Фебруар 2015. **Број 33**

ЛиБРЕ!

Часопис о слободном софтверу

Изгубљени хероји „Блечли парка”

ЈОШ ИЗДВАЈАМО

bgfx - Cross-platform rendering library

То су само метаподаци?

Реч уредника

Транскрипција

Приметили сте да је већина страних речи у претходном броју часописа писана прилагођено. Одлучили смо да почнемо са транскрипцијом. Одлуци је претходило разматрање о томе да ли је транскрипција корисна и могућа, и како би наши читаоци реаговали.

У нашем језику има много речи страног порекла. Ми смо их временом прихватили и прилагодили их. Како време протиче, стичемо утисак да су поједине речи одувек део нашег језика. Следеће генерације не морају имати свест о томе да је нека реч страног порекла. Примећујете да страна реч прихватањем постаје реч страног порекла, а можемо их сретати једнако често као домаће речи. Прихватање речи - шта се под тим подразумева? Најпре се реч устали у народу и прилагоди се нашем изговору. Додељујемо јој род, број и падеж. Пишемо је нашим писмом. Да ли се транскрипцијом реч удаљава од изворног значења? Одговор је одричан. Ајнштајн (нем. *Einstein*) је познати физичар, Лавоазје (фр. *Lavoisier*) је хемичар, а свима су нам познате Езопове басне (грч. *Αἴσωπος — Аисо̄пос*). Састављачи наших уџбеника одувек су вршили транскрипцију страних речи. Подразумева се да су макар у фусноти наведени значење и изворна графија. Транскрипција је саставни део нашег језика.

С друге стране, противници транскрипције су мишљења да је најбитније оптичко препознавање страних речи, а да би транскрипција могла да збуни читаоца у смислу да не зна о чему се ради, иако се раније сусретао с истим појмом али у изворно написаном облику. Да је могућност збуњивања мала указује вишедеценијска пракса у нашој научној литератури да је страна реч код првог спомињања написана изворним писмом. Транскрипција није пуко пресловљавање с једног писма на други, већ подразумева адекватно преношење звукова и њихово

записивање.

ЛиБРЕ! је стране речи до 32. броја часописа, чак и у текстовима на ћирилици, писао изворним писмом. Масивно уплитање латинице у ћирилицу је непрактично, а такав текст је непрегледан. Транскрибован облик речи нам показује како се оне читају, те нам отклања недоумице. Добар пример је име познатог графичког окружења Мате. Реч потиче из шпанског језика те се чита Мате, а не Мејт. Мате је биљка која расте у Јужној Америци од које се справља чај. ЛиБРЕ! ће следити Правопис и препоруке наших познатих филолога те ће већину устаљених речи писати у транскрибованом облику. Код првог писања речи у загради ће се навести изворни облик да би се читаоцу омогућила даља претрага интернета о појму. Кад би транскрипција битно отежала препознавање неког садржаја (као, например, код акронима), такве речи писаћемо надаље у изворном облику уз повремено писање препоруке како се реч чита. Мај-ес-кју-ел би било непрегледно и исувише дугачко, стога пишемо *MySQL*. Исто важи за мање познате називе програма и за команде унутар програма.

Од изласка прошлог броја нисмо добили ниједну критику на рачун транскрипције. То бисмо могли да протумачимо као знак да читаоци немају замерку у вези са транскрипцијом. Добили смо одговоре на питања постављена у Речи уредника из претходног броја. Овом приликом вам се захваљујемо на одговорима. Сви ваши предлози су добродошли и разматрају се на редовним састанцима. И даље очекујемо ваше критике и предлоге, које нам можете послати на већ познату адресу електронске поште libre [at] lugons [dot] org

До следећег броја,

ЛиБРЕ! тим.

Садржај

Моћ слободног софтвера

Број: 33
Периодика излажења: месечник

Извршни уредник: Стефан Ножинић
Главни лектор:
Александар Божиновић

Лектура:
Јелена Мунћан Сашка Спишјак
Милена Беран Милана Војновић
Адмир Халилкановић

Графичка обрада:
Дејан Маглов Иван Радељић

Дизајн: White Circle Creative Team

Аутори у овом броју:
Ненад Марјановић Дејан Чугаљ
Никола Харди Бранимир Караџић
Криптопанк

Остали сарадници у овом броју:
Марко Новаковић Михајло Богдановић

Почасни чланови редакције:
Жељко Попивода Жељко Шарић
Владимир Попадић
Александар Станисављевић

Контакт:
IRC: #floss-magazin на irc.freenode.net

Е-пошта: libre@lugons.org

Вести

21. јануар 2015.

Каноникал покренуо верзију Убунту Кора за Интернет ствари

Каноникал, компанија задужена за развој Убунту Линукс дистрибуције, покренула је своју верзију Убунту Кора (*Core*) за Интернет ствари (*Internet of Things*).

Користан линк: http://t.co/tN3VmZe5PK

---

26. јануар 2015.

Масовно надгледање прети људским правима

У европском извештају масовно шпијунирање прети дигиталној сигурности и људским правима.

Користан линк: http://t.co/9gcrmXw44q

---

27. јануар 2015.

Јутјуб користи *HTML5* као подразумевани начин репродукције

Јутјуб је заменио Флеш технологију новијом, отворенијом и напреднијом технологијом - *HTML5* која ће се користити као подразумевани начин пуштања видео садржаја.

Користан линк: http://t.co/zBpJh8Op6x

---

28. јануар 2015.

GHOST - безбедносни проблем

GHOST - нови безбедносни проблем у ГНУ-овој *C* библиотеци који доводи у опасност све апликације које користе функције *gethostbyname* и *gethostbyname2* и омогућава удаљеном нападачу да добије дозволе које има корисник који извршава ту апликацију.

Користан линк: http://t.co/GDYKsH2nOo

---

1. фебруар 2015.

Пајрат Беј се вратио

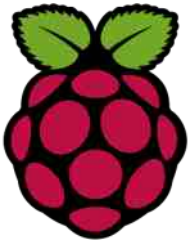
Најпопуларнији торент сајт Пајрат Беј (*The Pirate Bay*) се поново вратио после двомесечног периода када је био недоступан.

Користан линк: http://t.co/rSAPCVXWJi

---

2. фебруар 2015.

Расбери-Пај 2 је изашао



Доступан је у продаји Расбери-Пај 2 (*Raspberry PI 2*) са 4 процесорска језгра.

Користан линк: http://t.co/RGlOdoeioc

Вести

3. фебруар 2015.

Америчка војска отвара свој програм за детекцију напада

Америчка војска је одлучила да свој програм за детекцију напада постави на Гитхаб и тиме омогући целом свету да користе софтвер који су они користили за детекцију напада на мрежу министарства одбране. Они овим потезом очекују повратну информацију од људи који не раде у влади и надају се унапређењу свог програма.

Користан линк: http://t.co/v3PYlKF2sz

6. фебруар 2015.

GPG у опасности

Вернер Кох (*Werner Koch*), програмер који је задужен за развој *GPG*-а (Ји-Пи-Ји, engl. GNU Privacy Guard), софтвера који сви користимо за енкрипцију како наше електронске поште тако и осталих битних података, полако остаје без новчаних средстава. Позивају се сви заинтересовани и они који су у могућности да пошаљу донацију у виду новца.

Користан линк: http://t.co/jGac5emwPX

9. фебруар 2015.

Ардуино развојно окружење - ново издање

Изашло је ново развојно окружење за Ардуино у верзији 1.6

Користан линк: http://t.co/4A0pFcIsVf

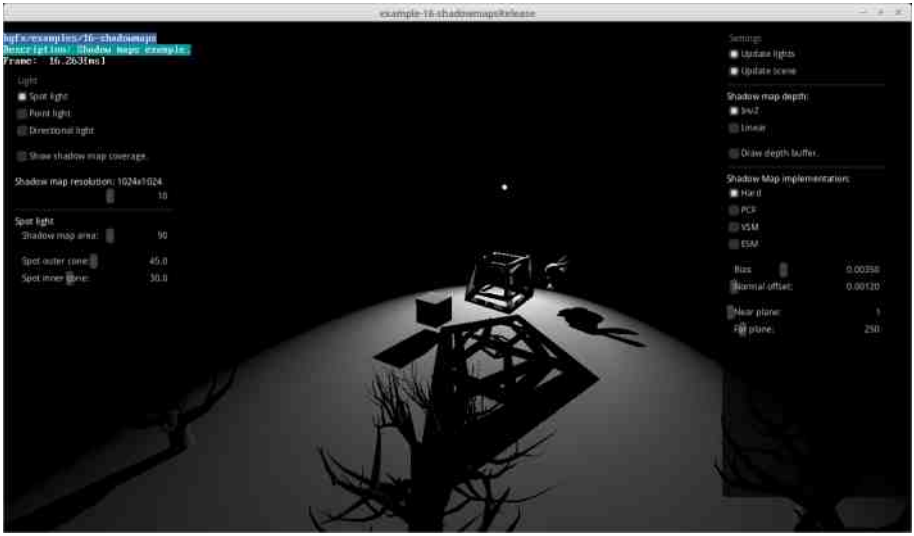
bgfx - Cross-platform rendering library

Аутор: Бранимир Караџић

bgfx је рендеринг библиотека отвореног кода са подршком за више различитих графичких *API*-ја [1] (енг. *Application Programming Interface*) и оперативних система [2]. Библиотека служи за једноставнији приступ графици и развој игара на више платформи. Највећи проблем са компјутерском графиком је што постоји више различитих *API*-ја који када се користе директно онемогућавају портовање игара на друге платформе. Правећи игру која приступа само *bgfx* библиотеци а не директно *API*-у, аутор игре омогућава себи да ту игру на једноставан начин може пребацити на било који подржан оперативни систем, односно платформу.

Представљамо

У почетку *bgfx* је био само хоби пројекат и зато има немаштовито име. Била је то само једна од многих библиотека које сам направио за прављење игара. Када сам радио на једној игри, постојала је потреба да се игра пребаци са старог *DX9* (енг. *DirectX 9*) рендеринга на *OpenGL* (енгл. Опен-џи-ел), па сам том приликом извукао део моје хоби библиотеке за рендеровање и запаковао је као целину. После успешне примене ове библиотеке у мојим пројектима, одлучио сам да је објавим као пројекат отвореног кода на Гитхабу (енг. *Github*) под лиценцом BSD 2-clause . По мом мишљењу, овај део рендеринга је требало већ одавно да буде решен проблем, као и да буде могуће да се игре без веће муке издају на било којој платформи по жељи аутора. Намењена је превасходно мањим ауторима који желе да имају потпуну контролу над својим кодом, без обзира на платформу на којој желе да објаве игру. Код оваквих комерцијалних производа увек постоје нека ограничења, а ако фирме изгубе интерес за производом, могу оставити ауторе на цедилу, без подршке и одржавања. Са друге стране, пројекти отвореног кода омогућују сарадњу аутора и унапређење функционалности пројекта на задовољство свих.



Употребом *bgfx* библиотеке корисник може да занемари конкретну имплементацију различитих технологија као што су *OpenGL*, *OpenGL ES* и *Direct3D*, без компромиса са перформансама. Ова библиотека није танка апстракција или емулатор неког постојећег *API*-ја, типа Гугловог Енгл (енг.

ANGLE) пројекта који емулира *OpenGL ES* преко *Direct3D*-а, или Валвов ТуЏи-Ел (енг. *ToGL*), који емулира *Direct3D* 9 преко *OpenGL*-а, него засебан *API* вишег нивоа. Иако је у прошлости било сличних пројеката, сви они су углавном имали претензију да буду енџин (*engine*) или фрејмворк који би творци игара користили као такав. Такав приступ доноси доста компликација и веома је тешко такав пројекат интегрисати са постојећим кодом. Из овог разлога *bgfx* пројекат нема претензије да буде цео енџин или фрејмворк, него само библиотека коју свако може лако да интегрише у сопствени пројекат.

bgfx библиотека се тренутно користи у неколико комерцијалних игара, али има и доста интересовања за пројекат као основе за прављење корисничких апликација, па чак и алата за програмере.

На пројекту је до сада радило око десет људи. Углавном су додавали подршку за нове платформе, примере, или су повезивали библиотеку са другим језицима. Што се будуће сарадње на овом пројекту тиче, све је добродошло, чак и сâмо коришћење библиотеке, а захтеви за увођење нових могућности унапређују саму библиотеку.

Представљамо

[1] Подржава графичке *API*-је:

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

[2] Подржава платформе:

- Андроид *(14+, ARM, x86, MIPS)*
- *asm.js/Emscripten (1.25.0)*
- *iOS*
- Линукс *(x86, x86_64)*
- Нејтиве клајент *(Native Client: PPAPI 37+, ARM, x86, x86_64, PNaCl)*
- ОС Икс *(OSX 10.9)*
- Расбери-пај *(RaspberryPi - ARM)*
- Виндоуз *(Икс-пе, Виста, 7, 8, 10)*
- Вин-ар-ти *(WinRT, WinPhone 8.0+)*

Представљамо

Аутор: Стефан Ножинић

Увод

rEFInd је софтверска апликација за *EFI* (*Extensible Firmware Interface*) системе која омогућава избор више оперативних система на једноставан начин. Ово је само управник бута (*boot*) и у овом броју вам представљамо о каквом пројекту је реч и како се може применити.

Шта је *(U)EFI*?

Пре него што представимо сам пројекат, требало би прво да скренемо пажњу и детаљније да представимо проблематику. *EFI* је настао као замена за досадашњи БИОС (*Basic Input/Output System*) и развила га је компанија Интел. Овај приступ је касније замењен *UEFI* стандардом. *EFI* технологија има предност у случајевима као што су:

- Покретање система на дисковима преко 2*TB* меморије коришћењем *GPT* технологије.
- Независност од специфичне архитектуре процесора.
- Модуларан дизајн.
- Добро окружење пре учитавања оперативног система што подразумева и мрежну подршку.

Технологија омогућава употребу 32-битних и 64-битних адреса на процесорима који то подржавају. Раније, БИОС технологија то није омогућила, већ се увек користила 16-битна имплементација из историјских разлога.

Партициона шема

EFI технологија има другачију шему партиција. Док је стари приступ подржавао само *MBR* (*Master Boot Record*) шему, нова технологија подржава *GPT* шему која омогућава већи број примарних партиција. Са старим приступом тај број био је ограничен на четири примарне партиције, од којих је свака морала да има капацитет мањи од 2*TB*.

Апликације

UEFI подржава израду апликација које би се покретале независно од оперативног система. Ове апликације се могу накнадно инсталирати. Пример таквих апликација су покретачи (*loader*) оперативних система као и управници бутовања.

EFI системска партиција

EFI подржава посебну партицију за смештање разних потребних информација као што су потребни подаци за руковање рачунаром и специфичне апликације. Ова партиција је најчешће *FAT32* систем датотека.

Представљамо

Бут менаџер и бут лоудер - има разлике

Од популаризације *GRUB* управника многи корисници су остали без објашњења каква је разлика између бут лоудера (*boot loader*) и бут менаџера (*boot manager*). Ово се дешава јер *GRUB* пружа обе услуге у склопу једног програма, па се тиме теже види разлика између ова два појма. Бут менаџер је управник помоћу којег је могуће изабрати оперативни систем за покретање. Он не зна ништа специјално о оперативном систему, па тиме не зна ни како да га покрене. Он само зна да треба да покрене бут лоудер за тај систем. Када је систем изабран, покреће се специфичан бут лоудер за тај систем који је задужен за даљи процес покретања кернела. За разлику од *GRUB* управника, *rEFInd* пружа само услуге бут менаџера, што у почетку може звучати поражавајуће за овај пројекат. Када узмемо у обзир да већина система данас има бут лоудер у склопу кернела (укључујући и Линукс од новијих издања) овај проблем је превазиђен.

Инсталација

rEFInd постоји као *.deb* и као *.rpm* пакет, али постоји и запакован као *.zip* архива. Захваљујући томе, пакет је могуће инсталирати употребом *dpkg* команде на

дистрибуцијама базираним на Дебијану као и употребом *rpm* команде на дистрибуцијама које користе тај начин паковања пакета. За Убунту постоји и *PPA* репозиторијум па, *rEFInd* може бити инсталиран на следећи начин:

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

Независно од тога који начин од горе описаних користите, *rEFInd* ће аутоматски бити пребачен на *ESP* и бити постављен као подразумевани управник.

Конфигурација

rEFInd би требало да вам пружи задовољавајућу функционалност без додатне конфигурације после инсталационог процеса. Он ће аутоматски детектовати лоудере оперативних система коришћењем претраге по свим партицијама које препозна. У случају да сте напреднији корисник и да желите да модификујете понашање *rEFInd*-а, то можете урадити изменом његове конфигурационе датотеке. Постоји глобална конфигурација и конфигурација за сваки систем посебно. Овај текст би постао превише дугачак када бисмо улазили у детаље саме конфигурације па вам зато препоручујемо, ако имате потребу за додатном конфигурацијом, да посетите документацију пројекта или, ако имате неке недоумице, да контактирате с нама преко наше е-поште.

За крај

Желели бисмо да овај текст закључимо оценом да је овај пројекат доста занимљив, да пружа задовољавајућу функционалност и да омогућава корисницима који користе овај систем покретања лако покретање својих оперативних система. Још једном желимо да вам скренемо пажњу да нам се обратите и укажете ако смо нешто важно пропустили да поменемо или ако нам се негде поткрала грешка.

Корисни линк: http://www.rodsbooks.com/refind/

Како да...?

Увод у програмски језик *C*
Рад са стринговима (9. део)

Аутор: Никола Харди

Опште је познато да су рачунари добри у раду са бројевима. Међутим, временом су се и потребе корисника и алати за програмирање рачунара развијали, и рад са другачијим типовима података је постао уобичајен и знатно једноставнији него раније. Сви смо имали прилику да радимо са програмима за обраду фотографија, видео материјала, или звучних записа. Мултимедија је ипак тема која превазилази један уводни курс о *C*-у, али до краја овог серијала ћемо се дотаћи и тих тема. За почетак, осврнућемо се на рад са текстуалним подацима на *C* начин.

Стрингови на *C* начин

Дефиниција текстуалног податка у сфери рачунарства се углавном своди на то да је текст низ карактера, а карактери су знаковни симболи. Иако имамо могућност да радимо са текстом или мултимедијом, рачунари ипак разумеју само бројеве па морамо тако да им представимо и текст. Постоје разни стандарди како се поједини карактери представљају у меморији рачунара, између осталог најпознатији су *ASCII*, *EBCDIC*, *UTF* итд. *EBCDIC* је стандард који је присутан у Епловом (енг. *Apple*) свету и више нема толико важну улогу. *ASCII* је стандард који је врло присутан и данас. *UTF* је настао као проширење *ASCII* стандарда, јавља се у више варијанти и уводи подршку за писма различита од абецеде (кинеско, јапанско, ћирилица, арапско и многа друга).

Важно је напоменути да је оригинални *ASCII* стандард прописивао 7 битова за један карактер, што је дало простора за 128 карактера. Међу тим карактерима се налазе цифре, мала и велика слова енглеског алфабета и низ контролних карактера. Контролни карактери су ознаке за нови ред, празно место, табулатор, управљање штампачима (враћање главе штампача на почетак), рад са терминалима (аларм или обавештење да се нешто догодило). *ASCII* стандард је

проширен на 8 битова, односно цео један бајт и сада подржава 256 карактера. Потражите на интернету термин „*ASCII* табела" да бисте стекли увид у то како изгледа тзв. табела *ASCII* карактера.

Рад са појединачним знаковима не обећава много. Постоји реална потреба за радом са речима, реченицама, линијама текста итд. Подаци који се састоје од више од једног карактера се у домаћој литератури називају знаковним нискама, мада одомаћен је и назив „стринг", који је и много чешће у употреби. У програмском језику *C* се стрингови представљају обичним низовима карактера (**char niz[]**), а крај текста се означава **NULL** карактером, односно специјалним знаком **0**, чија бројевна представа је, погађате, нула. Оваква представа текстуалних података носи са собом и поједине компликације. Готово сви савремени језици познају тип стринга, док су у *C*-у стрингови ипак само обични низови.

Да се подсетимо, низови у *C*-у су заправо показивачи на део у меморији. Када покушамо да приступимо једном елементу низа, тада се врши операција „дереференцирања" са задатим одступањем. Следи неколико примера који илуструју рад са низовима на примеру низова карактера.

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

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

При декларацији низа 1 није задата дужина зато што је у једном изразу задата и иницијализација низа која гласи *"zdravo"*. Преводилац може сам да закључи колико меморије је потребно.

У другом примеру је креиран низ карактера дужине 5. Међутим, попуњена су само прва три места. Примећујете разлику између првог и трећег елемента? Први је цифра **'0'** (уоквирен је наводницима), а трећи је бројчана вредност **0** која означава крај стринга.

Може се приметити још једна разлика у дата два примера. У првом примеру су коришћени двоструки наводници, док су у другом коришћени једноструки. Разлог

за то је што се знаковни низови који су уоквирени двоструким наводницима сматрају стринговима и подразумевано им се додаје „терминатор", односно знак **NULL**, који означава крај стринга. Једноструки наводници се искључиво користе за представљање појединачних карактера. Ово значи да записи **"a"** и **'a'** нису једнаки. Први запис је дужине 2, јер има и знак **NULL** којим је завршен.

Исписивање и учитавање стрингова

Пре него што наставимо даље, биће објашњен једноставан начин за исписивање и учитавање стрингова помоћу стандардног улаза/излаза. Да се подсетимо, библиотека за стандардни улаз/излаз (испис података у конзоли односно терминалу) је **stdio.h**. Уколико се користи форматирани испис, потребно је функцијама **printf()** и **scanf()** проследити формат стринг који садржи специјалан знак **%s**, чиме је означено да желимо да испишемо знаковни низ све до **NULL** карактера. Други начин је употребом функција **puts()**, **gets()** и **getline()** из библиотеке **string.h**. Следи пример:

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

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

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

  return 0;
}
```

Може се приметити да је у случају функције printf() потребно нагласити специјалним контролним карактером \n да након исписа треба прећи у нови ред. Функција puts() то подразумевано ради.

За ове и све друге функције додатну помоћ је могуће пронаћи у Линуксовим *man* страницама у одељку 3, који је посвећен стандардним библиотекама. Рецимо, **man getline** садржи детаљно упутство о функцији **getline()**.

Копирање стрингова

Као што је у претходном одељку напоменуто, стрингови су у *C*-у само обични низови. То значи да следећи код вероватно неће урадити оно шта би било очекивано.

```
niz1 = niz2;
```

Овиме смо показивачу **niz1** „рекли" да показује тамо где и показивач **niz2**. Ефективно, они сад показују на исти текст, међутим ако изменимо **niz1**, то ће се одразити и на **niz2**, а важи и обрнуто. Овиме смо добили само два показивача на исти део меморије, а углавном то није био циљ.

Други начин како бисмо садржај другог низа могли да копирамо у први низ је петља којом бисмо копирали елемент по елемент. Пошто је операција копирања стрингова врло често потребна, постоји стандардна библиотека за рад са стринговима која се зове **string.h**, а функција за копирање стрингова је **strcpy()**.

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

Дати пример копира садржај низа 2 у низ 1. Копира се елемент по елемент све док се у низу 2 не наиђе на **NULL** карактер. Овакво копирање (елемент по елемент) се иначе у страној литератури назива и *„deep copy"* и врло често се среће у раду са структурама података, где је потребно обићи целу структуру података и сваки елемент копирати, а структуру опет изградити у другом делу меморије.

Занимљиво је што је копирање стрингова могуће поједноставити уколико дефинишемо нови тип податка као структуру која садржи низ карактера и потом извршимо копирање. Ово је трик који се углавном не користи у пракси и споменут је само као занимљивост.

```
#include  <string.h> struct str {
  char niz[50];
};

int main()
{
  struct str s1;
```

```
  struct str s2;

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

  s2.niz[0] =  'Z';

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

  return 0;
}
```

Креирана је структура *str* која садржи поље низа карактера под називом низ. Позивом функције **strcpy()** копирамо стринг *"zdravo"* у **s1** чиме смо „напунили" садржај објекта **s1**. Потом је извршена редовна додела чиме је садржај објекта **s1** копиран у **s2**. Затим је измењен само први карактер објекта **s2** и оба стринга су исписана. Примећујете да су копирани сви знакови, а да је измењен само други објекат? Занимљив трик.

Поређење, спајање и претрага стрингова

Друга врло честа ситуација је поређење два стринга. Уколико би поређење било извршено употребом редовног оператора за поређење **==**, тада би у ствари биле поређене само адресе на које показују показивачи тих стрингова, јер, да се подсетимо, стрингови су у *C*-у само обични низови. Исправан начин за поређење два стринга је позивањем функције **strcmp()** чија је повратна вредност 0 уколико су јој прослеђени стрингови са једнаким садржајем.

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

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

```
  return 0;
}
```

Функција **strcmp()** пореди сваки елемент прослеђених низова појединачно, а добра је вежба имплементирање нове функције **strcmp()** која ће унутар петље да пореди одговарајуће елементе прослеђених низова, и потом вратити резултат **0** уколико су сви елементи једнаки. Потребно је припазити на ситуацију када су стрингови различитих дужина.

У свету рачунара, спајање два садржаја се зове конкатенација. Одатле и назив функције за спајање два стринга, **strcat()**. Важно је уверити се да је стринг у који се додаје садржај довољно велик да га прихвати, иначе може да дође до нежељеног преписивања у меморији, уништавања других података и неисправног понашања програма.

Претрага унутар стринга се може обавити функцијом **strstr()**. Ова функција као параметре очекује показивач на низ карактера (стринг) у којем је потребно пронаћи стринг који је прослеђен као други параметар. Повратна вредност је показивач на део меморије где се жељени стринг први пут појавио.

Следи мало опширнији пример који илуструје спајање, претрагу и копирање стрингова.

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

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

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

  puts(niz_a);
  return 0;
}
```

Како да...?

Претварање типова података у стрингове и претварање из стрингова

Претварање података из једног типа у други или кастовање (енг. *type casting*) се за већину типова врши једноставно писањем типа у заградама, на пример **(int)5.23**. Стрингови су и по овом питању другачији. Поставља се питање, шта претворити у целобројни тип - да ли адресу на коју показује показивач низа који представља стринг? Први елемент? Шта ако тај стринг није број? Представићемо решење за конверзију бројева у стрингове и обрнуто.

Испис података на екрану се такође може сматрати конверзијом. Број са покретним зарезом је у меморији приказан на помало чудан начин, а позивом функције **printf()** када је као формат задат **%f** на екрану ћемо добити разумљив низ знакова. Сличан резултат можемо добити функцијом **sprintf()**, а једина разлика у односу на **printf()** је та што **sprintf()** као први аргумент очекује показивач на стринг, док су други (формат стринг) и трећи (аргументи који се повезују са форматним стрингом) слични као код функције **printf()**.

Код претварања стрингова у бројевне типове може послужити функција **sscanf()**, која се понаша слично као функција **scanf()**, с тим што је први аргумент адреса (показивач на стринг) коју желимо да скенирамо у нади да ће се поклопити са формалним стрингом који је други аргумент, а резултати ће бити сачувани у адресама које су прослеђене као 3, 4 и остали параметри. Други начин за претварање стрингова у целобројне бројеве и бројеве са покретним зарезом су функције **atoi()** и **atof()**.

Функције које користе форматни стринг су честе у језику *C* и заслужују посебну пажњу, међутим то је прича за посебан чланак. У ову класу функција спадају, рецимо, и функције за рад са датотекама.

Безбедност

Рад са стринговима може да буде изузетно опасна ствар по питању безбедности програма. Уколико се низ карактера не завршава **NULL** карактером, свашта може да пође по злу при позиву функција као што су **strcpy()**, **strcmp()** и сличне. Цела класа тих функција се ослања на постојање знака који означава крај стринга. Када стринг није прописно завршен, тада те функције могу да залутају у део меморије ван стринга и свашта може да пође по злу. Овакав тип напада се иначе

зове прекорачење бафера (енг. *buffer overflow*). Препоручен начин како да избегнете такве непријатне ситуације при раду са стринговима је да користите верзије функција које су ограничене максималном дужином стринга, као што су **strncpy()**, **strncat()**, **strncmp()** итд. Све те функције као трећи параметар захтевају број који одређује максималну очекивану дужину стринга. Још једна врло корисна функција је **strlen()** чији задатак је да одреди дужину стринга. Будите пажљиви, ово је прво место где се траже безбедносни пропусти у програмима. Не верујте подацима који споља улазе у ваш програм, а поготово када је реч о стринговима.

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

Distrowatch

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

Пад <
Пораст >
Исти рејтинг =
(Коришћени подаци са Дистровоча)

Ослобађање

У потрази за идеалном дистрибуцијом:

Социјални критеријуми за избор идеалне дистрибуције (6. део)

Аутор: Дејан Маглов

Пројекти слободног софтвера обично почињу као пројекти појединца или мале групе програмера, а који раде на развоју софтвера чији је циљ да задовољи неке од њихових личних потреба. Пројекат ће се сматрати успешним уколико се шира заједница заинтересује за њега након што препозна корисност софтвера не само за појединце који су започели пројекат, већ и за ширу заједницу. Када је у питању слободни софтвер, временом је постала стална пракса да се заједница умеша у развој пројеката. Добри пројекти сами привлаче заједницу људи који су заинтересовани за њих и она, у већини случајева, преузима на себе тестирање, побољшање функционалности, одржавање и техничку подршку пројекта.

Заједнице слободног софтвера могу да се окупе: око пројекта, слободног софтвера, по територијалном принципу или да настану комбинацијом ових принципа.

Примери:

- **ЛиБРЕ!** пројекат окупља заједницу око пројекта без територијалног ограничења.
- **ЛУГоНС** (*Linux User Group of Novi Sad*) је заједница корисника Линукса по територијалном принципу без обзира на конкретан пројекат.
- **Убунту-рс** заједница која је комбинација окупљања заједнице око конкретног пројекта (Убунту) и територијалног принципа који се ограничава на територију Србије.

Ова синергија пројеката и корисника слободног софтвера има више предности и за кориснике и за пројекат. Пројекат се на овај начин брже развија, добија на функционалности, грешке се брзо исправљају, итд. Корисници добијају осећај да припадају групи, друже се са истомишљеницима, учествују у промоцији квалитетних појединаца итд.

Две најважније улоге локалних заједница за ГНУ/Линукс дистрибуције су:

1. пружање техничке подршке и едукације на матерњем језику;
2. рад на локализацији и прилагођавање дистрибуције локалним потребама.

Информација да постоје локалне заједнице слободног софтвера је од нарочитог значаја за почетнике и кориснике који недовољно добро знају стране језике (нарочито енглески језик). Ако изузмемо комерцијалне пројекте отвореног кода (Ред Хет, СУСЕ), не постоје званичне школе и курсеви за слободни софтвер. За едукацију почетника је углавном задужена заједница, а јасно је да је учење на матерњем језику увек лакше него на било ком другом страном, ма колико добро га познајете.

Форуми и вики странице локалних заједница су праве библиотеке у којима корисници могу да нађу већину одговора на своја питања, а за сва остала ту су форуми и *IRC* канали заједница на којима увек има некога и који имају већ готова решења за велики број проблема.

Ослобађање

Локалне заједнице као критеријум за избор идеалне дистрибуције

Почетницима и корисницима који не владају добро енглеским језиком, топло препоручујемо да бирају ГНУ/Линукс дистрибуције које имају активну локалну заједницу. Овладавање новим оперативним системом је много лакше уколико вам „леђа чува" локална заједница на коју се увек можете ослонити.

Већ смо споменули да су популарније ГНУ/Линукс дистрибуције намењене почетницима. Случај је исти и са локалним заједницама окупљеним око њих. Ово међутим не значи да је квалитет услуга које нуде мање популарне локалне заједнице, окупљене око дистрибуција намењених напреднијим корисницима, лошији. Мање популарне дистрибуције окупљају мање људи, али зато они поседују веће знање. Искуство са заједницама окупљеним око слободног софтвера је углавном позитивно и указује на то да квалитет услуга које оне нуде кориснику није условљен популарношћу и масовношћу заједнице него искључиво њеном (не)активношћу.

У сваком случају, досадашњи савети за избор идеалне дистрибуције важе и даље. Социјални аспект слободног софтвера је само додатни фактор који вам може помоћи при доношењу одлуке када се двоумите између неколико могућих решења.

Још увек не постоји на једном месту потпуни и ажурирани списак свих активних српских заједница. Нешто најближе томе може да се нађе на адреси https://pulsslobode.wordpress.com/ . За информације о неким другим локалним заједницама које можда постоје али нису на овом списку мораћете сами да претражите интернет.

Локализација као критеријум за избор

За локализацију ГНУ/Линукс дистрибуција и ГНУ програма одговорни су појединци и локалне заједнице. Сасвим солидну српску локализацију могу имати и пројекти који уопште немају своју локалну заједницу, а исто тако, лошу локализацију могу имати и пројекти са добром локалном заједницом. Све је ствар приоритета заједница и активности појединаца.

Праву и контролисану локализацију може имати само локални пројекат. Оваквих

пројеката у Србији има мало и тешко се пробијају, чак и у локалу у конкуренцији светских пројеката. Истина је да многе веће ГНУ/Линукс дистрибуције имају српску локализацију у понуди, па се можда слични локални пројекти који су усмерени само на стриктну локализацију сматрају непотребнима. Узводна локализација у случају Србије врло често буде веома лоше, чак накарадно, спроведена. Једна половина програма је локализована српском ћирилицом, а друга латиницом. Ако постоји још делова система који уопште није локализован, онда се појављују и делови програма који су на енглеском. Зато ћемо се вратити на прву реченицу овог пасуса и поново утврдити да само локални пројекат може потпуно контролисати праву српску локализацију.

У Србији за сада имамо два активна пројекта ГНУ/Линукс дистрибуције – **ТриОС** (http://trios.rs/) и **Сербиан** (http://www.debian-srbija.iz.rs/p/serbian.html). Оба пројекта су млада и имају својих предности и мана. Коме треба локализован оперативни систем може да проба један од ова два пројекта.

Иако ове две домаће дистрибуције нису баш најидеалније, интересантно је пробати их и упоредити са другим дистрибуцијама.

За крај серијала

Намера нам је била да кроз овај серијал о критеријумима при избору идеалне дистрибуције дефинишемо већину критеријума при избору и покажемо зашто је тешко одговорити на једноставно питање: „Коју ми дистрибуцију препоручујете?"

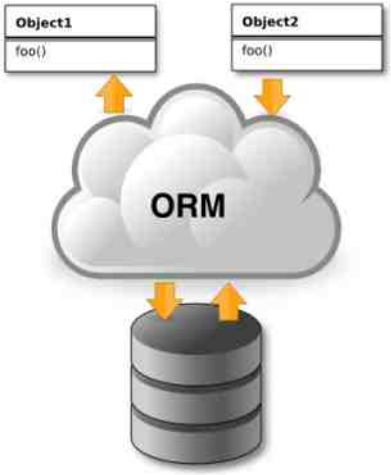
Ни на самом крају нисмо дали конкретан одговор на питање која је идеална ГНУ/Линукс дистрибуција. Сувише је субјективних оцена да би се могао дати конкретан одговор. Сматрамо да је јако тешко написати неки алгоритам избора по критеријумима које смо идентификовали и дефинисали, а можда је чак и немогуће, пошто се излази стално мењају. Излази из тог алгоритма су ГНУ/Линукс дистрибуције којих из дана у дан има све више а и оне старе се стално мењају.

Сам серијал „У потрази за идеалном дистрибуцијом" неће бити завршен само зато што се више неће бавити критеријумима при избору, него ће покушати да представља ГНУ/Линукс дистрибуције које су по нечему „идеалне".

HIBERNATE ORM (2. део)

Аутор: Дејан Чугаљ

Object relation mapping (ORM)



ORM (енг. *Object Relation Mapping*) је начин постизања трајности (енг. *persistence*) објеката унутар релационих база података. Модел делује као посредник на логичком делу апликације. Аутоматски пресликава податке из објеката у базу и назад, на захтев апликације. Да бисмо објаснили како ово све ради, морамо да вам представимо *ORM* фрејмворк (енг. *framework*) програма отвореног кода - Хајбернејт (енг. *Hibernate* - http://hibernate.org/orm/).

Шта је *ORM*?

Укратко, *ORM* је аутоматизовано, транспарентно пресликавање објеката коришћењем метаподатака који описују мапирање везе између објекта и базе података у апликацијама које су написане у објектно оријентисаним програмским језицима. Апликација комуницира са *ORM*-ом преко његовог властитог *API*-ја (енг.

application programming interface) и одређених објеката. Ова комуникација је потпуно одвојена од комуникације која се одвија између *ORM*-а и базе података преко *SQL*-а и *DBC*-а.

Да би се постигла и достигла потпуна снага *ORM*-а у апликацији коју пројектујемо, никако се не сме искључити знање релационих модела, *SQL* језика, те познавање *DBMS*-а у коме се ради. *ORM* није ништа друго до средњи слој између горе поменутих технологија и објектно оријентисаног програмирања, као што смо то до сада већ више пута нагласили.

Имплементација *ORM*-а

Постоје различити начини имплементације *ORM*-а. Марк Фасл (*Mark Fussel*) је 1997. дефинисао четири нивоа:

1. „Чист" релациони начин
2. Слабо мапирање објеката
3. Умерено мапирање објеката
4. Потпуно мапирање објеката

„Чист" релациони начин

Цела апликација, укључујући и кориснички интерфејс, дизајнирана је релационим моделом и *SQL* релационим операторима. Овакав приступ је одличан за мале пројекте. Директан приступ *SQL*-у омогућава фино подешавање сваког аспекта упита који се поставља бази података. Основне мане су портабилност (преносивост) и одржавање оваквих система, поготово ако се планира коришћење апликације на дужи рок. Овакви системи масовно користе сачуване процедуре (енг. *stored procedures*), што значи да је пословна логика (енг. *busines logic*) пребачена на саму базу података.

Слабо мапирање објеката

Ентитети су представљени као класе и мапирање се ради експлицитно ка релационим табелама. *SQL/JDBC* је сакривен од пословне логике коришћењем познатих шаблона (*patterns*).

Умерено мапирање објеката

Дизајн апликација се своди на објектне моделе. *SQL* се генерише у време компајлирања (енг. *build-time*) или у време извршавања (енг. *run-time*). Спецификација упита над базама података се ради на објектно оријентисаном језику. Овакви системи се користе у апликацијама средњих величина код којих је битна портабилност између различитих система за управљање базама података (*DBMS*).

Потпуно мапирање објеката

Подржава софистицирано објектно моделирање, наслеђивање и полиморфизам. Ефикасност добијања података је на високом нивоу и користе се методе као што су лењо учитавање (енг. *lazy loading*) и стратегија кеширања (енг. *caching*). Имплементација је транспарентна у апликацијама. Овај ниво је јако тешко достићи. Појединцу су потребне године развоја да постигне овакву флексибилност приступа и манипулација у релационим базама података. Управо овај ниво се постиже ако се користе решења *ORM*-а као што је Хајбернејт.

Зашто *ORM*?

Имплементација *ORM* решења, поготово ако се први пут сусрећете са овим моделом, може да буде веома фрустрирајућа и мало тежа за имплементацију. Када ово кажемо, мислимо да програмер који имплементира *ORM* и хоће да исти буде квалитетно урађен, мора да поседује предзнање објектно оријентисаног програмирања и напредних техника као што су наслеђивање и полиморфизам.

Ово иницира фундаментална питања:

- Па зашто би ико онда имплементирао овако нешто комплексно?
- Вреди ли труда упуштати се уопште у *ORM*?

Одговор није једноставан. Досадашња искуства у коришћењу *ORM*-а и његова популарност већ дају одговор на постављено питање. Очигледно је да су

бенефити које доноси овај модел већи од комплексности његове имплементације, а то су:

Већа продуктивност

ORM смањује количину кода потребног за остваривање трајности података. Ово омогућава програмеру да време које би утрошио на ручно мапирање података, утроши на друге делове апликације.

Лакше одржавање

Мање линија кода осигурава бољу разумљивост (и можда бољи квалитет кода) јер наглашава апликацијску логику. *ORM* такође ублажава утицај насталих промена као што су промене у бази, структури табела, итд.

Преносивост

Већина *ORM* решења подржава системе за управљање базама података разних произвођача. *ORM* у потпуности одваја слој апликације од стварне *SQL* комуникације која се одвија у позадини. Тиме омогућава да се пишу апликације које ће моћи да се извршавају на различитим базама података без потребе за додатним кодирањем. Ова комуникација се одвија са програмским интерфејсима *ORM*-а независно о *DBMS*-у са којим је у било ком моменту повезан. *ORM* такође омогућава лакши прелаз са мањих на веће базе података током развоја апликације.

* * *

Увидели смо да тренутно постоје две парадигме које се „сударају" и које су подједнако добре саме за себе. Релациона парадигма потврђује да све што је изграђено на математичким основама је и трајно, у смислу квалитета које поседује. У домену у ком релациона парадигма егзистира, она је, ако је то у ово време уопште могуће тврдити, незамењива.

Оно што отвара питања и даје разлог за сумњу је, која је крајња граница употребљивости релационе парадигме, тј. колики су капацитети сложености захтева апликација са којом може да се носи. Пракса је показала да је сасвим могуће „извести" сложене пројекте - али одржавање, скалабилност, разумљивост, преносивост - питања су са којим се релациона парадигма тешко

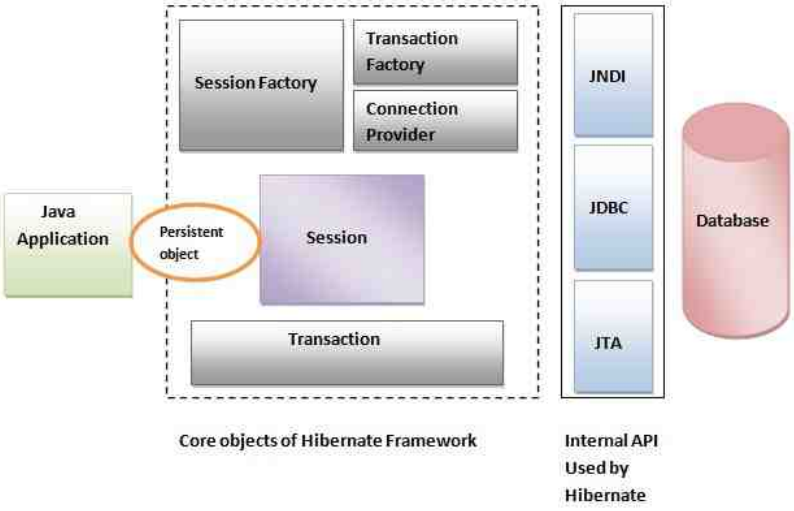
може изборити.

Са друге стране, свој хабитат у информационим технологијама је нашла и објектна парадигма. Иако на први поглед ова два модела немају ништа заједничко - што је у неку руку и истинито гледајући природу података којим манипулишу - пракса је показала да су уско повезане и скоро да имају синаптичку везу.

Питање „како спојити те синапсе" довело је до жучних расправа у уским научним, програмерско-администраторским круговима. Родили су се разни модели, али један је искочио и полако улази у широку примену - а то је *ORM*. Својим суптилним приступом проблему некомпатибилних парадигми некако је успео да створи тај недостајући „електрични сигнал" који их је повезао. Ово је омогућило да две стране, које су свака за себе у свом домену оставиле и остављају дубоки траг у информационим технологијама, сарађују и донесу нови квалитет.

Корисни линкови:

[1] Хибернејт: http://hibernate.org/
[2] Вики: http://goo.gl/LlZulw
[3] Туторијали: http://goo.gl/CGzvnc



То су само метаподаци?

Аутор: Криптопанк

Шта су, заправо, метаподаци? Најкраће речено, то су подаци о подацима или информације које описују податке.

Најпре да направимо разлику између података и метаподатака. Подаци су, например, снимак (или транскрипт) ваших телефонских разговора, садржај порука које пишете, примате или шаљете, подаци које качите на интернет или преузимате, претраге Гуглом, постови на форумима, дописивања на IRC каналима или другим четовима, аудио и видео снимци Скајп разговора, и тако даље.

Хајде да видимо шта би били одговарајући метаподаци за ове примере:

- За телефонске разговоре метаподаци могу бити локација учесника, бројеви телефона, трајање разговора, *IP* и *MAC* адреса.
- За поруке, четове и Скајп, или друге видео-аудио комуникације, слично је као код телефонских разговора.
- Код форума и претрага Гуглом има мало мање метаподатака, али Гугл и разни сајтови и форуми ионако чувају логове о корисницима што су такође метаподаци.

То су само метаподаци?

Ако до сада, којим случајем, нисте увидели опасност по приватност у овим метаподацима, које свако од нас смртника свакодневно оставља за собом у свету интернета и осталих дигиталних комуникација, надамо се да ће вас овај пример макар подстаћи на размишљање, а можда и на нешто више.

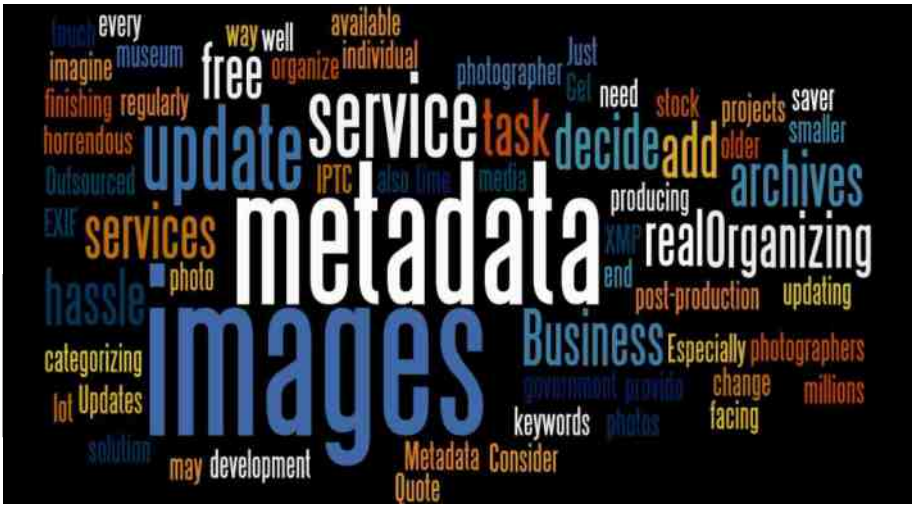
Замислите да неко (зваћемо га посматрач) стално сакупља метаподатке које за собом остављате као мрвице хлеба. Посматрач, рецимо, зна да се налазите на аутопуту недалеко од фамозног Фекетића, да се не крећете, да зовете АМСС, затим шлеп службу и можда такси, али не зна шта сте разговарали. Или посматрач види да сте прво звали аеродром Никола Тесла, па одмах потом такси, али не зна шта сте разговарали. Или посматрач види да сте звали прво шест бројева чије идентитете такође зна, па види да се они затим крећу ка вама - *GPS* (енг. *Global Positioning System*), па затим пица сервис, Мекдоналдс, и неку познату продавницу кинеске хране, али не зна шта сте разговарали.

Посматрач не мора да буде много паметан нити да прислушкује ваше разговоре да би схватио шта се дешава, или шта ће се десити. Сасвим му је јасно да вам је у првом примеру ауто стао на аутопуту, затим да сте се у другом случају упутили на аеродром и да можда путујете ван земље, или сте послали такси по некога са аеродрома, а да у трећем случају окупљате журку код вас. Више можете погледати на: http://goo.gl/7vUjP .

Иако су то „само" метаподаци, треба бити свестан да ти метаподаци о вама стварају једну дигиталну слику, или боље рећи дигитални мозаик, и на неки начин вас јединствено одређују. Поготово када се плански прикупљају на више нивоа (компјутер, паметни телефон, паметни аутомобил и кредитна картица), метаподаци чине ваш дигитални идентитет. Али, да ствар буде још гора, та слика о вама не мора нужно бити истинита, јер се лако може лажирати. Рецимо, можете заменити телефоне или аутомобиле са неким намерно или случајно, или вам неко може једноставно украсти телефон, лаптоп, или кредитну картицу. Занимљив видео: http://goo.gl/KO1fKf .

Ево једног примера који вас може увући у велике невоље. Рецимо да пролазите неком забаченом улицом и да вам неприметно испадне телефон, али ви то не приметите и наставите даље својим путем. У тој улици се само неколико минута касније деси неко убиство, и вас позову у полицију као осумњиченог. Ако којим случајем немате добар алиби, или немате сведока да посведочи да нисте били у тој улици, него тамо где сте стварно били, а других сведока нема - можете јако

лоше проћи. Занимљиви видео: http://goo.gl/spS5OZ .



Метаподаци се такође налазе и у фајловима/подацима које свакодневно преузимамо са интернета, правимо и користимо. Сваки тип података може да садржи одређене метаподатке који боље описују сам податак, као, например, време настанка података, колико је времена потрошено за његово креирање, колико пута му је приступљено, кад је последњи пут приступљено, и тако даље. Иако се овакви метаподаци можда још увек чине безазленим, они могу садржати и *IP* адресу рачунара са ког су направљени, *GPS* координате (најчешће *PNG* формат слика са телефона), ко га је направио (личност или компанија/организација), назив уређаја и модел са кога је направљен као и *MAC* адресу уређаја, мејл адресу креатора, софтвер којим је направљен, и тако даље. Све ово може много да нашкоди приватности изворног креатора фајла, поготово ако је садржај поверљив. Уколико неко пресретне тај податак на интернету, може сазнати од кога је потекао. Да ово може бити катастрофа по креатора таквог фајла, може илустровати пример торента. Ако сте дошли у посед неких слика или видео материјала или текста који желите да поделите са светом преко неког торент сајта, морате тај податак прво очистити од метаподатака, да не би свако ко скине то што сте окачили сазнао да ви стојите иза тога.

То су само метаподаци?

Људи се константно прате, метаподаци се сакупљају и то није сада већ ништа ново, али можемо да се трудимо да иза себе остављамо мало мање нежељених метаподатака, и да их такорећи чистимо из података које размењујемо. На срећу постоје програми који ово добро раде.

Најпре је потребно да видимо које све метаподатке неки фајл садржи, како бисмо се уверили да ови метаподаци стварно постоје и да је фајл безбедан након њиховог брисања.

Ако користите оперативне системе ГНУ/Линукса, треба да из терминала преузмете и инсталирате програм под називом **exiftool**:

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

Овај програм ће нам пружити увид у све метаподатке које фајл садржи. Коришћење је из терминала, али је зато веома једноставно, само га покрените на следећи начин:

```
exiftool File
```

или

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

Добићете приказ метаподатака које је програм пронашао. Сада је потребно да видимо да ли је податак „прљав", тј. да ли има у њему нешто да се брише, и да, ако има, из њега све непотребно обришемо. Ту ће нам помоћи други програм звани **MAT** (енг. *Metadata Anonymisation Toolkit*) кога можете наћи на сајту: https://mat.boum.org или једноставно за кориснике Убунтуа и Дебијана у Софтверском центру, или га можете инсталирати из терминала:

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

MAT ће проверити да ли је податак „прљав" или не, и за вас га очистити од нежељених метаподатака.

Када очистите фајл, он је спреман за дељење са другима, а ви сте безбедни јер нико не може рећи да је фајл потекао са вашег рачунара.

Изгубљени хероји „Блечли парка”

Аутор: Дејан Чугаљ

> Г.Х. Харди: „Права математика нема утицај на рат.” - одбрана математичара, 1940. година.

Бил Тат - британски математичар, вероватно нисте чули за њега, 1942. године је извео огроман подухват који је скратио рат две године (барем по речима Ајзенхауера) и спасио милионе живота. Нажалост, његова величина је преминула 2002. године обавијена велом тајне и никада званично није признато његово достигнуће.

Томи Флауерс - бивши инжењер поштанског саобраћаја, претворио је математичке идеје Тата у први рачунар, а то није био *„ENIAC”* [1]. Преминуо је 1998. године, а претпостављамо да никада нисте чули ни за његову величину.

[1] енгл. ***Electronic Numerical Integrator And Computer*** - први дигитални електронски рачунар

Ови умни гиганти „Блечли парка” омогућили су Британији да дешифрује строго поверљиву машину коју је користио Хитлер да диригује другим светским ратом, а то није била „ЕНИГМА”, него нешто много тајније и значајније. Велика је вероватноћа да нисте чули ни за ово. Неки кажу да је то био Хитлеров Блекбери (*BlackBerry*).

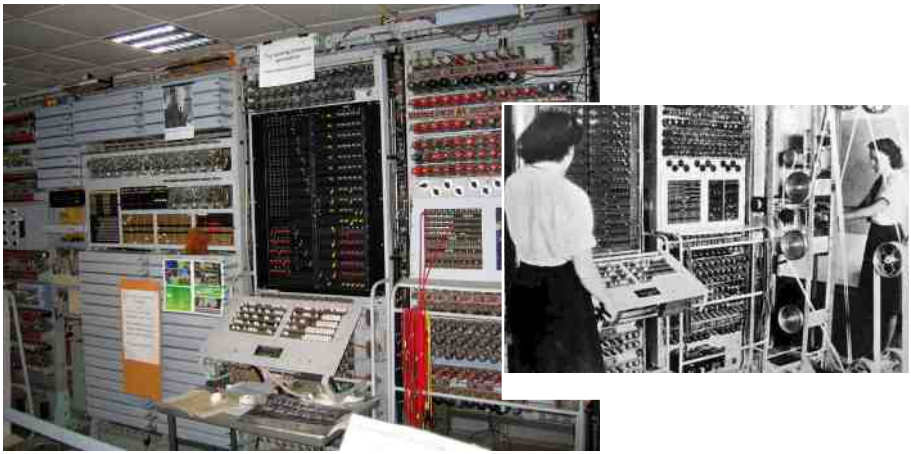
* * *

„Блечли парк” је 1939. године постао ратни штаб *MI6* (британске војне обавештајне службе). До данас је процурело у јавност и документовано бројним филмским документарцима само да је ту Алан Тјуринг успео да дешифрује немачку морнаричку шифру познату као „ЕНИГМА” и увелико допринео победи савезника, а то је само део приче који је испричан.

У Блечли парку су била три хероја: Алан Тјуринг, Бил Тат и Томи Флауерс. Од њих тројице једино Алану Тјурингу је признат део заслуга за победу савезника у Другом светском рату. Прегледом откривених података, данас можемо да тврдимо да је „ЕНИГМА” већ са почетка другог светског рата била застарела технологија. Због тога је било лако скинути вео тајне са овог успеха хероја Блечли парка.

Интернет мреже и комуникације

Бил Тат, који је дешифровао „ТАНИ систем”, (*TANI*) и Томи Флауерс, који је само својом интуицијом и знањем склопио први рачунар на свету („КОЛОС”) у сврху бржег долажења до информација разбијеног кода „ТАНИ” тако да застарелост шифрата буде сведена на минимум - радили су на новој генерацији машина за шифровање. Ова технологија је била препозната као технологија будућности па и данас изнад информација о тим технологијама стоји натпис „врхунска тајна” (енг. *Top Secret*). Управо овај вео тајни око шифрата „ТАНИ” и првог рачунара је резултирао искривљењу историје информатике.



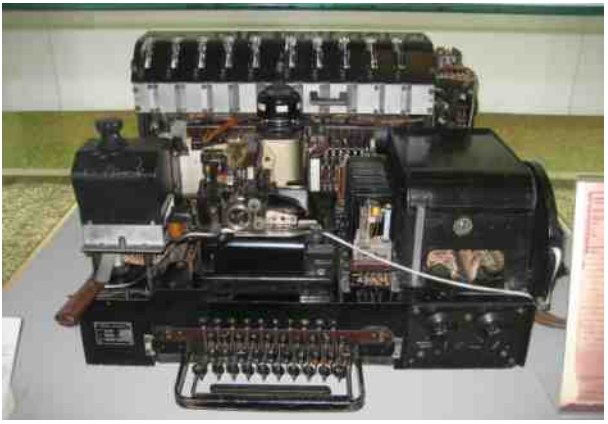
За успехе у првим годинама рата је заиста заслужно разбијање „ЕНИГМА” шифрата и то није спорно, али крајем 1941. године у етру планете се зачуо нови звук. То је била нова машина за шифровање; није била заснована на „Морзеу”, већ је радила на принципу телепринтера.

Једна од кључних идеја Хитлеровог ратовања је била у великој мобилности и брзом деловању трупа на терену, а да би се то извело, комуникација је исто тако морала да буде у најмању руку путем радио сигнала.

Разлика је огромна. Разлика је у количини података који може да се пренесе овим „тајним” путем. Први светски рат је био занимљив јер су се шифроване поруке базирале на речима, док је Други у потпуности отворио врата данашњој криптографији, а то је математика.

Оригиналан немачки кôдни назив за овај шифрат био је „ЛОРЕНЦ”. Британска

обавештајна служба му је дала надимак „ТАНИ”. Чак ни дан данас није баш познато како је све настало, али поуздано се зна да је тајну машину Хитлер назвао „Гехајмшрајбер” (нем. *Geheimschreiber*) - машина тајни.



У срж и технологију шифровања „ЛОРЕНЦ” шифрата нећемо улазити у овом чланку, али верујте нам на реч - није била једноставна. Може се претпоставити да је инспирација Шеноновог [2] рада случајности била управо ова технологија из Другог светског рата.

Оно чиме се Трећи рајх водио је „сигурна” шифрована веза, коју нико није смео да дешифрује. Преношење шифрованих порука помоћу радио сигнала је дало обавештајним службама могућност пресретања порука послатих радио таласима. Да ли се ово и данас дешава? Имамо ли сигурност у нашим шифрованим подацима данашњице?

Оно што је уследило након пробоја овог „неухватљивог” шифрата је управо оно што је принцип данашњице, као нека врста протокола који мора да се испоштује у перфектном шифарском Шеноновом систему:

1. НИКАДА, НИКАДА, НЕ КОРИСТИТИ ДВА ПУТА ИСТИ КЉУЧ ЗА ШИФРОВАЊЕ.
2. НИКАДА, НИКАДА, КЉУЧ НЕ СМЕ ДА СЕ ШАЉЕ ИСТИМ КОМУНИКАЦИОНИМ

[2] **Клод Елвуд Шенон** (енг. *Claude Elwood Shannon*) (1916-2001) - Амерички научник и инжењер

КАНАЛОМ КАО ПОРУКА.
3. НИКАДА, НИКАДА, КЉУЧ НЕ СМЕ ДА БУДЕ МАЊИ НЕГО ШТО ЈЕ ПОРУКА (енгл. *one time pad*).
4. НИКАДА, НИКАДА, НЕ СМЕМО ДА САКРИВАМО СИСТЕМ ШИФРОВАЊА, СВЕ МОРА ДА ЈЕ ОТВОРЕНОГ КÔДА.

Највећи пропусти криптологије, који су уочени баш у Другом светском рату, јесу покушај сакривања средстава којим се генерише тајна порука и непознавање Шенонових крајњих граница информација. Када кажемо „средство и сакривање", мислимо на алгоритме. Поставља се питање зашто је данас криптологија сигурнија него четрдесетих и педесетих година прошлог века?

Данас, упркос томе што су алгоритми за криптографију отвореног кôда, тајност информација је на вишем нивоу него што је то било док је алгоритам био строго поверљив. Тајност тајних података се не заснива на тајности алгоритама који генеришу тај исти „тајни излаз" (енг. *cipher text*) него у немогућности математичке инверзне функције која би шифрован текст отворила. Овде се види да је сврха структуре отвореног кôда као једног од значајних делова данашње криптологије она која га чини сигурним и свима доступним.

Сада, када се са ове дистанце осврнемо на Други светски рат, можемо видети колико смо били близу апокалипсе. Наиме, Алберт Ајнштајн је већ 1944. године имао у рукама рецепт за нуклеарну бомбу, Шенон је могао већ у то време да нам предочи 100% сигуран шифрат (тзв. Хитлерову супер шифру), али ипак, срећа нас је послужила да човечанство тада није било спремно да употреби сву расположиву технологију.

> Алберт Ајнштајн, „Четврти светски рат ће се водити тољагама."

* * *

Заборављени хероји Блечли парка, иако су одиграли важну улогу у победи савезника у светском рату, постали су жртве „*top secret*" доктрине, потпуно су заборављени и нестали су из историје.

Закључак овог чланка је да је идеологија отвореног кôда сасвим сигурно ту да нам помогне и да отвори врата ка већој сигурности, новим идејама и квалитету саме инфраструктуре система коју сви заједно развијамо.

LaTeX презентација:

Beamer (4. део)

Аутор: Никола Харди

Набрајања

Без бројева

За набрајање појмова у Бимеру (енг. *Beamer*) користи се окружење **itemize**, а унутар тог окружења можемо да додајемо нове ставке помоћу **\item**. Пример изгледа овако:

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

- Stavka A
- Stavka B
- Stavka C

Са бројевима

Постоје ситуације када уз набрајање треба да стоје и редни бројеви - за то се користи **enumerate** окружење, а нове ставке се такође додају помоћу **\item**.

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

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

Дефиниције

Трећа ситуација је набрајање парова термин - дефиниција. У том случају се користи окружење **description**. Већ можете да претпоставите како се додају нове ставке.

Пример:

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

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

Остала подешавања и занимљивости

Сва претходна окружења могу да буду угнеждена једно унутар другог, до трећег нивоа. Треба бити пажљив јер слајдови врло лако могу да постану нечитљиви. Осим тога, могуће је изабрати и формат ознаке или редних бројева - на пример овако:

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

Такође, могуће је управљати појављивањем појединих ставки на слајдовима. Другим речима, можемо да одредимо да се набрајања постепено откривају тако што ћемо уз ставку додати **<1->**, односно број од којег слајда желимо да се та ставка појављује. Тада ће једно набрајање бити „развучено" на више слајдова. Једноставнија могућност је да у заглавље окружења додамо **{+-}**, што значи да желимо да се сваким слајдом открива по једна нова ставка из набрајања. Знак „-"

у оба примера значи да желимо да се ставка појави на следећем слајду и да остане присутна. Можемо да изоставимо знак „-” и тада ће ставка бити присутна само на једном слајду, или да напишемо распон слајдова у којима желимо да та ставка буде присутна.

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

Рад са колонама и блоковима

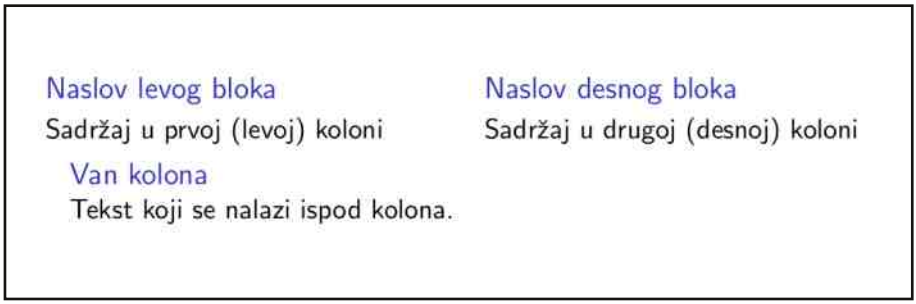
До сада су сви примери подразумевали да је садржај написан преко целе ширине слајда, али Бимер (енг. *Beamer*) има и једноставне механизме за размештање садржаја по колонама. За рад са колонама се користи окружење **columns**, унутар којег можемо да додајемо нове колоне помоћу **\column{širina}**. Садржај је могуће потом писати и ван окружења са колонама.

У пару са колонама, често се среће и окружење **block**, које се параметризује насловом. Примером су илустровани и окружење **column** и окружење **block**.

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

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

```
  \begin{block}{Van kolona}
    Tekst koji se nalazi ispod kolona.
  \end{block}
\end{frame}
```



Табеле

Иако се у раду са презентацијама садржај најчешће може уредно распоредити помоћу колона и набрајања, некада то није довољно јер су табеле прави формат. Бимер подржава стандардно Латех (*LaTeX*) окружење за рад са табелама. Окружење за рад са табелама се зове табулар. При креирању нове табеле могуће је подесити које колоне ће бити раздвојене линијом и да ли ће њихов садржај бити центриран, или уз леву или уз десну ивицу. Додавање новог реда се једноставно ради додавањем **\hline**. Латех пружа заиста јако флексибилан рад са табелама и сигурно ће доскочити свим могућим и немогућим замислима. Потражите било које Латех упутство за рад са табелама и радиће и са Бимером.

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

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

Теореме и истакнут текст

У настави су честа излагања теорема, њихових доказа и примера. У Бимер је уграђена подршка за елегантно представљање ове тројке. Механизам такође подржава постепено откривање садржаја слајда, тако да слушаоцима не буде одмах откривен доказ теореме. Механизам за постепено откривање се, као и код набрајања, постиже помоћу **„<1->”** конструкције. Ево како то изгледа:

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

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

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

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

Theorem
Teorema kaže da su beamer prezentacije zanimljive.

Example
Ovaj kod kreira slajd beamer prezentacije.

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

За крај

Овим чланком завршавамо серијал о Бимеру. Надамо се да смо успели да вас заинтересујемо за ову врло корисну класу Латеха. Трудили смо се да вам презентујемо најважније делове Бимера које ћете најчешће користити приликом креирања својих презентација.

Напомињемо још једном да је Бимер само једна од класа Латеха што значи да и за Бимер важи све оно што сте већ могли да прочитате у овом часопису кад смо писали серијал „Увод у //LaTeX//" (ЛиБРЕ! бројеви 17 до 21). Штавише, овај серијал о Бимеру је био наставак управо тог серијала. У овом серијалу нисмо писали о самом форматирању текста у презентацијама јер се подразумева да важе иста правила као и у Латеху, а о томе смо писали у претходном серијалу.

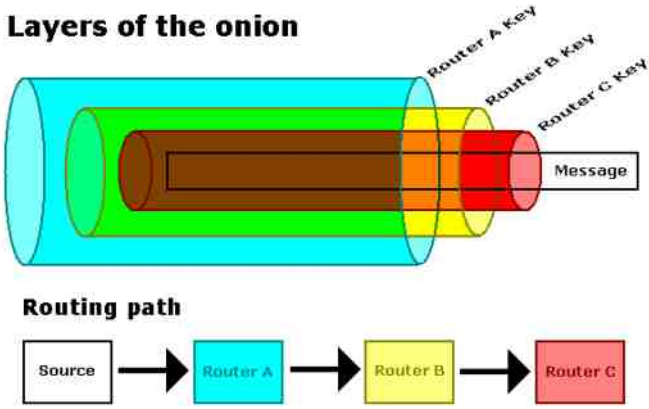
Хвала вам што сте пратили овај серијал и желимо вам пуно успеха у креирању Бимер презентација.

BeagleBone Black Rev C

Водич од првог дана (3. део) – *BeagleBone Black* као Тор релеј

Autor: Nenad Marjanović

Када смо се упознали са подешавањима система и умрежавањем, време је за наш први Биглбон Блек (ББББ - енг. BeagleBone Black) пројекат. Сваком од нас ће овај изазов дати различите идеје, а једна од њих може бити и заштита приватности интернет корисника. Нажалост, у Србији не постоји ниједан излазни тор сервер (енг. *node*) и то је помало тужна слика с обзиром да говоримо о популацији од седам милиона становника. Али ову слику можемо променити удруженим снагама и, за почетак, креирањем Тор трансфер и излазног сервера. У овде приказаном графикону фокусираћемо се на „Раутер Б" тачку, односно трансфер сервер конфигурацију.



Тор представља мрежу сервера који се повезују ради боље енкрипције података и заштите *IP* адресе корисника. Наш циљ је да данас постанемо део овог предивног пројекта и да се нађемо на светској карти Тор сервера.

Брзина интернета и законске регулативе

Потребно је имати на уму неколико битних ствари:

1. Наш сервер = наша одговорност
2. Обавезно назначите да сте ви администратор а не корисник истог сервера
3. Да поседујете интернет брзину од 2 Mb/s

Ако испуњавамо три наведена правила и одговорности, можемо се посветити административним пословима.

Инсталација и подешавања

У овом примеру користимо Дебијан Визи (енг. Wheezy), односно Дебијан 7 верзију на нашем ББББ уређају. Ви можете тестирати и друге Линукс платформе. Прво додајемо изворну библиотеку у **/etc/apt/sources.list** фајл покретањем следеће команде у терминалу:

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

Затим додајемо изворни кључ, вршимо ажурирање система и инсталирамо **tor** и **tor-arm** (Тор мониторинг систем).

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

Овим смо успешно завршили инсталацију Тор сервера и система за мониторинг, односно надгледања нашег Тор сервера. Следећом командом радимо на подешавањима Тор сервера која се налазе у **/etc/tor/torrc** фајлу. Обратите

пажњу и извршите потребне измену у пољима *IP* адреса (енг. *Address*), надимка (енг. *Nickname*) и контакт података (енг. *ContactInfo*) пре покретања ове команде. Копирамо и покрећемо команду у целости у терминалу.

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

Детаље о свим вредностима нећемо износити у овом броју ЛиБРЕ! часописа, али ћемо о њима писати у следећем, у коме ћемо причати о заштити Тор сервера и могућности подешавања одређених излазних правила (енг. *ExitPolicy*).

Након овога можемо приметити да се у фолдеру **/etc/tor/** не налази **tor.html** фајл. Овај фајл није од великог значаја и можете га лично направити, или преузети једну копију - например, са следеће локације:

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

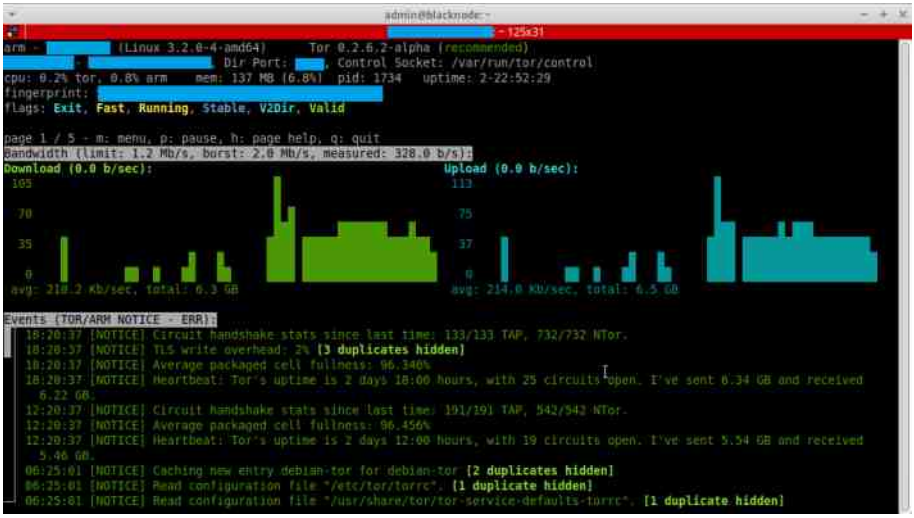
Опције и Тор Арм мониторинг

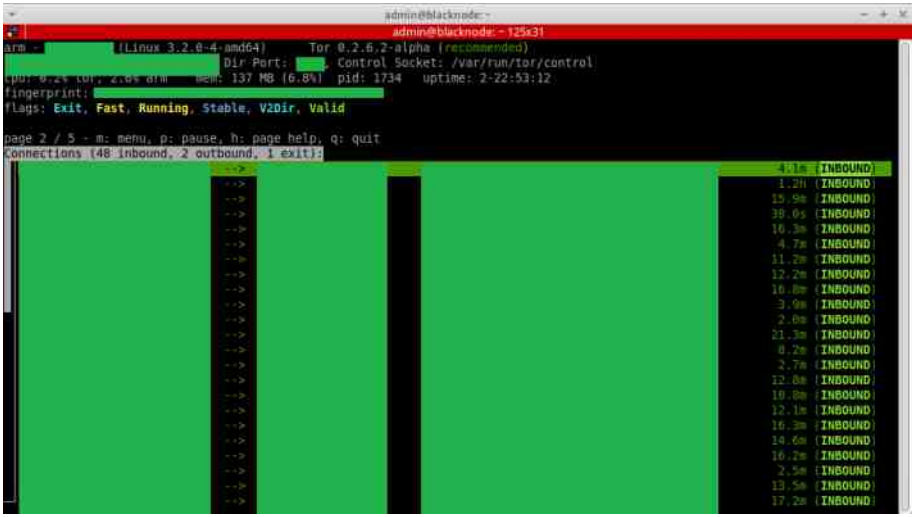
Да бисмо исправно користили Тор Арм, потребно је уредити још једном **/etc/tor/torrc** фајл и рестартовати сервис.

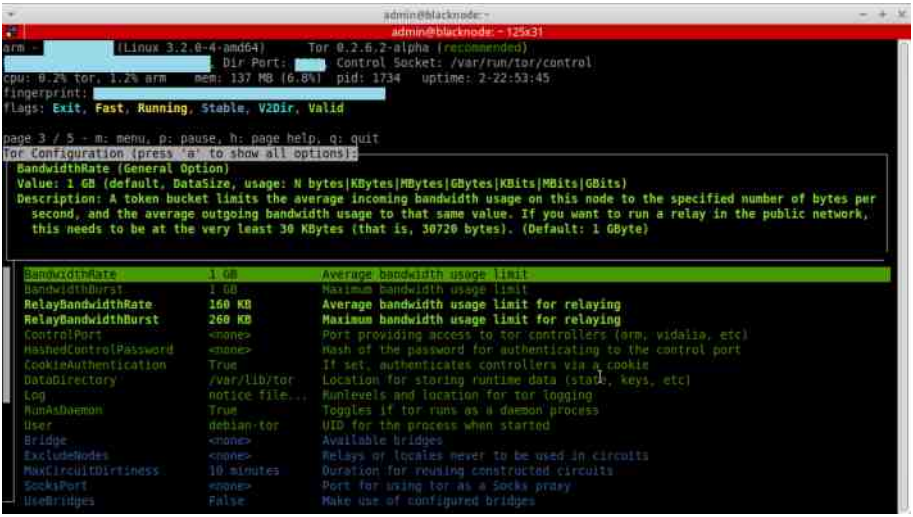
```
echo -e “DisableDebuggerAttachment 0\n AvoidDiskWrites”  >>
/etc/tor/torrc
sudo service tor reload
```

Уколико желите користити Тор Арм за надгледање активности на серверу покрените програм **screen**, кликните на „Enter” и куцајте **sudo -u debian-tor arm**. На екрану ћете имати сличан приказ:

Хардвер


admin@blacknode: ~
admin@blacknode: ~ 125x31
arm ~ (Linux 3.2.0-4-amd64) Tor 0.2.6.2-alpha (recommended)
Dir Port: , Control Socket: /var/run/tor/control
cpu: 0.2% tor, 0.8% arm mem: 137 MB (6.8%) pid: 1734 uptime: 2-22:52:29
fingerprint:
Flags: Exit, Fast, Running, Stable, V2Dir, Valid
page 1 / 5 - m: menu, p: pause, h: page help, q: quit
Bandwidth (limit: 1.2 Mb/s, burst: 2.0 Mb/s, measured: 328.0 b/s):
Download (0.0 b/sec):
Upload (0.0 b/sec):
105
70
35
0
113
75
37
0
avg: 218.2 Kb/sec, total: 6.3 GB
avg: 214.0 Kb/sec, total: 6.5 GB
Events (TOR/ARM NOTICE - ERR):
18:20:37 [NOTICE] Circuit handshake stats since last time: 133/133 TAP, 732/732 NTor.
18:20:37 [NOTICE] TLS write overhead: 2% [3 duplicates hidden]
18:20:37 [NOTICE] Average packaged cell fullness: 96.340%
18:20:37 [NOTICE] Heartbeat: Tor's uptime is 2 days 18:00 hours, with 25 circuits open. I've sent 6.34 GB and received 6.22 GB.
12:20:37 [NOTICE] Circuit handshake stats since last time: 191/191 TAP, 542/542 NTor.
12:20:37 [NOTICE] Average packaged cell fullness: 96.456%
12:20:37 [NOTICE] Heartbeat: Tor's uptime is 2 days 12:00 hours, with 19 circuits open. I've sent 5.54 GB and received 5.46 GB.
06:25:01 [NOTICE] Caching new entry debian-tor for debian-tor [2 duplicates hidden]
06:25:01 [NOTICE] Read configuration file "/etc/tor/torrc". [1 duplicate hidden]
06:25:01 [NOTICE] Read configuration file "/usr/share/tor/tor-service-defaults-torrc". [1 duplicate hidden]


admin@blacknode: ~
admin@blacknode: ~ 125x31
arm ~ (Linux 3.2.0-4-amd64) Tor 0.2.6.2-alpha (recommended)
Dir Port: , Control Socket: /var/run/tor/control
mem: 137 MB (6.8%) pid: 1734 uptime: 2-22:53:12
fingerprint:
Flags: Exit, Fast, Running, Stable, V2Dir, Valid
page 2 / 5 - m: menu, p: pause, h: page help, q: quit
Connections (48 inbound, 2 outbound, 1 exit):
4.1s (INBOUND)
1.2h (INBOUND)
15.9s (INBOUND)
38.6s (INBOUND)
16.3m (INBOUND)
4.7m (INBOUND)
11.2m (INBOUND)
12.2m (INBOUND)
16.8m (INBOUND)
3.9m (INBOUND)
2.0m (INBOUND)
21.3m (INBOUND)
8.2m (INBOUND)
2.7m (INBOUND)
12.8m (INBOUND)
10.8m (INBOUND)
12.1m (INBOUND)
16.3m (INBOUND)
14.6m (INBOUND)
16.2m (INBOUND)
2.5m (INBOUND)
13.5m (INBOUND)
17.2m (INBOUND)



Да бисмо напустили **screen** програм, користимо **Ctrl + a + d** и, на тај начин, када се следећи пут пријавимо на сервер, куцамо само **screen -r** у терминалу - и нађи ћемо се поново на чувеном Тор Арм мониторинг екрану.

Након сваке промене конфигурационих фајлова, ако случајно нисмо нешто подесили како треба, рестартујемо сервисе.

У следећем броју часописа говорићемо о минималним подешавањима и конфигурисању излазног Тор сервера (енг. *Tor Exit Node*), опцијама, сигурности, и сличним детаљима у вези са нашим првим пројектом. До читања у следећем броју!

LUGoNS BarCamp №4

субота 7. март 2015. године

Четврти ЛУГоНС-ов Баркемп (*BarCamp*) одржаће се у Новом Саду у суботу 7. марта 2015. године на Факултету техничких наука са почетком у 12 часова.

Улаз је слободан!

Позив за предају радова

ЛУГоНС позива све заинтересоване да пошаљу своје радове (предавања, презентације, радионице и дискусије) до 28. фебруара 2015. године.

Области из којих можете да пријавите ваше радове су:

- Слободан софтвер
- Занимљив хардвер
- Хакерски закони
- Сигурносне ноћне море
- Анонимност и приватност на интернету
- Дубоки веб (енг. *Deep Web*)
- Социјалне мреже
- Социјални инжењеринг
- Хаковање мобилних уређаја
- Интернет у облаку (енг. *Clouds*) – хаковање, разбијање и неочекивано коришћење
- *DPI* (енг. *deep packet inspection*)
- Мрежа и мрежна неутралност – власништво, цензура, надмудривање и политика „де факто” стандарда
- Програмирање и програмски језици

Немојте се устручавати и слободно нам се јавите ако имате рад на горе наведене или сличне теме који сматрате интересантним.

Страница за предају радова налази се на: https://events.lugons.org/?p=1658