Август/Септембар 2014.

ЛиБРЕ!

Часопис о слободном софтверу

број 28

07. септембар, 2014.

Одржан је други *BalCCon 2k14.*

12. септембар. 2

Турин прелази на *GNU/Linux*

Реч уредника

Повратак енергије

На почетку морамо прво да се извинимо због закашњења овог броја. Комерцијални часописи нити се извињавају за кашњење, нити објашњавају због чега је до њега дошло. Њихови читаоци очекују нови број сваког месеца и на време, и не занимају их разлози кашњења. ЛиБРЕ! има обавезу да се извини и да објасни кашњење, јер то занима чланове заједнице. Чланови наше заједнице нису само наши читаоци него практично - породица. Породица чији појединци нису само пуки читаоци, него се по потреби и жељи укључују у активно креирање овог пројекта. Из тих разлога, добра комуникација са нашим читаоцима од кључне је важности.

Кроз ова наша објашњења разлога кашњења, осликава се не само стање у пројекту, него и у самој заједници, што ако се правилно разуме, може да доведе само до побољшања стања. Овог пута постоји више фактора за кашњење броја. Можда је најважнији фактор пад енергије који је захватио целу заједницу слободног софтвера, и то не само у Србији него и у региону, па самим тим и овај пројекат. Замор материјала старијих (по стажу) чланова редакције, слабија комуникација и дезорганизација млађих чланова редакције, довела је до пада продуктивности. Оно што смо завршавали за петнаест дана, сад нам је за то требало месец дана. До додатног опуштања је дошло јер смо унапред планирали да каснимо седам дана због *BalCCona 2k14* чије се одржавање поклопило са нашим уобичајеним термином објављивања новог броја (прва недеља у месецу). Ово планирано кашњење се поклопило и са септембарским испитним роком, тако да смо са планираног прешли и на непланирано кашњење.

Питање је када ћемо надокнадити оволико кашњење и поново се врати у стандардни ритам месечног изласка часописа. Једно је добро, да се енергија полако вратила у заједницу и пројекат. За то је, пре свега, заслужан *BalCCon 2k14*.

Одлука да занемаримо све обавезе, да као редакција и појединци што масовније учествујемо на овогодишњем *BalCCon*-у, била је добра. Захваљујемо се организаторима што су нам то и омогућили. И поред тога што смо унапред знали да смо као редакција потпуно неспремни да овако велики догађај професионално новинарски сервисирамо, само учешће је било јако корисно. Нико у редакцији ЛиБРЕ! није професионални новинар који би могао да се у тренутку снађе и направи прави интервју са обичним пролазником, а камоли са људима као што су Мич Алтман или Бернд Фикс који знају шта је прави интервју и када професионални новинар прилази неспреман за интервју. Чак и школовани новинари испадају смешни ако нису припремљени за разговор. Редакцији ЛиБРЕ! је додатно било тешко да спреми „новинаре“ јер већина никад није ни била на овако великом догађају и нису могли ни да замисле шта их тамо чека. Зато је ово искуство непроценљиво. Не само да смо заказали као новинарски сервис *BalCCon*-а, него смо пропустили и прилику за бољу промоцију часописа. Имали смо у плану да поставимо штанд часописа са пригодним материјалом, али га нисмо реализовали због слабе организације, незнања и недостатка новца. Сама презентација је дело више појединца а не организације читаве редакције. Потонуће до самог дна, тотални фијаско, понекад и није тако лош положај. Бар имамо чврсто дно од којег можемо да се

одгурнемо и почнемо да испливавамо на површину. Уз то *BalCCon* са својом енергијом, прави је додатни балон који враћа енергију и може да помогне да брже испливамо ако се чврсто ухватимо за њега. Сад је само битно да направимо добру анализу и кренемо у правом смеру. *BalCCon* не враћа енергију само пројекту, него и целој српској заједници. У условима недостатка енергије у самим заједницама, увоз енергије из иностранства је непроценљив. *BalCCon* је недвосмислено показао да *FLOSS* филозофија није мртва, како је то изгледало у последње време. Вести са оптужбама да *Canonical* вуче *Ubuntu* ка комерцијалном софтверу, повратак минхенских рачунара на *Windows*, смањење активности у локалним *FLOSS* заједницама, проблеми са хостингом *Ubuntu-rs* и нешто раније *Archlinux-rs*, фактори су који су трошили домаћу *FLOSS* енергију. Хвала *LUGoNS*-у, а посебно Милошу Красојевићу што су организовали овај конгрес и донели нову *FLOSS* енергију у Србији.

ЛиБРЕ! иде даље са новом енергијом. Позивамо све оне који осећају прилив ове нове енергије да нам се јаве на нашу, већ познату адресу електронске поште libre [et] lugons [dot] org.

До читања

ЛиБРЕ! тим

Моћ слободног софтвера

Број: 28
Периодика излажења: месечник

Извршни уредник: Стефан Ножинић

Главни лектор: Јелена Мунћан

Лектура:
Александар Божиновић
Милена Беран
Маја Панајотовић
Александра Ристовић

Редакција:
Дејан Чугаљ
Александар Тодоровић
Марко Кажић
Гаврило Продановић
Вељко Симић
Александар Брковић
Никола Харди
Михајло Богдановић
Петар Симовић
Владимир Цицовић
Златан Васовић
Марко Новаковић
Александар Весић

Сарадници:
Горан Мекић
Сандрина Димитријевић
Жељко Попивода
Недељко Стефановић
Јоаким Јањатовић
Стефан Стојановић
Јелена Георгијевић
Владимир Попадић

Почасни чланови редакције:
Александар Станисављевић
Жељко Шарић

Графичка обрада:
Дејан Маглов
Иван Радељић

Дизајн:
Младен Шћекић
Зоран Лојпур
White Circle Creative Team

Контакт:
IRC: #floss-magazin на irc.freenode.net
Е-пошта: libre@lugons.org

http://libre.lugons.org

ЛиБРЕ! пријатељи

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 фондација нуди стипендије онима који би помогли у развоју слободног софтвера
4. август 2014.

Linux фондација и ове године нуди стипендије студентима који развијају слободан софтвер.

Користан линк: http://j.mp/1saRcbu

Elementary OS Freya бета издање
11. август 2014.

Изашло је ново бета издање ове *Linux* дистрибуције.

Користан линк: http://j.mp/1wwC0U4

Fedora ради на новом менаџеру партиција
7. септембар 2014.

Fedora израђује нови менаџер партиција који ће подржавати системе који до сада нису подржани у графичким програмима за ову намену.

Користан линк: http://j.mp/1mmt4js

Одржан је други интернационални Балкански хакерски конгрес
7. септембар 2014.

Од 5. до 7. септембра одржан је други по реду интернационални Балкански хакерски конгрес у Новом Саду у организацији LUGoNS удружења и *Wau Holland* фондације.

Користан линк: http://j.mp/XvGVrZ

Фондација за слободан софтвер и пројекат Debian су покренули базу хардвера који подржава Linux
8. септембар 2014.

Free Software Foundation и *Debian* су покренули базу хардвера који подржава *Linux*. За разлику од осталих база које се баве овом тематиком, њима је циљ промоција хардвера који не захтева власничке драјвере.

Користан линк: http://j.mp/1Dnobfp

Хакован званичан сајт за Bitcoin
9. септембар 2014.

Хакован је званичан сајт за *Bitcoin* као и адреса електронске поште *Satoshi Nakamoto*, оснивача ове валуте.

Користан линк: http://j.mp/1AVf69u

GCC 5 ће имати пуну подршку за Cilk Plus
10. септембар 2014.

GCC ће имати пуну подршку за ову *Intel*-ову технологију од верзије 5.

Користан линк: http://j.mp/1o9ppBj

Counter-Strike: Global Offensive ускоро на Linux-у
11. септембар 2014.

Valve планира портовање ове популарне игре на *Linux*. Ова игра постоји за *Windows* и *OS X* већ две године.

Користан линк: http://j.mp/1uOiCBU

Турин прелази на Linux
12. септембар 2014.

Овај град у Италији предвиђа уштеду од шест милиона евра преласком на слободан софтвер.

Користан линк: http://j.mp/1yjyDVk

Android Wear ће „прегазити" Apple Watch
12. септембар 2014.

Аналитичари предвиђају да ће *Android Wear* прегазити *Apple Watch* на овом пољу.

Користан линк: http://j.mp/1sDtCP1

2k14 - Second Base

Први утисци

Аутор: Стефан Ножинић

Као што сви добро знате, почетком септембра, одржан је највећи догађај на Балкану намењен промоцији слободног софтвера, слободи образовања и хактивизму. Овогодишњи *BalCCon* је други по реду а организован је под слоганом „*Second Base*" који је одличан наставак прошлогодишњег слогана „*First Contact*". Иако је тешко пренети сјајну атмосферу и огромну енергију људи који су присуствовали овогодишњем *BalCCon*-у, покушаћемо да вам у овом тексту пренесемо макар трачак читавог доживљаја. Делује помало грубо да читав догађај поделимо у три дана, јер је хакерска енергија владала и ноћима, стога ћемо описати сваки дан засебно.

Пред *BalCCon*

BalCCon је, за неке, почео дан раније, него што је најављено. У Музеју савремене уметности Војводине састали су се 4. септембра у 11 часова главни организатори догађаја и волонтери како би се договорили о плановима за наредна три дана да би се конгрес одвијао предвиђеним током. После кратког састанка, остатак дана пратила је опуштена атмосфера уз пиће, добар провод и, наравно, хаковање.

Петак - први дан

Јутро је помало кишовито, али то није могло да спречи овако сјајан догађај где се људи друже, забављају и размењују знање. Већ од раних сати могле су се у Музеју видети гомиле људи како дискутују на разне теме од којих неке ретко када чујете у вашем свакодневном окружењу. Сви су чекали прва предавања и радионице који су биле одржани у две сале. Обе сале су добиле и своје називе по нашим највећим научницима: Тесли и Пупину.

Мало по мало, и кренула су прва предавање. Прво предавање је било на тему квантне криптографије. Иако је ово сасвим нова дисциплина, за доста људи, помало комплексна, многи су ово слушали са одушевљењем и радозналошћу шта све могу да ураде квантни рачунари. *Bernd Fix* нам је указао на важност развоја алгоритама за енкрипцију који узимају у обзир моћ квантних рачунара и указао је на чињеницу да данашњи крипто алгоритми могу бити лако пробијени коришћењем квантних рачунара.

После овог сјајног предавања, уследила је реч о патентима и ауторским правима. Указано је шта је патент, шта је ауторско право и дефинисане су многе ствари које се користе у правним водама. Након увода, Жарко Птичек говорио је о томе где се ту налази Србија, али исто тако је поменуо занимљивости везане за остале земље као и проблеме које постоје у системима тих земаља. Паралелно са овим предавањем, Ђорђе Марковић је причао о *3D* штампачима, *bio-printing*-у и комбинацији тродимензионалног скенирања и тродимензионалног штампања.

После је уследило предавање Васила Колева о системима датотека, сигурности тих система и начинима како такве системе учинити сигурним коришћењем

енкрипције. Паралелно је у Тесли о фракталима говорио Предраг Бокшић. Он је говорио о хаосу и о томе како је свет око нас хаотичан и како можемо владати таквим системима.

Спава ли вам се? *Christina Johanna* нам је причала о спавању. Највећи проблем код хакера јесте спавање. Како функционише сан? Како да спавамо квалитетније и како да контролишемо нашу поспаност? Или, боље рећи, како да ухакујемо наш сан? И тако, док су неки спавали, други су разговарали о стартапима (енг. *startup*). Небојша Вислављски и Никола Новаковић причали су о великим компанијама, како изабрати најбољу технологију за стартап и где се отворене технологије налазе у трци са комерцијалним.

David Oswald нам је причао о електронском закључавању које се све више користи у разним зградама и о томе како се могу такви системи пробити. Паралелно са тим, одржани су *Lighting talks* где су представљени разни пројекти у говору од пет до десет минута. Између осталог, Никола Харди је представио ЛиБРЕ! и указао на циљ пројекта, планове као и проблеме са којима се суочавамо. Већ у касним сатима сте могли чути предавање о пројекту *Tor*, о ком је писано и у нашем часопису. *Moritz Bartl* нам је причао зашто је *Tor* важан и како се користи. Иако смо покушали да кренемо хронолошки, намерно смо изоставили једно предавање и оставили га за крај. *Mitch Altman* нам је причао о слободном хардверу, али не само о томе. Од овог сјајног хакера и иноватора, могли се чути доста паметних ствари.

Altman је причао о томе зашто је важно да радимо оно што волимо. Причао је о *hackerspace*-у и поменуо је један такав у Сан Франциску: *Noise Bridge*. *Altman* је нудио и кључеве како би сви имали приступ просторијама независно да ли је неко ту или није. Већ на почетку предавања није се могло ући у салу јер су сва места била заузета. *Mitch Altman* је указао на много битне ствари и пренео свој ентузијазам. Причао је и о својој иновацији „*TV-B Gone*", уређају намењеном за даљинско искључивање телевизора на јавним местима. Да је ово на многе оставило огроман позитиван утисак, показао је и велики аплауз задивљене већине.

Субота - други дан

Викенд - доста људи не ради викендом, а и ученици и студенти су слободни. О томе сведочи и чињеница да је број људи овог дана значајно порастао. Оно што вам можемо одмах рећи, јесте да су волонтери задужени за стављање наруквица за посетиоце, имали пуне руке посла овог суботњег јутра.

Од раног јутра кренули смо право у мету: Вакцинација *Android*-а. Да ли можемо веровати мобилним апликацијама данас? Како да их тестирамо и какве аплете (енг. *applets*) можемо користити? О томе су нам причали Милан Габор и Данијел Грах. Паралелно се у Тесли говорило о аутоматизацији домова за педесет евра. Звучи немогуће? Не за Николу Расовића. Он нам је причао како је аутоматизовао свој дом коришћењем *Raspberry Pi*-а.

Александр Тиморин је причао о *SCADA* протоколима и њиховој сигурности, а на крају нам је показао и све то у пракси кроз демонстрацију. За то време, Ивица Коленкаш је говорио о структуирању података коришћењем *MongoDB*-а и о важности свега тога. Лука Герзић нам је причао о развоју *WEB* апликација са фокусом на безбедност. Указао је на честе грешке које, понекад и најбољи програмери, могу да направе. Предавање је било пропраћено забавним темама и можемо слободно рећи да смо у себи рекли „Човече, ово се и мени једном десило".

Патроклос Аргуроудис је одржао предавање о експлоатацији меморије. Помињао је честе грешке код овог нивоа као што су *buffer overflow*, *use-after-free* и

други. За све оне који нису присуствовали радионицама почетком године о *OpenStreetMap*-у које је држао Хрвоје Богнер, могли су то урадити и на *BalCCon*-у. Он је одржао радионицу и говор о томе шта је *OSM* и како се користи. Ако се плашите да вам неко не украде кола, могли сте чути предавање о заштити аутомобила. Шта вам је потребно? Ништа специјално, један *Arduino* и неколико додатака. Ово потврђује и Мариан Маринов који нам је на свом примеру показао како да заштитимо свој аутомобил.

О мерењу је говорио Марјан Урекар. Говорио је о мултиметрима, како да меримо напон, струју, отпор и остале величине. Звучи једноставно? Нисте свесни колико ту има детаља и колико је битно знати како ти уређаји функционишу. Шта ћемо са сигналима и разним њиховим облицима? Марјан Урекар нам показује и како да користимо осцилоскоп и објашњава његову функционалност. Аустрија, земља демократије - никако! *Fin* нам је говорио о слободи информација и проблеме на које она наилази. Говорио нам је како Аустрија спречава новинаре и хакере у раду, зашто су новинарима потребни хакери, и зашто су хакерима неопходни новинари. *Anil Kurmus* је причао о нападима на *Linux* кернел. Он је говорио о површини напада на кернел и њеној редукцији уклањањем могућности кернелу. Представио је два приступа који то могу да изведу, један током компајлирања, а други током извршавања. Представио нам је и занимљиве статистике и поређења. Мирослав Штампар је говорио о сигурности и експлоатацији софтвера, некада и сада. Ако сте били на овогодишњем *BarCamp*-у у фебруару, могли сте чути *Silvan*-а *Gebhardt*-а о *BGP routing*-у. Он је пружио увид у ову тематику и на овогодишњем *BalCCon*-у. Могли смо чути и предавање о *RIPE Atlas API*-у и сазнати чему служи и како га можемо користити за мерења и истраживања. Ово предавање је одржала Весна Манојловић. На крају дана, уследио је „*Rakija workshop*". Од дуње до крушке, могли сте пробати наше национално пиће на чије неке варијације Србија има регистрован *trademark*. Окупили су се сви у соби за радионице с циљем да пробају разне врсте ракије и да се друже после дугог дана.

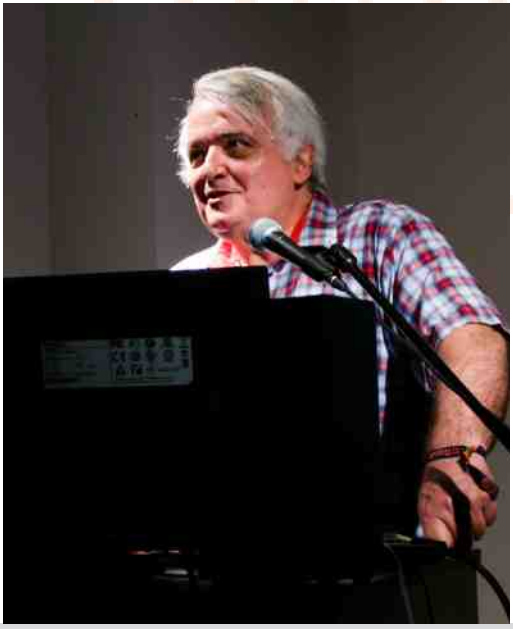
Недеља - трећи дан

Од раног јутра кренуло се радно. *Mitch* нам је одржао радионицу о лемљењу и полазници су добили прилику да сами направе сопствени *TV-B gone* уређај који су могли да понесу својој кући. Било је тридесет места за ову радионицу и већ на самом почетку све је било попуњено, а половина места је већ била резервисана.

Поред ове радионице, у току су била предавања о слободи говора и улози интернета које је одржао Андреј Петровски, предавање о конфликтима права на приватност, заштиту података и на интелектуалну својину које је одржала Јелена Јовановић. Предавање о Википедији, како се пишу добри чланци на њој и где се жене налазе у свему томе, одржала нам је *Greta Doci*. Након *Mitch*-ове фантастичне радионице, Александар Пејић и Андрија Прчић су нам причали о *Linux* драјверима (енг. *driver* - управљачки програм) за уграђене (енг. *embedded*) системе. Тонимир Кисасонди је говорио о крековању шифри и паметном генерисању листа речи за ту намену. Дарко Ивковић је говорио о људском сату, како појединци доприносе глобалној мрежи и сличним стварима. За то време *Anand Buddhdev* нас је увео у *Ansible*, систем управљања сервера али и осталих система па чак и вашег кућног рачунара. Показао је његове предности и начин коришћења. Ми планирамо да у наредним бројевима напишемо текст о овом систему и тако покажемо и вама како се овај алат може паметно користити за олакшавање разних послова код куће али и у предузећима. Влатко Костурјак нам је говорио о *77FEh* и приказао какви га све уређаји користе и објаснио шта је то и чему служи. Чаба Пардовички нам је показао шта је *Docker* и зашто је то један од омиљених алата људи који се баве развојем софтвера у последњој деценији, како се користи и које су му предности. Жарко Живанов нам је причао како су настали кућни рачунари у Југославији и, као да је циљано, а можда и јесте, направљен је одличан увод за наредно предавање, последње ове године на *BalCCon*-у.

То последње предавање је можда оставило онолики утисак колико је оставио *Mitch* првог дана. Воја Антонић, човек који је са тринаест година, без огромног знања електронике, направио систем који је служио људима да науче где се налазе велики градови у Југославији. Да

ли је ту стао? Није. Упркос чињеници да је тада у Југославију било тешко увести било шта што је било мало скупље, то га није спречило да набави себи свој први рачунар из САД. Ако се питате како, покушајте да замислите сценарио где му је достављан један рачунар из више делова путем више пакета. На његову срећу, али и на нашу, то се исплатило. Он није био један од оних који су узимали рачунаре да их користе, већ један од оних који су узимали рачунаре да би научили како они функционишу. Мало по мало настаје и први кућни рачунар у Југославији - Галаксија. Воја Антонић је испричао разне анегдоте из његове каријере, а публика је то са задовољством слушала. Током предавања сви су се добро насмејали и прекидали Антонића огромним аплаузима. На крају је уследио највећи аплауз, а затим и сликање са овим сјајним иноватором.



Ван сала за предавања

На конгресу је било могуће видети и неке примерке старих рачунара и посетиоци су имали прилику да се опробају игрицама из осамдесетих година прошлог века, али и да виде *quadcopter*, малу летелицу коју покрећу четири ротора.

Поред свега овога, на конгресу су били и штандови за заједнице између којих је био и штанд за наш часопис где се могао узети *CD* са архивом досадашњих бројева нашег часописа.

За крај

Можемо са сигурношћу рећи да је ове године *LUGoNS* оправдао наша очекивања и да се надамо да ће следеће године бити још боље. Такође признајемо да смо се као часопис могли више ангажовати, али на грешкама се учи тако да не вреди да жалимо што нисмо направили интервју са предавачима.

У сваком случају можемо да закључимо да су овакви догађаји најбољи пример промоције слободе образовања и хактивизма и да их треба организовати и већем броју.

Преглед популарности *GNU/Linux /BSD* дистрибуција за месец септембар

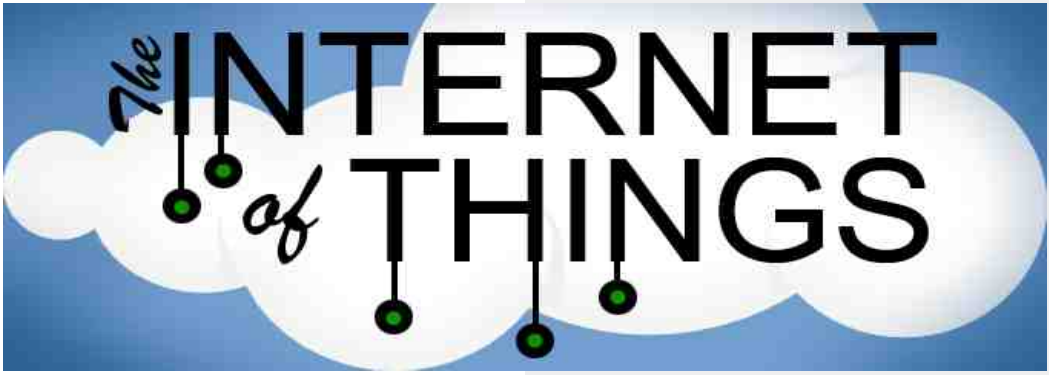
Distrowatch

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>

Пад <
Пораст >
Исти рејтинг =
(Коришћени подаци са *Distrowatch*-а)

Слободан софтвер и интернет ствари

(1. део)



Аутор: Александар Тодоровић

Да ли можемо рачунамо на софтверске гиганте?

Интернет је постао распрострањенији него што је ико могао претпоставити. Данашњи паметни телефони имају више снаге него супер-рачунари које смо користили пре само 20 година. Компанија *Cisco* предвиђа да ћемо до 2020. имати преко 50 милијарди уређаја спојених на интернет. Фаза прикључивања рачунара, лаптопа и телефона на интернет је већ иза нас, а сада су дошли на ред и остали уређаји: паметне наочаре, паметни сатови, паметне куће, паметни аутомобили... Нова решења и нови „паметни" предмети се појављују на тржишту рекордном брзином и то је већ свима врло познато.

Како се *Linux* сналази на новом тржишту које је веће него икада до сада?

Отворене технологије имају огромну предност над власничким решењима у овој фази развоја интернета. Више се не ради о једној врсти уређаја и о једном оперативном систему који треба да комуницира са истим типом уређаја. Сада се ради на томе да се повеже што више различитих уређаја и да се омогући њихова међусобна комуникација и управљање. Да би једна компанија ту преузела превласт, морала би да у врло кратком времену избаци много квалитетних различитих производа на тржиште и да увери потрошаче да су баш њихови производи оно што им треба. Иако не желимо да потцењујемо велике

компаније као што су *Google*, *Microsoft* и *Apple*, мишљења смо да су шансе да се то догоди веома мале, поготово узимајући у обзир да је та трка већ у току, а од *Microsoft*-а и *Apple*-а још увек нисмо видели ништа ван традиционалног тржишта које обухвата рачунаре, лаптопове, таблете и паметне телефоне. *Google* и те како води у тој бици јер је до сада на тржиште већ избацио паметне телевизоре, сатове и наочаре (све базирано на *Android*-у), а поред тога је урадио огроман посао и када је у питању пласирање на тржиште првог аутомобила који не захтева људско управљање. Једина предност *Google*-а у односу на остала два гиганта на тржишту јесте управо тај што је *Google*-ов софтвер базиран на отвореном коду, те га је као таквог лакше прилагодити за различите намере и различите уређаје.

Међутим, да ли заједница која воли отворен софтвер може да рачуна само на *Google*? Сви знамо да та заједница (из већ добро познатих разлога) није превише одушевљена ни са *Android*-ом за паметне телефоне и биће им врло тешко да прихвате да им нека прилагођена верзија *Android*-а покреће и остале уређаје.

Склањајући *Google* на страну за сада, остаје нам *Red Hat*, који се не фокусира на борбу за гигантима, *Canonical* још увек покушава да прати корак те да се пробије на тржиште паметних телефона и таблета, и *Mozilla* која поред пробијања на тржиште паметних телефона покушава да уради нешто потпуно другачије – да нам осигура отворени интернет.

Како раде паметни уређаји?

Да бисмо схватили како да направимо потпуно отворене уређаје, морамо научити како они раде. Процесори у данашњим паметним телефонима имају више снаге него што су имали супер-рачунари прије само 20 година, што је врло импресиван податак који доказује колико смо напредовали. Међутим, иако су данас минијатурни процесори врло јаки, нису бесплатни. Већина уређаја који ће у будућности бити спојени на интернет већ су одавно у употреби у свом „глупом" издању (у контексту „анти-паметни" упоређујући са појмом паметни телефони). Наша тренутни будилник не може да зна да смо будни већ два сата, наш фрижидер не може да зна да нам је понестало млека у њему, а апарат за прављење кафе не може да претпостави каква би нам кафа одговарала овог јутра. У њима се не налази процесор, него микроконтролер, мали уређај који служи искључиво да уређаје укључимо и искључимо и да јави остатку система шта да раде када је уређај укључен. Сада, замислимо да све микроконтролере у свим уређајима одједном заменимо процесорима и дамо им неке „паметне" особине. По навици, када устанемо, почнемо да радимо нешто на нашем

телефону, било да проверавамо колико је сати, проверавамо мејлове, друштвене сајтове или нешто слично. Уколико радимо неку мало комплекснију радњу (радњу коју није могуће урадити уколико нам се спава), наш телефон би тада могао да пошаље информацију нашем будилнику да се искључи, фрижидер би могао да провери имамо ли довољно млека и да нас обавести о томе да ли треба да одемо до продавнице, а апарат за кафу би могао да зна да смо будни већ неко време, те на основу тога да претпостави да нам није потребна јака кафа да нас разбуди овог јутра. Са данашњом технологијом ми смо одавно у могућности да тако нешто изведемо, па шта нас то у томе спречава?

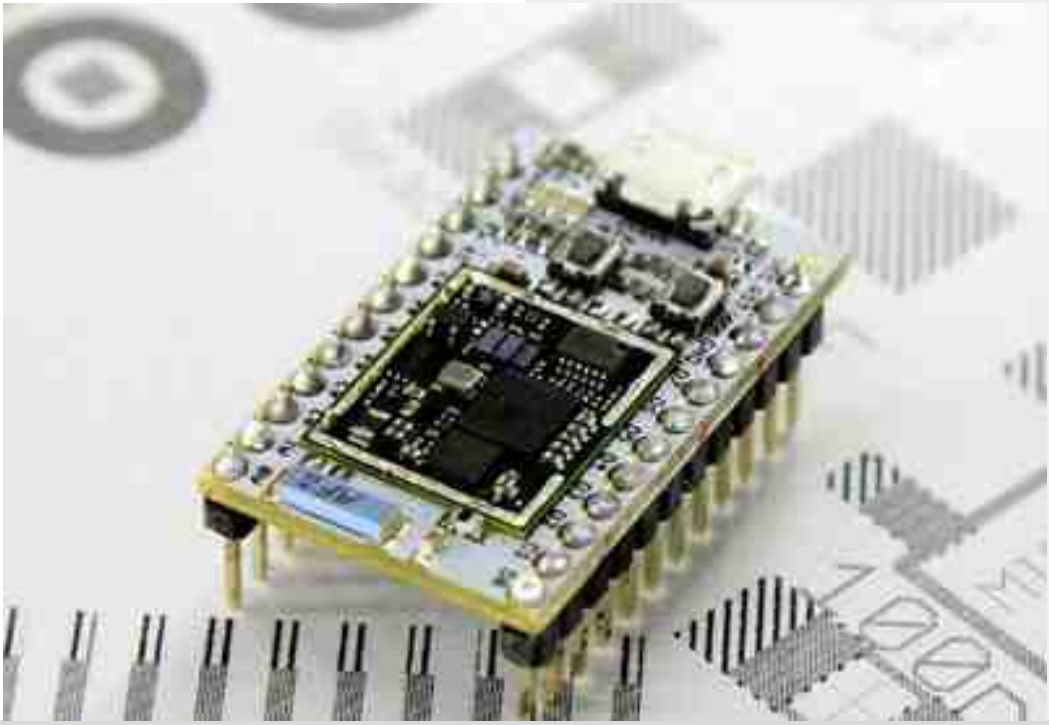
Оно што нас спречава јесте управо цена процесора. Цена на којој се крећу процесори из паметних телефона јесте једноставно неприхватљива за коришћење у неким кућним уређајима из тог разлога што би се цена тих уређаја морала подићи на вишеструко већи износ у односу на онај на којем се тренутно налази. Са друге стране, микроконтролери немају снаге да се носе са свим задацима које им можемо поставити са циљем да нам кућни уређаји постану „паметни". Међутим, шта ако микроконтролерима поставимо нешто лакши задатак?

Ту се за помоћ можемо обратити једној од технологија коју користимо од када постоји интернет: „*Cloud*" комуницирање. Шта ако наши уређаји не морају да спремају огромне количине података? Шта ако се одлуке доносе ван наших кућних уређаја? Цена за изградњу процесора који треба искључиво да се повеже са интернетом и преузме малу количину података је много мања него цена стављања пунокрвних процесора у паметне уређаје. Решења о томе шта уређај заправо треба да уради се налазе у „*Cloud*"-у, наш процесор их само треба преузети и на основу датих решења извршити операције. Чини се као поприлично једноставно. Сада када смо вас увели у ово једноставније решење и објаснили како паметни уређаји раде, време је да покажемо повезаност између овог решења и слободног софтвера.

Повезаност између овог приступа и слободног софтвера јесте у томе што различити уређаји не морају да покрећу потпуно различите оперативне системе. Сви уређаји би могли да користе врло сличне оперативне системе, само програмиране на начин да се управља другачијим деловима уређаја да би се извршила хардверска операција за коју је тај уређај намењен. Сам процес прилагођавања једноставног оперативног система за тачно одређену употребу јесте далеко лакши када пред собом имамо слободно софтверско решење него када пред собом имамо власнички софтвер. Можемо да произведемо много више паметних уређаја у краћем времену него што би то извела нека компанија, и врло брзо би сви уређаји у нашем дому могли бити повезани на централни сервер у нашем дому на којем би се смештали подаци и доносиле одлуке. Подаци о нама никада не би требало да напусте наш дом, процес набавке нових уређаја и њихово повезивање на нашу мрежу би било тривијално, а уређаји би имали интеракцију са нама, онакву какву смо могли само да замислимо пре неколико година, а морамо признати да и сада

буди неки осећај далеког футуризма у нама.

Међутим, тај футуризам је већ остварљив за особе које имају нешто више новца, желе да своје снове претворе у стварност и знања да програмирају како би остварили то све. Интернет ствари („The Internet of Things") је на путу и фанови слободног софтвера имају неколико решења доступних већ сада. Међутим, та решења још увек нису достигла ниво да су лака за коришћење. Свако ко покуша да имплементира паметне особине у доступним уређајима ће морати добро да се потруди да би то остварио, а ми ћемо вам у следећим бројевима представити неколико пројеката који се фокусирају на интернет ствари користећи првенствено слободне компоненте, те представити опасности које нам у будућности доноси интернет. У следећем броју вас очекује чланак о *Spark* пројекту.



Conky Manager

Аутор: Милош Миладиновић

О *Conky*-ју

Conky је програмска апликација која вам омогућава да пратите софтверско и хардверско стање вашег рачунара, поред других опција у које спадају провера поште, *RSS* вести, временске прогнозе и многих сервиса чија примарна сврха не мора бити уско повезана са *Conky*-јем. Генерално гледано, *Conky* олакшава праћење процеса за које би нам иначе било потребно неколико апликација. У свега неколико редова можете имати часовник, датум, активне процесе и оне који највише црпе ресурсе система, температуру рачунара, број непрочитаних имејлова и временску прогнозу. Не може се рећи да је планирано, али *Conky* од почетка тежи да остане лаган и једноставан, са поставкама које стају у двадесетак редова (или много мање) текстуалне датотеке. Овакав начин обично (скоро обавезно) почетницима представља проблем, нарочито ако долазе са *Windows*-а, али се лако превазилази у готово само једном дану и

уз мало стрпљења. Наравно, за оне који немају стрпљења да куцкањем подешавају своју апликацију, ту је *Conky Manager* који има посебну моћ да текстуалне редове претвори у графичко сучеље које чак и апсолутним почетницима омогућава препознатљив доживљај подешавања. Занимљиво је да је *Conky* добио име по лику из ТВ серије *Trailer Park Boys* која се снима од 2001. године. Серија је иначе доживела 8 сезона, док се од 5. марта 2014. снима и 9. сезона ове серије.

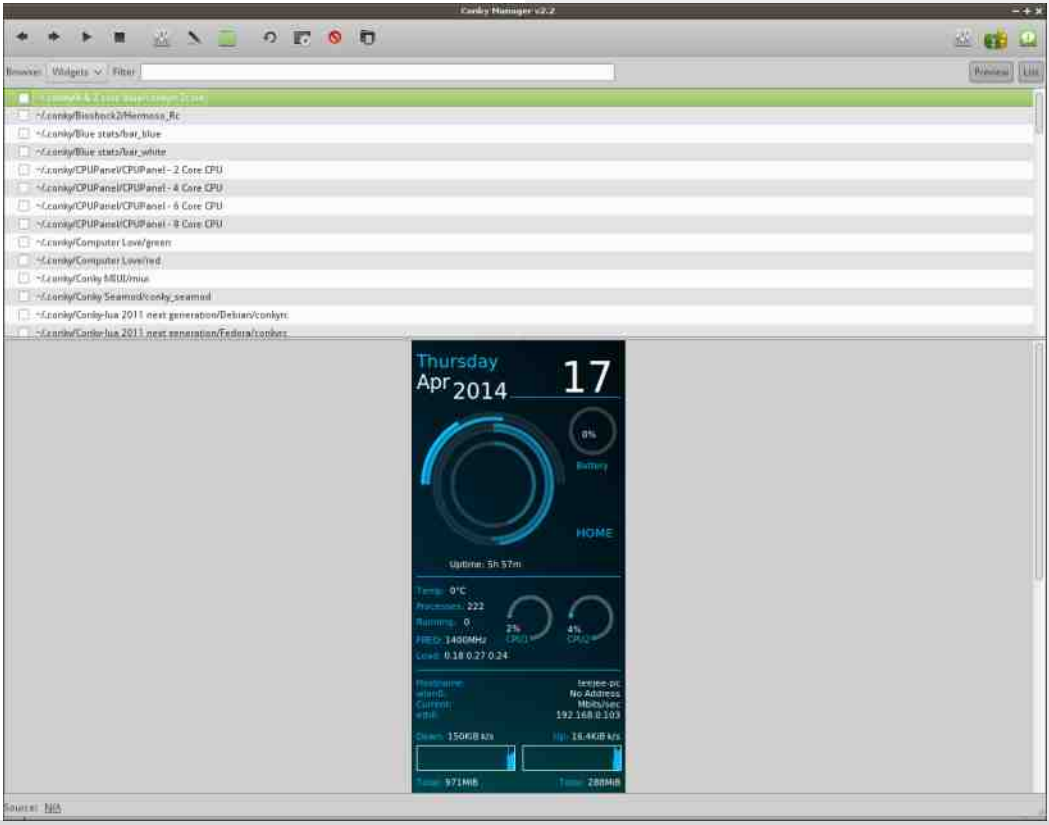
Conky Manager

Неке од предности *Conky Manager*-а су:

- Покретање/заустављање, претрага и подешавање *Conky* тема;
- Покретање *Conky*-ја са системом;
- Опције за мењање локације, транспарентности и величине *Conky*-јевог прозора;
- Опције за мењање времена и избора мреже (*wlan0/eth0*).

Инсталација

Инсталацију је могуће извршити на *Ubuntu* базираним дистрибуцијама



(*Ubuntu*, *Linux Mint*, итд). Уколико користите *Ubuntu* или неки од његових деривата (*Xubuntu*, *Lubuntu*, *Kubuntu*, итд), инсталацију можете извршити путем путем *Lanchpad PPA* на следећа *Ubuntu* издања:

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

За нека од предходних издања можете користити *DEB* датотеке:

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

Да би сте инсталирали *Conky* користећи *PPA*, у терминалу откуцајте следеће команде, једну по једну:

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

Подешавање

Прозор *Conky Manager*-а дизајниран је тако да врло лако можете поставити и подесити омиљени *Conky*. Оно што ћете прво приметити приликом отварања прозора је листа основне понуде *Conky*-ја. Изнад листе су вам понуђени алати за подешавање *Conky*-ја. Од алата, у понуди су следеће опције:

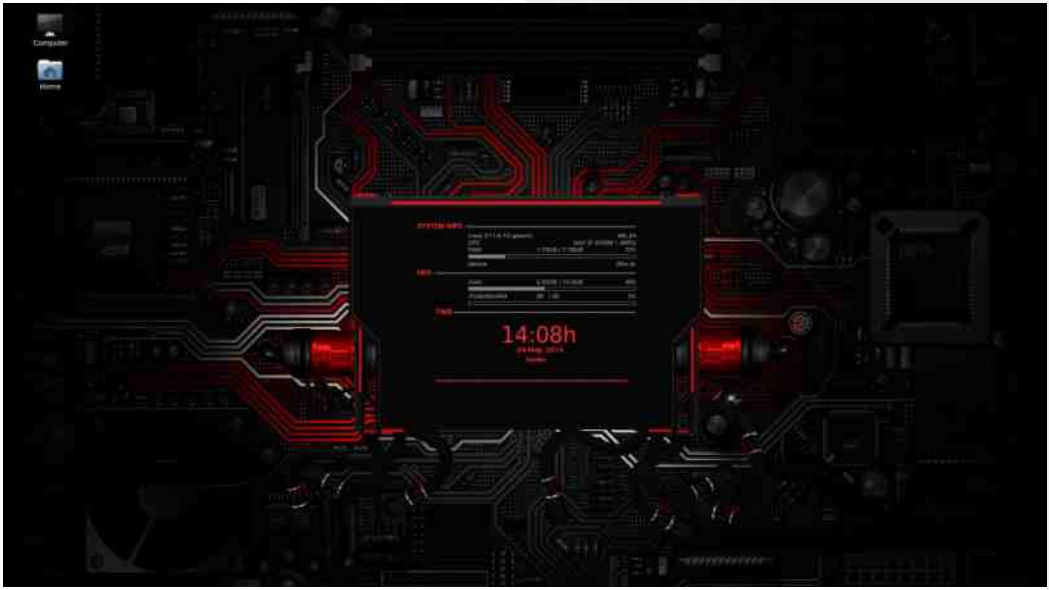
- Стрелице - служе за преглед вицета;
- Паљење/ресетовање вицета;
- Заустављање вицета;
- Подешавање;
- Подешавање *Conky*-ја помоћу текстуалног едитора;

- Отварање фасцикле у којој се налазе теме;
- Претрага нових тема;
- Прављење прегледа тема;
- Комплетно искључивање свих покренутих вицета;
- Додавање нових тема.

У основној инсталацији, избор *Conky*-ја и тема је мали, али са овог линка можете преузети још одличних тема: http://www.teejeetech.in/2014/06/conky-manager-v2-themes.html

Закључак

Као што смо навели у уводу, *Conky Manager* је направљен пре свега за почетнике, за оне који не желе да се претерано баве подешавањем у самом кôду. У сваком случају, *Conky Manager* ће вас упутити, олакшати рад са самим *Conky*-јем, али ће вам и олакшати разумевање начина функционисања *Conky*-ја. Свакако заслужује нашу препоруку.



libGDX „Java game development framework” (4. део)

Аутор: Гаврило Продановић

У прошлим бројевима, говорили смо о *LibGDX*-у у контексту графике и улаза. У овом броју ћемо рећи нешто више о осталим помоћним класама које се не односе директно на ово двоје, а ту су да убрзају и помогну развој игрице и дотјеривање *gameplay*-а. Започећемо звуком чиме смо заокружили једну цјелину која је потребна за „комплетну” игрицу (графика, улаз и звук).

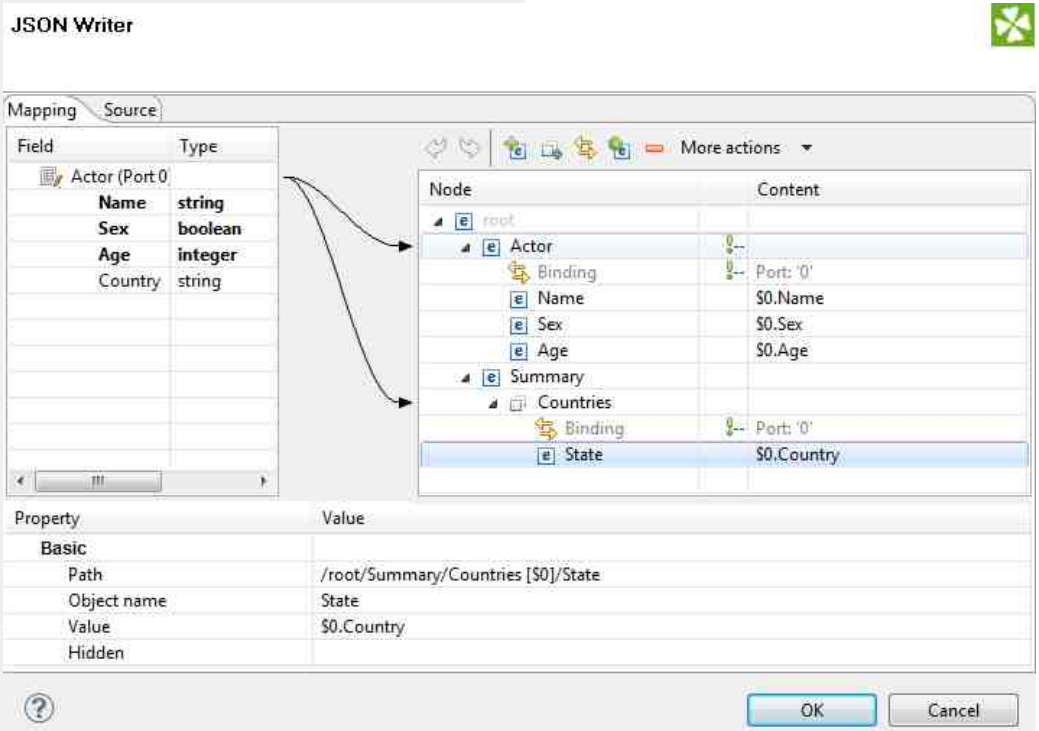
Од формата за аудио, *LibGDX* подржава *OGG*, *MP3* и *WAV*, а за манипулацију овим класама постоје класе *Sound* и *Music*. Основна разлика ове двије класе у примјени је та да прву користимо за репродукцију звучних ефеката, а другу за *streaming* неке пјесме у позадини. Технички гледано, *Sound* класа декодује датотеку и декодирани *stream* учитава у *RAM*, што d *омогућује пуштање ефекта у тачно потребном тренутку. Из једне* Sounинстанце могуће је стимулативно репродуковати исти ефекат више пута, а свака нова инстанца добија свој *ID* који враћа *play* метода. Свакој новој инстанци можемо посебно да подесимо волумен, *loop*, *pan* и *pitch*. На *Android* платформи, *Sound* инстанца не може да буде већа од 1 *MB*, док на *iOS*-у није подржан *OGG*

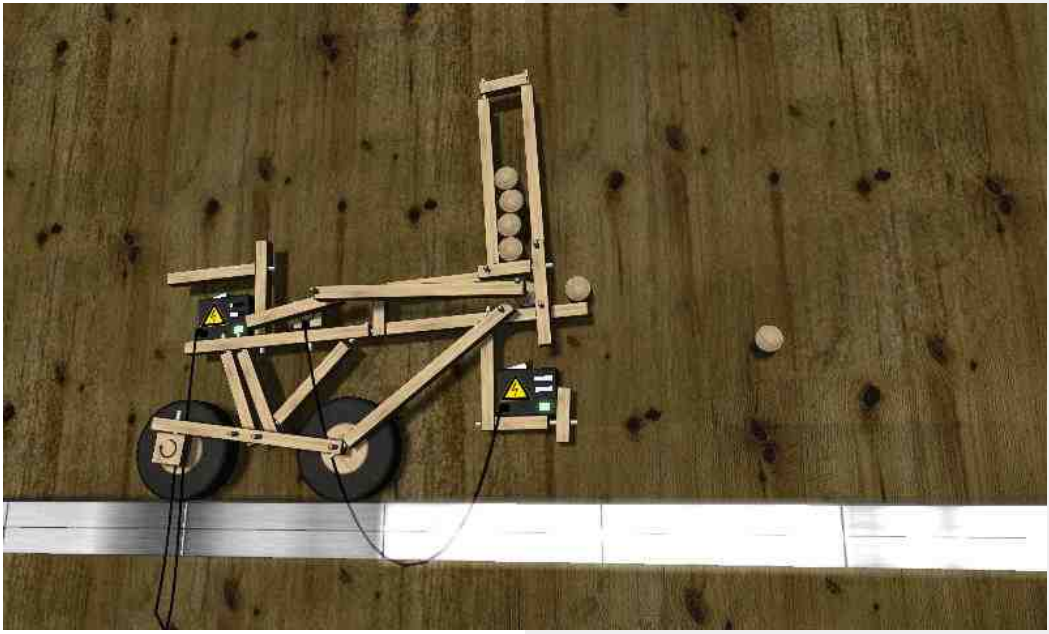
формат. Ако желимо да репродукујемо звук који је дужи од неколико секунди, као што је музичка пратња, коришћење *Music* класе је много бољи избор јер се датотека *stream*-ује са диска по потреби умјесто да се учита читав у *RAM*. Из једне инстанце *Music* објекта није могуће пустити више узастопних нумера као у *Sound* класе. У току репродукције можемо да промјенимо волумен и *pan* или позицију у секундама или да подесимо да *playback* иде у круг. Такође, могуће је имплементирати *listener* који ће да „окине" када се *playback* заврши. Једна од веома лијепих опција по питању аудија је могућност директног приступа хардверу помоћу класе *AudioDevice* преко чијих метода можемо да пошаљемо директно *PCM* „узорак" у бафер. Ако желимо да добијемо улаз са микрофона у томе ће нам помоћи *AudioRecorder*. *AudioDevice* и *AudioRecorder* нису доступни на *JavaScript/WebGL backend*-у. За све класе које смо овде навели потребно је позвати *dispose()* када постану непотребне да би се ослободили ресурси.

JSON и *XML* су често коришћени за организацију података, па се међу многобројним класама налазе и класе за читање и писање ових формата. За *JSON* можемо да се послужимо следећим класама: *JsonWriter*, *JsonReader*, *JsonValue* и *Json*. Функција класа *JsonWriter* и *JsonReader* је очигледна; *JsonValue* описује *Json* објекат, а *Json* класа ће нам помоћи да неки *java* објекат серијализујемо у *Json* објекат или да из *Json* објекта да десерализујемо у *java* објекат.

Захваљујући аутоматској серијализацији у *LibGDX*, могуће је без много муке сачувати или учитати игру из датотеке ако је потребно. У случају да се јави потреба да контролишете серијализацију или десеријализацију постоје интерфејси који ће вам у томе помоћи, да ли на нивоу своје *java* класе тако што ћете имплементирати *Json.Serializable* или на нивоу читавог процеса серијализације тако што ћете у *Json* објекту постави серијализер преко *setSerializer()* методе. За оне којима је потребан *XML*, постоји *XmlReader* и *XmlWriter*. Ако мислите да је то неки од познатих *java XML* паресера имплементиран у *LibGDX* преварили сте се. Према ријечима аутора, он је написао намјенски *XML* парсер за *LibGDX* јер је имао предосјећај да је смислио нешто мало и брзо и хтјео је да тестира своје вјештине у *Ragel*-у. У тестовима *LibGDX*-ов *Xml* парсер се показао много бољи од *javax.xml.parsers.DocumentBuilder* у датотекама од 1 *MB* на *desktop*-у и *Android*-у, док је на датотеци од 100 *MB* *javax* парсер добио трку за нешто мање од једне секунде, *Android* је био искључен у тесту на великим датотекама. Нећемо залазити у коришћење парсера, пошто је *XML* нешто компликованији од *JSON*-а.

Математика у игрицама не може бити запостављена, а за чисту математику *LibGDX* обезбједио пакет *com.badlogic.gdx.math* у којем постоје разноврсне класе да нам помогну. Прво ћемо





споменути *MathUtils* у којој постоје статичке методе за уобичајене математичке операције као што је заокруживање, израчунавања синуса и косинуса, конвертовање степена у радијане. Споменућемо да постоји неколико згодних метода за генерисање насумичних бројева. Постоје класе које дефинишу основне геометријске облике у равни као што су круг, квадрат, елипса и класа која ће нам помоћи да одредимо да ли постоји пресјек између њих. Постоје класе и за интерполацију (познато и као *tweening*) које нам могу помоћи у састављању анимације. Споменућемо да су подржани вектори, матрице и кватерниони. Имплементиран су *Ear-Clipping* алгоритам и још неки, али не планирамо залазити у дубину математичких способности *LibGDX*-а и овде ћемо завршити.

У платформском или *RPG* жанру потребне су мапе које могу бити алгоритамски генерисане или ручно израђене за вријеме развоја игрице. У нашем *framework*-у постоји читав један скуп класа за рад са мапама. Мапа је скуп слојева (енг. *layer*), а сваки слој посједује своје објекте. Један од типова су *Tile maps* које су састављене од плочица или ћелија и карактеристичне су за *RPG*, платформске и сличне игре. *Tile* мапе су једини комплетно имплементирани тип мапа, али урађене су основне апстрактне класе које дефинишу уопштено мапу, слој и објекат тако да би имплементирање било којег другог типа 2Д мапе било једноставно. За *tile* мапе главна класа је *TiledMap* у којој се налазе инстанце *TiledMapTileLayer* класе. У *layer*-има се налазе ћелије које посједују референцу на *TiledMapTile*, а *tile*-ови су обично дјељени међу ћелијама. *Tile* може бити статична текстура или анимација. За

рендеровање ових мапа постоји више рендера. За реднеровање ортогоналних мапа користићемо *OrthogonalTiledMap Renderer*. Рендеру се у конструктору прослеђује мапа како би се извршила оптимизација и кеширање мапе. За изометричне мапе постоји *IsometricTiled MapRenderer* чије је коришћење аналогно ортогоналном рендеру. Постоји још четири рендера, али су они експериментални и нећемо их спомињати. На крају, оно што је најважније међу *tile* мапама јесу формати који су подржани за учитавање. Постоје два едитора, односно формата која су подржани. Први је *open source* едитор *Tiled* са својим *TMX (Tile Map XML)* форматом, а други је формат едитора затвореног кода *Tide*.

LibGDX нам је обезбједио *scene2d*, да логику своје игре организујемо унутар графа који нам помаже да створимо хијерархију између наших „глумаца" (енг. *actor*). Оно што нам *scene2d* омогућује су олакшице као што је управљање групама, на примјер: ротација или транслација коју примјенимо на родитеља одразиће се и на потомке. Сваки потомак посједује свој сопствени кординатни систем који је релативан у односу на родитеља. Кроз организацију нам долази лакша могућност за исцртавање објеката кроз *SpriteBatch*, а могућност управљање улаза такође постоји, што нам даје могућност да родитељ ухвати догађај прије својих потомака или послије њих. Могућност да се *actor*-има задају акције, као што је кретање до одређене позиције је или да пређу одређену дужину у неком смјеру релативно на своју позицију. Трансформација и многе друге је можда главна предност *scene2d*-а да се користи у *gameplay*-у. Такође постоји и пакет *scene2d.ui* који посједује класе потребне за грађење менија или *HUD*-а што може много да нам уштеди вријеме.

За крај, споменућемо два *engine*-а за физику које наш *framework* подржава. Први *engine* је *Box2D* намјењен као што му име говори за *2D* физику. Од верзије 1.0 је издвојен као екстензија док је прије био укључен у изворни код. *Box2D* је *C++ engine* док је у *LibGDX* имплементиран преко лаког *java wrapper*-а. Нећемо залазити у дубине, зато што је *Box2D* читава прича за себе, али скоро свако искуство које посједујете у *C++* језику по питању овог *engine*, може се превести на *java*-у. По питању перформанси, *Box2D* се показао и више него солидан колико смо ми имали личног искуства у њему. Поред *Box2D*, постоји и подршка за *Bullet Engine* који је намјењен за *3D* физику. Он је писан у *C++*-у, а за *LibGDX* постоји *Java wrapper*.

Увод у програмски језик *C*

(5. део)

Аутор: Вељко Симић

У прошлом броју, причали смо о низовима и матрицама, и сада имамо једно питање: Шта радити уколико желимо да испишемо пет различитих низова? Код би изгледао овако:

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

Тај код бисмо искуцали пет пута. Мана овога је то што код постаје знатно непрегледнији. Ако бисмо хтели на пет места у коду да испишемо пет низова, имали бисмо 50 линија кода и прилично би било лако да се изгубимо у том коду. Ово је један прост пример где бисмо могли да применимо функције. Функција која исписује низ изгледала би овако:

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

Објаснимо сада ред по ред ове функције: *void ispisiNiz(int niz], int n)* - Ово је заглавље функције, *void* означава тип функције. То је тип који враћа функција. Питате се сада који је то тип и зашто га нисмо споменули када смо причали о

типовима података. То је непостојећи тип, дакле функција не враћа никакву вредност. После типа функције следи њено име, па заграда у којој се наводе аргументи функције. Функција може, а и не мора да има аргументе. Аргументе функције наводимо тако што наведемо прво тип аргумента, па име. После тога, у витичастим заградама се налази тело функције. Ту се налази све што функција треба да одради. Последња наредба која се извршава јесте повратна вредност функције нпр. *return 0*. Када напишемо функцију, све што је потребно јесте да је позовемо у функцији *main* и проследимо јој параметре и она ће бити извршена.

Разлика између параметра и аргумента.

Ова два термина се веома често мешају. Параметри функције су оне вредности које се прослеђују при позиву функције, док су аргументи функције они који се наводе при дефиницији функције, нпр. када смо малопре навели за испис низа *„niz“* и *„n“*, то су били аргументи те функције, док су *„“%d”,nizi]“* параметри функције *printf*.

Можда нисте приметили, али до сада смо и писали и користили функције. Функцију *main* сваки програм мора да има. То је главна функција од које почиње рад нашег програма. На крају те функције, писали смо *return 0;*. Функција *main* враћа вредност 0 уколико је све добро извршено. Користили смо функције *printf* и *scanf*. Оне се налазе у стандардној библиотеци *stdio.h* у којој се налазе улазно/излазне функције.

Рекурзивне функције

Рекурзивне функције су оне функције које у свом телу позивају саму себе. Рекурзивне функције се састоје из два дела: тривијалног случаја и рекурзивног позива. Појаснићемо на примеру:

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

Тривијалан случај ове функције је провера да ли је *n* (тј. број за који израчунавамо факторијел) мањи од један, или је једнак један. Тада се рекурзивна функција завршава. Рекурзиван позив је у *else* грани. Да бисте лакше схватили како се рекурзивна функција извршава, функцију ћемо изменити да нам испише када улази, а када излази из функције. Тако да ћемо *else* грану написати другачије:

```
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;
  }
}
```

Када бисмо позвали ову функцију у функцији *main* са параметром 4 и покренули програм као излаз, добили бисмо ово:

```
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
```

Као што видите, при сваком позиву покрене се нова функција. Као да умотавамо и одмотавамо папир. У општем случају, када унутар једне функције позивамо другу, прва наставља да се извршава тек када се заврши друга функција.

У потрази за идеалном дистрибуцијом:

Почетак

Аутор: Дејан Маглов

Овај серијал, који смо започели у прошлом броју часописа, замишљен је као серијал који кроз наша искуства помаже *FLOSS* почетницима да пронађу идеалну дистрибуцију за себе. Због велике дисперзије *FLOSS* оперативних система, питање свих питања за почетнике јесте „Која је идеална дистрибуција за мене". Идеално не постоји. Оно што је за једну особу идеално није идеално за неког другог. Према томе, ова наша потрага за идеалним је у ствари „прича која се никад не завршава" (*NeverEnding Story*).

Ако сте почетник у *FLOSS*-у или вас је само наш часопис „заголицао" и заинтересовао за *FLOSS*, можда вам сад и није баш најјасније шта значи тражити идеалну дистрибуцију. Као прво, морамо разјаснити шта је „дистрибуција" у овом контексту. За почетак претпоставимо да сте се заинтересовали за слободни софтвер. Као прва „раскрсница" стоји вам одабир *FLOSS* оперативног система. У *FLOSS* понуди вам је *Linux*, *BSD* и неколико много мањих пројеката. Сразмера популарности, међу обичним корисницима, између *Linux*-а и *BSD*-а је отприлике као и између *Windows*-а и *Linux*-а. Међу обичним корисницима *Linux* је много популарнији од *BSD*-а, па претпоставимо да сте се определили за неку варијанту *Linux*-а.

Када кажемо *Linux*, то значи да користимо неки оперативни систем чији

кернел је *Linux*. Кернел је „срце" оперативног система. Упроштено, он представља везу између корисника и хардвера. Ако сте читали наш двадесет и трећи број и чланак „*Linux* унатрашке", могли сте да видите да је и сам оперативни систем сложенији и да је кернел само један његов део. *Linux* кернел јесте најважнији део оперативног система, али без апликација, сервера и менаџера, оперативни систем не би постојао као целина која обавља одређени посао за корисника. *FLOSS* чистунци, стога, овај оперативни систем зову *GNU/Linux*, где *GNU* представља слободне апликације, менаџере и сервере, а *Linux*-у остаје заслуга само за кернел (срце система). У наставку текста испоштоваћемо ову конвенцију код именовања, па ћемо комплетан оперативни систем звати *GNU/Linux*, а ако мислимо само на кернел зваћемо га просто *Linux*.

Спајањем различитог слободног софтвера са *Linux* кернелом може се добити огроман број комбинација. Ако томе додамо различиту „шминку" и подешавања, број комбинација је неограничен. Свако од нас, ако има довољно знања, може сам да направи своју комбинацију *Linux*-а, слободног софтвера, „шминке", подешавања и направи свој *GNU/Linux*. Ако свој сопствени *GNU/Linux* понудите и другима на коришћење, то ће се звати вашом *Linux* дистрибуцијом јер даље дистрибуирате (делите) *Linux* кернел. Због тога се варијанте *GNU/Linux*-а зову дистрибуцијама *Linux*-а по његовом најважнијем делу – кернелу.

Огромна већина нас, не само почетника већ и прилично искусних корисника *GNU/Linux*-а, нема довољно знања да састави идеалну дистрибуцију за себе из саставних делова, зато користимо услуге програмера који су то већ урадили за нас и понудили нам своје производе. Ти програмери су искомбиновали идеалну дистрибуцију за себе, али су и надоградили још неке ствари како би задовољили и неке шире укусе. То што је идеално за њих можда није идеално и за нас. Свака дистрибуција је специфична на свој начин и има акценат на оном што је било битније програмеру који ју је комбиновао. На нама је сад да у мору већ готових дистрибуција пронађемо ону која најбоље ради на нашем хардверу, која има нагласак на функције које су нама битне, додамо шминку и подешавања по свом укусу. Сад је, ваљда, мало јасније зашто се сви у *GNU/Linux* свету претварамо у истраживаче у потери за идеалном дистрибуцијом.

Потрага за идеалном дистрибуцијом по кернелу

Сви *GNU/Linux*-и имају „једнак" *Linux* кернел који се развија контролисано под надзором његовог творца Линуса Торвалдса. Кернел није измишљотина *GNU/Linux*-а: поседују га и сви остали оперативни системи (*Windows*, *MacOS*, *BSD*).

Разлика између *Windows* и *Linux* кернела је у транспарентности тог дела оперативног система и у одвојеност *Windows* кернела од управљачког софтвера за хардвер (*drivers*-а). За разлику од *Windows*-а, *Linux* има уграђене управљачке програме за већину познатог хардвера.

Код инсталације *OS Windows*-а, корисник прво што мора да одради јесте да инсталира све припадајуће покретачке програме да би хардвер прорадио. Код *GNU/Linux*-а, ако је све прошло како треба (одабран је одговарајући *Linux* кернел), сав стандардни хардвер треба да ради одмах по инсталацији система. Неки специфични хардвери нуде додатне покретачке програме за *Linux*, ако је то произвођач предвидео. Осим тога, кориснику је дато да бира да ли ће да користи слободне покретачке програме за графичку картицу или власничке покретачке програме које нуде произвођачи графичких картица (*Nvidia*, *AMD-Radeon*).

На почетку овог поглавља, ставили смо под наводнике реч „једнак”. У принципу *Linux* кернел има једнак принцип рада али, углавном због различитих управљачких програма, постоје више верзија *Linux* кернела у оптицају који се редовно одржавају. У тренутку писања овог чланка, најстарији *Linux* кернел који се редовно одржава је са ознаком 2.6, а најновији стабилни је 3.16. Поред њих са продуженим одржавањем, од стране

kernel.org-а, у оптицају су и верзије 3.2, 3.4, 3.10, 3.12, 3.14. Осим ових верзија које одржава *kernel.org* и саме веће дистрибуције за своје потребе одржавају неку међуверзију кернела, као на пример кернел 3.13 који ће бити одржаван од стране *Canonicala* за потребе *Ubuntu 14.04 LTS* до 2019. године.

Од кернела зависи функционалност оперативног система, зато треба водити рачуна да се изабере кернел који одговара хардверу. Ми нећемо сами компоновати *GNU/Linux* систем, већ ћемо бирати дистрибуцију из гомиле већ понуђених. Кернел је, вероватно, први и најважнији критеријум за одабир одговарајуће дистрибуције.

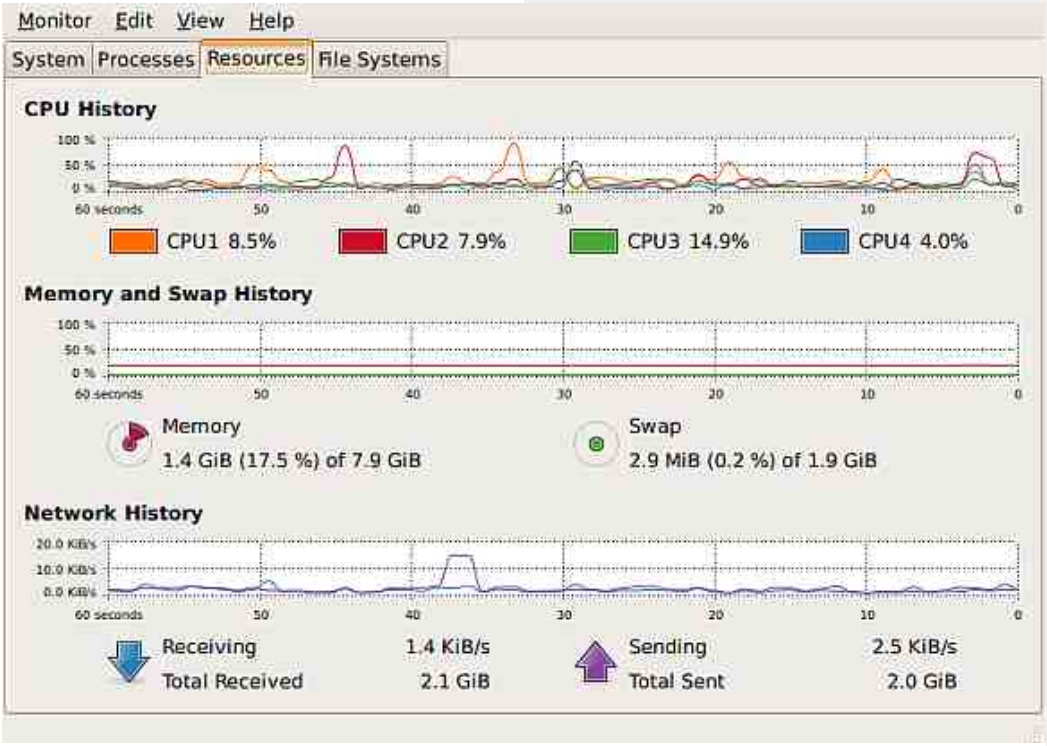
Како бирати? Код *Linux*-а не значи да је најновији кернел и најбољи. Хардвер се мења, а са тим мењају се и неки принципи рада тако да најновији хардвер неће радити са старијим кернелима јер не постоје управљачки програми за њега, али исто тако и старији хардвер често одбија да буде покренут најновијим кернелом. Поуздано знамо да стари рачунар *Pentium 2* класе може да покрене кернел 3.10 и старији. Новији кернели неће радити на тој старој машини.

Проблеми око кернела углавном се односе на екстремне случајеве хардвера, као што су потпуно нов хардвер и много стари хардвер. Ако поседујете неки од актуелних хардвера, што претпоставља хардвер који није „последњи крик" технологије и не старији од четири године, највероватније нећете имати проблема са кернелом. Мада можда, треба избегавати најновији кернел због могућих багова који су се „провукли" и који ће тек бити уочени и отклоњени али исто тако и најстарији кернел који се одржава јер је он баш намењен за застарели хардвер.

Ресурси хардвера као критеријуми за одабир идеалне дистрибуције

Од кернела зависи функционалност оперативног система. Сама функционалност не подразумева и угодан рад. Предуго чекање на одзив система на постављени захтев корисника може да буде иритирајуће. На „живахност”, односно брзи одзив система, утичу хардверски ресурси. Најважнији хардверски ресурси на које треба обратити пажњу, јесу количина *RAM* меморије, брзина процесора, број језгара процесора, величина хард диска, брзина хард диска, квалитет графичке картице и други мање уочљиви хардверски ресурси за корисника. Идеално би било имати брз процесор (*2.5 GHz* и више) са два језфра или више њих, више хард дискова великог капацитета (преко *250GB*) за складиште, *SSD* (eng. *solid-state drive*) – хард диск брзог одзива који је идеалан за оперативни систем и апликације, доста брзе системске меморије (преко *4GB RAM*-а), квалитетну графичку картицу са добрим графичким процесором и што више сопствене брзе графичке меморије (преко *1 GB*). Обичан корисник рачунара нема потребе за толиком хардверском снагом. За удобан свакодневни канцеларијски рад (рад у *office* пакету програма, интернет претраживачу, музичком и видео плејеру) довољан је и много слабији хардвер.

Хардверски минимум за мало захтевније кориснике који може да „потера” било



коју до сада познату *Linux* дистрибуцију је рачунар *Pentijum 4* класе са процесором *2.5GHz* са 2 језгра, *4 GB RAM*-а, хард диском од *250GB*, са екстерном графичком картицом класе *Gforce 8* са *1 GB* графичке меморије. Овакве спецификације задовољава хардвер стар 4-5 година.

И много скромнији хардвер може успешно да ради под *GNU/Linux*-ом али захтева мало рационализације и бирања софтвера који није толико захтеван за хардверским ресурсима. Један од споредних делова *GNU/Linux*-а чијим правилним избором може доста да се уштеди на потрошњи хардверских ресурса је графичко окружење.

GNU/Linux оперативни систем може, теоретски, да функционише и потпуно без графичког окружења. Наравно то подразумева и употребу апликација без графичког окружења. Пошто се савремени оперативни систем не може замислити без графичког окружења и апликација (програма) са графичким окружењем, и *GNU/Linux* има графичко окружење и то не једно него више њих. Нека графичка окружења су усмерена ка дисплејима осетљивима на додир (*Gnome*, *Unity*), нека дају пуну графичку прилагодљивост кориснику уз нешто већу потрошњу хадверских ресурса (*KDE*, *Cinnamon*), затим компромисна решења који су балансирани однос графичке прилагодљивости и потрошње ресурса (*Xfce*, *Mate*), лака графичка окружења (*LXDE*. *Enlightenment*), графичка окружења која се заснивају само на менаџерима прозора (*Openbox*, *Fluxbox*) и још многа друга. Ово набрајање је управо ишло од захтевнијих ка мање захтевним графичким окружењима. Ако сте испробали неку *GNU/Linux* дистрибуцију и нисте задовољни брзином њеног рада, највероватније је за то кривац управо уграђено графичко окружење. Нека од ових окружења троше превише *RAM*-а, нека сувише оптерећују *GPU* (графичку процесорску јединицу – графички процесор на графичкој картици) и/или *CPU* (централну процесорску јединицу - процесор рачунара). Једно од решења

проблема са презахтевним графичким окружења је инсталација истог *GNU/Linux*-а са мање захтевним графичким окружењем. На пример, уместо *Ubuntu*-а са *Unity* графичким окружењем *Xubuntu* са *Xfce* графичким окружењем. Сам хардвер диктира идеално графичко окружење за тај систем. Ако сте корисник доброг хардвера који успешно може да користи било које графичко окружење, на избор идеалног окружења утичу други фактори. Напоменули бисмо да графичко окружење са собом повлачи и специфичне апликације које су потпуно прилагођене том графичком окружењу. Понекад су те апликације само са промењеним именом. Пример: управљач датотекама *Nautilus* (као *File Explorer* у *Windows*-у) се у *Cinnamon*-у зове *Nemo* а у *Mate – Caja*. У већини других окружења управљачи датотекама су потпуно другачије апликација, на пример у *KDE* управљачн датотекам је *Dolphin*, у *Xfce – Thunar* а у *LXDE – PCManFM*. Иако раде сви једнак посао, разлике нису само у „шминки", него и сваки од управљача има неке специфичне функције. Све то збуњује нове *Linux* кориснике, поготову што морају да се навикну на промењена имена апликација, па сад кад промене и *GNU/Linux* радно окружење и добију потпуно нова имена апликација то буде прилично збуњујуће, као да је у питању потпуно нови оперативни систем.

Управљач датотекама је само један од примера различитих апликација у различитим радним окружењима. Могуће је инсталирати апликације и из другог графичког окружења, али то повлачи доста зависности оригиналног графичког окружења што може да оптерети ресурсе хардвера нарочито ако имате мало простора на хард диску. Осим оптерећивања складишног простора на хард диску, инсталацијом зависности везаних за оригинално графичко окружење, могуће је повући и неке функције које оптерећују ресусе *RAM*-а и *CPU*-а што је корисник у старту хтео да избегне инсталирањем лакшег окружења. Зато треба избегавати мешање подразумеваних апликација из различитих радних окружења ако имате проблем ограничених ресурса.

Избор идеалне дистрибуције преко избора пакет менаџера

До сада смо бирали идеалну дистрибуцију условљени расположивим хардвером. Постоји још један критериј по којем можемо да бирамо идеалну дистрибуцију. То је избор према пакет менаџеру.

GNU/Linux дистрибуције и апликације се испоручују корисницима путем интернета у облику пакета изворног кода или путем пакета већ компајлираног бинарног кода. За разлику од *Windows* инсталационих програма, *GNU/Linux* пакети нису монолитни пакети који садрже све функције тог програма. Врло често поједине функције *GNU/Linux* апликација су у посебним пакетима, а и изглед апликације је у посебном пакету. Тако се обезбеђује да поједине функције могу да користе неке друге апликације, а и изглед се прилагођава изабраном графичком окружењу. Све то ствара проблем за ручну инсталацију нових програма на *GNU/Linux*-у.

Зато постоје пакет менаџери, чија је

улога да препознају захтев корисника за инсталацијом тачно одређене апликације, проналазак његових пакета у интернет ризницама дистрибуције, препознавање потребних међузависности са пакетима који нису директно везани за апликацију али су неопходни апликацији (нпр. изглед апликације у графичком окружењу – енг. *GUI*), препознавање који су од међузависних пакета већ инсталирани и које тек треба инсталирати, распакивање, компајлирање (ако пакети нису бинарни), инсталација, уклањање непотребног софтвера и међузависности које не користи неки други инсталирани програм и на крају, „апдејт" (енг. update) инсталираног софтвера.

Ако од кернела зависи уопште функционисање оперативног система, од графичког окружења зависи удобност рада, онда од пакет менаџера зависи једноставност одржавања система. Стога је прави избор пакет менаџера један од битних критеријума за избор *OS*-а.

Данас су напознатија три формата *GNU/Linux* пакета: *DEB* карактеристичан за *Debian* и његове деривате, *RPM* карактеристичан за *Red Hat* и његове деривате и обични *TXZ*, *TGZ* пакети који могу бити пакети изворног кода, али и пакети предкомпајлираног кода. Са овим форматима се „боре": *DEB – dpgk* и *APT*, *RPM – yum*, *TXZ – slackpkg*, *TAR – pacman*. Ово је само неколико најпознатијих пакет менаџера. Треба нагласити да немају сви ови менаџери све раније набројане функције. Неки се добро „боре" са међузависностима док други ни не покушавају да их решавају. Затим, неки од ових менаџера имају и своје апликације са графичким интерфејсом, као на пример *Synaptic* који је графичка апликација за *APT* или *Pamac* – графичка апликација за *pacman*. Ове графичке апликације много помажу, нарочито почетницима да лако нађу, инсталирају и одржавају свој софтвер. Искуснији корисници ће понекад радо жртвовати једноставност пакет менаџера зарад неке повећане сигурности, прецизности и стабилности софтвера одређене дистрибуције.

За крај епизоде

Тек смо загребали по површини проблема како изабрати идеални *GNU/Linux* оперативни систем. Поменули смо три најважнија критеријума, а томе бисмо могли додати и избор према начину инсталације система (графички или текст инасталер), па избор према врсти инсталационог диска (прости инсталациони диск или живи диск), па према начину надоградње система (системи са периодичним стабилним верзијама, системи са *rolling update*-ом...) и још много других критеријума. наша потрага ће се наставити. Вероватно ћемо у наредним епизодама овог серијала поменути и те друге битне критеријуме за избор идеалне дистрибуције.

Енкриптована електронска пошта

(3. део)

Аутор: Петар Симовић

Ова тема постаје све популарнија, па се тако сада уједињују *Lavabit* и *Silent Circle* и формирају заједничку алијансу под именом *Darkmail* (https://www.darkmail.info/). Појављују се и хардвери који ће целу енкрипцију радити аутоматски за вас (https://kinko.me/). Ту је и будући *webmail* клијент *Mailpile* (https://www.mailpile.is/) који је још у алфа фази тестирања, али га је могуће скинути код са *github*-а и испробати. Корисницима који су навикли да пошту проверавају из интернет претраживача, највероватније би највише одговарао неки програм у виду додатка претраживачу који је у исто време и менаџер њихових кључева. Два таква су сигурно *WebPG* (https://webpg.org/) и *Mailvelope* (https://www.mailvelope.com/), веома су интуитивни и погодни за почетнике (поред *Thunderbird*-а и *Clawsmail*-а). Нажалост, аутор овог текста их не препоручује јер умањују ниво сигурности и безбедности из простог разлога што у том случају морате да се поуздате да ни сам додатак претраживача (екстензија/плугин) ни претраживач неће да вас изневере и одају трећој страни вашу тајну, тј. да немају неки пропуст или неку рањивост која би нападачу одала ваш тајни кључ. __

Наравно, цео процес око кључева, шифровања и сертификата могуће је било одрадити преко команди из терминала, а програми попут *Thunderbird*-а и *Clawsmail*-а су само интерфејс за кориснике који још нису напредовали до чина хакера. Тако је, на пример, следећих шест команди све што вам је потребно да бисте већ описани процес из претходног дела одрадили из терминала за вежбу, ако имате још неки имејл налог вишка:

- Генерисање кључева:

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

- Извоз јавног кључа:

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

на екран или у датотеку ако на ову команду додате још

```
>>mypubkey.txt
```

- Извоз приватног кључа:

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

на екран или у датотеку ако на ову команду додате још

```
>>myprivkey.txt
```

- Генерисање сертификата за опозив кључева („*Revocation certificate*") уколико је тајност приватног кључа угрожена или сте једноставно изгубили уређај на коме сте га чували:

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

- Слање јавног кључа на сервер кључева:

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

где је *ID* уствари *ID* број вашег јавног кључа и треба да изгледа слично овом *ID*-у: *6382285E*.

- Претрага јавног кључа особе којој желите слати шифровану пошту на серверу јавних кључева:

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

Лоше стране *PGP*-а

Пре него што поменемо још неке занимљиве програме и пројекте, објаснићемо чему служи такозвани „сертификат за опозив" кључева који смо малопре споменули, а у прошлом броју смо саветовали да га направите и сачувате. Наиме, ако вас задеси несрећан случај да изгубите лаптоп на коме се налазио ваш приватни кључ, а претходно сте копију истог сачували и на некој другој спољној меморији заједно са овим сертификатом и тајном фразом, онда можете друге упозорити на могућу угроженост вашег тајног кључа користећи овај сертификат. Ако нападач (или лопов хакер) дође до вашег тајног кључа, онда може дешифровати све ваше претходне поруке. Употребом сертификата дајете до знања осталима који желе да са вама размењују приватну пошту (слањем опозваних кључева на сервер јавних кључева), да су кључеви неупотребљиви и да није

сигурно да их користе. У том случају, морате генерисати нове и послати их на сервер (више на http://goo.gl/IWoS4v). Команде су следеће:

- Генерисање серификата уколико то нисте раније урадили:

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

- Увоз сертификата за опозив кључева:

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

- Слање опозваних кључева на сервер - „*update*" кључева:

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

Ово је само један од, условно говорећи, недостатака садашњег начина функционисања *PGP*-а, тј. један тајни кључ откључава све предходне тајне шифроване поруке. Наравно, решење постоји употребом *PFS* (*Perfect Forward Secrecy*) протокола који спречава да се овакав сценарио одигра, јер се свака нова комуникација обавља новим паром кључева након које се кључеви за ту трансакцију одбацују и више никада не користе. Тако да ма ког кључа са нападач домогао, не може дешифровати њиме све пређашње комуникације зато што кључеви међу собом немају никакве везе. Наравно, ово је много лакше рећи него урадити зато што постоји велика разлика у архитектури тј. када је *PGP* настао 1991.године прошлог века, није још постојао *PFS* протокол, а сада се користи у *„instant messaging"* програмима. Ко користи овакав протокол, а ко не можете видети овде: http://goo.gl/GaiE1V.

Други овакав недостатак је познати проблем са дистрибуцијом јавних кључева и њихова верификација. Објаснићемо ово на примеру. Рецимо да на *Twitter*-у нађете особу (новинара) којој би послали шифрован мејл са поверљивим доказима против неког. На *Twitter*-у је та особа оставила свој *PGP* отисак (*Fingerprint*) или *ID* кључа, па је логично да поседује и пар кључева, а јавни кључ се највероватније налази на неком од сервера јавних кључева и можете га добити командом из терминала:

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

Осим што у неким верзијама *PGP*-а нема провере на вашем рачунару да ли се отисак (*fingerprint*) преузетог јавног кључа поклапа са оним који је тражен, овде морате да „верујете" серверу кључева да вам прослеђује прави кључ. Звучи помало параноично, зар не? Размислите још једном, осим што свако може стартовати свој сервер кључева и укључити га у постојећу мрежу, већ постојећи сервери нису и не морају бити нарочито безбедни јер не чувају никакве тајне, те постоји реална могућност да их неко хакује јер са већом популарношћу технологије долазе и већа опасност и већи ризик. Још се није овако нешто десило и не знамо шта би све нападач могао постићи и какву штету нанети, али је сигурно да не треба имати потуно поверење у информације са ових сервера и ваља их проверавати. На страну све то, мана је у самој архитектури система јер је делом централизована, а није дистрибуирана на саме кориснике, па корисници морају да верују серверу који

не могу да провере. Наиме, сервери јавних кључева међусобно синхронизују своје базе података кључева и ту су да нам олакшају претрагу кључева, али никако нису неизбежни. Тако да се овај проблем може решити тако што се организују јавна састајања (познатије као *Key-signing party*) и онда се физички размењују и потписују кључеви са особама које лично знате и на тај начин стварате своју „мрежу поверења" (*Web of Trust - WOT*). Иако је ово мало непрактично, има и својих добрих страна - упознавање нових људи и размена вештина.

Да не би било никакве забуне, *PGP* пружа **приватност** и **безбедност** шифрујући поруке тј. њихов садржај, **аутентичност** употребом хеш (*hash*) функција и дигиталног потписивања порука и уопштено сваке врсте датотека. Али никако **анонимност**, јер се хедери (*headers*) поруке тј. адреса пошиљаоца и примаоца не шифрују и за страног посматрача су доступни. Технички речено, *PGP* не сакрива метаподатке. Тако да ако користите *Gmail*, *Google* зна да сте јуче послали поруку „том и том" кориснику, тачно време слања, ИП адресу са које сте слали, али уколико сте шифровали поруку не знају сам њен садржај. Занимљиво је поменути да су *Yahoo* и *Google* увидели потенцијалну корист од „енкрипције" своје поште, па су најавили да ће догодине имплементирати исту у своје сервисе, а *Google* је већ поставио и код свог будућег софтвера слободног за преузимање са *Git*-а (више на: http://goo.gl/P33PN7, http://goo.gl/quc0cs, http://goo.gl/xIw23o, http://goo.gl/v3Zvoo). Ово није никаква револуционарна новина коју је „свемоћни" *Google* управо измислио, већ га примењују неко време мање познате компаније попут *Openmailbox*-а који има и занимљив једнократни мејл да не бисте морали сумњивим сајтовима да дајете праву адресу. Ствар је у томе што су ови гиганти увидели прилику да се убаце и на ово тржиште јер постаје све популарније, оно би им обезбедило позицију централих поузданих ауторитета. Ако се тога домогну, не верујемо да ће безбедност бити ништа боља пре свега зато што би под одређеним околностима могли да изводе текозване *MITM* (*Man-in-the-middle*) нападе и заобићу дешифровање порука. Више на http://goo.gl/AdlCXa и http://goo.gl/dh1C7e.

Светла будућност

На нашу срећу, није све баш тако црно и нове, боље идеје се рађају скоро свакодневно, поготово када смо на то принуђени и инспирисани *Snowden*-овим објављивањем. Из тих идеја се понекад и „скува" нека нова и боља апликација или програм, па погледајмо неке. Пре свега, једна од најновијих је свакако *Keybase* (https://keybase.io/), то је онлајн програм али такође се може њиме управљати из командне линије односно терминала. *Keybase* је на неки начин сервер јавних кључева, али умногоме олакшава баратање кључевима и сертификатима, њихову верификацију, као и тражење особе са којом желите да се дописујете. Оно што га издваја од обичног сервера кључева је повезивање ваших кључева са вашим налозима на *Reddit*-у, *Twitter*-у, *Hacker news* сајтом, *GitHub*-ом и вашом *bitcoin* адресом или вашим сајтом како би

други били што сигурнији да сте то ви, и да употребом ваших кључева заправо пишу вама. Откривање и повезивање ваших идентитета би иначе било проблем, али пошто *PGP* ионако не пружа анонимност онда то нема штетних ефеката. Програм је још увек у алфа степену развоја и мали број је позван на тестирање. Мада можете да се придружите пројекту и резервишете име на сајту и бићете тек неки 1600-ти, али ако сте озбиљно заинтересовани за овај пројекат администратори су неким тестерима подарили позивнице за будуће кориснике који су озбиљни и заиста желе да допринесу програму. Аутор овог теста има пет позивница, тако да ће пет најбржих који пошаљу мејл са насловом *Keybase* на „*keybase@openmailbox.org*" добити мејл са линком за регистрацију. Мејл ми је неопходан, јер морам унети мејл особе коју желим да позовем.

Ту су и напреднији пројекти попут *Pond*-а (http://goo.gl/axqNwx), који није ни класични *mail* протокол ни *IM* (*instant messaging*) апликација, већ једноставно асинхрони *P2P* програм који је у дописивање успео да убаци и *PFS*, а поруке се аутоматски бришу након недељу дана. Ово је и даље у развоју и немојте овај програм још користити ни за шта озбиљно. Веома сличан пројекат је и *Flowingmail* (https://goo.gl/) који иако није остварио жељену суму као *start-up* на *Indiegogo* сајту (http://goo.gl/psWb6m) пре скоро годину дана, то га није обесхрабрило и нису одустали од своје замисли *P2P* децентализованог, сигурног мејл протокола, где се адресе рачунара на који шаљете рачунају по јавном кључу тако да не могу бити фалсификоване.

Оно што предходна два пројекта и *Darksmail* покушавају да ураде, јесте да нам поред наведених добрих особина *PGP*-а, пруже и ону једну која недостаје - анонимност. *Darkmail* то постиже *end-to-end* енкрипцијом, чиме сакрива све метаподатке и постиже анонимност из перспективе страног посматрача.

У следећем делу ћемо се упознати са анонимним *email* системима који су настали још давних деведесетих, и после неколико фаза развоја и унапређења у употреби су и данас.

MARY Text To Speech

Аутори: Дејан Чугаљ и Дејан Маглов

Недељно послеподне некако је резервисано за досаду и убијање времена необавезним стварима као што је необавезно „сурфовавње" интернетом. Како може да се заврши и куда може да одведе то досадно недељно послеподне, никада се не зна. Електронска пошта која стиже недељом или је спам, или су обавештења на која сте се претплатили. Регистрацијом на разне блогове, *web IT* часописе, претплатили сте се на потенцијална *inbox* обавештења која пристижу недељом. Ако сте притом програмер - *freelancer*, *blog-freelancer*... сигурни смо да се баш у том истом *inbox*-у недељом појављују и обавештења како сте баш ви тај који би могао да добије одређен посао. Пословна обавештења су корисна, али досадна. Понекад се и у том пословном садржају крије потенцијална „иницијална каписла" која би могла да „истисне" чланак у ЛиБРЕ! часопису. Једна од таквих случајности је повод писања овог чланка, а уско је везан за „*multilingual text-to-speech synthesis*", преведено, мулти-лингвистичко окружење (енг. *framework*) за имплементацију *TTS* (енг. скраћеница од *Text-to-speech* - текст у говор) синтезе. На први поглед ништа компликовано: откуцаш текст и неки „робот – синтетички глас" (енг. *synthesis*) то исто изговори. Наиме, проблем је „мало" већи него што се чини. Сваки језик, свака фонетска разлика језика, уско је везана за фино подешавање алгоритма имплементације *TTS*-а у којој се језичка синтеза диверзибилности разликује сама по себи, као и свака фонетска дисторзија која чини исти јединственим самом говорном подручију (енг. *slang*).

У којој мери је ова технологија значајна, веома је дискутабилно, само они који имају проблема са видом су ти који могу објективно да кажу колико је то значајно и корисно, док остали ... ЛиБРЕ! се ограничио на слободан софтвер. Назив „слободан софтвер" не значи само слободно коришћење него слободан развој, унапређење и модификовање софтвера. *TTS* софтвер управо показује једну од главних мана оваквог развоја софтвера, барем у Србији. Софтвер који зависи од

локализације подразумева већу ангажованост локалне заједнице на њеном развоју, јер не можемо да очекујемо од странаца да развијају и нашу локализацију. Чињеница је да скоро сви пројекти слободног софтвера, код нас настају као последица задовољавања личних потреба самих програмера. У оваквим условима, развој *TTS*-а можемо очекивати само од програмера који су лично заинтересовани за тај софтвер. То значи да се круг могућих развијатеља своди само на слабовиде програмере или на програмере који имају неког свог који је слабовид, а то је права реткост у Србији. У Србији је развој слободног софтвера, за неког другог, без личне сатисфакције мисаона именица. Чак ни хуманитарни разлози нису довољан мотив или се још нико није сетио, а то у овом случају ускраћује слободу читавој једној групацији људи – људима оштећеног вида и слабовидим.

Зато ЛиБРЕ! тим жели мало да „заталаса", подсети да и хуманитарни разлози могу да буду мотив за развој слободног софтвера.

Тренутно, једна од најбољих имплементација *TTS*-а поседује српска компанија „*AlfaNum*" http://www.alfanum.co.rs/, која наравно, *TTS* услуге и наплаћује. Кôд су затворили и држе га у својим „сигурним рукама". Горе наведени линк то и демонстрира. (Имао сам прилику да видим форум пост управо једног од *CEO* „*AlfaNum*"-а који тврди да је много људи укључено у саму реализацију пројекта, те да мора да се наплаћује. Нисмо сигурни да ли линк ка форуму ради, али пробајте: http://www.elitesecurity.org/t38544-SERBIAN-Speech-engines)

Мислимо да би то могло да се промени, јер управо овде представљамо пројекат отвореног кода, који би тај проблем у Србији могао да преусмери на пут слободног софтвера и слободе.

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

Онако, на први „програмерски" поглед (енг. *first view*), документација је више него добра: https://github.com/marytts/marytts/wiki, тако да то у целом неком будућем пројекту не би био проблем. Наиме, мало да „закочимо" и да објаснимо шта је овде стваран проблем. Развој текст-у-говор (енг. *TTS*) система је еквивалентан развоју система где особа, која зна да чита и изговара правилно, чита текст наглас. Иако сви мислимо да је то лако, и да смо у убеђењу да знамо то да радимо, чињенично стање је мало другачије. Другим речима, скоро 80% грађана не чита по лингвистичким правилима и то представља један од првих проблема у развоју *TTS*-а. Тако да би један од првих модела у имплементацији *TTS*-а за српски језик, био знање како нешто прочитати (енг. *knowing how to read*). Без тог лингвистичког знања, имплицитне радње алгоритма би по аутоматизму прерасле у експлицитно, а самим тим и сложеност истог би експоненцијално порасла. Како би то свели на неки минимум, у *TTS*-у постоје правила која се деле на:

1. прелом реченица
2. нормализација текста

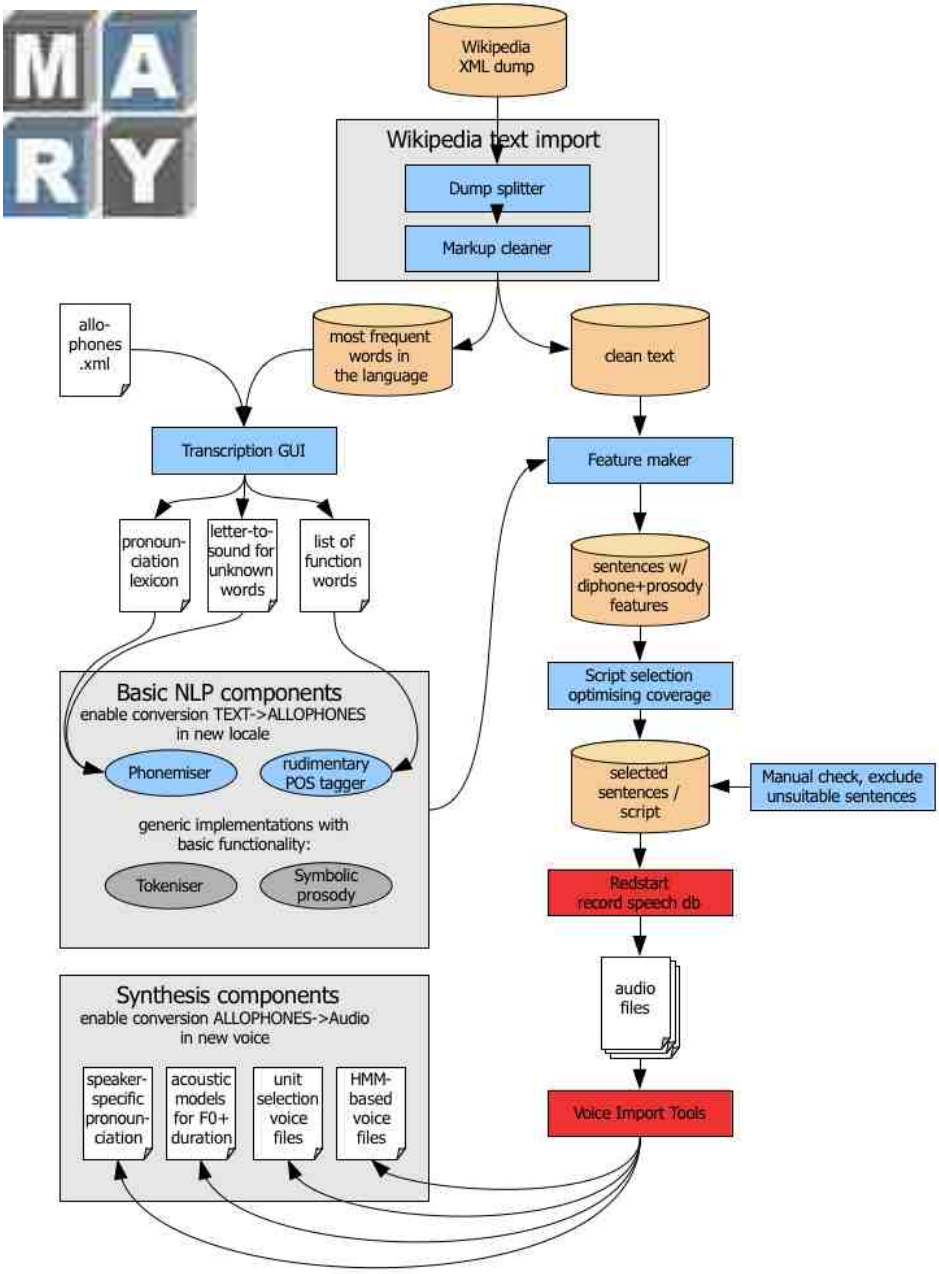
Прелом реченица у фонетском контексту је један од битнијих делова развоја *TTS*-а, јер се наставак изговора реченице веома разликује после интерпункцијских знакова, као што су зарези, знакови узвика, тачке... па чак ни ту нема правила, нпр. „др Мирко Мирковић", није исто када се изговара у средини или на крају реченице. Нормализација текста је везана за раздвајање речи из реченица, тј. другим речима требало би да постоји база изговорених речи коју користи *TTS* систем за сам изговор и то у конјукцији са преломом реченица и правилом интерпункцијских знакова српског језика. Само малу смерницу о комплексности можете да погледате на линку: http:www.srpskijezik.rs/gramatika/podela-reci-na-slogove.

Само смо загребали врх леденог брега, комплексност имплементације *TTC* система. Покушај индивидуалног имплементирања би био ништа мање до програмерске велике авантуре, али, срећом, не морамо да почнемо из почетка, јер пројекат „*MARY TTS*", који је отвореног кода (енг. *open source*) нам даје огромну одскочну даску.

Прегледом *API* целог система „*MARY TTS*", стиче се утисак да не би требало да буде тешко имплементирати подршку за нови језик. Никако нећемо рећи да нема посла, али са сигурношћу можемо да потврдимо да је могуће, и то бесплатно. Надамо се да смо макар мало заголицали машту неким будућим програмерима, којим би „*MARY TTS*" пројекат био одскочна даска. Један такав пројекат мотивисан хуманитарним разлозима био би значајан за српски *FLOSS* покрет.

Корисни линкови:

1. Пројекат: https://github.com/marytts/marytts
2. Подршка новом језику: https://github.com/marytts/marytts/wiki/New-Language-Support
3. Идеје: 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

Аутор: Стефан Ножинић

Android је већ одавно заузео огроман проценат тржишта мобилних уређаја и много је апликација развијено за њега. Ако сте се икада бавили развојем апликација за *Android* уређаје, онда сте највероватније за то користили *Eclipse* и *ADT* додатак за исти, који вам је то олакшао. Можемо се запитати како је *Google* толико форсирао *Eclipse*, уместо да је направио своје окружење. После толико времена, засебно окружење је стигло.

Android Studio у тренутку писања овог чланка је још увек у бета фази, али се доста добро развија и већ је употребљив. Базиран је на *IntelliJ IDEA*. Он пружа нове могућности које *ADT* додатак за *Eclipse* није пружио и постаће званично развојно окружење чим буде готова прва стабилна верзија.

Неке од карактеристика су:

1. Систем изградње базиран на *Gradle*-у уместо на *ant*-у.
2. Генерисање више варијанти *APK* архива у зависности од уређаја.
3. Уредник графичког интерфејса са подршком измене тема.
4. Алати за проверу компатибилности са различитим верзијама.
5. Алати за мерење перформанси.
6. Потписивање апликација.
7. ...

Миграција на *Android Studio*

Као што смо већ поменули, *Android Studio* је базиран на систему изградње *Gradle*, а не *ant*. То значи да је потребно пребацити наше пројекте на нови систем. *ADT* додатак за *Eclipse* у верзијама после 2.2 има опцију експортовања пројекта са новим *Gradle* системом, па је само потребно да урадимо надоградњу додатка ако већ немамо новију верзију. Битно је напоменути да ће *Android Studio* радити и са старим *ant* системом, али се препоручује прелазак на нови систем како бисмо били у могућности да користимо неке додатне и напредније опције у будућности.

Употреба

Android Studio нам омогућава креирање апликације за различите типове уређаја. Овако је могуће прављење апликација за телефон, таблет, ТВ, *Google* наочаре и *Google Wear*. Чаробњак за прављење новог пројекта нуди избор за који уређај желимо да правимо нашу апликацију, а тиме ће нас упитати коју верзију *API*-а желимо да користимо. Поред ових могућности, у чаробњаку имамо и опције да изаберемо одређени *Activity*, и подесимо га по нашој вољи.

После креирања пројекта, потребно је приметити да је структура директоријума мало другачија, него до сада у

Eclipse-у. За ово је „крив“ *Gradle* систем изградње, који се користи у *Android Studio*-у. У принципу, све функционише као и до сада, само су неки директоријуми премештени у *„src/“*, па је тако лакше сналажење. Ово ће пуно поједноставити рад на пројекту, у смислу да програмери неће бити збуњени које датотеке могу да мењају, а који се мењају/пишу приликом изградње извршних датотека.

Из самог програма, могуће је директно креирање виртуелних уређаја, који служе за покретање апликације у емулатору у случају да не желимо да је покренемо на физичком уређају. Ово је корисно у случајевима када немамо одређену верзију *Android*-а на физичком уређају, а желимо да тестирамо рад наше апликације на тој верзији.

Додавање нових датотека је постало заиста једноставно. Потребно је само кликнути на одређени директоријум у пројекту, и притиснути комбинацију тастера *ALT + INSERT*, и *Android Studio* ће се постарати да покрене чаробњака за креирање одређеног типа датотека. На пример, уколико селектујемо *„layout/“* директоријум, *Android Studio* ће понудити креирање новог *Activity*-ја.

Уклањање грешака је могуће позивањем *adb logcat* из самог окружења, и тиме добијамо лог поруке које бележе апликације на уређају или емулатору.

Закључак

Право је чудо да *Google* није раније урадио овакву платформу, с обзиром на то да развој *Android* апликација постаје све учесталији и да су се многи проблеми *Eclipse*-а са додатком почели назирати када је реч о комплекснијим пројектима. *Android Studio* нуди једноставност коришћења, а опет много више могућности него претходни подржани приступ развоја (*Eclipse* са додатком за развој *Android* апликација).

Raspberry Pi B++ & HummingBoard

Аутор: Симовић Петар

Raspberry Pi B+

Сви смо до сада добро упознати са *Raspberry Pi* моделима *A* и *B* и вероватно смо купили неки да се забављамо код куће, или да нам служи као музички сервер или *proxy*, мада списак његових намена не би могао да стане у овај чланак (http://goo.gl/5UqAkC). Иако је на тржишту већ две године, неки би помислили да је мало застарео, али се пројекат итекако развија и прилагођава тржишту и потребама купаца. Тако је на пример у априлу ове године изашла и нова верзија намењена пословним и индустријским корисницима звана *Raspberry Pi Compute module*. „*Compute module*" је уствари само плочица која се повезује на *Compute Module IO Board* преко стандардног улаза за *DDR2 SO-DIMM* меморију (као оне у лаптоп рачунарима). Више на http://goo.gl/SfbKJr.

То није све, после ове трансформације, модел *B* је добио мала унапређења у виду *B+* модела на основу „*feedback*-а" (енг. повратна информација) самих корисника. Најважнија унапређења се односе на повећан број *USB* 2.0 портова са два (*B* модел) на четири (*B+* модел), онда *MicroSD* подршка уместо стандардне *SD*

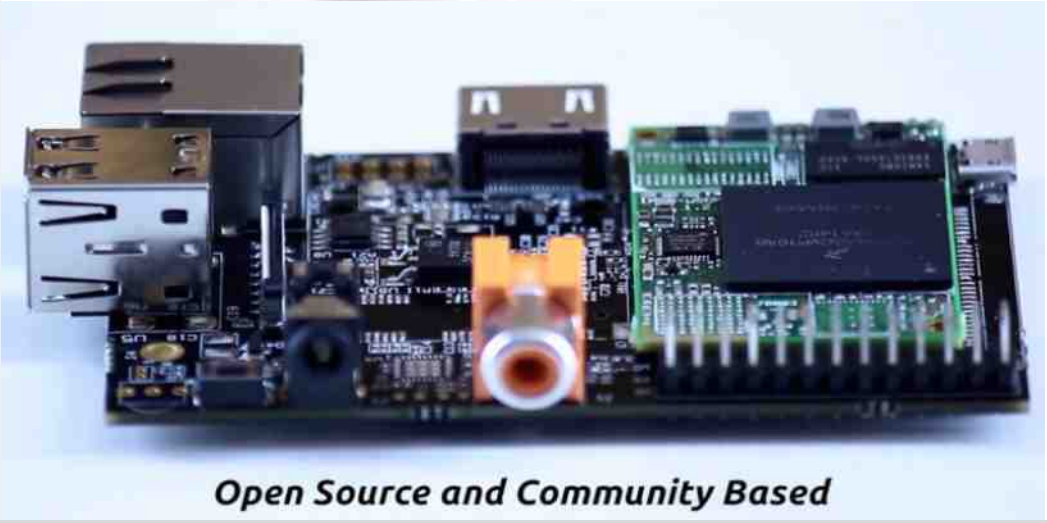
картице и повећан је број пинова *GPIO* (*General Purpose I/O*) са двадесет шест (*B* модел) на четрдесет (*B+* модел), с тим што је двадесет шест основних пинова задржано, само је додато четрнаест нових са новим могућностима. Између осталог, смањена је потрошња струје са 3.5 W (*B* модел) на 2.5W (*B+* модел), или од 0.5W до 1W зависно од употребе у односу на *B* модел или 600 *mA* у односу на 750 *mA*. Аудио и видео конектори са *B* модела су спојени у један конектор на супротној страни. Оно што се није променило, јесте 10/100 *lan* порт, меморија је остала на 512*MB* и *ARMv6* процесор је остао исти (700 *MHz* - 1.1 *GHzoverclock*) са чипсетом *Broadcom BCM2835*, као и *Micro USB Power supply* и *HDMI* конектори. Важно је напоменути да се цена није променила и да *B+* модел кошта колико и *B* модел, али пластичне кутијице које су прављене за *B* модел не одговарају *B+* моделу. Видео можете погледати на http://goo.gl/7u7WWo и http://goo.gl/R21wUG .

Најинтересантније је да је *Raspberry 2* најављен за 2017. годину, а да ће се до тада развој фокусирати на унапређење софтвера.

HummingBoard

Можда мислите да се *Raspberry Pi* мало размазио и одомаћио јер већ две године држи монопол рачунара мале потрошње величине кредитних картица, али у трку улазе и нови играчи са својим моделима за такмичење. Такав је на пример и нови *HummingBoard* компаније *SolidRun* који се појавио на тржиште мало пре новог *B+* модела *Raspberry Pi*, па је *B+* донекле одговор на *HummingBoard*. На први поглед изгледа исто као и обичан *Raspberry Pi* (*RPi*) *B* модел, а то је и била намера, пошто може да стане у било које кућиште намењено за *RPi*. Оно што издваја *HB* од *RPi*-а, јесу јачи процесор

1*GHz ARMv7* насупрот 700 *MHz ARMv6* код *RPi*-ја, затим подршка за више оперативних система и условно већа меморија пошто цене модела иду од 45$ до 100$. Такође, ту је подршка за *mSATA*, *PCIe mini* и *LVDS display*, а процесор је (мобилан) одвојен на засебну плочицу која се прикључује основној, тако да се лако може заменити или унапредити у јачи модел. Зависно од модела, меморија „иде" од 512*MB* до 1*GB*, процесорска језгра од 1 до 2, тако да постају већ озбиљнији мали рачунари. Детаљније на http://goo.gl/Z7AgHJ и видео на http://goo.gl/nN7W14 .



ЛиБРЕ!

Часопис о слободном софтверу

на

BALCCON 2k14