Avgust/Septembar 2014.

LiBRE!

Časopis o slobodnom softveru

broj 28

07. septembar, 2014.

Održan je drugi *BalCCon 2k14.*

12. septembar. 2

Turin prelazi na *GNU/Linux*

Reč urednika

Povratak energije

Na početku moramo prvo da se izvinimo zbog zakašnjenja ovog broja. Komercijalni časopisi niti se izvinjavaju za kašnjenje, niti objašnjavaju zbog čega je do njega došlo. Njihovi čitaoci očekuju novi broj svakog meseca i na vreme, i ne zanimaju ih razlozi kašnjenja. LiBRE! ima obavezu da se izvini i da objasni kašnjenje, jer to zanima članove zajednice. Članovi naše zajednice nisu samo naši čitaoci nego praktično - porodica. Porodica čiji pojedinci nisu samo puki čitaoci, nego se po potrebi i želji uključuju u aktivno kreiranje ovog projekta. Iz tih razloga, dobra komunikacija sa našim čitaocima od ključne je važnosti.

Kroz ova naša objašnjenja razloga kašnjenja, oslikava se ne samo stanje u projektu, nego i u samoj zajednici, što ako se pravilno razume, može da dovede samo do poboljšanja stanja. Ovog puta postoji više faktora za kašnjenje broja. Možda je najvažniji faktor pad energije koji je zahvatio celu zajednicu slobodnog softvera, i to ne samo u Srbiji nego i u regionu, pa samim tim i ovaj projekat. Zamor materijala starijih (po stažu) članova redakcije, slabija komunikacija i dezorganizacija mlađih članova redakcije, dovela je do pada produktivnosti. Ono što smo završavali za petnaest dana, sad nam je za to trebalo mesec dana. Do dodatnog opuštanja je došlo jer smo unapred planirali da kasnimo sedam dana zbog *BalCCona 2k14* čije se održavanje poklopilo sa našim uobičajenim terminom objavljivanja novog broja (prva nedelja u mesecu). Ovo planirano kašnjenje se poklopilo i sa septembarskim ispitnim rokom, tako da smo sa planiranog prešli i na neplanirano kašnjenje.

Pitanje je kada ćemo nadoknaditi ovoliko kašnjenje i ponovo se vrati u standardni ritam mesečnog izlaska časopisa. Jedno je dobro, da se energija polako vratila u zajednicu i projekat. Za to je, pre svega, zaslužan *BalCCon 2k14*.

Odluka da zanemarimo sve obaveze, da kao redakcija i pojedinci što masovnije učestvujemo na ovogodišnjem *BalCConu*, bila je dobra. Zahvaljujemo se organizatorima što su nam to i omogućili. I pored toga što smo unapred znali da smo kao redakcija potpuno nespremni da ovako veliki događaj profesionalno novinarski servisiramo, samo učešće je bilo jako korisno. Niko u redakciji LiBRE! nije profesionalni novinar koji bi mogao da se u trenutku snađe i napravi pravi intervju sa običnim prolaznikom, a kamoli sa ljudima kao što su Mič Altman ili Bernd Fiks koji znaju šta je pravi intervju i kada profesionalni novinar prilazi nespreman za intervju. Čak i školovani novinari ispadaju smešni ako nisu pripremljeni za razgovor. Redakciji LiBRE! je dodatno bilo teško da spremi „novinare“ jer većina nikad nije ni bila na ovako velikom događaju i nisu mogli ni da zamisle šta ih tamo čeka. Zato je ovo iskustvo neprocenljivo. Ne samo da smo zakazali kao novinarski servis *BalCCona*, nego smo propustili i priliku za bolju promociju časopisa. Imali smo u planu da postavimo štand časopisa sa prigodnim materijalom, ali ga nismo realizovali zbog slabe organizacije, neznanja i nedostatka novca. Sama prezentacija je delo više pojedinca a ne organizacije čitave redakcije. Potonuće do samog dna, totalni fijasko, ponekad i nije tako loš položaj. Bar imamo čvrsto

dno od kojeg možemo da se odgurnemo i počnemo da isplivavamo na površinu. Uz to *BalCCon* sa svojom energijom, pravi je dodatni balon koji vraća energiju i može da pomogne da brže isplivamo ako se čvrsto uhvatimo za njega. Sad je samo bitno da napravimo dobru analizu i krenemo u pravom smeru. *BalCCon* ne vraća energiju samo projektu, nego i celoj srpskoj zajednici. U uslovima nedostatka energije u samim zajednicama, uvoz energije iz inostranstva je neprocenljiv. *BalCCon* je nedvosmisleno pokazao da *FLOSS* filozofija nije mrtva, kako je to izgledalo u poslednje vreme. Vesti sa optužbama da *Canonical* vuče *Ubuntu* ka komercijalnom softveru, povratak minhenskih računara na *Windows*, smanjenje aktivnosti u lokalnim *FLOSS* zajednicama, problemi sa hostingom *Ubuntu-rs* i nešto ranije *Archlinux-rs*, faktori su koji su trošili domaću *FLOSS* energiju. Hvala *LUGoNS*-u, a posebno Milošu Krasojeviću što su organizovali ovaj kongres i doneli novu *FLOSS* energiju u Srbiju.

LiBRE! ide dalje sa novom energijom. Pozivamo sve one koji osećaju priliv ove nove energije da nam se jave na našu, već poznatu adresu elektronske pošte libre [et] lugons [dot] org.

Do čitanja

LiBRE! tim

Moć slobodnog softvera

Broj: 28
Periodika izlaženja: mesečnik

Izvršni urednik: Stefan Nožinić

Glavni lektor: Jelena Munćan

Lektura:
Aleksandar Božinović
Milena Beran
Maja Panajotović
Aleksandra Ristović

Redakcija:
Dejan Čugalj
Marko Kažić
Veljko Simić
Nikola Hardi
Petar Simović
Zlatan Vasović
Aleksandar Vesić
Aleksandar Todorović
Gavrilo Prodanović
Aleksandar Brković
Mihajlo Bogdanović
Vladimir Cicović
Marko Novaković

Saradnici:
Goran Mekić
Željko Popivoda
Joakim Janjatović
Jelena Georgijević
Sandrina Dimitrijević
Nedeljko Stefanović
Stefan Stojanović
Vladimir Popadić

Počasni članovi redakcije:
Aleksandar Stanisavljević
Željko Šarić

Grafička obrada:
Dejan Maglov
Ivan Radeljić

Dizajn:
Mladen Šćekić
Zoran Lojpur
White Circle Creative Team

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

http://libre.lugons.org

LiBRE! prijatelji

ubuntu SRBIJA

Slackware Srbija

WHITE CIRCLE & CREATIVE TEAM
GRAPHIC DESIGN & WEB DEVELOPMENT

ЈУГоНС
УДРУЖЕЊЕ ЛИНУКС КОРИСНИКА НОВОГ САДА

ЛИНог
Линукс блог | вести и мишљења

linux zasve
REGIONALNI LINUX PORTAL
linuxzasve.com

archlinux.rs

Linux fondacija nudi stipendije onima koji bi pomogli u razvoju slobodnog softvera
4. avgust 2014.

Linux fondacija i ove godine nudi stipendije studentima koji razvijaju slobodan softver.

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

Elementary OS Freya beta izdanje
11. avgust 2014.

Izašlo je novo beta izdanje ove *Linux* distribucije.

Koristan link:
http://j.mp/1wwC0U4

***Fedora* radi na novom menadžeru particija**
7. septembar 2014.

Fedora izrađuje novi menadžer particija koji će podržavati sisteme koji do sada nisu podržani u grafičkim programima za ovu namenu.

Koristan link: http://j.mp/1mmt4js

Održan je drugi internacionalni Balkanski hakerski kongres
7. septembar 2014.

Od 5. do 7. septembra održan je drugi po redu internacionalni Balkanski hakerski kongres u Novom Sadu u organizaciji LUGoNS udruženja i *Wau Holland* fondacije.

Koristan link: http://j.mp/XvGVrZ

Fondacija za slobodan softver i projekat Debian su pokrenuli bazu hardvera koji podržava Linux
8. septembar 2014.

Free Software Foundation i *Debian* su pokrenuli bazu hardvera koji podržava *Linux*. Za razliku od ostalih baza koje se bave ovom tematikom, njima je cilj promocija hardvera koji ne zahteva vlasničke drajvere.

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

Hakovan zvaničan sajt za Bitcoin
9. septembar 2014.

Hakovan je zvaničan sajt za *Bitcoin* kao i adresa elektronske pošte *Satoshi Nakamoto*, osnivača ove valute.

Koristan link: http://j.mp/1AVf69u

GCC 5 će imati punu podršku za Cilk Plus
10. septembar 2014.

GCC će imati punu podršku za ovu *Intelovu* tehnologiju od verzije 5.

Koristan link: http://j.mp/1o9ppBj

Counter-Strike: Global Offensive uskoro na Linuxu
11. septembar 2014.

Valve planira portovanje ove popularne igre na *Linux*. Ova igra postoji za *Windows* i *OS X* već dve godine.

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

Turin prelazi na Linux
12. septembar 2014.

Ovaj grad u Italiji predviđa uštedu od šest miliona evra prelaskom na slobodan softver.

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

Android Wear će „pregaziti" Apple Watch
12. septembar 2014.

Analitičari predviđaju da će *Android Wear* pregaziti *Apple Watch* na ovom polju.

Koristan link: http://j.mp/1sDtCP1

2k14 - Second Base

Prvi utisci

Autor: Stefan Nožinić

Kao što svi dobro znate, početkom septembra, održan je najveći događaj na Balkanu namenjen promociji slobodnog softvera, slobodi obrazovanja i haktivizmu. Ovogodišnji *BalCCon* je drugi po redu a organizovan je pod sloganom „*Second Base*" koji je odličan nastavak prošlogodišnjeg slogana „*First Contact*". Iako je teško preneti sjajnu atmosferu i ogromnu energiju ljudi koji su prisustvovali ovogodišnjem *BalCConu*, pokušaćemo da vam u ovom tekstu prenesemo makar tračak čitavog doživljaja. Deluje pomalo grubo da čitav događaj podelimo u tri dana, jer je hakerska energija vladala i noćima, stoga ćemo opisati svaki dan zasebno.

Pred *BalCCon*

BalCCon je, za neke, počeo dan ranije, nego što je najavljeno. U Muzeju savremene umetnosti Vojvodine sastali su se 4. septembra u 11 časova glavni organizatori događaja i volonteri kako bi se dogovorili o planovima za naredna tri dana da bi se kongres odvijao predviđenim tokom. Posle kratkog sastanka, ostatak dana pratila je opuštena atmosfera uz piće, dobar provod i, naravno, hakovanje.

Petak - prvi dan

Jutro je pomalo kišovito, ali to nije moglo da spreči ovako sjajan događaj gde se ljudi druže, zabavljaju i razmenjuju znanje. Već od ranih sati mogle su se u Muzeju videti gomile ljudi kako diskutuju na razne teme od kojih neke retko kada čujete u vašem svakodnevnom okruženju. Svi su čekali prva predavanja i radionice koji su bile održani u dve sale. Obe sale su dobile i svoje nazive po našim najvećim naučnicima: Tesli i Pupinu.

Malo po malo, i krenula su prva predavanje. Prvo predavanje je bilo na temu kvantne kriptografije. Iako je ovo sasvim nova disciplina, za dosta ljudi, pomalo kompleksna, mnogi su ovo slušali sa oduševljenjem i radoznalošću šta sve mogu da urade kvantni računari. *Bernd Fix* nam je ukazao na važnost razvoja algoritama za enkripciju koji uzimaju u obzir moć kvantnih računara i ukazao je na činjenicu da današnji kripto algoritmi mogu biti lako probijeni korišćenjem kvantnih računara.

Posle ovog sjajnog predavanja, usledila je reč o patentima i autorskim pravima. Ukazano je šta je patent, šta je autorsko pravo i definisane su mnoge stvari koje se koriste u pravnim vodama. Nakon uvoda, Žarko Ptiček govorio je o tome gde se tu nalazi Srbija, ali isto tako je pomenuo zanimljivosti vezane za ostale zemlje kao i probleme koje postoje u sistemima tih zemalja. Paralelno sa ovim predavanjem, Đorđe Marković je pričao o *3D* štampačima, *bio-printingu* i kombinaciji trodimenzionalnog skeniranja i trodimenzionalnog štampanja.

Posle je usledilo predavanje Vasila Koleva o sistemima datoteka, sigurnosti tih sistema i načinima kako takve sisteme učiniti sigurnim korišćenjem enkripcije. Paralelno je u Tesli o

fraktalima govorio Predrag Bokšić. On je govorio o haosu i o tome kako je svet oko nas haotičan i kako možemo vladati takvim sistemima.

Spava li vam se? *Christina Johanna* nam je pričala o spavanju. Najveći problem kod hakera jeste spavanje. Kako funkcioniše san? Kako da spavamo kvalitetnije i kako da kontrolišemo našu pospanost? Ili, bolje reći, kako da uhakujemo naš san? I tako, dok su neki spavali, drugi su razgovarali o startapima (eng. *startup*). Nebojša Vislavski i Nikola Novaković pričali su o velikim kompanijama, kako izabrati najbolju tehnologiju za startap i gde se otvorene tehnologije nalaze u trci sa komercijalnim.

David Oswald nam je pričao o elektronskom zaključavanju koje se sve više koristi u raznim zgradama i o tome kako se mogu takvi sistemi probiti. Paralelno sa tim, održani su *Lighting talks* gde su predstavljeni razni projekti u govoru od pet do deset minuta. Između ostalog, Nikola Hardi je predstavio LiBRE! i ukazao na cilj projekta, planove kao i probleme sa kojima se suočavamo. Već u kasnim satima ste mogli čuti predavanje o projektu *Tor*, o kom je pisano i u našem časopisu. *Moritz Bartl* nam je pričao zašto je *Tor* važan i kako se koristi. Iako smo pokušali da krenemo hronološki, namerno smo izostavili jedno predavanje i ostavili ga za kraj. *Mitch Altman* nam je pričao o slobodnom hardveru, ali ne samo o tome. Od ovog sjajnog hakera i inovatora, mogli se čuti dosta

pametnih stvari. *Altman* je pričao o tome zašto je važno da radimo ono što volimo. Pričao je o *hackerspaceu* i pomenuo je jedan takav u San Francisku: *Noise Bridge*. *Altman* je nudio i ključeve kako bi svi imali pristup prostorijama nezavisno da li je neko tu ili nije. Već na početku predavanja nije se moglo ući u salu jer su sva mesta bila zauzeta. *Mitch Altman* je ukazao na mnogo bitne stvari i preneo svoj entuzijazam. Pričao je i o svojoj inovaciji „*TV-B Gone*", uređaju namenjenom za daljinsko isključivanje televizora na javnim mestima. Da je ovo na mnoge ostavilo ogroman pozitivan utisak, pokazao je i veliki aplauz zadivljene većine.

Subota - drugi dan

Vikend - dosta ljudi ne radi vikendom, a i učenici i studenti su slobodni. O tome svedoči i činjenica da je broj ljudi ovog dana značajno porastao. Ono što vam možemo odmah reći, jeste da su volonteri zaduženi za stavljanje narukvica za posetioce, imali pune ruke posla ovog subotnjeg jutra.

Od ranog jutra krenuli smo pravo u metu: Vakcinacija *Androida*. Da li možemo verovati mobilnim aplikacijama danas? Kako da ih testiramo i kakve aplete (eng. *applets*) možemo koristiti? O tome su nam pričali Milan Gabor i Danijel Grah. Paralelno se u Tesli govorilo o automatizaciji domova za pedeset evra. Zvuči nemoguće? Ne za Nikolu Rasovića. On nam je pričao kako je automatizovao svoj dom korišćenjem *Raspberry Pia*.

Aleksandr Timorin je pričao o *SCADA* protokolima i njihovoj sigurnosti, a na kraju nam je pokazao i sve to u praksi kroz demonstraciju.

Za to vreme, Ivica Kolenkaš je govorio o struktuiranju podataka korišćenjem *MongoDB*-a i o važnosti svega toga. Luka Gerzić nam je pričao o razvoju *WEB* aplikacija sa fokusom na bezbednost. Ukazao je na česte greške koje, ponekad i najbolji programeri, mogu da naprave. Predavanje je bilo propraćeno zabavnim temama i možemo slobodno reći da smo u sebi rekli „Čoveče, ovo se i meni jednom desilo".

Patroklos Argyroudis je održao predavanje o eksploataciji memorije. Pominjao je česte greške kod ovog nivoa kao što su *buffer overflow*, *use-after-free* i drugi. Za sve one koji nisu prisustvovali radionicama početkom godine o *OpenStreetMapu* koje je držao Hrvoje Bogner, mogli su to uraditi i na *BalCConu*. On je održao radionicu i govor o tome šta je *OSM* i kako se koristi. Ako se plašite da vam neko ne ukrade kola, mogli ste čuti predavanje o zaštiti automobila. Šta vam je potrebno? Ništa specijalno, jedan *Arduino* i nekoliko dodataka.

Ovo potvrđuje i Marian Marinov koji nam je na svom primeru pokazao kako da zaštitimo svoj automobil. O merenju je govorio Marjan Urekar. Govorio je o multimetrima, kako da merimo napon, struju, otpor i ostale veličine. Zvuči jednostavno? Niste svesni koliko tu ima detalja i koliko je bitno znati kako ti uređaji funkcionišu. Šta ćemo sa signalima i raznim njihovim oblicima? Marjan Urekar nam pokazuje i kako da koristimo osciloskop i objašnjava njegovu funkcionalnost. Austrija, zemlja demokratije - nikako! *Fin* nam je govorio o slobodi informacija i probleme na koje ona nailazi. Govorio nam je kako Austrija sprečava novinare i hakere u radu, zašto su novinarima potrebni hakeri, i zašto su hakerima neophodni novinari. *Anil Kurmus* je pričao o napadima na *Linux* kernel. On je govorio o površini napada na kernel i njenoj redukciji uklanjanjem mogućnosti kernelu. Predstavio je dva pristupa koji to mogu da izvedu, jedan tokom kompajliranja, a drugi tokom izvršavanja. Predstavio nam je i zanimljive statistike i poređenja. Miroslav Štampar je govorio o sigurnosti i eksploataciji softvera, nekada i sada. Ako ste bili na ovogodišnjem *BarCampu* u februaru, mogli ste čuti *Silvana Gebhardta* o *BGP routingu*. On je pružio uvid u ovu tematiku i na ovogodišnjem *BalCConu*. Mogli smo čuti i predavanje o *RIPE Atlas API*-u i saznati čemu služi i kako ga možemo koristiti za merenja i istraživanja. Ovo predavanje je održala Vesna Manojlović. Na kraju dana, usledio je „*Rakija workshop*”. Od dunje do kruške, mogli ste probati naše nacionalno piće na čije neke varijacije Srbija ima registrovan *trademark*. Okupili su se svi u sobi za radionice s ciljem da

probaju razne vrste rakije i da se druže posle dugog dana.

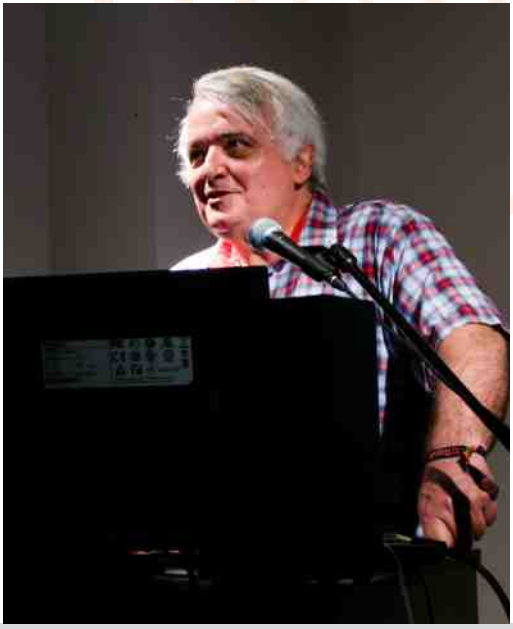
Nedelja - treći dan

Od ranog jutra krenulo se radno. *Mitch* nam je održao radionicu o lemljenju i polaznici su dobili priliku da sami naprave sopstveni *TV-B gone* uređaj koji su mogli da ponesu svojoj kući. Bilo je trideset mesta za ovu radionicu i već na samom početku sve je bilo popunjeno, a polovina mesta je već bila rezervisana.

Pored ove radionice, u toku su bila predavanja o slobodi govora i ulozi interneta koje je održao Andrej Petrovski, predavanje o konfliktima prava na privatnost, zaštitu podataka i na intelektualnu svojinu koje je održala Jelena Jovanović. Predavanje o Vikipediji, kako se pišu dobri članci na njoj i gde se žene nalaze u svemu tome, održala nam je *Greta Doci*. Nakon *Mitchove* fantastične radionice, Aleksandar Pejić i Andrija Prčić su nam pričali o *Linux* drajverima (eng. *driver* - upravljački program) za ugrađene (eng. *embedded*) sisteme. Tonimir Kisasondi je govorio o krekovanju šifri i pametnom generisanju lista reči za tu namenu. Darko Ivković je govorio o ljudskom satu, kako pojedinci doprinose globalnoj mreži i sličnim stvarima. Za to vreme *Anand Buddhdev* nas je uveo u *Ansible*, sistem upravljanja servera ali i ostalih sistema pa čak i vašeg kućnog računara. Pokazao je njegove prednosti i način korišćenja. Mi planiramo da u narednim brojevima napišemo tekst o ovom sistemu i tako pokažemo i vama kako se ovaj alat može pametno koristiti za olakšavanje raznih poslova kod kuće ali i u preduzećima. Vlatko Kosturjak nam je govorio o *77FEh* i prikazao kakvi ga sve uređaji koriste i objasnio šta je to i čemu služi. Čaba Pardovički nam je pokazao šta je *Docker* i zašto je to jedan od omiljenih alata ljudi koji se bave razvojem softvera u poslednjoj deceniji, kako se koristi i koje su mu prednosti. Žarko Živanov nam je pričao kako su nastali kućni računari u Jugoslaviji i, kao da je ciljano, a možda i jeste, napravljen je odličan uvod za naredno predavanje, poslednje ove godine na *BalCConu*.

To poslednje predavanje je možda ostavilo onoliki utisak koliko je ostavio *Mitch* prvog dana. Voja Antonić, čovek

koji je sa trinaest godina, bez ogromnog znanja elektronike, napravio sistem koji je služio ljudima da nauče gde se nalaze veliki gradovi u Jugoslaviji. Da li je tu stao? Nije. Uprkos činjenici da je tada u Jugoslaviju bilo teško uvesti bilo šta što je bilo malo skuplje, to ga nije sprečilo da nabavi sebi svoj prvi računar iz SAD. Ako se pitate kako, pokušajte da zamislite scenario gde mu je dostavljan jedan računar iz više delova putem više paketa. Na njegovu sreću, ali i na našu, to se isplatilo. On nije bio jedan od onih koji su uzimali računare da ih koriste, već jedan od onih koji su uzimali računare da bi naučili kako oni funkcionišu. Malo po malo nastaje i prvi kućni računar u Jugoslaviji - Galaksija. Voja Antonić je ispričao razne anegdote iz njegove karijere, a publika je to sa zadovoljstvom slušala. Tokom predavanja svi su se dobro nasmejali i prekidali Antonića ogromnim aplauzima. Na kraju je usledio najveći aplauz, a zatim i slikanje sa ovim sjajnim inovatorom.



Van sala za predavanja

Na kongresu je bilo moguće videti i neke primerke starih računara i posetioci su imali priliku da se oprobajuu igricama iz osamdesetih godina prošlog veka, ali i da vide *quadcopter*, malu letelicu koju pokreću četiri rotora. Pored svega ovoga, na kongresu su bili i štandovi za zajednice između kojih je bio i štand za naš časopis gde se mogao uzeti *CD* sa arhivom dosadašnjih brojeva našeg časopisa.

Za kraj

Možemo sa sigurnošću reći da je ove godine *LUGoNS* opravdao naša očekivanja i da se nadamo da će sledeće godine biti još bolje. Takođe priznajemo da smo se kao časopis mogli više angažovati, ali na greškama se uči tako da ne vredi da žalimo što nismo napravili intervju sa predavačima.

U svakom slučaju možemo da zaključimo da su ovakvi događaji najbolji primer promocije slobode obrazovanja i haktivizma i da ih treba organizovati i većem broju.

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

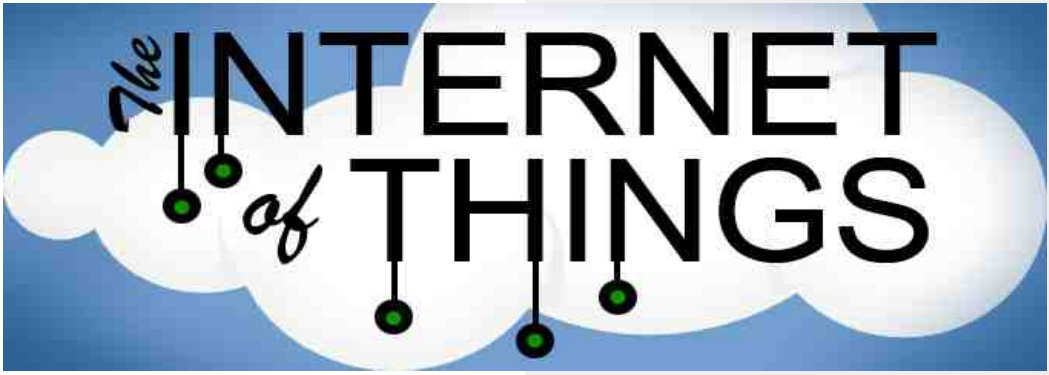
Distrowatch

1	Mint	2199<
2	Ubuntu	1874>
3	Debian	1548>
4	Fedora	1272>
5	openSUSE	1231>
6	Arch	1174>
7	Mageia	1102<
8	Kali	1016>
9	CentOS	991>
10	Deepin	921>
11	LXLE	811>
12	Puppy	803<
13	elementary	780<
14	Lubuntu	774=
15	Zorin	701<
16	Q4OS	653>
17	Gentoo	651>
18	PCLinuxOS	629<
19	Simplicity	624>
20	Absolute	611>
21	Ultimate	602<
22	Android-x86	596>
23	4MLinux	568<
24	SparkyLinux	544>
25	FreeBSD	544>

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

Slobodan softver i internet stvari

(1. deo)



Autor: Aleksandar Todorović

Da li možemo računamo na softverske gigante?

Internet je postao rasprostranjeniji nego što je iko mogao pretpostaviti. Današnji pametni telefoni imaju više snage nego super-računari koje smo koristili pre samo 20 godina. Kompanija *Cisco* predviđa da ćemo do 2020. imati preko 50 milijardi uređaja spojenih na internet. Faza priključivanja računara, laptopa i telefona na internet je već iza nas, a sada su došli na red i ostali uređaji: pametne naočare, pametni satovi, pametne kuće, pametni automobili... Nova rešenja i novi „pametni" predmeti se pojavljuju na tržištu rekordnom brzinom i to je već svima vrlo poznato.

Kako se *Linux* snalazi na novom tržištu koje je veće nego ikada do sada?

Otvorene tehnologije imaju ogromnu prednost nad vlasničkim rešenjima u ovoj fazi razvoja interneta. Više se ne radi o jednoj vrsti uređaja i o jednom operativnom sistemu koji treba da komunicira sa istim tipom uređaja. Sada se radi na tome da se poveže što više različitih uređaja i da se omogući njihova međusobna komunikacija i upravljanje. Da bi jedna kompanija tu preuzela prevlast, morala bi da u vrlo kratkom vremenu izbaci mnogo kvalitetnih različitih proizvoda na tržište i da uveri potrošače da su baš njihovi proizvodi ono što im treba. Iako ne želimo da potcenjujemo velike

kompanije kao što su *Google*, *Microsoft* i *Apple*, mišljenja smo da su šanse da se to dogodi veoma male, pogotovo uzimajući u obzir da je ta trka već u toku, a od *Microsofta* i *Appla* još uvek nismo videli ništa van tradicionalnog tržišta koje obuhvata računare, laptopove, tablete i pametne telefone. *Google* i te kako vodi u toj bici jer je do sada na tržište već izbacio pametne televizore, satove i naočare (sve bazirano na *Androidu*), a pored toga je uradio ogroman posao i kada je u pitanju plasiranje na tržište prvog automobila koji ne zahteva ljudsko upravljanje. Jedina prednost *Googla* u odnosu na ostala dva giganta na tržištu jeste upravo taj što je *Googlov* softver baziran na otvorenom kodu, te ga je kao takvog lakše prilagoditi za različite namere i različite uređaje.

Međutim, da li zajednica koja voli otvoren softver može da računa samo na *Google*? Svi znamo da ta zajednica (iz već dobro poznatih razloga) nije previše oduševljena ni sa *Androidom* za pametne telefone i biće im vrlo teško da prihvate da im neka prilagođena verzija *Androida* pokreće i ostale uređaje.

Sklanjajući *Google* na stranu za sada, ostaje nam *Red Hat*, koji se ne fokusira na borbu za gigantima, *Canonical* još uvek pokušava da prati korak te da se probije na tržište pametnih telefona i tableta, i *Mozilla* koja pored probijanja na tržište pametnih telefona pokušava da uradi nešto potpuno drugačije – da nam osigura otvoreni internet.

Kako rade pametni uređaji?

Da bismo shvatili kako da napravimo potpuno otvorene uređaje, moramo naučiti kako oni rade. Procesori u današnjim pametnim telefonima imaju više snage nego što su imali superračunari prije samo 20 godina, što je vrlo impresivan podatak koji dokazuje koliko smo napredovali. Međutim, iako su danas minijaturni procesori vrlo jaki, nisu besplatni. Većina uređaja koji će u budućnosti biti spojeni na internet već su odavno u upotrebi u svom „glupom" izdanju (u kontekstu „anti-pametni" upoređujući sa pojmom pametni telefoni). Naša trenutni budilnik ne može da zna da smo budni već dva sata, naš frižider ne može da zna da nam je ponestalo mleka u njemu, a aparat za pravljenje kafe ne može da pretpostavi kakva bi nam kafa odgovarala ovog jutra. U njima se ne nalazi procesor, nego mikrokontroler, mali uređaj koji služi isključivo da uređaje uključimo i isključimo i da javi ostatku sistema šta da rade kada je uređaj uključen. Sada, zamislimo da sve mikrokontrolere u svim uređajima odjednom zamenimo procesorima i damo im neke „pametne" osobine. Po navici, kada ustanemo, počnemo da radimo nešto na našem

telefonu, bilo da proveravamo koliko je sati, proveravamo mejlove, društvene sajtove ili nešto slično. Ukoliko radimo neku malo kompleksniju radnju (radnju koju nije moguće uraditi ukoliko nam se spava), naš telefon bi tada mogao da pošalje informaciju našem budilniku da se isključi, frižider bi mogao da proveri imamo li dovoljno mleka i da nas obavesti o tome da li treba da odemo do prodavnice, a aparat za kafu bi mogao da zna da smo budni već neko vreme, te na osnovu toga da pretpostavi da nam nije potrebna jaka kafa da nas razbudi ovog jutra. Sa današnjom tehnologijom mi smo odavno u mogućnosti da tako nešto izvedemo, pa šta nas to u tome sprečava?

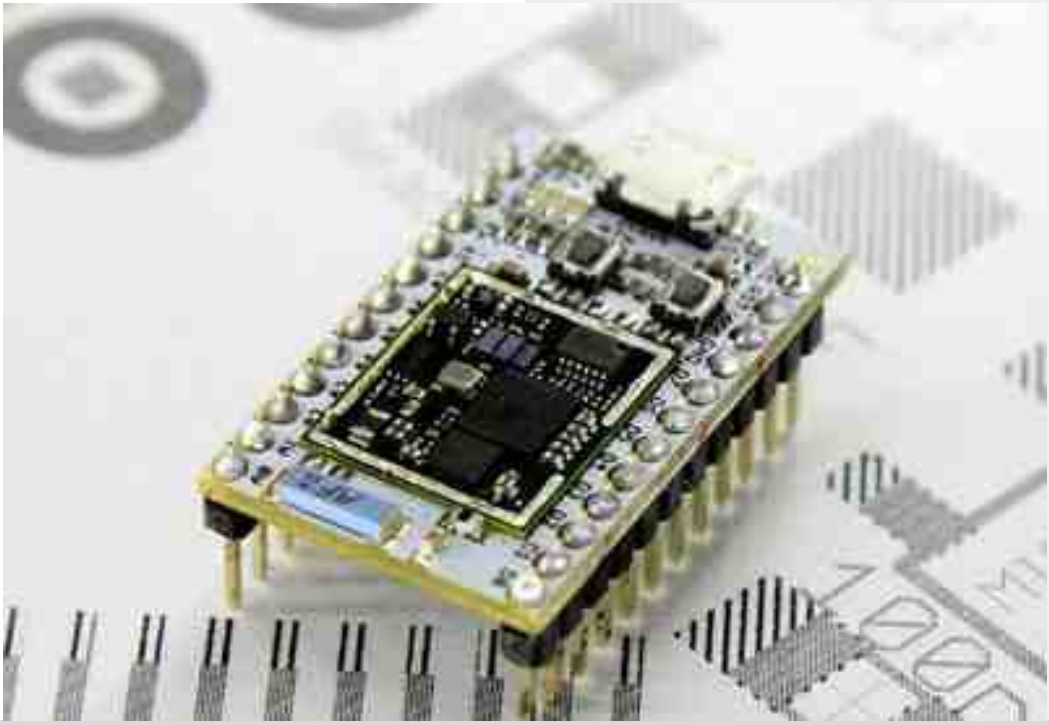
Ono što nas sprečava jeste upravo cena procesora. Cena na kojoj se kreću procesori iz pametnih telefona jeste jednostavno neprihvatljiva za korišćenje u nekim kućnim uređajima iz tog razloga što bi se cena tih uređaja morala podići na višestruko veći iznos u odnosu na onaj na kojem se trenutno nalazi. Sa druge strane, mikrokontroleri nemaju snage da se nose sa svim zadacima koje im možemo postaviti sa ciljem da nam kućni uređaji postanu „pametni". Međutim, šta ako mikrokontrolerima postavimo nešto lakši zadatak?

Tu se za pomoć možemo obratiti jednoj od tehnologija koju koristimo od kada postoji internet: „*Cloud*" komuniciranje. Šta ako naši uređaji ne moraju da spremaju ogromne količine podataka? Šta ako se odluke donose van naših kućnih uređaja? Cena za izgradnju procesora koji treba isključivo da se poveže sa internetom i preuzme malu količinu podataka je mnogo manja nego cena stavljanja punokrvnih procesora u pametne uređaje. Rešenja o tome šta uređaj zapravo treba da uradi se nalaze u „*Cloud*", naš procesor ih samo treba preuzeti i na osnovu datih rešenja izvršiti operacije. Čini se kao poprilično jednostavno. Sada kada smo vas uveli u ovo jednostavnije rešenje i objasnili kako pametni uređaji rade, vreme je da pokažemo povezanost između ovog rešenja i slobodnog softvera.

Povezanost između ovog pristupa i slobodnog softvera jeste u tome što različiti uređaji ne moraju da pokreću potpuno različite operativne sisteme. Svi uređaji bi mogli da koriste vrlo slične operativne sisteme, samo programirane na način da se upravlja drugačijim delovima uređaja da bi se izvršila hardverska operacija za koju je taj uređaj namenjen. Sam proces prilagođavanja jednostavnog operativnog sistema za tačno određenu upotrebu jeste daleko lakši kada pred sobom imamo slobodno softversko rešenje nego kada pred sobom imamo vlasnički softver. Možemo da proizvedemo mnogo više pametnih uređaja u kraćem vremenu nego što bi to izvela neka kompanija, i vrlo brzo bi svi uređaji u našem domu mogli biti povezani na centralni server u našem domu na kojem bi se smeštali podaci i donosile odluke. Podaci o nama nikada ne bi trebalo da napuste naš dom, proces nabavke novih uređaja i njihovo povezivanje na našu mrežu bi bilo trivijalno, a uređaji bi imali interakciju sa nama, onakvu kakvu smo mogli samo

da zamislimo pre nekoliko godina, a moramo priznati da i sada budi neki osećaj dalekog futurizma u nama.

Međutim, taj futurizam je već ostvarljiv za osobe koje imaju nešto više novca, želje da svoje snove pretvore u stvarnost i znanja da programiraju kako bi ostvarili to sve. Internet stvari („The Internet of Things") je na putu i fanovi slobodnog softvera imaju nekoliko rešenja dostupnih već sada. Međutim, ta rešenja još uvek nisu dostigla nivo da su laka za korišćenje. Svako ko pokuša da implementira pametne osobine u dostupnim uređajima će morati dobro da se potrudi da bi to ostvario, a mi ćemo vam u sledećim brojevima predstaviti nekoliko projekata koji se fokusiraju na internet stvari koristeći prvenstveno slobodne komponente, te predstaviti opasnosti koje nam u budućnosti donosi internet. U sledećem broju vas očekuje članak o *Spark* projektu.



Conky Manager

Autor: Miloš Miladinović

O *Conkyju*

Conky je programska aplikacija koja vam omogućava da pratite softversko i hardversko stanje vašeg računara, pored drugih opcija u koje spadaju provera pošte, *RSS* vesti, vremenske prognoze i mnogih servisa čija primarna svrha ne mora biti usko povezana sa *Conkyjem*. Generalno gledano, *Conky* olakšava praćenje procesa za koje bi nam inače bilo potrebno nekoliko aplikacija. U svega nekoliko redova možete imati časovnik, datum, aktivne procese i one koji najviše crpe resurse sistema, temperaturu računara, broj nepročitanih imejlova i vremensku prognozu. Ne može se reći da je planirano, ali *Conky* od početka teži da ostane lagan i jednostavan, sa postavkama koje staju u dvadesetak redova (ili mnogo manje) tekstualne datoteke. Ovakav način obično (skoro obavezno) početnicima predstavlja problem, naročito ako dolaze sa *Windowsa*, ali se lako prevazilazi u gotovo samo jednom danu

i uz malo strpljenja. Naravno, za one koji nemaju strpljenja da kuckanjem podešavaju svoju aplikaciju, tu je *Conky Manager* koji ima posebnu moć da tekstualne redove pretvori u grafičko sučelje koje čak i apsolutnim početnicima omogućava prepoznatljiv doživljaj podešavanja. Zanimljivo je da je *Conky* dobio ime po liku iz TV serije *Trailer Park Boys* koja se snima od 2001. godine. Serija je inače doživela 8 sezona, dok se od 5. marta 2014. snima i 9. sezona ove serije.

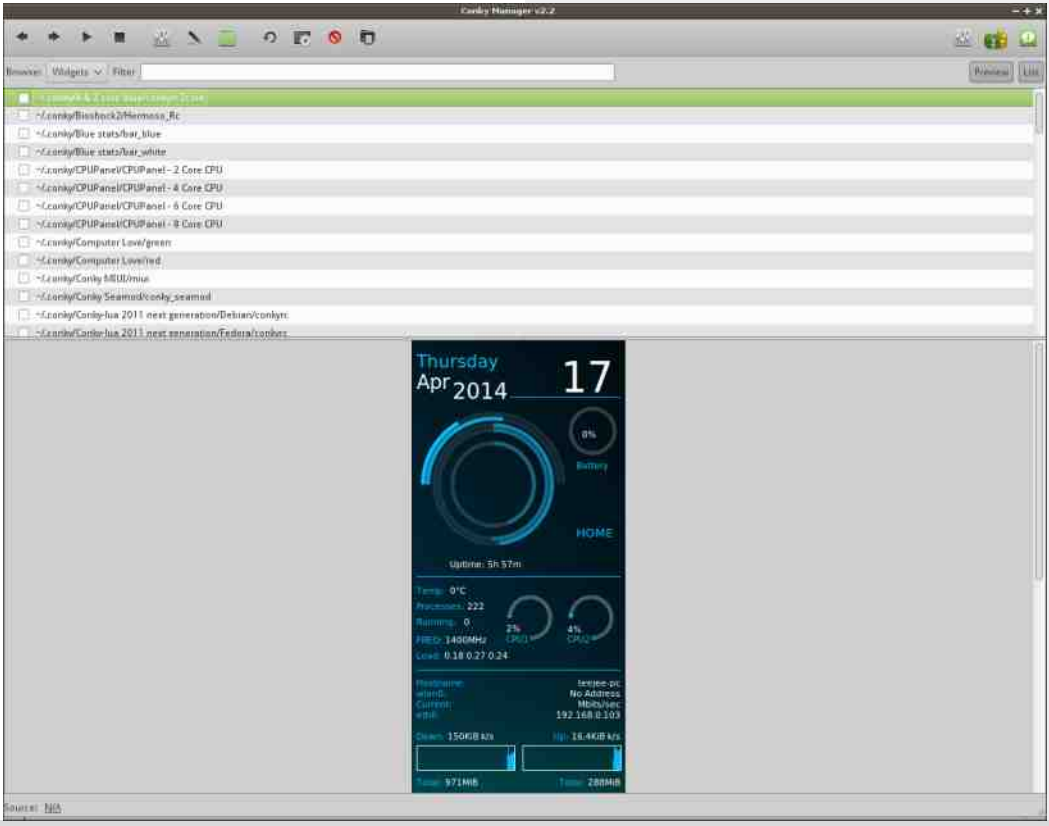
Conky Manager

Neke od prednosti *Conky Managera* su:

- Pokretanje/zaustavljanje, pretraga i podešavanje *Conky* tema;
- Pokretanje *Conkyja* sa sistemom;
- Opcije za menjanje lokacije, transparentnosti i veličine *Conkyjevog* prozora;
- Opcije za menjanje vremena i izbora mreže (*wlan0/eth0*).

Instalacija

Instalaciju je moguće izvršiti na *Ubuntu* baziranim distribucijama (*Ubuntu*,



Linux Mint, itd). Ukoliko koristite *Ubuntu* ili neki od njegovih derivata (*Xubuntu*, *Lubuntu*, *Kubuntu*, itd), instalaciju možete izvršiti putem putem *Lanchpad PPA* na sledeća *Ubuntu* izdanja:

- *13.10 (saucy)*
- *14.04 (trusty)*
- *14.10 (utopic)*

Za neka od predhodnih izdanja možete koristiti *DEB* datoteke:

conky-manager-latest-i386.deb (32-bit, 1 MB)conky-manager-latest-amd64.deb (64-bit, 1 MB)

Da bi ste instalirali *Conky* koristeći *PPA*, u terminalu otkucajte sledeće komande, jednu po jednu:

```
sudo apt-add-repository -y
ppa:teejee2008/ppa
sudo apt-get update
sudo apt-get install conky-
manager
```

Podešavanje

Prozor *Conky Managera* dizajniran je tako da vrlo lako možete postaviti i podesiti omiljeni *Conky*. Ono što ćete prvo primetiti prilikom otvaranja prozora je lista osnovne ponude *Conkyja*. Iznad liste su vam ponuđeni alati za podešavanje *Conkyja*. Od alata, u ponudi su sledeće opcije:

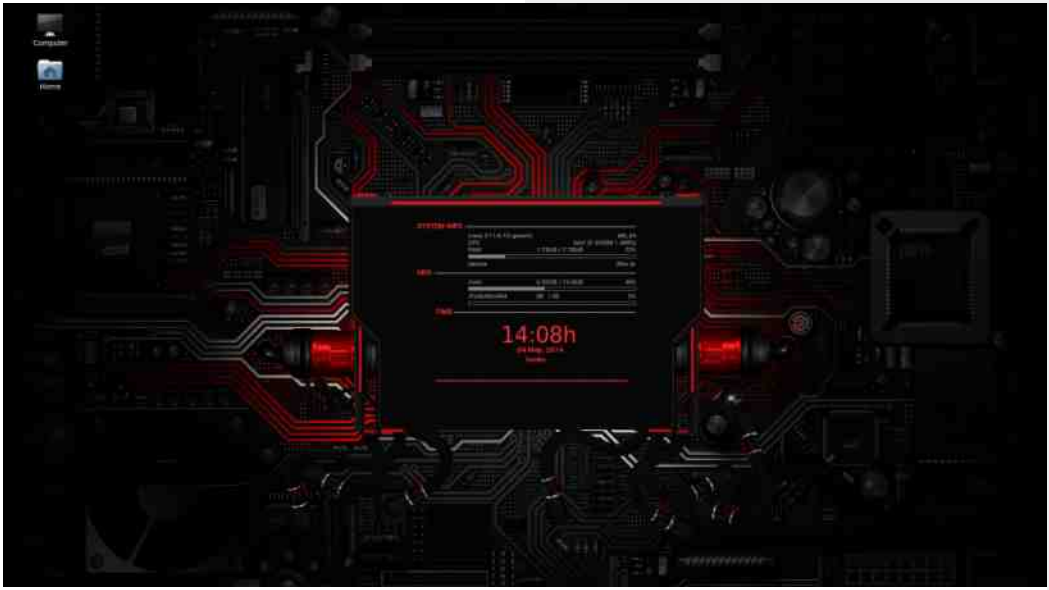
- Strelice - služe za pregled vidžeta;
- Paljenje/resetovanje vidžeta;
- Zaustavljanje vidžeta;
- Podešavanje;
- Podešavanje *Conkyja* pomoću

tekstualnog editora;
- Otvaranje fascikle u kojoj se nalaze teme;
- Pretraga novih tema;
- Pravljenje pregleda tema;
- Kompletno isključivanje svih pokrenutih vidžeta;
- Dodavanje novih tema.

U osnovnoj instalaciji, izbor *Conkyja* i tema je mali, ali sa ovog linka možete preuzeti još odličnih tema: http://www.teejeetech.in/2014/06/conky-manager-v2-themes.htm

Zaključak

Kao što smo naveli u uvodu, *Conky Manager* je napravljen pre svega za početnike, za one koji ne žele da se preterano bave podešavanjem u samom kôdu. U svakom slučaju, *Conky Manager* će vas uputiti, olakšati rad sa samim *Conkyjem*, ali će vam i olakšati razumevanje načina funkcionisanja *Conkyja*. Svakako zaslužuje našu preporuku.



libGDX „Java game development framework" (4. deo)

Autor: Gavrilo Prodanovic

U prošlim brojevima smo govorili o libgdx-u u kontekstu grafike i ulaza, u ovom broju ćemo reći nešto više o ostalim pomoćnim klasama koje se ne odnose direktno na ovo dvoje, a tu su da ubrzaju i pomognu razvoj igrice i dotjerivanju gameplay-a. Započećemo sa zvukom ćime smo zaokružili jednu cjelinu koja je potrebna za "kompletnu" igricu(grafika, ulaz i zvuk).

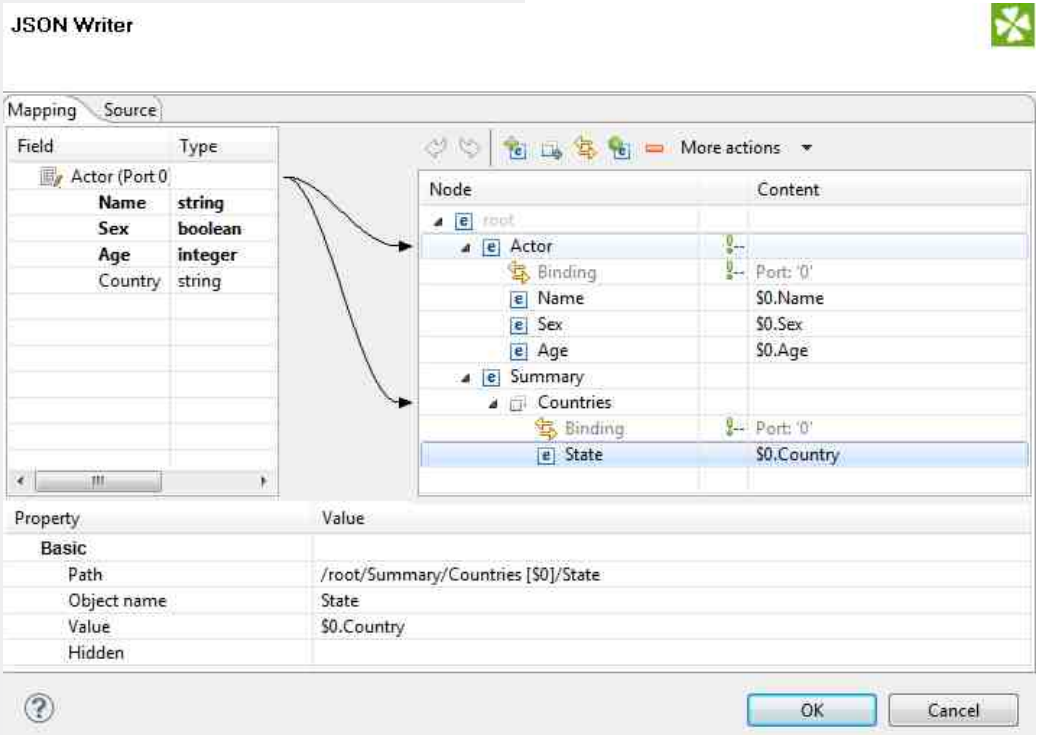
Od formata za audio LibGDX podržava OGG, MP3 i WAV, a za manipulaciju sa ovim klasama postoje klase Sound i Music. Osnovna razlika ove dvije klase u primjeni je ta da prvu koristimo za reprodukciju zvučnih efekata, a drugu za *streaming* neke pjesme u pozadini. Tehnički gledano Sound klasa dekoduje fajl i dekodirani stream učitava u ram, što omogućuje puštanje efekta u tačno potrebnom trenutku. Iz jedne Sound instance moguće stimulativno reprodukovati isti efekat više puta, a svaka nova instanca dobija svoj ID koji vraća play metoda. Svakoj novoj instanci možemo posebno da podesimo volumen, loop, pan i pitch. Na android platformi Sound instanca ne može da bude veća od 1MB, dok na iOS-u nije podržan OGG format. Ako želimo da

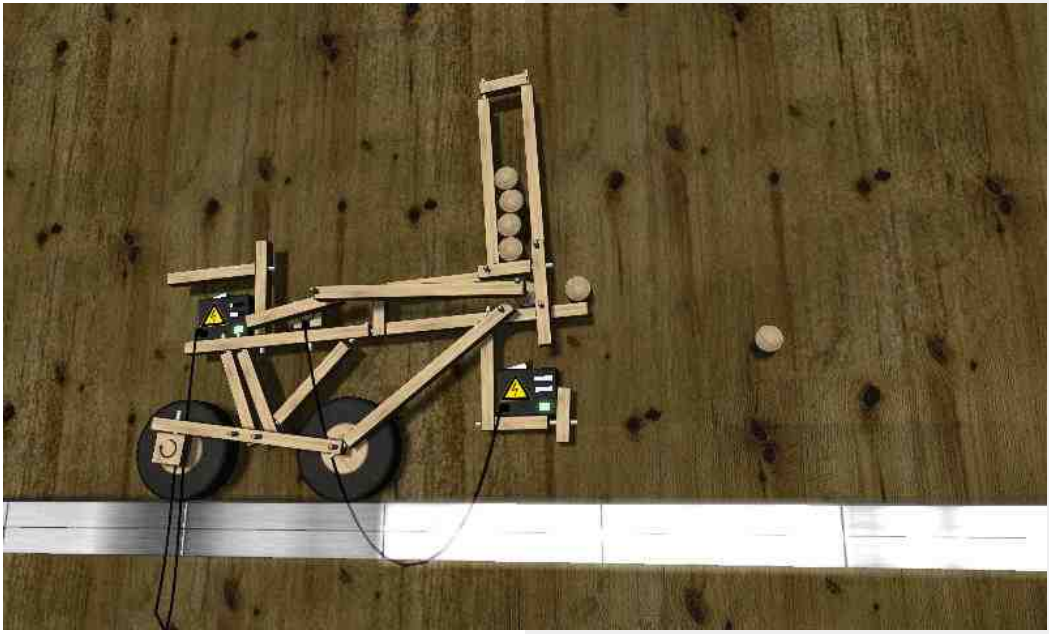
reprodukujemo zvuk koji je duži od nekoliko sekundi, kao što je muzička pranja, korišćenje Music klase je mnogo bolji izbor jer se fajl streamuje sa diska po potrebi umjesto da se učita čitav u ram. Iz jedne instance Music objekta nije moguće pustiti više uzastopnih numera kao u Sound klase. U toku reprodukcije možemo da promjenimo volumen i pan ili poziciju u sekundama ili da podesimo da playback ide u krug. Takođe moguće implementirati listener koji će da "okine" kada se playback završi. Jedna od veoma lijepih opcija po pitanju audija je mogućnost direktnog pristupa hardveru pomoću klase AudioDevice preko čijih metoda možemo da pošaljemo direktno PCM "uzorak" u bafer. Ako želimo da dobijemo ulaz sa mikrofona u tome će nam pomoći AudioRecorder. AudioDevice i AudioRecorder nisu dostupni na JavaScript /WebGL backendu. Za sve klase koje smo ovde naveli potrebno je pozvati dispose() kada postanu nepotrebne da bi se oslobodili resursi.

JSON i XML su često korišćeni za organizaciju podataka, pa među mnogobrojnim klasama nalaze se i klase za čitanje i pisanje ovih formata. Za JSON možemo da se poslužimo sa sledećim klasama: JsonWriter, JsonReader, JsonValue i Json. Funkcija klasa JsonWriter i JsonReader je očigledna; JsonValue opisuje Json objekat, a Json klasa će nam pomoći da neki java objekat serijalizujemo u Json objekat ili da iz Json objekta da deseralizujemo u java

objekat. Zahvaljujući automatskoj serijalizaciji u LibGDX je bez mnogo muke moguće sačuvati ili učitati igru iz fajla ako je potrebno. U slučaju da se javi potreba da kontrolišete serijalizaciju ili deseralizaciju postoje interfejsi koji će vam u tome pomoći da li na nivou svoje java klase tako što ćete implementirati Json.Serializable ili na nivou čitavog procesa serijalizacije tako što ćete u Json objektu postavi serijalizer preko setSerializer() metode. Za one kojima je potreban XML postoji XmlReader i XmlWriter. Ako mislite da je to neki od poznatih java XML paresera implementiran u LibGDX prevarili ste se, po riječima autora on je napisao namjenski XML parser za LibGDX jer je imao predosjećaj da je smislio nešto malo i brzo i htjeo je da testira svoje vještine u Ragelu. U testovima libGDXov Xml parser se pokazao mnogo bolji od *javax.xml.parsers.DocumentBuilder* u fajlovima od 1MB na desktopu i androidu. Dok na fajlu od 100MB javax parser je dobio trku za nešto manje od jedne sekunde, android je bio isključen u testu na velikim fajlovima. Nećemo zalaziti u korišćenje parsera, pošto je XML nešto komplikovaniji od JSON-a.

Matematika u igricama ne može biti zapostavljena, a za čistu matematiku LibGDX obezbjedio paket com.badlogic.gdx.math u kojem postoje raznovrsne klase da nam pomognu. Prvo





ćemo spomenuti MathUtils u kojoj postoje statičke metode za uobičajene matematičke operacije kao što je zaokruživanje, izračunavanja sinusa i kosinusa, konvertovanje stepena u radijane. Spoemnućemo da postoji nekoliko zgodnih metoda za generisanje random brojeva. Postoje klase koje definišu osnovne geometrijske oblike u ravni kao što su krug, kvadrat, elipsa i klasa koja će nam pomoći da odredimo da li postoji presjek između njih. Postoje klase i za interpolaciju (poznato i kao *tweening*) koje nam mogu pomoći u sastavljanju animacije. Spomenućemo da su podržani vektori, matrice i kvaternioni. Takođe je implementiran *Ear-Clipping* algoritam i još neki, ali ne planiramo zalaziti u dubinu matematičkih sposobnosti LibGDX-a i ovde ćemo završiti.

U platformskom ili RPG žanru potrebne su mape koje mogu biti algoritamski generisane ili ručno izrađene za vrijeme razvoja igrice. U našem frameworku postoji čitav jedan skup klasa za rad sa mapama. Mapa je skup slojeva(eng. layer), a svaki sloj posjeduje svoje objekte. Jedan od tipova su *Tile mape* koje su sastavljene od pločica ili ćelija i karakteristične su za RPG, platformske i slične igre. Tile mape su jedini kompletno implementirani tip mapa, ali urađene su osnovne apstraktne klase koje definišu uopsteno mapu, sloj i objekat tako da bi implementiranje bilo kojeg drugog tipa 2D mape bilo jednostavno. Za tile mape glavna klasa je TiledMap u kojoj se nalaze instance TiledMapTileLayer klase. U layerima se nalaze ćelije koje posjeduju referencu na *TiledMapTile*, a tile-ovi su obično djeljeni među ćelijama. Tile može biti

statična tekstura ili animacija. Za renderovanje ovih mapa postoji više rendera. Za rednerovanje ortogonalnih mapa koristićemo *OrthogonalTiledMapRenderer*. Renderu se u konstruktoru prosleđuje mapa kako bi se izvršila optimizacija i keširanje mape. Za izometrične mape postoji *IsometricTiledMapRenderer* čije je korišćenje analogno ortogonalnom renderu. Postoje još 4 rendera ali su oni eksperimentalni i nećemo ih spominjati. Na kraju što je najvažnije među *tile* mapama jeste formati koji su podržani za učitavanje. Postoje dva editora, odnosno formata koja su podržana. Prvi je open source editor Tiled sa svojim TMX(Tile Map XML) formatom, a drugi je format editora zatvorenog koda Tide.

LibGDX nam je obezbjedio *scene2d* da logiku svoje igre organizujemo unutar grafa koji nam pomaže da stvorimo hijerarhiju između naših "glumaca" (eng. actor). Ono što nam scene2d omogućuje su mnoge olakšice kao što je upravljanje grupama, na primjer rotacija ili translacija koju primjenimo na roditelja odraziće se i na potomke. Svaki potomak posjeduje svoj sopstveni kordinatni sistem koji je relativan u odnosu na roditelja. Kroz organizaciju nam dolazi lakša mogućnost za iscrtavanje objekata kroz SpriteBatch. A mogućnost upravljanje ulaza takođe postoji što nam daje mogućnost da roditelj uhvati događaj prije svojih potomaka ili poslije njih. Mogućnost da se actorima zadaju akcije kao što je kretanje do određene pozicijje ili da pređu određenu dužinu u nekom smjeru relativno na svoju poziciju, transformacija i mnoge druge je možda glavna prednost scene2d-a da se koristi u gameplay-u. Takođe postoji i paket scene2d.ui koji posjeduje klase potrebne za građenje menija ili HUDa što može mnogo da nam uštedi vrijeme.

Za kraj spomenućemo dva engine-a za fiziku koje naš framework podržava. Prvi engine je Box2D namjenjen kao što mu ime govori za 2D fiziku. Od verzije 1.0 je izdvojen kao ekstenzija dok je prije bio uključen u source kod. Box2D je C++ engine dok je u LibGDX implementiran preko lakog java *wrappera*. Nećemo zalaziti u dubine, zato što je Box2D čitava priča za sebe, ali skoro svako iskustvno koje posjedujete u C++ jeziku po pitanju ovog engine može se prevesti na javu. Po pitanju performansi Box2D se pokazao i više nego solidan koliko smo mi imali ličnog iskustva u njemu. Pored Box2D postoji i podrška za Bullet Engine koji je namjenjen za 3D fiziku. On je pisan u C++-u a za LibGDX postoji Java wrapper.

Uvod u programski jezik *C*

(5. deo)

Autor: Veljko Simić

U prošlom broju, pričali smo o nizovima i matricama, i sada imamo jedno pitanje: Šta raditi ukoliko želimo da ispišemo pet različitih nizova? Kod bi izgledao ovako:

```
for (int i=0; i<n; i++)
    printf ("%d  ",niz[i]);
```

Taj kod bismo iskucali pet puta. Mana ovoga je to što kod postaje znatno nepregledniji. Ako bismo hteli na pet mesta u kodu da ispišemo pet nizova, imali bismo 50 linija koda i prilično bi bilo lako da se izgubimo u tom kodu. Ovo je jedan prost primer gde bismo mogli da primenimo funkcije. Funkcija koja ispisuje niz izgledala bi ovako:

```
void ispisiNiz (int niz[], int
n){
    for (int i=0; i<n; i++)
        printf ("%d  ",niz[i]);
}
```

Objasnimo sada red po red ove funkcije: *void ispisiNiz(int niz], int n)* - Ovo je zaglavlje funkcije, *void* označava tip funkcije. To je tip koji vraća funkcija. Pitate se sada koji je to tip i zašto ga nismo spomenuli kada smo pričali o

tipovima podataka. To je nepostojeći tip, dakle funkcija ne vraća nikakvu vrednost. Posle tipa funkcije sledi njeno ime, pa zagrada u kojoj se navode argumenti funkcije. Funkcija može, a i ne mora da ima argumente. Argumente funkcije navodimo tako što navedemo prvo tip argumenta, pa ime. Posle toga, u vitičastim zagradama se nalazi telo funkcije. Tu se nalazi sve što funkcija treba da odradi. Poslednja naredba koja se izvršava jeste povratna vrednost funkcije npr. *return 0*. Kada napišemo funkciju, sve što je potrebno jeste da je pozovemo u funkciji *main* i prosledimo joj parametre i ona će biti izvršena.

Razlika između parametra i argumenta. Ova dva termina se veoma često mešaju. Parametri funkcije su one vrednosti koje se prosleđuju pri pozivu funkcije, dok su argumenti funkcije oni koji se navode pri definiciji funkcije, npr. kada smo malopre naveli za ispis niza *„niz“* i *„n“*, to su bili argumenti te funkcije, dok su *„“%d”,nizi]“* parametri funkcije *printf*.

Možda niste primetili, ali do sada smo i pisali i koristili funkcije. Funkciju *main* svaki program mora da ima. To je glavna funkcija od koje počinje rad našeg programa. Na kraju te funkcije, pisali smo *return 0;*. Funkcija *main* vraća vrednost 0 ukoliko je sve dobro izvršeno. Koristili smo funkcije *printf* i *scanf*. One se nalaze u standardnoj biblioteci *stdio.h* u kojoj se nalaze ulazno/izlazne funkcije.

Rekurzivne funkcije

Rekurzivne funkcije su one funkcije koje u svom telu pozivaju samu sebe. Rekurzivne funkcije se sastoje iz dva dela: trivijalnog slučaja i rekurzivnog poziva. Pojasnićemo na primeru:

```
int fakt (int n){
    if (n<=1)
        return 1;
    else
        return n*fakt(n-1);
}
```

Trivijalan slučaj ove funkcije je provera da li je *n* (tj. broj za koji izračunavamo faktorijel) manji od jedan, ili je jednak jedan. Tada se rekurzivna funkcija završava. Rekurzivan poziv je u *else* grani. Da biste lakše shvatili kako se rekurzivna funkcija izvršava, funkciju ćemo izmeniti da nam ispiše kada ulazi, a kada izlazi iz funkcije. Tako da ćemo *else* granu napisati drugačije:

```
int fakt(int n){
        if (n<=1){
                printf
("Izvrsava se trivijalan
slucaj");
                return 1;
        }else{
        printf ("Ulaz u %d
funkciju \n",n);
        int a =fakt(n-1);
        printf ("Izlaz iz %d
funkcije \n",n);
        return a*n;
  }
}
```

Kada bismo pozvali ovu funkciju u funkciji *main* sa parametrom 4 i pokrenuli program kao izlaz, dobili bismo ovo:

```
Ulaz u 4 funkciju
Ulaz u 3 funkciju
Ulaz u 2 funkciju
Izvrsava se trivijalan slucaj
Izlaz iz 2 funkcije
Izlaz iz 3 funkcije
Izlaz iz 4 funkcije
```

Kao što vidite, pri svakom pozivu pokrene se nova funkcija. Kao da umotavamo i odmotavamo papir. U opštem slučaju, kada unutar jedne funkcije pozivamo drugu, prva nastavlja da se izvršava tek kada se završi druga funkcija.

U potrazi za idealnom distribucijom:

Početak

Autor: Dejan Maglov

Ovaj serijal, koji smo započeli u prošlom broju časopisa, zamišljen je kao serijal koji kroz naša iskustva pomaže *FLOSS* početnicima da pronađu idealnu distribuciju za sebe. Zbog velike disperzije *FLOSS* operativnih sistema, pitanje svih pitanja za početnike jeste „Koja je idealna distribucija za mene". Idealno ne postoji. Ono što je za jednu osobu idealno nije idealno za nekog drugog. Prema tome, ova naša potraga za idealnim je u stvari „priča koja se nikad ne završava" (*Never-Ending Story*).

Ako ste početnik u *FLOSS*-u ili vas je samo naš časopis „zagolicao" i zainteresovao za *FLOSS*, možda vam sad i nije baš najjasnije šta znači tražiti idealnu distribuciju. Kao prvo, moramo razjasniti šta je „distribucija" u ovom kontekstu. Za početak pretpostavimo da ste se zainteresovali za slobodni softver. Kao prva „raskrsnica" stoji vam odabir *FLOSS* operativnog sistema. U *FLOSS* ponudi vam je *Linux*, *BSD* i nekoliko mnogo manjih projekata. Srazmera popularnosti, među običnim korisnicima, između *Linuxa* i *BSD*-a je otprilike kao i između *Windowsa* i *Linuxa*. Među običnim korisnicima *Linux* je mnogo popularniji od *BSD*-a, pa pretpostavimo da ste se opredelili za neku varijantu *Linuxa*.

Kada kažemo *Linux*, to znači da koristimo neki operativni sistem čiji

kernel je *Linux*. Kernel je „srce" operativnog sistema. Uprošteno, on predstavlja vezu između korisnika i hardvera. Ako ste čitali naš dvadeset i treći broj i članak „*Linux* unatraške", mogli ste da vidite da je i sam operativni sistem složeniji i da je kernel samo jedan njegov deo. *Linux* kernel jeste najvažniji deo operativnog sistema, ali bez aplikacija, servera i menadžera, operativni sistem ne bi postojao kao celina koja obavlja određeni posao za korisnika. *FLOSS* čistunci, stoga, ovaj operativni sistem zovu *GNU/Linux*, gde *GNU* predstavlja slobodne aplikacije, menadžere i servere, a *Linuxu* ostaje zasluga samo za kernel (srce sistema). U nastavku teksta ispoštovaćemo ovu konvenciju kod imenovanja, pa ćemo kompletan operativni sistem zvati *GNU/Linux*, a ako mislimo samo na kernel zvaćemo ga prosto *Linux*.

Spajanjem različitog slobodnog softvera sa *Linux* kernelom može se dobiti ogroman broj kombinacija. Ako tome dodamo različitu „šminku" i podešavanja, broj kombinacija je neograničen. Svako od nas, ako ima dovoljno znanja, može sam da napravi svoju kombinaciju *Linuxa*, slobodnog softvera, „šminke", podešavanja i napravi svoj *GNU/Linux*. Ako svoj sopstveni *GNU/Linux* ponudite i drugima na korišćenje, to će se zvati vašom *Linux* distribucijom jer dalje distribuirate (delite) *Linux* kernel. Zbog toga se varijante *GNU/Linuxa* zovu distribucijama *Linuxa* po njegovom najvažnijem delu – kernelu.

Ogromna većina nas, ne samo početnika već i prilično iskusnih korisnika *GNU/Linuxa*, nema dovoljno znanja da sastavi idealnu distribuciju za sebe iz sastavnih delova, zato koristimo usluge programera koji su to već uradili za nas i ponudili nam svoje proizvode. Ti programeri su iskombinovali idealnu distribuciju za sebe, ali su i nadogradili još neke stvari kako bi zadovoljili i neke šire ukuse. To što je idealno za njih možda nije idealno i za nas. Svaka distribucija je specifična na svoj način i ima akcenat na onom što je bilo bitnije programeru koji ju je kombinovao. Na nama je sad da u moru već gotovih distribucija pronađemo onu koja najbolje radi na našem hardveru, koja ima naglasak na funkcije koje su nama bitne, dodamo šminku i podešavanja po svom ukusu. Sad je, valjda, malo jasnije zašto se svi u *GNU/Linux* svetu pretvaramo u istraživače u poteri za idealnom distribucijom.

Potraga za idealnom distribucijom po kernelu

Svi *GNU/Linuxi* imaju „jednak" *Linux* kernel koji se razvija kontrolisano pod nadzorom njegovog tvorca Linusa Torvaldsa. Kernel nije izmišljotina *GNU/Linuxa*: poseduju ga i svi ostali operativni sistemi (*Windows*, *MacOS*, *BSD*).

Razlika između *Windows* i *Linux* kernela je u transparentnosti tog dela operativnog sistema i u odvojenost *Windows* kernela od upravljačkog softvera za hardver (*driversa*). Za razliku od *Windowsa*, *Linux* ima ugrađene upravljačke programe za većinu poznatog hardvera.

Kod instalacije *OS Windowsa*, korisnik prvo što mora da odradi jeste da instalira sve pripadajuće pokretačke programe da bi hardver proradio. Kod *GNU/Linuxa*, ako je sve prošlo kako treba (odabran je odgovarajući *Linux* kernel), sav standardni hardver treba da radi odmah po instalaciji sistema. Neki specifični hardveri nude dodatne pokretačke programe za *Linux*, ako je to proizvođač predvideo. Osim toga, korisniku je dato da bira da li će da koristi slobodne pokretačke programe za grafičku karticu ili vlasničke pokretačke programe koje nude proizvođači grafičkih kartica (*Nvidia*, *AMD-Radeon*).

Na početku ovog poglavlja, stavili smo pod navodnike reč „jednak”. U principu *Linux* kernel ima jednak princip rada ali, uglavnom zbog različitih upravljačkih programa, postoje više verzija *Linux* kernela u opticaju koji se redovno održavaju. U trenutku pisanja ovog članka, najstariji *Linux* kernel koji se redovno održava je sa oznakom 2.6, a najnoviji stabilni je 3.16. Pored njih sa produženim održavanjem, od strane

kernel.orga, u opticaju su i verzije 3.2, 3.4, 3.10, 3.12, 3.14. Osim ovih verzija koje održava *kernel.org* i same veće distribucije za svoje potrebe održavaju neku međuverziju kernela, kao na primer kernel 3.13 koji će biti održavan od strane *Canonicala* za potrebe *Ubuntu 14.04 LTS* do 2019. godine.

Od kernela zavisi funkcionalnost operativnog sistema, zato treba voditi računa da se izabere kernel koji odgovara hardveru. Mi nećemo sami komponovati *GNU/Linux* sistem, već ćemo birati distribuciju iz gomile već ponuđenih. Kernel je, verovatno, prvi i najvažniji kriterijum za odabir odgovarajuće distribucije.

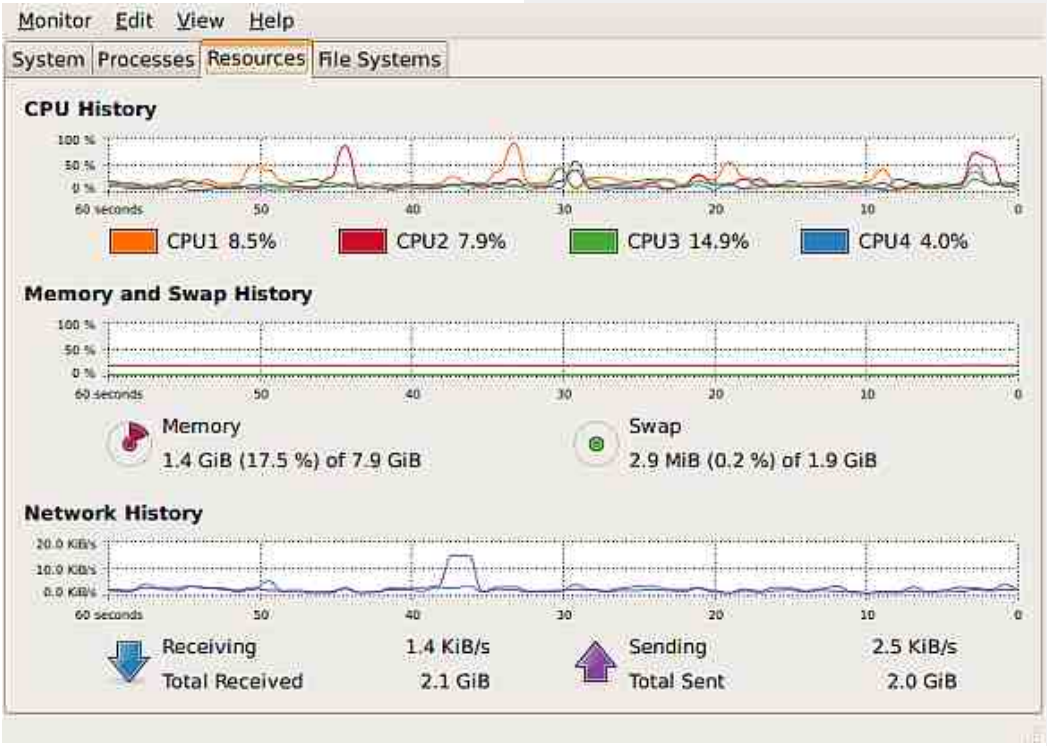
Kako birati? Kod *Linuxa* ne znači da je najnoviji kernel i najbolji. Hardver se menja, a sa tim menjaju se i neki principi rada tako da najnoviji hardver neće raditi sa starijim kernelima jer ne postoje upravljački programi za njega, ali isto tako i stariji hardver često odbija da bude pokrenut najnovijim kernelom. Pouzdano znamo da stari računar *Pentium 2* klase može da pokrene kernel 3.10 i stariji. Noviji kerneli neće raditi na toj staroj mašini.

Problemi oko kernela uglavnom se odnose na ekstremne slučajeve hardvera, kao što su potpuno nov hardver i mnogo stari hardver. Ako posedujete neki od aktuelnih hardvera, što pretpostavlja hardver koji nije „poslednji krik" tehnologije i ne stariji od četiri godine, najverovatnije nećete imati problema sa kernelom. Mada možda, treba izbegavati najnoviji kernel zbog mogućih bagova koji su se „provukli" i koji će tek biti uočeni i otklonjeni ali isto tako i najstariji kernel koji se održava jer je on baš namenjen za zastareli hardver.

Resursi hardvera kao kriterijumi za odabir idealne distribucije

Od kernela zavisi funkcionalnost operativnog sistema. Sama funkcionalnost ne podrazumeva i ugodan rad. Predugo čekanje na odziv sistema na postavljeni zahtev korisnika može da bude iritirajuće. Na „živahnost", odnosno brzi odziv sistema, utiču hardverski resursi. Najvažniji hardverski resursi na koje treba obratiti pažnju, jesu količina *RAM* memorije, brzina procesora, broj jezgara procesora, veličina hard diska, brzina hard diska, kvalitet grafičke kartice i drugi manje uočljivi hardverski resursi za korisnika. Idealno bi bilo imati brz procesor (*2.5 GHz* i više) sa dva jezfra ili više njih, više hard diskova velikog kapaciteta (preko *250GB*) za skladište, *SSD* (eng. *solid-state drive*) – hard disk brzog odziva koji je idealan za operativni sistem i aplikacije, dosta brze sistemske memorije (preko *4GB RAM*-a), kvalitetnu grafičku karticu sa dobrim grafičkim procesorom i što više sopstvene brze grafičke memorije (preko *1 GB*). Običan korisnik računara nema potrebe za tolikom hardverskom snagom. Za udoban svakodnevni kancelarijski rad (rad u *office* paketu programa, internet pretraživaču, muzičkom i video plejeru) dovoljan je i mnogo slabiji hardver.

Hardverski minimum za malo zahtevnije korisnike koji može da „potera"



bilo koju do sada poznatu *Linux* distribuciju je računar *Pentijum 4* klase sa procesorom *2.5GHz* sa 2 jezgra, *4 GB RAM*-a, hard diskom od *250GB*, sa eksternom grafičkom karticom klase *Gforce 8* sa *1 GB* grafičke memorije. Ovakve specifikacije zadovoljava hardver star 4-5 godina.

I mnogo skromniji hardver može uspešno da radi pod *GNU/Linuxom* ali zahteva malo racionalizacije i biranja softvera koji nije toliko zahtevan za hardverskim resursima. Jedan od sporednih delova *GNU/Linuxa* čijim pravilnim izborom može dosta da se uštedi na potrošnji hardverskih resursa je grafičko okruženje.

GNU/Linux operativni sistem može, teoretski, da funkcioniše i potpuno bez grafičkog okruženja. Naravno to podrazumeva i upotrebu aplikacija bez grafičkog okruženja. Pošto se savremeni operativni sistem ne može zamisliti bez grafičkog okruženja i aplikacija (programa) sa grafičkim okruženjem, i *GNU/Linux* ima grafičko okruženje i to ne jedno nego više njih. Neka grafička okruženja su usmerena ka displejima osetljivima na dodir (*Gnome*, *Unity*), neka daju punu grafičku prilagodljivost korisniku uz nešto veću potrošnju hadverskih resursa (*KDE*, *Cinnamon*), zatim kompromisna rešenja koji su balansirani odnos grafičke prilagodljivosti i potrošnje resursa (*Xfce*, *Mate*), laka grafička okruženja (*LXDE*. *Enlightenment*), grafička okruženja koja se zasnivaju samo na menadžerima prozora (*Openbox*, *Fluxbox*) i još mnoga druga. Ovo nabrajanje je upravo išlo od zahtevnijih ka manje zahtevnim grafičkim okruženjima. Ako ste isprobali neku *GNU/Linux* distribuciju i niste zadovoljni brzinom njenog rada, najverovatnije je za to krivac upravo ugrađeno grafičko okruženje. Neka od ovih okruženja troše previše *RAM*-a, neka suviše opterećuju *GPU* (grafičku procesorsku jedinicu – grafički procesor na grafičkoj kartici) i/ili *CPU* (centralnu procesorsku jedinicu - procesor računara). Jedno od rešenja

problema sa prezahtevnim grafičkim okruženja je instalacija istog *GNU/Linuxa* sa manje zahtevnim grafičkim okruženjem. Na primer, umesto *Ubuntua* sa *Unity* grafičkim okruženjem *Xubuntu* sa *Xfce* grafičkim okruženjem. Sam hardver diktira idealno grafičko okruženje za taj sistem. Ako ste korisnik dobrog hardvera koji uspešno može da koristi bilo koje grafičko okruženje, na izbor idealnog okruženja utiču drugi faktori. Napomenuli bismo da grafičko okruženje sa sobom povlači i specifične aplikacije koje su potpuno prilagođene tom grafičkom okruženju. Ponekad su te aplikacije samo sa promenjenim imenom. Primer: upravljač datotekama *Nautilus* (kao *File Explorer* u *Windowsu*) se u *Cinnamonu* zove *Nemo* a u *Mate* – *Caja*. U većini drugih okruženja upravljači datotekama su potpuno drugačije aplikacija, na primer u *KDE* upravljačn datotekam je *Dolphin*, u *Xfce* – *Thunar* a u *LXDE* – *PCManFM*. Iako rade svi jednak posao, razlike nisu samo u „šminki", nego i svaki od upravljača ima neke specifične funkcije. Sve to zbunjuje nove *Linux* korisnike, pogotovu što moraju da se naviknu na promenjena imena aplikacija, pa sad kad promene i *GNU/Linux* radno okruženje i dobiju potpuno nova imena aplikacija to bude prilično zbunjujuće, kao da je u pitanju potpuno novi operativni sistem.

Upravljač datotekama je samo jedan od primera različitih aplikacija u različitim radnim okruženjima. Moguće je instalirati aplikacije i iz drugog grafičkog okruženja, ali to povlači dosta zavisnosti originalnog grafičkog okruženja što može da optereti resurse hardvera naročito ako imate malo prostora na hard disku. Osim opterećivanja skladišnog prostora na hard disku, instalacijom zavisnosti vezanih za originalno grafičko okruženje, moguće je povući i neke funkcije koje opterećuju resuse *RAM*-a i *CPU*-a što je korisnik u startu hteo da izbegne instaliranjem lakšeg okruženja. Zato treba izbegavati mešanje podrazumevanih aplikacija iz različitih radnih okruženja ako imate problem ograničenih resursa.

Izbor idealne distribucije preko izbora paket menadžera

Do sada smo birali idealnu distribuciju uslovljeni raspoloživim hardverom. Postoji još jedan kriterij po kojem možemo da biramo idealnu distribuciju. To je izbor prema paket menadžeru.

GNU/Linux distribucije i aplikacije se isporučuju korisnicima putem interneta u obliku paketa izvornog koda ili putem paketa već kompajliranog binarnog koda. Za razliku od *Windows* instalacionih programa, *GNU/Linux* paketi nisu monolitni paketi koji sadrže sve funkcije tog programa. Vrlo često pojedine funkcije *GNU/Linux* aplikacija su u posebnim paketima, a i izgled aplikacije je u posebnom paketu. Tako se obezbeđuje da pojedine funkcije mogu da koriste neke druge aplikacije, a i izgled se prilagođava izabranom grafičkom okruženju. Sve to stvara problem za ručnu instalaciju novih programa na *GNU/Linuxu*.

Zato postoje paket menadžeri, čija je uloga da prepoznaju zahtev korisnika za instalacijom tačno određene aplikacije, pronalazak njegovih paketa u internet riznicama distribucije, prepoznavanje potrebnih međuzavisnosti sa paketima koji nisu direktno vezani za aplikaciju ali su neophodni aplikaciji (npr. izgled aplikacije u grafičkom okruženju – eng. *GUI*), prepoznavanje koji su od međuzavisnih paketa već instalirani i koje tek treba instalirati, raspakivanje, kompajliranje (ako paketi nisu binarni), instalacija, uklanjanje nepotrebnog softvera i međuzavisnosti koje ne koristi neki drugi instalirani program i na kraju, „apdejt" (eng. update) instaliranog softvera.

Ako od kernela zavisi uopšte funkcionisanje operativnog sistema, od grafičkog okruženja zavisi udobnost rada, onda od paket menadžera zavisi jednostavnost održavanja sistema. Stoga je pravi izbor paket menadžera jedan od bitnih kriterijuma za izbor *OS*-a.

Danas su napoznatija tri formata *GNU/Linux* paketa: *DEB* karakterističan za *Debian* i njegove derivate, *RPM* karakterističan za *Red Hat* i njegove derivate i obični *TXZ*, *TGZ* paketi koji mogu biti paketi izvornog koda, ali i paketi predkompajliranog koda. Sa ovim formatima se „bore": *DEB – dpgk* i *APT*, *RPM – yum*, *TXZ – slackpkg*, *TAR – pacman*. Ovo je samo nekoliko najpoznatijih paket menadžera. Treba naglasiti da nemaju svi ovi menadžeri sve ranije nabrojane funkcije. Neki se dobro „bore" sa međuzavisnostima dok drugi ni ne pokušavaju da ih rešavaju. Zatim, neki od ovih menadžera imaju i svoje aplikacije sa grafičkim interfejsom, kao na primer *Synaptic* koji je grafička aplikacija za *APT* ili *Pamac* – grafička aplikacija za *pacman*. Ove grafičke aplikacije mnogo pomažu, naročito početnicima da lako nađu, instaliraju i održavaju svoj softver. Iskusniji korisnici će ponekad rado žrtvovati jednostavnost paket menadžera zarad neke povećane sigurnosti, preciznosti i stabilnosti softvera određene distribucije.

Za kraj epizode

Tek smo zagrebali po površini problema kako izabrati idealni *GNU/Linux* operativni sistem. Pomenuli smo tri najvažnija kriterijuma, a tome bismo mogli dodati i izbor prema načinu instalacije sistema (grafički ili tekst inastaler), pa izbor prema vrsti instalacionog diska (prosti instalacioni disk ili živi disk), pa prema načinu nadogradnje sistema (sistemi sa periodičnim stabilnim verzijama, sistemi sa *rolling updateom*...) i još mnogo drugih kriterijuma. naša potraga će se nastaviti. Verovatno ćemo u narednim epizodama ovog serijala pomenuti i te druge bitne kriterijume za izbor idealne distribucije.

Enkriptovana elektronska pošta

(3. deo)

Autor: Petar Simović

Ova tema postaje sve popularnija, pa se tako sada ujedinjuju *Lavabit* i *Silent Circle* i formiraju zajedničku alijansu pod imenom *Darkmail* (https://www.darkmail.info/). Pojavljuju se i hardveri koji će celu enkripciju raditi automatski za vas (https://kinko.me/). Tu je i budući *webmail* klijent *Mailpile* (https://www.mailpile.is/) koji je još u alfa fazi testiranja, ali ga je moguće skinuti kod sa *githuba* i isprobati. Korisnicima koji su navikli da poštu proveravaju iz internet pretraživača, najverovatnije bi najviše odgovarao neki program u vidu dodatka pretraživaču koji je u isto vreme i menadžer njihovih ključeva. Dva takva su sigurno *WebPG* (https://webpg.org/) i *Mailvelope* (https://www.mailvelope.com/), veoma su intuitivni i pogodni za početnike (pored *Thunderbirda* i *Clawsmaila*). Nažalost, autor ovog teksta ih ne preporučuje jer umanjuju nivo sigurnosti i bezbednosti iz prostog razloga što u tom slučaju morate da se pouzdate da ni sam dodatak pretraživača (ekstenzija/plugin) ni pretraživač neće da vas iznevere i odaju trećoj strani vašu tajnu, tj. da nemaju neki propust ili neku ranjivost koja bi napadaču odala vaš tajni ključ. __

Naravno, ceo proces oko ključeva, šifrovanja i sertifikata moguće je bilo odraditi preko komandi iz terminala, a programi poput *Thunderbirda* i *Claws-maila* su samo interfejs za korisnike koji još nisu napredovali do čina hakera. Tako je, na primer, sledećih šest komandi sve što vam je potrebno da biste već opisani proces iz prethodnog dela odradili iz terminala za vežbu, ako imate još neki imejl nalog viška:

- Generisanje ključeva:

```
gpg --gen-key
```

- Izvoz javnog ključa:

```
gpg --armor --export
your@email.address
```

na ekran ili u datoteku ako na ovu komandu dodate još

```
>>mypubkey.txt
```

- Izvoz privatnog ključa:

```
gpg --armor --export-secret-key
your@email.address
```

na ekran ili u datoteku ako na ovu komandu dodate još

```
>>myprivkey.txt
```

- Generisanje sertifikata za opoziv ključeva („*Revocation certificate*") ukoliko je tajnost privatnog ključa ugrožena ili ste jednostavno izgubili uređaj na kome ste ga čuvali:

```
gpg --output revoke.asc --gen-
revoke  your@email.address
```

- Slanje javnog ključa na server ključeva:

```
gpg --send-keys ID
```

gde je *ID* ustvari *ID* broj vašeg javnog ključa i treba da izgleda slično ovom *ID*-u: *6382285E*.

- Pretraga javnog ključa osobe kojoj želite slati šifrovanu poštu na serveru javnih ključeva:

```
gpg --search-keys
your@email.address
```

Loše strane *PGP*-a

Pre nego što pomenemo još neke zanimljive programe i projekte, objasnićemo čemu služi takozvani „sertifikat za opoziv" ključeva koji smo malopre spomenuli, a u prošlom broju smo savetovali da ga napravite i sačuvate. Naime, ako vas zadesi nesrećan slučaj da izgubite laptop na kome se nalazio vaš privatni ključ, a prethodno ste kopiju istog sačuvali i na nekoj drugoj spoljnoj memoriji zajedno sa ovim sertifikatom i tajnom frazom, onda možete druge upozoriti na moguću ugroženost vašeg tajnog ključa koristeći ovaj sertifikat. Ako napadač (ili lopov haker) dođe do vašeg tajnog ključa, onda može dešifrovati sve vaše prethodne poruke. Upotrebom sertifikata dajete do znanja ostalima koji žele da sa vama razmenjuju privatnu poštu (slanjem opozvanih ključeva na server

javnih ključeva), da su ključevi neupotrebljivi i da nije sigurno da ih koriste. U tom slučaju, morate generisati nove i poslati ih na server (više na http://goo.gl/IWoS4v). Komande su sledeće:

- Generisanje serifikata ukoliko to niste ranije uradili:

```
gpg --gen-revoke KEY_ID
```

- Uvoz sertifikata za opoziv ključeva:

```
gpg --import revoke.asc
```

- Slanje opozvanih ključeva na server - „*update*" ključeva:

```
gpg --send-keys KEY_ID
```

Ovo je samo jedan od, uslovno govoreći, nedostataka sadašnjeg načina funkcionisanja *PGP*-a, tj. jedan tajni ključ otključava sve predhodne tajne šifrovane poruke. Naravno, rešenje postoji upotrebom *PFS* (*Perfect Forward Secrecy*) protokola koji sprečava da se ovakav scenario odigra, jer se svaka nova komunikacija obavlja novim parom ključeva nakon koje se ključevi za tu transakciju odbacuju i više nikada ne koriste. Tako da ma kog ključa sa napadač domogao, ne može dešifrovati njime sve pređašnje komunikacije zato što ključevi među sobom nemaju nikakve veze. Naravno, ovo je mnogo lakše reći nego uraditi zato što postoji velika razlika u arhitekturi tj. kada je *PGP* nastao 1991.godine prošlog veka, nije još postojao *PFS* protokol, a sada se koristi u *„instant messaging"* programima. Ko koristi ovakav protokol, a ko ne možete videti ovde: http://goo.gl/GaiE1V.

Drugi ovakav nedostatak je poznati problem sa distribucijom javnih ključeva i njihova verifikacija. Objasnićemo ovo na primeru. Recimo da na *Twitteru* nađete osobu (novinara) kojoj bi poslali šifrovan mejl sa poverljivim dokazima protiv nekog. Na *Twitteru* je ta osoba ostavila svoj *PGP* otisak (*Fingerprint*) ili *ID* ključa, pa je logično da poseduje i par ključeva, a javni ključ se najverovatnije nalazi na nekom od servera javnih ključeva i možete ga dobiti komandom iz terminala:

```
gpg --recv-key 6382285E
```

Osim što u nekim verzijama *PGP*-a nema provere na vašem računaru da li se otisak (*fingerprint*) preuzetog javnog ključa poklapa sa onim koji je tražen, ovde morate da „verujete" serveru ključeva da vam prosleđuje pravi ključ. Zvuči pomalo paranoično, zar ne? Razmislite još jednom, osim što svako može startovati svoj server ključeva i uključiti ga u postojeću mrežu, već postojeći serveri nisu i ne moraju biti naročito bezbedni jer ne čuvaju nikakve tajne, te postoji realna mogućnost da ih neko hakuje jer sa većom popularnošću tehnologije dolaze i veća opasnost i veći rizik. Još se nije ovako nešto desilo i ne znamo šta bi sve napadač mogao postići i kakvu štetu naneti, ali je sigurno da ne treba imati potuno poverenje u informacije sa ovih servera i valja ih proveravati. Na stranu sve to, mana je u samoj arhitekturi sistema jer je delom centralizovana, a nije distribuirana na same korisnike, pa korisnici moraju da

veruju serveru koji ne mogu da provere. Naime, serveri javnih ključeva međusobno sinhronizuju svoje baze podataka ključeva i tu su da nam olakšaju pretragu ključeva, ali nikako nisu neizbežni. Tako da se ovaj problem može rešiti tako što se organizuju javna sastajanja (poznatije kao *Key-signing party*) i onda se fizički razmenjuju i potpisuju ključevi sa osobama koje lično znate i na taj način stvarate svoju „mrežu poverenja" (*Web of Trust - WOT*). Iako je ovo malo nepraktično, ima i svojih dobrih strana - upoznavanje novih ljudi i razmena veština.

Da ne bi bilo nikakve zabune, *PGP* pruža **privatnost** i **bezbednost** šifrujući poruke tj. njihov sadržaj, **autentičnost** upotrebom heš (*hash*) funkcija i digitalnog potpisivanja poruka i uopšteno svake vrste datoteka. Ali nikako **anonimnost**, jer se hederi (*headers*) poruke tj. adresa pošiljaoca i primaoca ne šifruju i za stranog posmatrača su dostupni. Tehnički rečeno, *PGP* ne sakriva metapodatke. Tako da ako koristite *Gmail*, *Google* zna da ste juče poslali poruku „tom i tom" korisniku, tačno vreme slanja, IP adresu sa koje ste slali, ali ukoliko ste šifrovali poruku ne znaju sam njen sadržaj. Zanimljivo je pomenuti da su *Yahoo* i *Google* uvideli potencijalnu korist od „enkripcije" svoje pošte, pa su najavili da će dogodine implementirati istu u svoje servise, a *Google* je već postavio i kod svog budućeg softvera slobodnog za preuzimanje sa *Gita* (više na: http://goo.gl/P33PN7, http://goo.gl/quc0cs, http://goo.gl/xIw23o, http://goo.gl/v3Zvoo). Ovo nije nikakva revolucionarna novina koju je „svemoćni" *Google* upravo izmislio, već ga primenjuju neko vreme menje poznate kompanije poput *Openmailboxa* koji ima i zanimljiv jednokratni mejl da ne biste morali sumnjivim sajtovima da dajete pravu adresu. Stvar je u tome što su ovi giganti uvideli priliku da se ubace i na ovo tržište jer postaje sve popularnije, ono bi im obezbedilo poziciju centralih pouzdanih autoriteta. Ako se toga domognu, ne verujemo da će bezbednost biti ništa bolja pre svega zato što bi pod određenim okolnostima mogli da izvode tekozvane *MITM* (*Man-in-the-middle*) napade i zaobiđu dešifrovanje poruka. Više na http://goo.gl/AdlCXa i http://goo.gl/dh1C7e.

Svetla budućnost

Na našu sreću, nije sve baš tako crno i nove, bolje ideje se rađaju skoro svakodnevno, pogotovo kada smo na to prinuđeni i inspirisani *Snowdenovim* objavljivanjem. Iz tih ideja se ponekad i „skuva" neka nova i bolja aplikacija ili program, pa pogledajmo neke. Pre svega, jedna od najnovijih je svakako *Keybase* (https://keybase.io/), to je onlajn program ali takođe se može njime upravljati iz komandne linije odnosno terminala. *Keybase* je na neki način server javnih ključeva, ali umnogome olakšava baratanje ključevima i sertifikatima, njihovu verifikaciju, kao i traženje osobe sa kojom želite da se dopisujete. Ono što ga izdvaja od običnog servera ključeva je povezivanje vaših ključeva sa vašim nalozima na *Redditu*, *Twitteru*, *Hacker news* sajtom, *GitHubom* i vašom *bitcoin* adresom ili vašim sajtom kako bi drugi bili što

sigurniji da ste to vi, i da upotrebom vaših ključeva zapravo pišu vama. Otrkivanje i povezivanje vaših identiteta bi inače bilo problem, ali pošto *PGP* ionako ne pruža anonimnost onda to nema štetnih efekata. Program je još uvek u alfa stepenu razvoja i mali broj je pozvan na testiranje. Mada možete da se pridružite projektu i rezervišete ime na sajtu i bićete tek neki 1600-ti, ali ako ste ozbiljno zainteresovani za ovaj projekat administratori su nekim testerima podarili pozivnice za buduće korisnike koji su ozbiljni i zaista žele da doprinesu programu. Autor ovog testa ima pet pozivnica, tako da će pet najbržih koji pošalju mejl sa naslovom *Keybase* na „*keybase@openmailbox.org*" dobiti mejl sa linkom za registraciju. Mejl mi je neophodan, jer moram uneti mejl osobe koju želim da pozovem.

Tu su i napredniji projekti poput *Ponda* (http://goo.gl/axqNwx), koji nije ni klasični *mail* protokol ni *IM* (*instant messaging*) aplikacija, već jednostavno asinhroni *P2P* program koji je u dopisivanje uspeo da ubaci i *PFS*, a poruke se automatski brišu nakon nedelju dana. Ovo je i dalje u razvoju i nemojte ovaj program još koristiti ni za šta ozbiljno. Veoma sličan projekat je i *Flowingmail* (https://goo.gl/) koji iako nije ostvario željenu sumu kao *start-up* na *Indiegogo* sajtu (http://goo.gl/psWb6m) pre skoro godinu dana, to ga nije obeshrabrilo i nisu odustali od svoje zamisli *P2P* decentalizovanog, sigurnog mejl protokola, gde se adrese računara na koji šaljete računaju po javnom ključu tako da ne mogu biti falsifikovane.

Ono što predhodna dva projekta i *Darksmail* pokušavaju da urade, jeste da nam pored navedenih dobrih osobina *PGP*-a, pruže i onu jednu koja nedostaje - anonimnost. *Darkmail* to postiže *end-to-end* enkripcijom, čime sakriva sve metapodatke i postiže anonimnost iz perspektive stranog posmatrača.

U sledećem delu ćemo se upoznati sa anonimnim *email* sistemima koji su nastali još davnih devedesetih, i posle nekoliko faza razvoja i unapređenja u upotrebi su i danas.

MARY Text To Speech

Autori: Dejan Čugalj i Dejan Maglov

Nedeljno poslepodne nekako je rezervisano za dosadu i ubijanje vremena neobaveznim stvarima kao što je neobavezno „surfovavnje" internetom. Kako može da se završi i kuda može da odvede to dosadno nedeljno poslepodne, nikada se ne zna. Elektronska pošta koja stiže nedeljom ili je spam, ili su obaveštenja na koja ste se pretplatili. Registracijom na razne blogove, *web IT* časopise, pretplatili ste se na potencijalna *inbox* obaveštenja koja pristižu nedeljom. Ako ste pritom programer - *freelancer*, *blog-freelancer*... sigurni smo da se baš u tom istom *inboxu* nedeljom pojavljuju i obaveštenja kako ste baš vi taj koji bi mogao da dobije određen posao. Poslovna obaveštenja su korisna, ali dosadna. Ponekad se i u tom poslovnom sadržaju krije potencijalna „inicijalna kapisla" koja bi mogla da „istisne" članak u LiBRE! časopisu. Jedna od takvih slučajnosti je povod pisanja ovog članka, a usko je vezan za „*multilingual text-to-speech synthesis*", prevedeno, multilingvističko okruženje (eng. *framework*) za implementaciju *TTS* (eng. skraćenica od *Text-to-speech* - tekst u govor) sinteze. Na prvi pogled ništa komplikovano: otkucaš tekst i neki „robot – sintetički glas" (eng. *synthesis*) to isto izgovori. Naime, problem je „malo" veći nego što se čini. Svaki jezik, svaka fonetska razlika jezika, usko je vezana za fino podešavanje algoritma implementacije *TTS*-a u kojoj se jezička sinteza diverzibilnosti razlikuje sama po sebi, kao i svaka fonetska distorzija koja čini isti jedinstvenim samom govornom području (eng. *slang*).

U kojoj meri je ova tehnologija značajna, veoma je diskutabilno, camo oni koji imaju problema sa vidom su ti koji mogu objektivno da kažu koliko je to značajno i korisno, dok ostali ... LiBRE! se ograničio na slobodan softver. Naziv „slobodan softver" ne znači samo slobodno korišćenje nego slobodan razvoj, unapređenje i modifikovanje softvera. *TTS* softver upravo pokazuje jednu od glavnih mana ovakvog razvoja softvera, barem u Srbiji. Softver koji zavisi od

lokalizacije podrazumeva veću angažovanost lokalne zajednice na njenom razvoju, jer ne možemo da očekujemo od stranaca da razvijaju i našu lokalizaciju. Činjenica je da skoro svi projekti slobodnog softvera, kod nas nastaju kao posledica zadovoljavanja ličnih potreba samih programera. U ovakvim uslovima, razvoj *TTS*-a možemo očekivati samo od programera koji su lično zainteresovani za taj softver. To znači da se krug mogućih razvijatelja svodi samo na slabovide programere ili na programere koji imaju nekog svog koji je slabovid, a to je prava retkost u Srbiji. U Srbiji je razvoj slobodnog softvera, za nekog drugog, bez lične satisfakcije misaona imenica. Čak ni humanitarni razlozi nisu dovoljan motiv ili se još niko nije setio, a to u ovom slučaju uskraćuje slobodu čitavoj jednoj grupaciji ljudi – ljudima oštećenog vida i slabovidim.

Zato LiBRE! tim želi malo da „zatalasa", podseti da i humanitarni razlozi mogu da budu motiv za razvoj slobodnog softvera.

Trenutno, jedna od najboljih implementacija *TTS*-a poseduje srpska kompanija „*AlfaNum*" http://www.alfanum.co.rs/, koja naravno, *TTS* usluge i naplaćuje. Kôd su zatvorili i drže ga u svojim „sigurnim rukama". Gore navedeni link to i demonstrira. (Imao sam priliku da vidim forum post upravo jednog od *CEO* „*AlfaNuma*" koji tvrdi da je mnogo ljudi uključeno u samu realizaciju projekta, te da mora da se naplaćuje. Nismo sigurni da li link ka forumu radi, ali probajte: http://www.elitesecurity.org/t38544-SERBIAN-Speech-engines)

Mislimo da bi to moglo da se promeni, jer upravo ovde predstavljamo projekat otvorenog koda, koji bi taj problem u Srbiji mogao da preusmeri na put slobodnog softvera i slobode.

Predstavljamo vam: *MARY TTS – an open-source, multilingual text-to-speech synthesis system written in pure java* http://mary.dfki.de

Onako, na prvi „programerski" pogled (eng. *first view*), dokumentacija je više nego dobra: https://github.com/marytts/marytts/wiki, tako da to u celom nekom budućem projektu ne bi bio problem. Naime, malo da „zakočimo" i da objasnimo šta je ovde stvaran problem. Razvoj tekst-u-govor (eng. *TTS*) sistema je ekvivalentan razvoju sistema gde osoba, koja zna da čita i izgovara pravilno, čita tekst naglas. Iako svi mislimo da je to lako, i da smo u ubeđenju da znamo to da radimo, činjenično stanje je malo drugačije. Drugim rečima, skoro 80% građana ne čita po lingvističkim pravilima i to predstavlja jedan od prvih problema u razvoju *TTS*-a. Tako da bi jedan od prvih modela u implementaciji *TTS*-a za srpski jezik, bio znanje kako nešto pročitati (eng. *knowing how to read*). Bez tog lingvističkog znanja, implicitne radnje algoritma bi po automatizmu prerasle u eksplicitno, a samim tim i složenost istog bi eksponencijalno porasla. Kako bi to sveli na neki minimum, u *TTS*-u postoje pravila koja se dele na:

1. prelom rečenica
2. normalizacija teksta

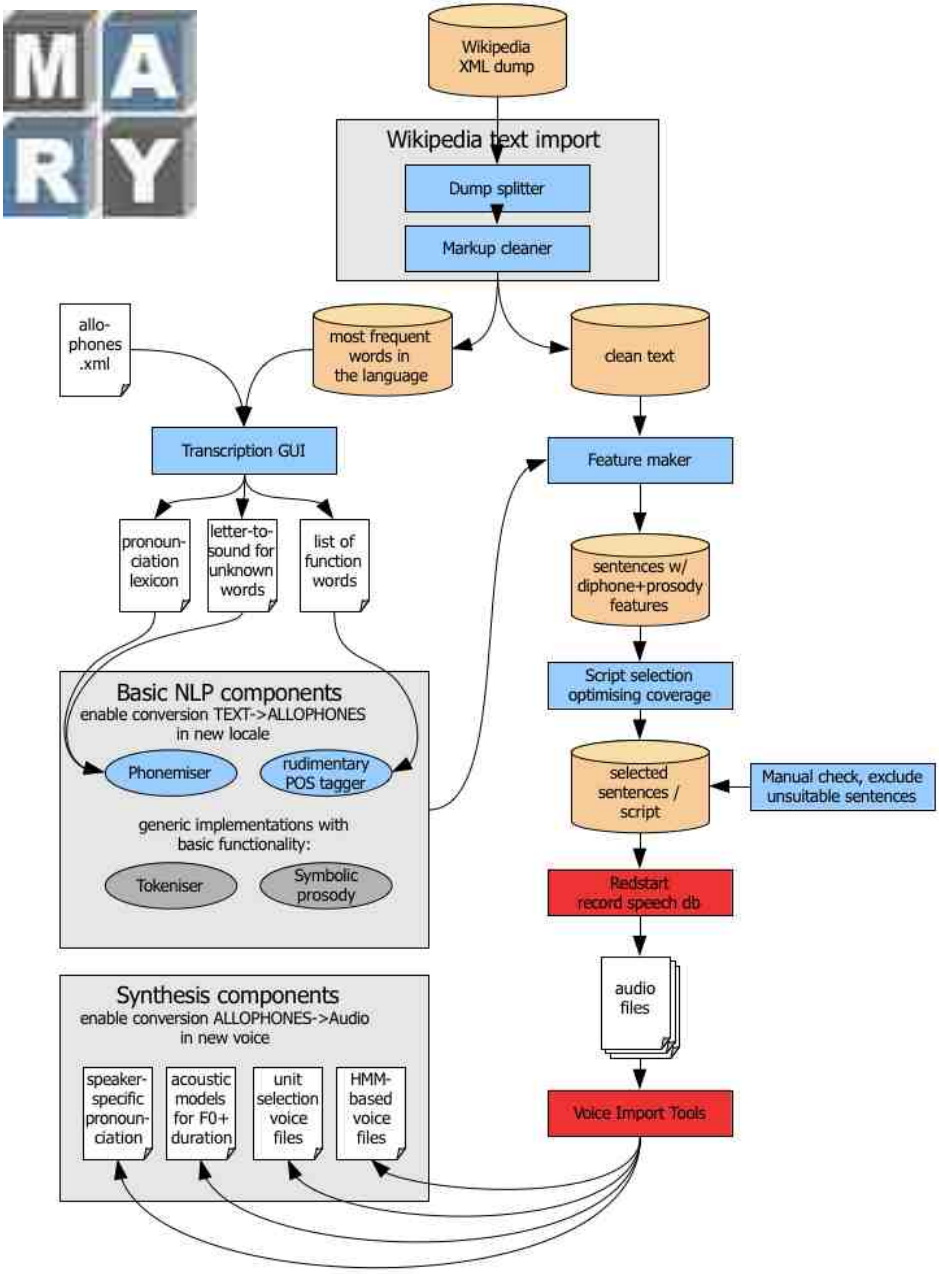
Prelom rečenica u fonetskom kontekstu je jedan od bitnijih delova razvoja TTS-a, jer se nastavak izgovora rečenice veoma razlikuje posle interpunkcijskih znakova, kao što su zarezi, znakovi uzvika, tačke... pa čak ni tu nema pravila, npr. „dr Mirko Mirković", nije isto kada se izgovara u sredini ili na kraju rečenice. Normalizacija teksta je vezana za razdvajanje reči iz rečenica, tj. drugim rečima trebalo bi da postoji baza izgovorenih reči koju koristi TTS sistem za sam izgovor i to u konjukciji sa prelomom rečenica i pravilom interpunkcijiskih znakova srpskog jezika. Samo malu smernicu o kompleksnosti možete da pogledate na linku: http://www.srpskijezik.rs/gramatika/podela-reci-na-slogove.

Samo smo zagrebali vrh ledenog brega, kompleksnost implementacije *TTS* sistema. Pokušaj individualnog implementiranja bi bio ništa manje do programerske velike avanture, ali, srećom, ne moramo da počnemo iz početka, jer projekat „*MARY TTS*", koji je otvorenog koda (eng. *open source*) nam daje ogromnu odskočnu dasku.

Pregledom *API* celog sistema „*MARY TTS*", stiče se utisak da ne bi trebalo da bude teško implementirati podršku za novi jezik. Nikako nećemo reći da nema posla, ali sa sigurnošću možemo da potvrdimo da je moguće, i to besplatno. Nadamo se da smo makar malo zagolicali maštu nekim budućim programerima, kojim bi „*MARY TTS*" projekat bio odskočna daska. Jedan takav projekat motivisan humanitarnim razlozima bio bi značajan za srpski *FLOSS* pokret.

Korisni linkovi:

1. Projekat:
   https://github.com/marytts/marytts
2. Podrška novom jeziku:
   https://github.com/marytts/marytts/wiki/New-Language-Support
3. Ideje:
   https://github.com/marytts/marytts/wiki/Ideas-for-future-work


Wikipedia XML dump
Wikipedia text import
Dump splitter
Markup cleaner
allo-phones .xml
most frequent words in the language
clean text
Transcription GUI
Feature maker
pronoun-ciation lexicon
letter-to-sound for unknown words
list of function words
sentences w/ diphone+prosody features
Basic NLP components
enable conversion TEXT->ALLOPHONES in new locale
Phonemiser
rudimentary POS tagger
Script selection optimising coverage
selected sentences / script
Manual check, exclude unsuitable sentences
generic implementations with basic functionality:
Tokeniser
Symbolic prosody
Redstart record speech db
audio files
Synthesis components
enable conversion ALLOPHONES->Audio in new voice
speaker-specific pronoun-ciation
acoustic models for F0+ duration
unit selection voice files
HMM-based voice files
Voice Import Tools

Android studio

Autor: Stefan Nožinić

Android je već odavno zauzeo ogroman procenat tržišta mobilnih uređaja i mnogo je aplikacija razvijeno za njega. Ako ste se ikada bavili razvojem aplikacija za *Android* uređaje, onda ste najverovatnije za to koristili *Eclipse* i *ADT* dodatak za isti, koji vam je to olakšao. Možemo se zapitati kako je *Google* toliko forsirao *Eclipse*, umesto da je napravio svoje okruženje. Posle toliko vremena, zasebno okruženje je stiglo.

Android Studio u trenutku pisanja ovog članka je još uvek u beta fazi, ali se dosta dobro razvija i već je upotrebljiv. Baziran je na *IntelliJ IDEA*. On pruža nove mogućnosti koje *ADT* dodatak za *Eclipse* nije pružio i postaće zvanično razvojno okruženje čim bude gotova prva stabilna verzija.

Neke od karakteristika su:

1. Sistem izgradnje baziran na *Gradlu* umesto na *antu*.
2. Generisanje više varijanti *APK* arhiva u zavisnosti od uređaja.
3. Urednik grafičkog interfejsa sa podrškom izmene tema.
4. Alati za proveru kompatibilnosti sa različitim verzijama.
5. Alati za merenje performansi.
6. Potpisivanje aplikacija.
7. ...

Migracija na *Android Studio*

Kao što smo već pomenuli, *Android Studio* je baziran na sistemu izgradnje *Gradle*, a ne *ant*. To znači da je potrebno prebaciti naše projekte na novi sistem. *ADT* dodatak za *Eclipse* u verzijama posle 2.2 ima opciju eksportovanja projekta sa novim *Gradle* sistemom, pa je samo potrebno da uradimo nadogradnju dodatka ako već nemamo noviju verziju. Bitno je napomenuti da će *Android Studio* raditi i sa starim *ant* sistemom, ali se preporučuje prelazak na novi sistem kako bismo bili u mogućnosti da koristimo neke dodatne i naprednije opcije u budućnosti.

Upotreba

Android Studio nam omogućava kreiranje aplikacije za različite tipove uređaja. Ovako je moguće pravljenje aplikacija za telefon, tablet, TV, *Google* naočare i *Google Wear*. Čarobnjak za pravljenje novog projekta nudi izbor za koji uređaj želimo da pravimo našu aplikaciju, a time će nas upitati koju verziju *API*-a želimo da koristimo. Pored ovih mogućnosti, u čarobnjaku imamo i opcije da izaberemo određeni *Activity*, i podesimo ga po našoj volji.

Posle kreiranja projekta, potrebno je primetiti da je struktura direktorijuma malo drugačija, nego do sada u *Eclipsu*. Za ovo je „kriv“ *Gradle* sistem izgradnje,

koji se koristi u *Android Studiu*. U principu, sve funkcioniše kao i do sada, samo su neki direktorijumi premešteni u *„src/“*, pa je tako lakše snalaženje. Ovo će puno pojednostaviti rad na projektu, u smislu da programeri neće biti zbunjeni koje datoteke mogu da menjaju, a koji se menjaju/pišu prilikom izgradnje izvršnih datoteka.

Iz samog programa, moguće je direktno kreiranje virtuelnih uređaja, koji služe za pokretanje aplikacije u emulatoru u slučaju da ne želimo da je pokrenemo na fizičkom uređaju. Ovo je korisno u slučajevima kada nemamo određenu verziju *Androida* na fizičkom uređaju, a želimo da testiramo rad naše aplikacije na toj verziji.

Dodavanje novih datoteka je postalo zaista jednostavno. Potrebno je samo kliknuti na određeni direktorijum u projektu, i pritisnuti kombinaciju tastera *ALT + INSERT*, i *Android Studio* će se postarati da pokrene čarobnjaka za kreiranje određenog tipa datoteka. Na primer, ukoliko selektujemo *„layout/“* direktorijum, *Android Studio* će ponuditi kreiranje novog *Activityja*.

Uklanjanje grešaka je moguće pozivanjem *adb logcat* iz samog okruženja, i time dobijamo log poruke koje beleže aplikacije na uređaju ili emulatoru.

Zaključak

Pravo je čudo da *Google* nije ranije uradio ovakvu platformu, s obzirom na to da razvoj *Android* aplikacija postaje sve učestaliji i da su se mnogi problemi *Eclipsa* sa dodatkom počeli nazirati kada je reč o kompleksnijim projektima. *Android Studio* nudi jednostavnost korišćenja, a opet mnogo više mogućnosti nego prethodni podržani pristup razvoja (*Eclipse* sa dodatkom za razvoj *Android* aplikacija).

Raspberry Pi B++ & HummingBoard

Autor: Simović Petar

Raspberry Pi B+

Svi smo do sada dobro upoznati sa *Raspberry Pi* modelima *A* i *B* i verovatno smo kupili neki da se zabavljamo kod kuće, ili da nam služi kao muzički server ili *proxy*, mada spisak njegovih namena ne bi mogao da stane u ovaj članak (http://goo.gl/5UqAkC). Iako je na tržištu već dve godine, neki bi pomislili da je malo zastareo, ali se projekat itekako razvija i prilagođava tržištu i potrebama kupaca. Tako je na primer u aprilu ove godine izašla i nova verzija namenjena poslovnim i industrijskim korisnicima zvana *Raspberry Pi Compute module*. „*Compute module*" je ustvari samo pločica koja se povezuje na *Compute Module IO Board* preko standardnog ulaza za *DDR2 SO-DIMM* memoriju (kao one u laptop računarima). Više na http://goo.gl/SfbKJr.

To nije sve, posle ove transformacije, model *B* je dobio mala unapređenja u vidu *B+* modela na osnovu „*feedbacka*" (eng. povratna informacija) samih korisnika. Najvažnija unapređenja se odnose na povećan broj *USB* 2.0 portova sa dva (*B* model) na četiri (*B+* model), onda ***MicroSD*** podrška umesto

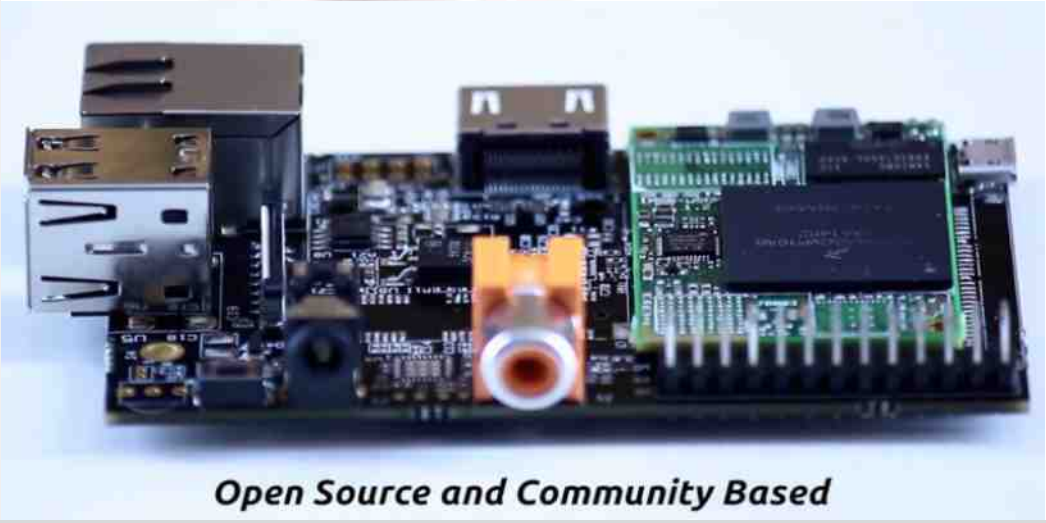
standardne *SD* kartice i povećan je broj pinova *GPIO* (*General Purpose I/O*) sa dvadeset šest (*B* model) na četrdeset (*B+* model), s tim što je dvadeset šest osnovnih pinova zadržano, samo je dodato četrnaest novih sa novim mogućnostima. Između ostalog, smanjena je potrošnja struje sa 3.5 W (*B* model) na 2.5W (*B+* model), ili od 0.5W do 1W zavisno od upotrebe u odnosu na *B* model ili 600 *mA* u odnosu na 750 *mA*. Audio i video konektori sa *B* modela su spojeni u jedan konektor na suprotnoj strani. Ono što se nije promenilo, jeste 10/100 *lan* port, memorija je ostala na 512*MB* i *ARMv6* procesor je ostao isti (700 *MHz* - 1.1 *GHzoverclock*) sa čipsetom *Broadcom BCM2835*, kao i *Micro USB Power supply* i *HDMI* konektori. Važno je napomenuti da se cena nije promenila i da *B+* model košta koliko i *B* model, ali plastične kutijice koje su pravljene za *B* model ne odgovaraju *B+* modelu. Video možete pogledati na http://goo.gl/7u7WWo i http://goo.gl/R21wUG .

Najinteresantnije je da je *Raspberry 2* najavljen za 2017. godinu, a da će se do tada razvoj fokusirati na unapređenje softvera.

HummingBoard

Možda mislite da se *Raspberry Pi* malo razmazio i odomaćio jer već dve godine drži monopol računara male potrošnje veličine kreditnih kartica, ali u trku ulaze i novi igrači sa svojim modelima za takmičenje. Takav je na primer i novi *HummingBoard* kompanije *SolidRun* koji se pojavio na tržište malo pre novog *B+* modela *Raspberry Pi*, pa je *B+* donekle odgovor na *HummingBoard*. Na prvi pogled izgleda isto kao i običan *Raspberry Pi* (*RPi*) *B* model, a to je i bila namera, pošto može da stane u bilo koje kućište namenjeno za *RPi*. Ono što

izdvaja *HB* od *RPia*, jesu jači procesor 1*GHz ARMv7* nasuprot 700 *MHz ARMv6* kod *RPija*, zatim podrška za više operativnih sistema i uslovno veća memorija pošto cene modela idu od 45$ do 100$. Takođe, tu je podrška za *mSATA*, *PCIe mini* i *LVDS display*, a procesor je (mobilan) odvojen na zasebnu pločicu koja se priključuje osnovnoj, tako da se lako može zameniti ili unaprediti u jači model. Zavisno od modela, memorija „ide" od 512*MB* do 1*GB*, procesorska jezgra od 1 do 2, tako da postaju već ozbiljniji mali računari. Detaljnije na http://goo.gl/Z7AgHJ i video na http://goo.gl/nN7W14.



LiBRE!

Časopis o slobodnom softveru

na

BALCCON 2k14