrowatch

1	Mint	3391>
2	Ubuntu	1795<
3	Debian	1643<
4	openSUSE	1228<
5	Arch	1172>
6	Fedora	1148<
7	Mageia	1114<
8	elementary	1021<
9	Zorin	968>
10	LXLE	893>
11	Kali	864>
12	Android-x86	801>
13	Lubuntu	781<
14	Puppy	776<
15	SteamOS	751>
16	Lite	740>
17	Pinguy	720<
18	CentOS	706>
19	Chakra	701>
20	wattOS	693>
21	Robolinux	679>
22	Antergos	641>
23	Sabayon	640<
24	Bodhi	608<
25	CrunchBang	599<

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

Powered by

Gentoo

Autor: Stefan Nožinić

Često imamo priliku da pišemo o novim distribucijama koje su, pre svega, namenjene ljudima koji se po prvi put sreću sa *Linux*om. Ovog puta ćemo napraviti izuzetak i predstaviti vam *GNU/Linux* distribuciju koja je upravo ono suprotno - nipošto nije za početnike. Kao što se može već naslutiti, ova distribucija nije toliko jednostavna za korišćenje na prvi pogled. Vredi postaviti pitanje otkud potreba za ovakvom distribucijom i kako se ona poredi sa ostalim distribucijama u ovom rangu.

Gentoo je nastao 2002. godine i predstavlja jednu od distribucija koja će vam omogućiti da vidite *GNU/Linux* baš onakav kakav jeste (stabilan, brz, transparentan ...) i iz toga, re svega, da naučite nešto novo i zanimljivo.

Najpre ćemo pokušati da odgovorimo otkud potreba za ovakvim pristupom. *Gentoo* se instalira tako što se preuzme osnovni sistem koji u sebi sadrži potreban softver za prevođenje nekih osnovnih programa (eng. *compiling* - prevođenje izvornog koda u izvršni) i koji služi kao osnova za dalju nadogradnju. Ako bi neko želeo da radi ovako nešto, to je sigurno proisteklo iz potrebe da sistem prilagodi po svojim potrebama od samog početka. Mogućnost da prevedete softver vam omogućava da uključite ili isključite neke njegove mogućnosti što vam daje priliku da izaberete samo ono što je vama zaista potrebno i time optimizujete vaš računar. Na primer ako ne koristite *Qt* aplikacije (*kde* recimo), možete isključiti *Qt* podršku za razne biblioteke. Obično kad se pomene čuvanje resursa, odmah se pomisli na stare računare. U ovom sluaju se nikako ne misli na stare računare već na novije računare čiji korisnici žele da iskoriste svu njihovu snagu na samo onaj deo koji je njima potreban. Ako na trenutak zaboravimo činjenicu da su nam perfomanse bitne, ovakav pristup ima i drugih prednosti. Jedna od dosta bitnih prednosti jeste minimalizacija sistema. Sistem od samo neophodnog softvera jeste dosta jednostavan za održavanje a i manja je verovatnoća da će nešto krenuti naopako. Ovo omogućava korisnicima, koji su najčešće i u ulozi sistem administratora na svojim ličnim računarima, omogući lako održavanje.

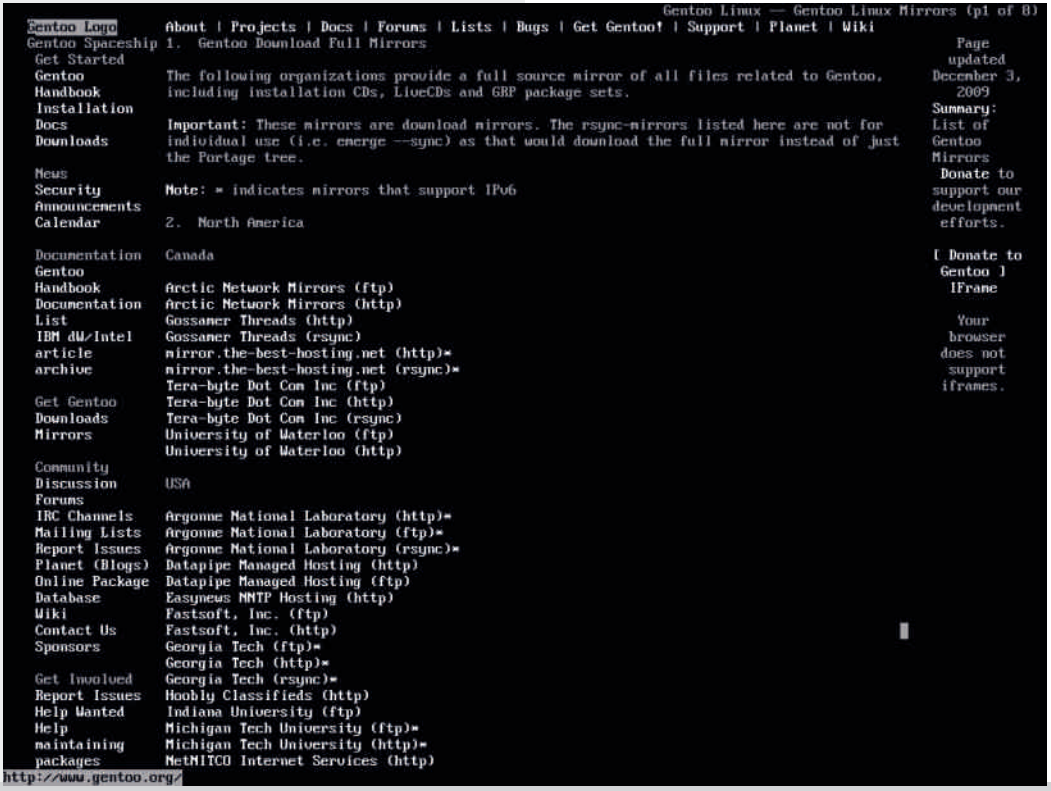
Sada kada smo videli kakve su prednosti ovakvog pristupa, vredi zapitati se da li postoje još neka slična rešenja kao

što je *Gentoo* i koja je suštinska razlika između njih. Mnogi će na pomenu gradnje sopstvenog sistema od nule prvo pomisliti na *LFS* (*Linux from scratch* - sistem od nule) koji daje mogućnost gradnje sistema od samog početka. Razlika između *Gentoo*a i *LFS*-a jeste ta što se uz *Gentoo* dobija malo obimnija osnova pa je time neki posao, koji je svakako neophodno uraditi na većini sistema ovakvog tipa, već urađen za vas. Uz *Gentoo* osnovni sistem ćete dobiti prevodilac (eng. *compiler*), *shell*, upravnik paketa o kojem će kasnije biti više reči i mnoge druge neophodne alatke. Ono što sigurno nećete dobiti jeste *kernel* (da, da, to ćete morati sami da prevedete) i okruženje radne površi (ovo ste možda i očekivali).

Sistemi koji koriste sličan metod upravljanja paketima su *BSD* sistemi. Oni koriste portove koji su slični *Gentoo*vom *Portage* stablu. Ovde treba napomenuti da *BSD* sistemi ne spadaju u *Linux* distribucije pa time ne koriste *Linux* jezgro.

Upravnik paketa

Na *Gentoo*u je generalno usvojen princip da se sav softver (sa par izuzetaka) direktno prevodi iz izvornog koda u krajni produkt (izvršni deo i dodatni podaci). Ovog se postiže pomoću sistema za upravljanje paketa



koji će kompajliranje uraditi umesto korisnika ali će vam opet dati mogućnost izbora koji biste imali i tokom ručnog kompajliranja. Ovaj sistem se naziva *Portage* i zvaničan je upravnik paketa. Izbor mogućnosti za svaki softver (npr da li će se uključiti podrška za *pulseaudio* u nekom softveru) se postiže tkz. *USE* flagovima. Oni mogu biti globalno za sve programe (recimo za sve programe želite da uključite podršku za *systemd*) a može biti pojedinačno za svaki. Postoji mogućnosti i *patch*ovanja softvera što je još jedna prednost ovakvog pristupa. Treba napomenuti da pored zvaničnih paketa postoje i tkz. *overlay*-i koji omogućavaju korišćenje softvera koji nije u zvaničnoj riznici. Svaki paket, bio on zvaničan ili ne, jeste najčešće jedna skripta (*ebuild*) koja opisuje kako se taj softver prevodi i instalira tako da upravnik paketa zna za svaki paket kako da ga prevede, podesi i instalira. Pored ove skripte nalaze se još i dodatni fajlovi kao što su to *dekstop* fajlovi koji su bitni za grafička okruženja a koji obično ne dolaze od autora samih programa. Kao što ste već mogli da predpostavite, paket ne sadrži izvorni kod programa već je njegova lokacija zabeležena u *ebuild*-u i tokom instalacije se ona preuzima sa interneta ako već nije preuzeta od ranije. *Portage* nudi i automatsko razrešavanje zavisnosti što dosta olakšava čitavu proceduru instalacije i nadogradnje. Ono što je dosta za pohvalu jeste i mogućnost maskiranja paketa. Ova mogućnost pruža dosta jednostavan metod da se paket vrati na prethodnu verziju ili da se preskoči neka verzija određenog paketa. Ovo može biti dosta

korisno ako želite da imate stabilan sistem. *Gentoo* se jednom instalira i potom je potrebno samo raditi nadogradnju paketa, što se zove *rolling release*, pa tako možete dobiti sistem koji, ako niste ništa zabrljali, možete koristiti dok vaš računar normalno funkcioniše.

Dokumentacija

Iako ovo sve zvuči komplikovano, težina instalacije i podešavanja se ogleda u dokumentaciji a za *Gentoo* možemo slobodno da kažemo da je ona i više nego odlična ali i da podrazumeva neko predznanje. Ovde treba pohvaliti i domaću zajednicu LUGoNS koja se postarala da *Gentooov* zvanični priručnik, koji je obavezno štivo tokom instalacije, prevede na srpski jezik i time olakša posao mnogima kojima jezik može biti prepreka.

Zaključak

Možemo reći da je *Gentoo* odličan sistem za sve one koji žele apsolutnu kontrolu nad svojim operativnim sistemom i koji su spremni da odlučuju za svaku sitnicu što se na kraju dosta isplati. Isto tako se nadamo da će *Gentoo* isprobati i neki sistemski administrator u firmi jer je ovakav pristup zaista pogodan za takve uslove. Ako ste već otvorili *Gentoo* priručnik nema boljeg zaključka nego da vam poželimo srećno kompajliranje!

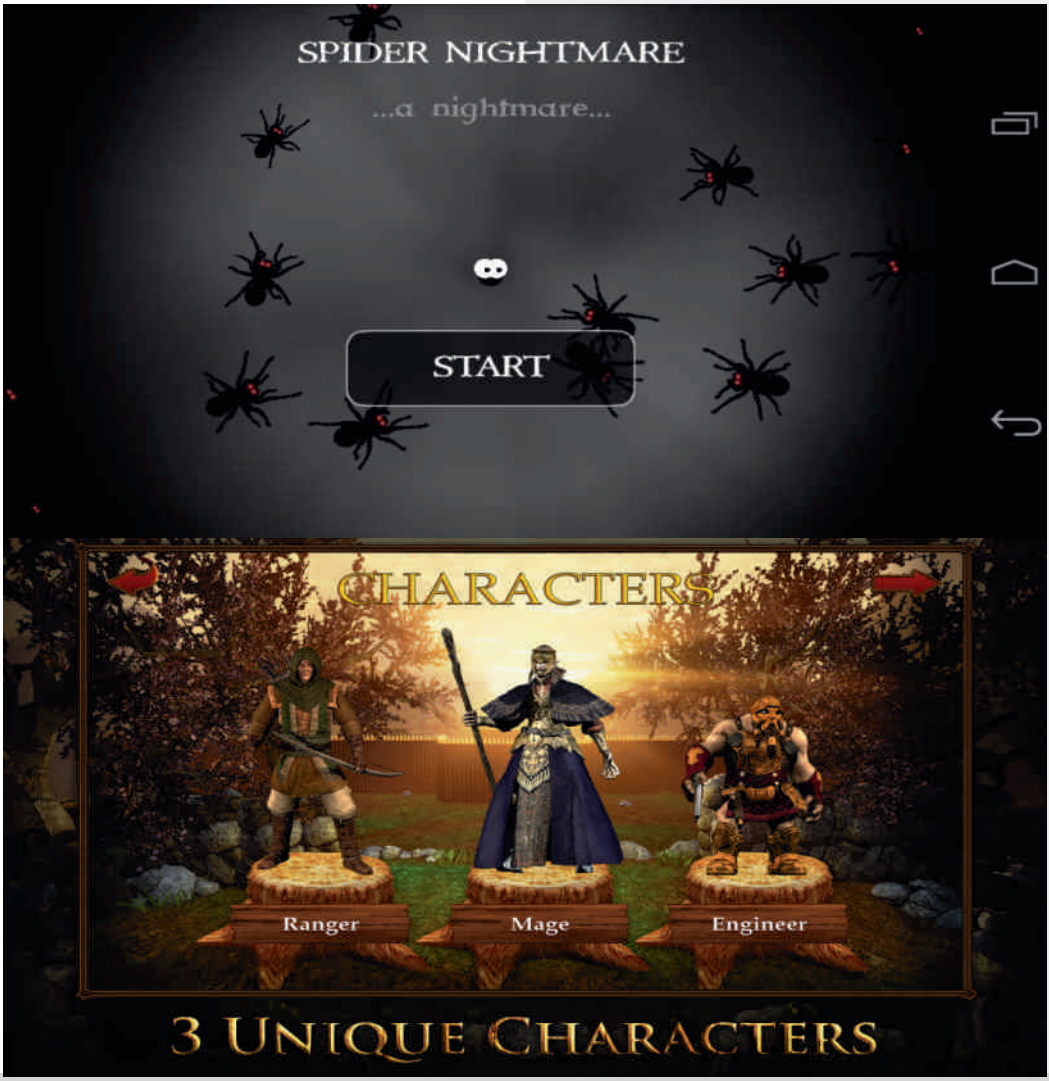
Linkovi:

[1] http://www.gentoo.org/
[2] http://www.gentoo.org/doc/en/handbook/
[3] https://gentoo-handbook.lugons.org/doc/sr/ handbook/

libGDX

„Java game development framework”

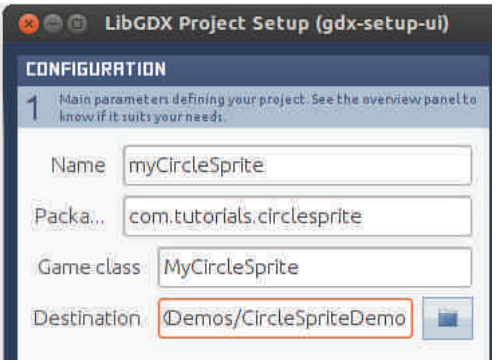
(1. deo)



Autor: Gavrilo Prodanović

Skoro čitavoj mladoj populaciji igrice su alatka za opuštanje i portal za uživanje u nekoj drugoj dimenziji. Mnogi odrasli često igraju igrice sa svojom djecom ili samostalno u slobodno vrijeme. Da bi se sastavila igrica, potrebne su određene tehničke sposobnosti i osjećaj za umjetnost i kreativnost da bi se igrica dočarala kroz grafiku, zvučne efekte i kroz sam *gameplay*. Mi ćemo ovde predstaviti alat koji će vam olakšati pisanje koda za igricu, a ime mu je *LibGDX* od *badlogicgamesa*.

Nekada prije, trebalo je vjerovatno duboko razumjevanje *low-level* stvari da bi se napisala i neka osnovna igrica. Da bi igrica bila dostupna na više platformi, obično se preskakao korak. Danas je sve mnogo lakše, a podrška za više platformi dolazi spontano i podrazumjevano, a to su, može se reći, glavni aduti *LibGDX*-a. *LibGDX* je *java framework* koji podržava sledeće platforme: *Desktop (Linux, Mac OSX, Windows), Mobile (Android, iOS)* i *web (HTML5)*. Od *IDE*-a podržani su *Eclipse*, *Intellij IDEA* i *NetBeans*. Moguće je koristiti samo komandnu liniju za razvoj. *LibGDX* svaki projekat konceptualno djeli u pet djelova (projekata): *core*, *desktop*, *android*, *ios* i *html*. *Core* sadrži zajednički kôd, a ostali projekti sadrže kôd specifičan za platformu. *LibGDX* posjeduje generator projekta koji će podesiti sve potrebne potprojekte i skinuti sve dodatne libove (eng. *libraries*) koji su potrebni. Možemo odabrati koje platforme želimo da koristimo, a u ponudi su ostale biblioteke koje će da prošire funkcije *libGDX*-a. Koristi *gradle*, što automatski omogućava importovanje u sve poznate *IDE*-ove.



Za portabilnost *LibGDX*-a vjerovatno su najviše zaslužni *OpenGL* i *Java* koji su dostupni za različite platforme. Programer će kôd logike igre da stavi u *core* projekat, dok su ostali projekti generisani vraperi (eng. *wrappers*), a u njima se i ne mora mnogo toga mjenjati. Za *desktop* bismo postavili podrazumjevanu rezoluciju, za *android* podrazumjevanu orjentaciju, a u *html* projekat kôd *landing* stranice. Za sve platforme biće vam dovoljan *Linux OS* za programiranje, osim za *iOS* platformu gdje će vam biti potreban *Mac* računar sa *OSX* sistemom i sa *xCode* razvojnim okruženjem.Znajući da se struktura sistema datoteka konceptualno razlikuje između *Windowsa* i ostalih *UNIX* sistema, a isto tako se i načini smještanja podataka na *Androidu* i na *Linux desktopu* razlikuju, prikaz *LibGDX* počećemo upravo o tome kako ovaj *framework* generalizuje sistem datoteka. *LibGDX* generalizuje sistem datoteka u sledećih pet grupa:

1. ***Classpath***: Omogućuje pristupanje

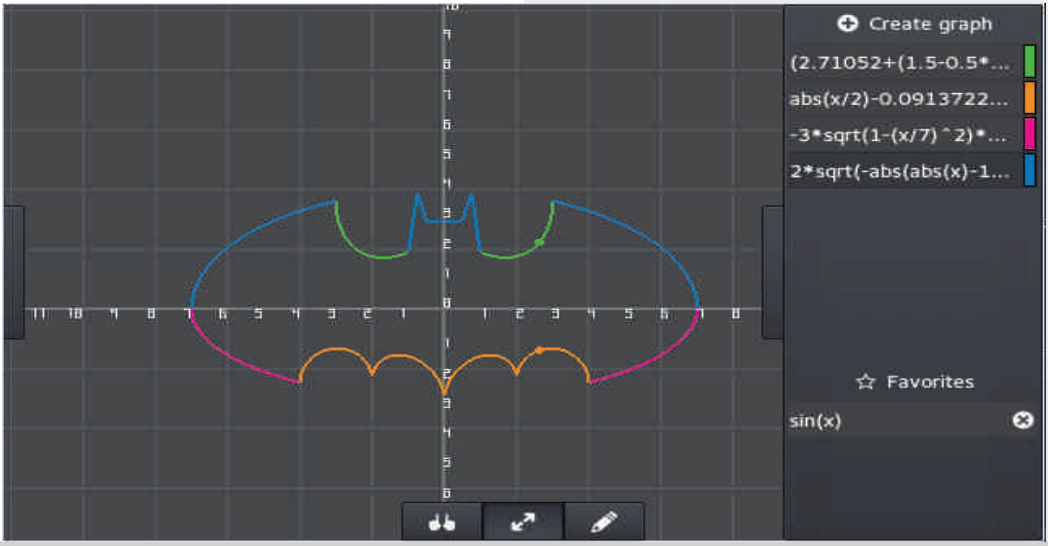
datotekama koje su upakovane u arhivu aplikacije (npr. *jar* ili *apk*). Pristup ovim datotekama je *read-only*.

2. ***Internal***: U interne datoteke spadaju one datoteke čija je putanja relativna na *root* putanju aplikacije i radnog direktorijuma za *Desktop* i za datoteke koje se nalaze unutar *assets* direktorijuma u *Android* projektu koji je uključen u arhivu aplikacije. Interne datoteke su obično *read-only* ako se odnose na datoteke upakovane u arhivu.
3. ***Local***: Lokalne datoteke na *desktopu* su smještene u *root* aplikacije, a za mobilne uređaje unutar privatnog direktorijuma u zavisnosti od sistema. Preporučeno ga je koristiti za čuvanje manjih datoteka kao što je *game state*.
4. ***External***: Za *desktop* su smještene unutar *home* direktorijuma, dok je za mobilne uređaje to *SD* kartica. Namjenjen je za čuvanje velikih datoteka, ako ima potrebe.
6. ***Absolute***: Omogućava pristup datotekama na osnovu apsolutne putanje. Ovakav način nije preporučljiv jer postoji mogućnost za narušavanje portabilnosti.

Za *html5* dostupno je samo interno skladište (eng. *storage*), dok se za sve ostale platforme mogu koristiti sva skladišta. Lokalno i eksterno skladište mogu biti fizički nedostupni na nekim platformama, pa su tu i funkcije kojima možemo provjeriti dostupnost. Moguće je brisati i mjenjati naziv datoteke unutar lokalnog ili eksternog skladišta ili kopirati i premještati datoteku između lokalnog i eksternog skladišta.

Kao primjer vrapera za sistem datoteka učitaćemo tekstualnu datoteku iz internog skladišta koja je smještena unutar *assets* fascikle u *android* projektu.

```
FileHandle file =
Gdx.files.internal("myfile.txt"
```



```
);
String text =
file.readString();
```

Program u *LibGDX*, možemo reći, počinje u klasi koja naslijeđuje *Game super* klasu. Prije toga, pokrene se kôd specifičan za platformu, npr. *main* funkcija za *desktop* ili *onCreate* u *MainActivityu* za *android*. Najvažnija naslijeđena metoda je *render()* koja se poziva u beskonačan *loop*. U njoj se pozivaju funkcije za renderovanje, a može biti smješten i kôd logike igrice. U praksi nije praktično implementirati čitavu igricu unutar glavne *Game* klase, a *libGDX* nam tu pomaže jer posjeduje *interface Screen* koji implementira neku našu klasu koja je zadužena za određenu usku logiku (npr. jedan *screen* za glavni meni, drugi *screen* za sam *gameplay*). Interfejs definiše veći broj metoda: *render*, *resize*, *show*, *hide*, *pause*, *resume* i *dispose*. One svojim imenom opisuju svoju namjenu. Zahvaljujući ovakvom pristupu, lako možemo da smjenjujemo logičke cjeline jednu za drugom.U sledećem broju udubićemo se u metode kojima možemo da renderujemo grafiku u *libgdx*, njegove *low-level opengl* vrapere i *high-level* klase koje su tu da nam olakšaju život. Govorićemo o tome kako je *input* klasifikovan kroz platforme i ukratko ćemo reći šta je u ponudi za reprodukovanje zvučnih efekata i melodija. Kasnije ćemo govoriti o mogućnostima koje su integrisane u *libgdx*, a koje nam mogu omogućiti da napravimo što bolji *gameplay*.

Uvod u programski jezik *C*

(3. deo)

Autor: Stefan Nožinić

U prošlom delu smo se dotakli tipova podataka, onih promenljivih nad kojima se mogu izvršiti neke operacije. U ovom delu ćemo diskutovatio uslovnom grananju i kontrolama toka programa i pokazaćemo neke zanimljive primere.

Uslovno grananje

Iako smo do sada naučili osnovne stvari o *C*-u, nismo bili u mogućnosti da napišemo neke zanimljive programe. Nismo mogli da napišemoprogram čiji tok zavisi od vrednosti neke promenljive, odnosno od njegovog ulaza. Prva stvar koju ćemo u ovom tekstu objasniti, jeste *if*naredba koja prima određeni uslov i na osnovu toga da li je uslov ispunjen, izvršava neki određen kod. Ovaj uslov jeste relacijadve promenljive ili promenljive i neke konstantne vrednosti. Pogledajmo sintaksu *if* naredbe:

```
if (uslov)
{
  /* .... naredbe koje će biti
izvršene ako je uslov ispunjen
*/
}
```

Kao što smo i rekli, uslov je relacija pa su neki primeri uslova:

- **a == b** - poređenje jednakosti dve promenljive tipa *int*
- **a != b** - tačno u slučaju da su dve promenljive različite vrednosti
- **a > 0** - tačno ako je promenljiva **a** (tipa *int*) veća od nule

Isto tako postoje i „dodaci" za ovu naredbu, a to su *else if* i *else*. Prvo se stavlja jedna *if* naredba, zatim nekoliko *else if* naredbi i potom jedna *else* naredba. Naredba *else if* zadaje drugi uslov u slučaju da onaj pre njega nije ispunjen, dok *else* obuhvata sve što nije obuhvaćeno u gornjim slučajevima.

Primer: program koji ispisuje da li je broj pozitivan, negativan ili nula:

```
#include <stdio.h>

int main()
{
  int n; // broj u koji
unosimo vrednost
  scanf(" %d",  &n); //
učitavamo broj
  if (n>0)
  {
  printf("Broj je
```

```
pozitivan\n");
  }
  else if (n < 0)
  {
  printf("Broj je negativan\n");
  }
  else
  {
  printf("Broj je nula\n");
  }
  return 0;
}
```

Petlje

Sada je vreme da pokažemo kako se neka određena sekvenca naredbi u našem programu može ponavljati, odnosno izvršavati u krug.Za to služe petlje i najčešće se koriste *for* i *while* petlje.

Petlja *for* se koristi u slučajevima kada znamo tačan broj ponavljanja i tu imamo jednu promenljivu kao brojač, ona dobija početnu vrednost, uslov koji treba da bude ispunjen kako bi se petlja izvršavala u naredbu koja se izvršava posle svakog izvršavanja petlje, odnosno iteracije. Ukoliko uslov potreban za izvršavanje više nije ispunjen, petlja se zaustavlja i nastavlja se dalje izvršavanje programa.

Sintaksa *for* petlje je:

```
for (inicijalizacija
promenljive; uslov; naredba
posle svake iteracije)
{
  /* ... naredbe u petlji
koje se izvršavaju tokom
svake iteracije ... */
}
```

Evo primer programa koji ispisuje „0 1 2 3 4 5 6 7 8 9”:

```
#include <stdio.h>

int main()
{
  int brojac;
  for (brojac = 0; brojac <
10; brojac++) // brojac++ je
isto što i brojac = brojac +
1
  {
    printf("%d ", brojac);
  }
  return 0;
}
```

Za razliku od *for* petlje, *while* petlje se izvršavaju sve dok je jedan određen uslov ispunjen. Nema brojača pa se tako ne može unapred znati koliko će se puta odrediti, odnosno to nije lako razaznati. Kod ove petlje se često dešava da se program „zaglavi” jer je neki uslov stalno ispunjen pa se tako petlja neprekidno izvršava.

Sintaksa:

```
while (uslov)
{
  /* ... naredbe ... */
}
```

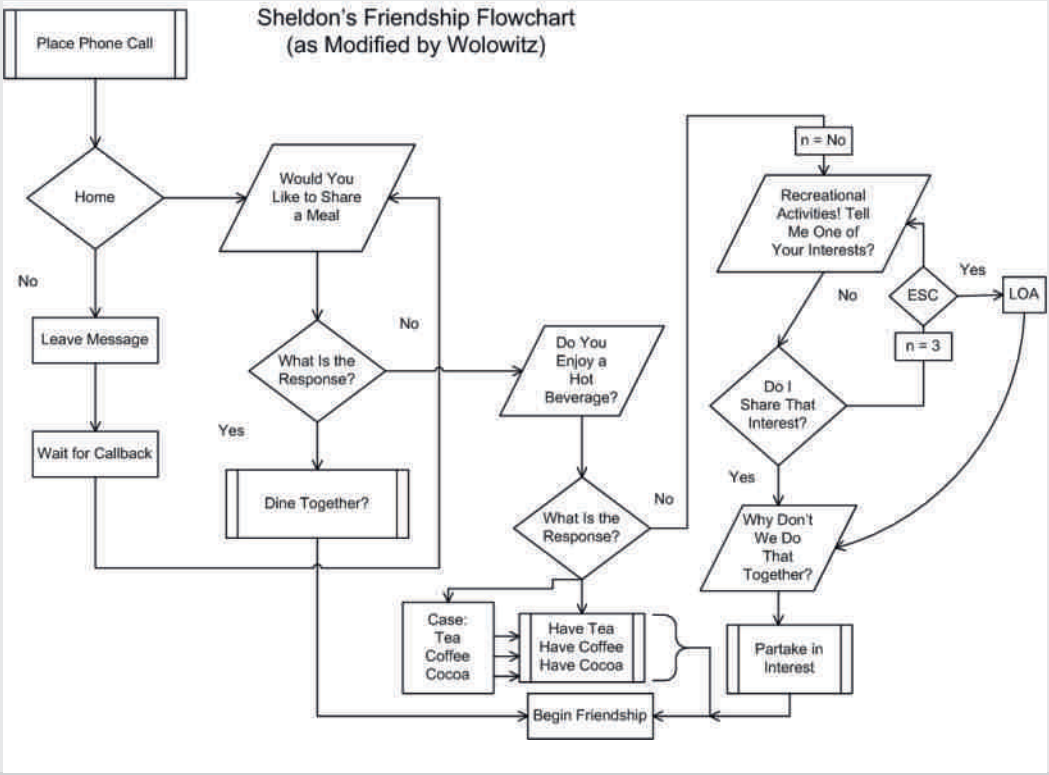
Evo i primera kako se gore napisani kod sa *for* petljom mogao napisati korišćenjem *while* petlje:

```
#include <stdio.h>

int main()
{
```

```
  int brojac = 0;
  while (brojac < 10)
  {
    printf("%d ", brojac);
    brojac++;
  }
  return 0;
}
```

Za kraj

Za kraj ovog dela možemo samo da vam preporučimo da pokušate sami da otkucate nešto i da vidite kako rade petlje jer ćete tako ući u „šemu”.



Unit tests i JUnit

Autor: Dejan Čugalj

Komparacijom industrijske revolucije izazvane parnom mašinom 18. veka (1781, *James Watt*) i informacionih tehnologija 20. i 21. veka primećujemo bitnu razliku. Neke od osnovnih su neopipljivost i apstrakcija 21. veka - virtuelizacija i *cloud*. Iako po pitanju energetske skalabilnosti, upotrebe i iskorišćenja nismo daleko odmakli, kako kaže *Michio Kaku*, „od TIP*0 civilizacije", (https://www.youtube.com/watch?v=JdILmgJGuvw), nekako se može videti ta strukturna razlika napredaka u samoj informacionoj tehnologiji.

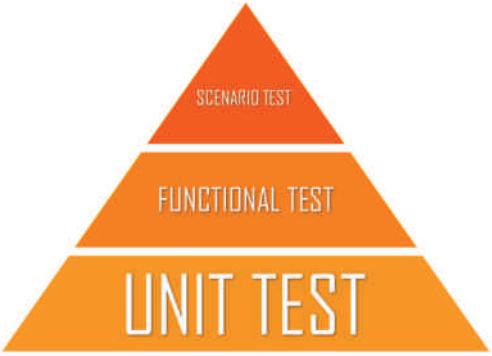
Iako postoji razlika, problemi isplivavaju sami po sebi. Trenutna situacija u *IT*-ju je ta da vlada potpuni raskol, nesrazmera u razvoju hardvera i softvera koji bi trebalo da ga prati. Naime, hardver je toliko napredovao da softver nikako ne može da isprati taj eksponencijalni razvoj hardverskih komponenti, tako da su programeri došli na „metu", i to sagledavajući kroz filozofsku prizmu, stiče se utisak kao da ljudski rod još uvek nije spreman na tako nagli razvoj.

Da bude još gore, uvideli smo još veći problem koji se odnosi na količinu podataka napravljenih u jedinici vremena, globalno gledajući. Nekako kao da smo ostali sami, nepripremljeni za sve što možemo da stvorimo. Svi ti podaci koji čekaju na obradu i koji nisu procesuirani, moraju da prođu kroz neku vrstu algoritma i da budu obrađeni, svrstani i postavljeni zavisno od važnosti u neki *cloud* ili medijum za skladištenje. Upravo ti algoritmi su od izuzetne važnosti jer jedan pogrešan parametar prosleđen algoritmu može da prouzrokuje totalno neočekivane rezultate gde bi krajnji rezultat bio skladištenje nekih medicinskih informacija u npr. katastar za zemljište.

Sa programerske tačke gledišta i tačke gde čovek kao individua ima udela u celom sistemu, greške su neminovne, jer ipak smo samo ljudi, a poznato je svima da svaki čovek greši dok računar sledi instrukcije dobijene upravo od istih, tih „koji greše". Programerska borba protiv ovih tipova grešaka vodi se od samog početka, početka komuniciranja sa računarom, od prvih programskih jezika (1842-1843, *Ada Lovelace*). Jedna od danas najviše korišćenih tehnika za proveru i najzastupljenijih jeste *Unit testing*, a pošto ćemo primer implementirati u *Java*

programskom jeziku, koristićemo *JUnit*.

Unit Testing



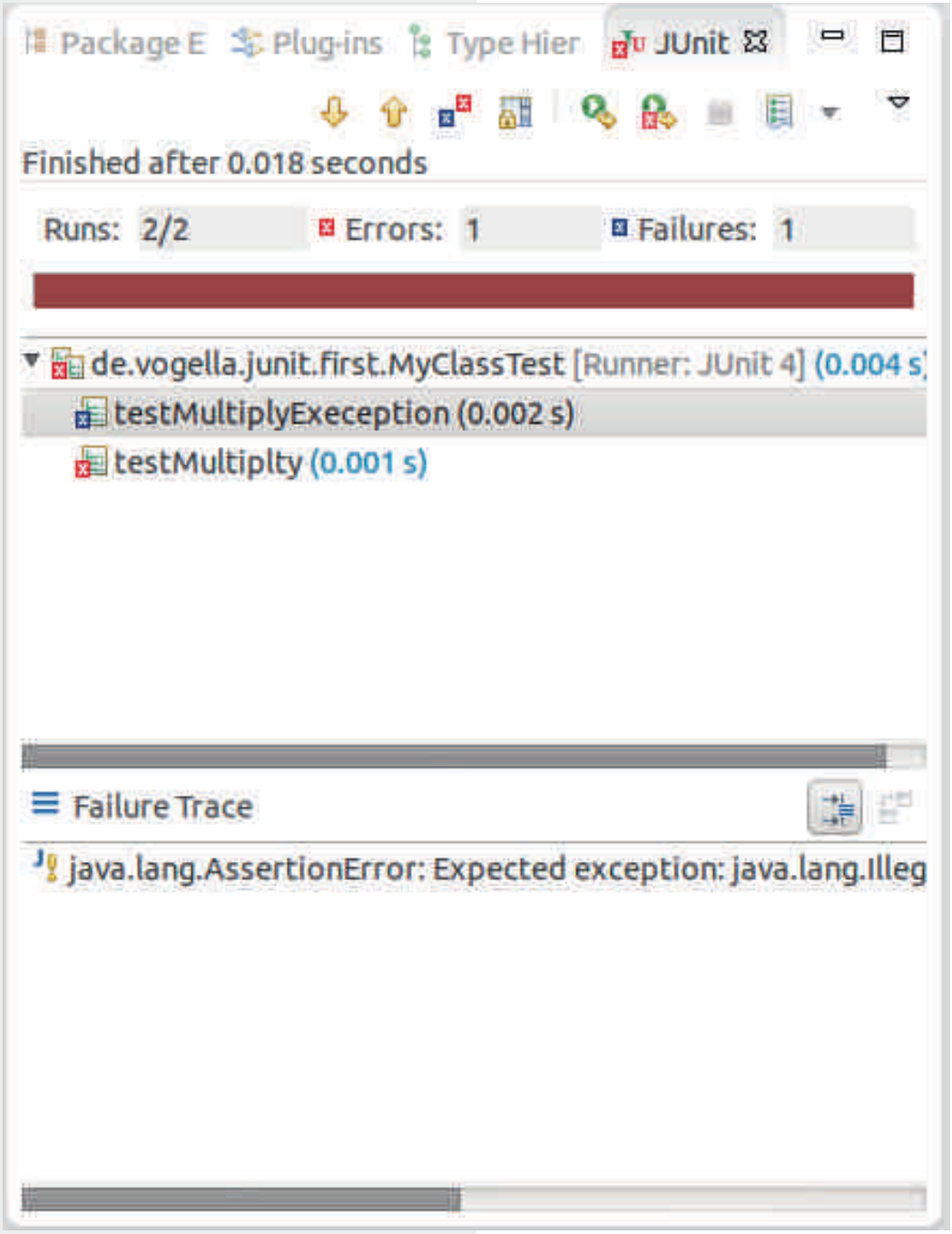
To je deo kôda, namenski napisan od strane programera, koji izvršava specifičan deo aplikacije i proverava dobijeni (vraćeni) rezultat sa sigurno očekivanim podacima ili ponašanjem.(http://www.vogella.com/tutorials/JUnit/article.html)

Laički rečeno, tokom razvoja aplikacije, jedna od osnovnih stvari je da se sa vremena na vreme napisani kôd prevodi u mašinski kôd, izvršava i proverava očekivan rezultat. Ovakvo testiranje je uglavnom i dovoljno, jer se greške odmah mogu uvideti i mnogi programeri upravo to i rade. Iako ova metoda uzima dosta vremena, pogotovo ako se radi o razvoju modula neke aplikacije koja ima veze sa bazom podataka, napišete *query*, pokrenete, proverite bazu i proverite da li je upisan takozvani „*hard coded*" podatak, pa nastavite. Međutim, zamislite situaciju da svako prevođenje na mašinski jezik nekog modula koji razvijate, traje i po nekoliko sati, kakav bi vaš učinak bio u nekoj kompaniji (prim.aut.)?

Ukoliko iole razmišljate na duže staze i vidite sebe kao programera koji radi na nekim većim projektima, trebalo bi da nas ispratite do kraja. U većim softverskim kompanijama testiranje je jedan od krucijalnih delova razvoja određenog softvera. Neki ga vole, a neki ne, to je činjenica, a dok određeni deo napisane aplikacije ne prođe testove, koji opet neki drugi tim sprovodi, napisani modul ne može nikako da uđe u produkciono okruženje, ma koliko on bio dobro implementiran.

JUnit

Kent Beck i *Erich Gamma* uvideli su ovu važnost i napisali su *open source framework* 1997. godine za *Javu* poznat kao *JUnit*. Važnost spomenutih imena ne bismo ovde „proširivali", dok više informacija o *JUnit frameworku* možete naći na adresi: (http://www.junit.org).


Package E
Plug-ins
Type Hier
JUnit
Finished after 0.018 seconds
Runs: 2/2
Errors: 1
Failures: 1
de.vogella.junit.first.MyClassTest [Runner: JUnit 4] (0.004 s)
testMultiplyExeception (0.002 s)
testMultiplty (0.001 s)
Failure Trace
java.lang.AssertionError: Expected exception: java.lang.Illeg

JUnit je objavljen pod *IBM's Common Public License Version 1.0* i postavljen je na *GitHub* repozitorijum. Ubrzo nakon objavljivanja postaje standard za skoro sve jezike pod nazivom *xUnit* i podržava: *ASP*, *C++*, *C#*, *Eiffel*, *Delphi*, *Perl*, *PHP*, *Python*, *REBOL*, *Smalltalk* i *Visual Basic*.

Ne ulazeći u „dubinu" mogućnosti ovog *frameworka*, primer testiranja ćemo pokazati malim *Java* primerom. Testiranje se implementira anotacijom određene metode, klase i potpuno je zaseban pod-modul-aplikativno izvršni program. Zamislimo da imamo prostu klasu *CalculatorTest* i u njoj metodu sabiranja:

```
public class Calculator {
   public double add(double
number1, double number2) {
     return number1 + number2;
   }
}
```

Testiranje *JUnit frameworkom* :

```
import static
org.junit.Assert.*;
import org.junit.Test;

public class CalculatorTest {
   @Test
   public void testAdd() {
     Calculator calculator =
new Calculator();
     double result =
calculator.add(10, 50);
     assertEquals(60, result,
0);
   }
}
```

Možda ovo trenutno ne izgleda kao neka velika pomoć, međutim, razvojem aplikacije i sve većim brojem implementiranih modula, ovo je od krucijalne važnosti kako za tim, tako i za individualne programere.

Korisni linkovi:

- **Zvanični sajt:** http://junit.org/
- ***GitHub* repozitorijum:** https://github.com/junit-team/

Uticaj matematike na nastanak i temelje računarstva (1. deo)

Uvod

Autor: Nedeljko Stefanović

Fundamentalna matematička istraživanja tridesetih godina dvadesetog veka omogućila su nastanak računara i ukazala su na granice njihove primenjivosti, koje se ni danas ne dovode u pitanje. Štaviše, rezultati tih istraživanja i danas predstavljaju stub teorijskog računarstva. Neki od problema koji još uvek nisu rešeni, od velike su važnosti za kriptografiju i očekuje se da njihovo rešavanje dovede do nove epohe u ovoj nauci.

Pojam algoritma i njegov razvoj kroz istoriju

Naivno shvatanje pojma algoritma je da je to mehanički, deterministički postupak, koji polazeći od nekakvih ulaznih podataka, proizvodi nekakav izlaz (rezultat) u konačnom broju koraka. Izraz „deterministički" je upotrebljen u značenju da taj postupak primenjen više puta za iste ulazne podatke, proizvodi uvek isti izlaz do koga stiže na potpuno isti način sa istim koracima. Međutim, ispostaviće se da je upotrebljeno značenje izraza „mehanički" mnogo dublje, a uprošćeno znači da postupak ne zahteva razmišljanje da bi se obavio, već da na osnovu pravila postupka u svakoj situaciji znamo kako da nastavimo postupak, sve dok on ne bude u potpunosti završen.

Algoritmi se javljaju kod vrlo starih naroda, koji su zbog potreba premeravanja površine zemljišta i zapremine ambara i sudova imali postupke za računanje površine figura i zapremina tela. Osim toga, u tom računu su morali nekako da predstavljaju brojeve i vrše operacije sa njihovim predstavljanjem, za šta su im opet nekakvi postupci bili neophodni.

Neki postupci koji u određenoj meri (mada ne potpuno) ispunjavaju navedene uslove, stariji su od brojeva. Stari pastiri nisu umeli da broje, a imali su potrebu da znaju da li su sve ovce na broju. Bilo je više postupaka sličnih ovome. Na primer, u praznu torbu se

može bacati kamenje prilikom izlaska ovaca iz tora (za svaku ovcu po jedan), a pri vraćanju bi se kamenje vadilo iz torbe (za svaku ovcu po jedan). Ako u torbi ostane neki kamen, neka ovca nedostaje.

U staroj Grčkoj su bili otkriveni algoritmi za rešavanje kvadratne jednačine (koji imaju razgranatu strukturu, tj. ima više slučajeva koji se rešavaju na različite načine), algoritam za određivanje prostih brojeva do neke gornje granice (tzv. „Eratostenovo sito" koje ima cikličnu strukturu, tj. neki koraci se ponavljaju), algoritmi za računanje najvećeg zajedničkog delioca i najmanjeg zajedničkog sadržaoca celih brojeva ili polinoma (tzv. „Euklidov algoritam"), kao i razni drugi.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Iako su otkrivani razni algoritmi, prošli su vekovi dok pojam algoritma nije uočen kao zaseban pojam i dok nije dobio svoje ime. To se dogodilo zbog jedne prevodilačke greške.

Persijski naučnik al-Hovarizmi (780-850) napisao je delo u kome je opisao indijski desetični brojni sistem (koji danas koristimo) zajedno sa postupcima za računanje u tom sistemu, koji se danas uče u nižim razredima osnovne škole. Kao autor dela se potpisao ispod naslova, ali je njegovo ime zbunilo prevodioce, koji su mislili da se radi o nekoj njima nepoznatoj arapskoj reči a ne o vlastitom imenu, pa su naslov preveli kao „algoritmi sa indijskim brojevima". Zatim su na osnovu sadržaja pokušali da shvate šta su to algoritmi. Na taj način, osim reči „algoritam", nastala je i reč „algorizam", koja označava upravo pomenute algoritme za vršenje računskih operacija.

Nakon toga su u matematici otkriveni mnogi algoritmi, ali je do sledećeg ključnog koraka došlo tek tridesetih godina dvadesetog veka. Naime, savremeno shvatanje pojma računara je da je to mašina koja može da izvrši program po bilo kom algoritmu, tj. da programski jezik, na kome se programira, obuhvata sve algoritme. Matematičari takve matematičke sisteme zovu potpunim algoritamskim sistemima. Da bi se to ostvarilo, neophodno je svesti sve algoritme na konačan broj jednostavnih pravila.

Nastaviće se.

Vaš posao, *open-source* posao

(2. deo)

Autor: Marko Kažić

Prodaja pretplate i usluga podrške – prodaja stabilnosti

U prošlom broju smo videli kako *open-source* prodire u naš lični ekosistem softvera. Videli smo njegov uticaj na razvoj današnje *IT* industrije i obećali smo da ćemo nastaviti sa roditeljem svih modela poslovanja zasnovanih na *open-sourceu*, a to je prodaja pretplate i usluga. Začetnik same ideje poslovanja zasnovanog na *open-sourceu* i jedan od pionira prodaje usluga i edukacije, svakako je *Red Hat Inc.*Jedan od najvećih problema u razvoju *open-sourcea* je, suprotno ubeđenju mnogih, nesposobnost programera i firmi da *open-source* naplate. Veći deo razvoja zasniva se na dobročiniteljima koji razvijaju besplatno nadajući se ponekoj donaciji.

Prodajmo vrednost a ne proizvod

Jedan od najranijih odgovora i rešenja na ovaj problem došao je iz *Red Hata* i glasio je: „Prodajmo vrednost a ne proizvod." *Red Hat* je celokupan svoj model poslovanja podredio poklanjanju glavnog proizvoda - operativnog sistema, a naplatom svih sekundarnih usluga, kao što su korisnička i tehnička podrška, edukacija i sertifikacija. Ipak, sam model nije došao kao posledica revolucionarnog razmišljanja, već kao potreba da se naplati kôd koji nije bio u vlasništvu *Red Hata*. Suštinski, kôd je bio tuđ.

Ipak, genijalnost ovog modela leži upravo u prodaji vrednosti. *Red Hat* zavisi od *Linuxa*, njegovog razvoja i uspeha, i tako *Red Hat* bezrezervno pomaže njegov razvoj i prodornost na tržište. Time omogućava sebi pristup novoj bazi poslovnih korisnika koji će u budućnosti svoje poslove migrirati na sertifikovane sisteme koje podržava *Red Hat*, jer poslovna arhitektura često ne ostavlja prostor za eksperimente, a često veruje samo čvrstom dokazu kao što je sertifikat. Naravno, ugovor o pretplati[1] koji *Red Hat* sklapa sa korisnicima, uokviruje celu sliku sistema i podrške koji dobijate kao rešenje. Iako možete i bez njega sami da sklopite *Red Hat Enterprise Linux* i koristite ga, koncept je takav da vam se to jednostavno ne isplati. Pretplata takođe osigurava dugoročan i siguran priliv sredstava i osigurava stalnu vezu korisnika sa *Red Hatom*. Okosnicu i završni udarac *Red Hat* zadaje pomoću *Red Hat Networka*, mreže kroz koju plasira ažuriranja i bezbednosne zakrpe i pruža tehničku podršku.

[1] *Red Hat Enterprise Agreement EMEA*

Apsolutni vladar *Linux Enterprise* sveta

Oko 60% implementacije *Red Hatovih* proizvoda, uže *RHEL*-a i usluga koje iz njega proizlaze, odlazi na saradnju sa velikim vendorima i *OEM*-ima, kao što su *Dell*, *Hewlett-Packard* i *IBM*, dok veći deo preostalih 40% odlazi na velike entitete koji koriste *RHEL*. *Red Hat* značajno cilja ka poslovnom svetu, sarađuje sa *OEM*-ima na integraciji svojih rešenja u hardver i popularizaciji svoje distribucije kroz taj hardver. Takođe, veoma je zastupljen i u internoj upotrebi *OEM*-a, vendora i drugih *IT*

kompanija. Sam pristup privatnom korisniku je marginalizovan jer *Red Hat* upravo i cilja na *Enterprise* svet. *Red Hat* je ovim modelom uspeo da se udalji od vendora i *OEM* firme kao što je *Canonical*, koji, uzgred rečeno, kombinuje gotovo sve modele poslovanja sa *open-sourceom*. Pouke koji osnivači, preduzetnici i programeri mogu da izvuku iz ovog modela, jeste da prodajom vrednosti i stvaranjem ekosistema oko vašeg proizvoda možete uspešno da naplatite drugi nivo proizvoda, ali i da finansirate dalji razvoj projekta. U slučaju *Red Hata* udica za korisnika je *Fedora* i upravo kroz nju se i gradi platforma, povećava njena vrednost i pruža poligon za testiranje daljeg razvoja komercijalnih platformi *Red Hata*.Svakako je ovo kratka priča i postoji još mnogo primera prodaje usluga podrške, edukacije i sertifikacije u *Linux* i *open-source* svetu i verovatno ćemo ih pomenuti dalje u serijalu. Kao što smo već rekli, pouke ovog modela za budući razvoj su dodavanje i prodavanje vrednosti i korišćenje prednosti otvorenih platformi na kojima se može izgraditi stabilnost koju je relativno lako naplatiti u turbulentnom svetu promena kakav je *IT* svet. Nadovezujući se na poslednju konstataciju, u sledećem broju pričaćemo o modelu koji je najpopularniji danas, a to je *SaaS* model, i o primeni kakvu ima u *open-source* svetu. Do tada, počnite da radite na nekom novom *Red Hatu*, znamo da umete.

OpenSSL:

Sigurnost ili pretnja

Autor: Petar Simović

Šta je u stvari *OpenSSL* koji je prouzrokovao neviđeno zlo nazvano *Heartbleed*?

OpenSSL je *open-source* implementacija kriptografskog protokola *SSL/TLS* (*SSL - Secure Socket Layer* i njegovog naslednika *TLS - Transport Layer Security*). Ovi protokoli služe za razmenu simetričnog ključa između dva računara kojim se na dalje šifruje sva međusobna komunikacija, ali samo tokom te sesije. Za svaku sledeću sesiju, stari se odbacuju i formiraju sasvim novi ključevi. Možete videti da li se koristi ovakav tip sigurne komunikacije po tome što veb-adrese počinju sa https:// umesto sa http:// .

Ranjivost je otkrivena prvog aprila ove godine od strane Codenomicon, firme za veb-sigurnost zajedno sa *Googleovim* sigurnosnim inženjerima, a publikovana je tek sedam dana kasnije, kada je izašla i prva zakrpa za ranjivost. Sama ranjivost je u okviru ekstenzije zvane „*Heartbeat*" za *TLS* protokol koja je dodata još 2012. godine.

Heartbeat, ili prevedeno „otkucaj srca", jeste dodatak koji omogućava održavanje uspostavljene sigurnosne komunikacije i u trenucima kada nema aktivne razmene poruka između servera i klijenta. Odlika ovog dodatka je da klijent-korisnik šalje poruku sa brojem koji predstavlja dužinu same poruke serveru na kome je *OpenSSL*, server odgovara vraćanjem podataka iz bafera u dužini koja mu je prosleđena uz samu poruku korisnika, što bi trebalo da bude ista poruka koju je i dobio od istog. Problem je što nema provere da li je informacija o dužini poruke ispravna, tj. da li je sama poruka baš toliko duga.

Ovakav problem se može eksploatisati tako što napadač koji se na server konektuje, može serveru da šalje „otkucaj srca" sa kratkom porukom od nekoliko bajtova, a da uz to prosleđuje informaciju da je dužina poruke 64 kilobajta. U tom slučaju, server prima poruku i informaciju o njenoj dužini, vraća je napadaču uz proizvoljne podatke do ukupne količine od *64kb* (*64kb* - nekoliko bajtova) iz bafera koji je rezervisan za potrebe *OpenSSL*-a. Postavlja se pitanje zašto bi napadač želeo *64kb* iz memorije *OpenSSL*-a. Odgovor je zato što baš ta memorija može da sadrži privatni ključ servera (*server's private key*), enkripcione ključeve korisnika logovanih na server, šifre korisnika i kolačiće korisnika (eng. *cookies*). Jednom rečju, to je sve ono što ne bi smeo napadač da zna i zbog čega i postoje sigurnosni enkripcioni protokoli.

Treba napomenuti da je katastrofa još veća time što se napad može ponoviti više puta i tokom svakog napada napadač dobija još 64 „zlatnih" kilobajta, pri tome iza sebe ne ostavlja nikakve tragove samog upada. Međutim, to nije ono najgore, jer se uz malo sreće mogao dočepati i vašeg privatnog ključa i time dešifrovati sav vaš prošli i budući „šifrovani" saobraćaj, ili je možda mogao napraviti lažni sajt (kopiju pravog) od koga je uzeo privatni ključ pretvarajući se na mreži da je pravi pomoću identifikacije privatnim ukradenim ključem i time je mogao da nanese još veću štetu jer može prevariti korisnike originalnog sajta i od njih prikupljati još poverljivih informacija. Zamislite da je otet identitet neke banke ili nekog servisa za trgovinu *bitcoinom*, ili ko zna čega još.

Popularni servisi ugroženi ovim *bagom* su *LastPass* i *Yahoo mail*, a više o tome

na http://goo.gl/59Ct1n i http://goo.gl/og2Pge. Problem je veći jer se softver brzo može zakrpiti, za razliku od hiljade drugih uređaja koji su na mreži i predstavljaju veliki problem jer je neophodna akcija korisnika istih, a među njima su i ruteri, štampači, serveri za skladištenje (*storage servers*), video kamere, vatreni zidovi (*firewalls*) i još mnogi drugi.

Međutim, ovaj problem se mogao izbeći omogućavanjem *PFS*-a (*Perfect Forward Secrecy*) u okviru *OpenSSL*-a, što se i preporučuje. On onemogućava zloupotrebu dobijenog ključa za dekriptovanje prošlog i budućeg šifrovanog saobraćaja, a koji nije podrazumevano omogućen, već mora biti ručno. Za one koji žele da omoguće ovu opciju na njihovim serverima, preporučujemo da prvo pročitaju ovaj dokument (http://goo.gl/-SkAz5v), a vlasnike *Apache* i *NGINX* servera - uputstvo za omogućavanje *PFS*-a koji se može naći na ovim stranama: http://goo.gl/a5vJVV i http://goo.gl/MGHi50. Proveru da li je vaš ili neki drugi sajt otporan na ovu ranjivost, možete izvršiti na ova tri sajta:

1. http://goo.gl/bOQDdx,
2. http://goo.gl/LTN5Uz i
3. http://goo.gl/VofjAx.

Dobra vest je to što je ranjivost samo u okviru *OpenSSL 1.0.1* (*1.0.1f*) i *1.0.2-beta* (*1.0.2-beta1*) verzije i što ostale verzije *OpenSSL*-a nemaju ovakav propust, pa se smatraju bezbednim, barem što se *Heartbleed* ranjivosti tiče, iako je preporučljivo imati uvek najnoviju verziju što je moguće ranije, najbolje pri samom izlasku, kao i omogućen *PFS*. U razvoju su i alternative *OpenSSL*-u, u vidu LibreSSL-a i PolarSSL-a, što je odlično jer će biti veća konkurencija i dve trećine interneta neće zavisiti od jedne jedine slobodne implementacije sigurnosnog protokola.

LibreSSL

POLARSSL
Straightforward, Secure Communication

Korak do *Googla* (5. deo)

Autor: Dejan Čugalj

Posle pauze koja je potrajala, vraćamo se implementaciji modula koji su nam neophodni za ostvarenje krajnjeg cilja koji smo zadali sebi još u 12. broju LiBRE! časopisa.

Poslednji članak o ovoj temi objavljen je u 15. broju, i zbog toga ćemo se podsetiti šta je sve urađeno do sada. U 12. broju smo predstavili neke uporedne podatke ljudske potrebe za pretraživačima u informacionim tehnologijama i ukratko smo spomenuli *Lucene* biblioteku. Broj 13. smo iskoristili za detaljnije upoznavanje sa istom, dok smo u broju 14. potpuno ušli u priču i predstavili studiju slučaja predstojećeg projekta. U 15. broju LiBRE! časopisa, tj. u četvrtom nastavku serije članaka „*Lucene* – korak do *Googla*", počinjemo sa implementacijom projektnih modula *Lucene* projekta. U tom izdanju smo implementirali prva dva modula: „Pronalazak svih *PDF* datoteka" i „*Tika* ekstrakcija". Takođe, ovaj izuzetan *Apache Tika framework* našao je svoje mesto u zasebnom članku i približio nam svoje mogućnosti.

S obzirom na količinu kôda koji će biti napisan, odlučili smo se za *GitHub* javni repozitorijum (Koristan link: https://github.com/libreoss/lucene-moduli) u koji je postavljen celokupan kôd, dok one najbitnije delove objavljujemo u časopisu. Nekako, kada sve saberemo i oduzmemo, to je ono što bi predstavilo kratku rekapitulaciju serijala „*Lucene* – korak do *Googla*".

Peti deo serijala nam donosi nastavak implementacije modula i to su: „Analiza dokumenata", „Indeksiranje *PDF* datoteka" i „Indeks (pregled *Lucene* strukture datoteka)". Ovo je dobar momenat da podsetimo da svi moduli iz studije slučaja obojeni zelenom bojom jesu tačnije oni delovi gde nam *Lucene* pruža svoje usluge i dovoljno je samo biti upoznat sa njenim *API*-jem (engl. *Application Programming Interface*) da bi se mogla i koristiti. Upravo taj *API* (http://bit.ly/SBcNKf) i dokumentacija dostupni su na adresi: http://lucene.apache.org/core/4_8_0/index.html (verzija aktuelna u toku pisanja ovog članka je 4.8.0).

Pre nego što počnemo, potrebno je dodati *Lucene* biblioteke u *Eclipse IDE* okruženje, koje su dostupne za preuzimanje sa zvaničnog sajta na adresi: http://lucene.apache.org/core/ Dodavanje biblioteka u projekat *Eclipse IDE* okruženja detaljno je objašnjeno na adresi: http://bit.ly/1qfFRpb

4. Analiza datoteka

Analiza datoteka

U prvom delu serijala, ako ste pažljivo čitali, dotakli smo se tokenizacije i šta bi ona ukratko trebalo da predstavlja (12. broj LiBRE! časopisa). Upravo je ovo deo gde ona dolazi do značaja. Pretraživači ne indeksiraju tekst direktno, već se sadržaj, kandidat za indeksiranje, rastavlja na manje delove (*Atomic part*), koji se nazivaju *tokeni*. Upravo se ovaj proces odvija u ovom modulu. Kako se ovo odvija implicitno, oko ovog modula u ovom momentu ne moramo previše da brinemo, ali ako bismo se udubili u problematiku, videli bismo da je od velikog značaja i da je jedan od ozbiljnijih problema. Ukratko, prilikom ovog procesa rešavaju se krucijalna pitanja kao što su: da li je potrebno obraćati pažnju na smisao reči (problem sinonima), da li je potrebno prilikom pretraživanja obratiti pažnju na semantičku stranu prirode reči kao što su laptop i računar, da li je potrebno uzimati u obzir infinitiv? Problemi i pitanja se mogu dovesti do filozofsko-filološkog nivoa, tako da bismo ovu temu ostavili za neke druge, filološke nauke. Ono što je bitno za nas, jeste to da *Lucene* daje niz alata, sa kojima makar iole možemo da se približimo nekim realnim rezultatima. I konačno, dolazimo do najbitnijeg dela i srži ovog serijala, a to je „indeksiranje".

5. Indeksiranje

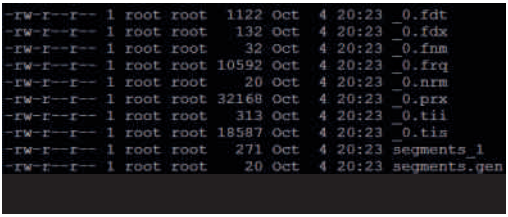
Indeksiranje PDF datoteka

Do sada smo, možda i prečesto, spominjali ovu famoznu reč „indeksiranje", tako da ovo zauzima centralni i najbitniji deo celog serijala. Kako bi hteli da ovaj deo bude respektivno, koliko je to moguće, približan kvalitetu uloženog truda u razvijanju *Lucene*, ovde stajemo i primer dajemo malim „školskim primerom" implementacije *Lucene* indeksiranja napisanim u *Java* programskom jeziku. S obzirom da je ovo centralni (*core*) deo serijala, respektivno zaslužuje i mesto u njemu, tako da ćemo ga detaljno predstaviti u sledećem nastavku.

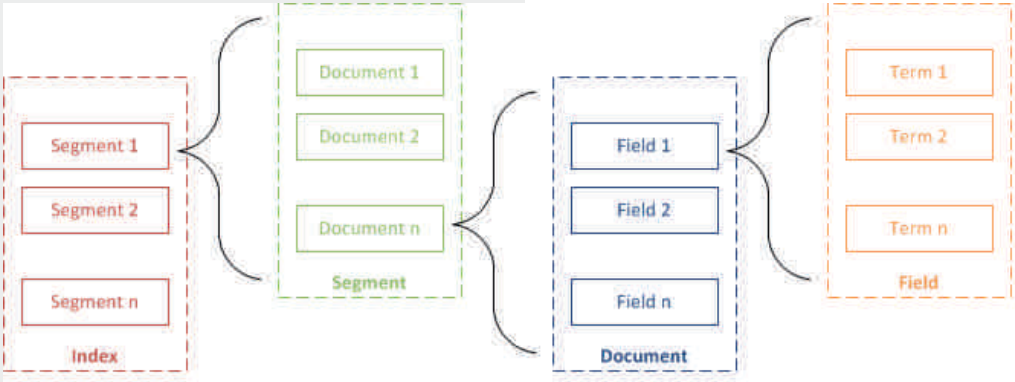
```
import java.io.File;

import org.apache.lucene.analysis .Analyzer;
import org.apache.lucene. analysis.standard.StandardAnalyzer;
import org.apache.lucene. document.Document;
import org.apache.lucene. document.Field;
import org.apache.lucene. document.StringField;
import org.apache.lucene. index.IndexWriter;
import org.apache.lucene.index. IndexWriterConfig;
import org.apache.lucene.store. Directory;
import org.apache.lucene.store. FSDirectory;
import org.apache.lucene.util. Version;

/** * */
public class LuceneIndexExample {
  public static void main(String args[]) throws Exception {
    String text = "Ovo je tekst indeksiran sa Lucene";

    String indexDir = System.getProperty("user.dir")
      + System.getProperty("file. separator") + "index";
    System.out.println(indexDir);
    Directory dir = FSDirectory. open(new File(indexDir));
    Analyzer analyzer = new StandardAnalyzer(Version. LUCENE_48);

    IndexWriterConfig iwc = new IndexWriterConfig(Version.LUCENE_48,
analyzer);

    IndexWriter writer = new IndexWriter(dir, iwc);
    Document document = new Document();

    Field pathField = new StringField("ime_polja", text,
Field.Store.YES);
document.add(pathField);

    writer.addDocument(document);
    writer.close();
  }
}
```

6. Indeks (pregled *Lucene* strukture datoteka)

Osnovni i fundamentalan koncept *Lucene* jesu: indeks (*index*), dokument (*document*), polje (*field*) i pojam (*term*). Kada bismo to „sklopili", dobili bismo strukturu da indeks sadrži sekvence dokumenta. Dokument je sekvenca polja, polje je imenovana sekvenca pojma, dok je pojam sekvenca bajtova.



Vidimo se u sledećem broju sa našim „*Lucene* indeksima".



Index Structure

Uglavnom, cela magija se odvija pod „*Lucene* haubom" i, ukoliko bismo želeli da uđemo dublje u tematiku, sigurno bi nam ponestalo prostora. Ovo je sasvim dovoljno za sticanje slike i saznanja da posle indeksiranja, *Lucene* pravi svoju strukturu fajlova koja se posle koristi za pretragu.

Korisni linkovi:

- **Zvanični sajt:** http://bit.ly/LdDxwN
- **Izvorni kôd:** http://bit.ly/1pIFmQo
- **Sistemski zahtevi:** http://bit.ly/1kN0bXP

http://2014.dorscluc.org