Открытые технологии

Материалы тринадцатой международной конференции Linux Vacation / Eastern Europe 2017

ОТКРЫТЫЕ ТЕХНОЛОГИИ

Сборник материалов тринадцатой
международной конференции
разработчиков и пользователей свободного
программного обеспечения
Linux Vacation / Eastern Europe 2017

(г. Гродно, 22–25 июня 2017 г.)

Брест
«Альтернатива»
2017

УДК 004.45(082)
ББК 32.973.26-018.2я43
О-83

Под общей редакцией **Д.А. Костюка**
Редакционная коллегия: **Д.А. Костюк, Н.Н. Маржан, Д.А. Пынькин, А.О. Шадура**

Рецензенты: кандидат технических наук, доцент, заведующий кафедрой электронных вычислительных машин Белорусского государственного университета информатики и радиоэлектроники **Д.И. Самаль;**
кандидат технических наук, доцент, заведующий кафедрой электронных вычислительных машин и систем Брестского государственного технического университета **С.С. Дереченник**

О-83 **Открытые технологии** : Сборник материалов тринадцатой международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2017, Гродно, 22-25 июня 2017 г. / под общей ред. Д.А. Костюка. – Брест : Альтернатива. – 132 с.

ISBN 978-985-521-591-3.

В сборник вошли материалы тринадцатой международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2017. Материалы докладов распространяются под лицензией Creative Commons Attribution-ShareAlike 3.0. Статьи посвящены новым технологиям и разработкам в сфере свободного (открытого) программного обеспечения. Тематические направления включают разработку, администрирование программных систем и создание аппаратного обеспечения под свободными лицензиями, а также правовые и организационные особенности их использования.

Сборник может быть интересен специалистам в области информационных технологий, занимающимся разработкой и сопровождением свободного программного обеспечения, а также подготовкой кадров высшей квалификации по профильным специальностям.

УДК 004.45(082)
ББК 32.973.26-018.2я43

ISBN 978-985-521-591-3

Содержание

Многослойные и многоуровневые системы хранения данных.*

Александр Клыга, Minsk, Belarus

In multilayers data storages the storage drivers are grouped into separate layers according to their technical characteristics. The first layer contains high-speed solid-state drivers (SSDs), whereas the other layers can have HDDs or others storage devices (tape, optical devices). The multilayer data storage provides high speed data access and can be formed by using hardware or software solutions. In multilevel software-define storages (SDS), the separate data storage are grouped with respect to levels of their technical characteristics, the type of stored data, and availability requirements for the end user. Multilevels SDS allows for optimization of the data placement in the storage. The main node of SDS is represented by a controller. The SDS controller is a software platform (open source and other) which includes solution tools for management, monitoring and migration of data between layers.

Многослойные и многоуровневые системы хранения данных. Обзор архитектуры и решений на базе open source.

В настоящее время развитие систем хранения данных (СХД) обусловлено следующими тенденциями (по материалам конференции [1]):

- постепенно разработка новых накопителей для хранения данных на базе магнитных дисков замедляется, при этом отмечаются технологические проблемы с производством дисков большой емкости;
- стремительное развитие технологий производства устройств хранения данных на базе flash памяти. При этом их емкость постоянно увеличиваются, а надежность хранения данных гарантируется на высоком уровне;

*alex_kls@mail.ru, http://lvee.org/ru/abstracts/238

– цена устройств на базе flash памяти больше не является сдерживающим фактором их применения в системах хранения данных;
– использованием нового интерфейса NVM Express (NVMe, NVMHCI – Non-Volatile Memory Host Controller Interface Specification [2]) для твердотельных устройств хранения данных (SSD);
– использование решений с открытым кодом для построения программно-определяемых систем хранения данных.

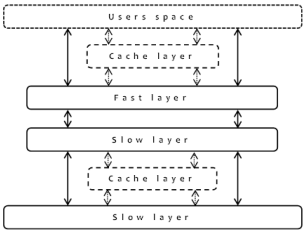
Популярным архитектурным решением СХД являются многослойные хранилища данных разработанные на базе технологии тиринга (от анг. tiering) и использующие интеллектуальные алгоритмы распределения данных между слоями. В основе технологии тиринга лежит принцип разделения хранилища данных на слои, отличающихся производительностью в зависимости от используемых типов запоминающих устройств, из которых оно состоит (например, твердотельные диски SSD или разноскоростные шпиндельных HDD). Для управления размещением данных по слоям СХД могут использоваться как аппаратные решения, например, RAID контроллеры с поддержкой технологии тиринга, так и различные программные решения (в том числе на базе open source). Одним из преимуществ использования программных решений является, то что в качестве слоев могут быть использованы дополнительные хранилища данных подключенные к контроллеру СХД.

При многоуровневом построении СХД отдельные хранилища данных группируются по уровням на основании их технических характеристик, типа хранимых данных, и требований доступности для конечного потребителя. Многоуровневые СХД, как правило, реализуются виде программно-определяемых систем хранения данных или software-define storages (SDS).

Многослойные системы хранения данных.

Технология тиринга применяемая при создании многослойных СХД позволяет группировать устройства хранения данных в отдельные слои в зависимости от их типа и характеристик. Для улучшения производительности хранилища данных между сломи могут располагаться дополнительные cache-layer, основное назначение которых повышение производительности при миграции данных между слоями, или уменьшение латентности отклика данных между

пространством пользователя и СХД. Общая структура СХД с многослойной структурой приведена на рисунке ниже.



Обычно в качестве cache-layer используют RAM-диски созданные в ОЗУ или SSD диски, если необходимо кэширование между слоями с медленными накопителями. В слой fast-layer, как правило, объединяют быстрые диски, типа SSD, но при этом важно учитывать их технические характеристики. Так, например, при использовании SSD с интерфейсом NVMe, рекомендуется делать несколько слоев fast-layer, объединяя в каждый слой SSD накопители одного типа. В слои slow-layer как правило, объединяют шпиндельные накопители на магнитных дисках (HDD), группируя исходя из их технических характеристик, либо иные типы накопителей, например, ленточные.

Суммарный объем многослойного СХД равен сумме объемов каждого слоя, при этом необходимо учитывать, что по мере заполнения слоя данными, часть его объема может быть зарезервировано для операций миграции данных (до 5% или более от общей емкости слоя).

Существуют две основные концепции создания многослойных СХД с использованием программных решений. В рамках первой концепции с помощью программных средств установленных на контроллере СХД для пространства пользователя создается интерфейс доступа к хранилищу данных как устройства блочного типа, таким образом пользователь оперирует с ним как с обычным накопителем. Управление контроллером СХД осуществляется через программный интерфейс, а структура слоев скрыта от пользователя.

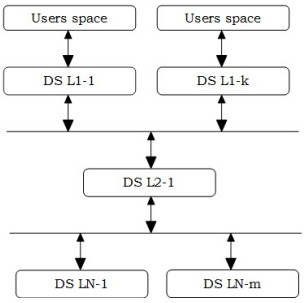
В рамках второй концепции возможности создания многослойных СХД реализуется с помощью программного менеджера накопителей в файловой системе (наиболее популярное решения для

хранилища данных модели DAS). Такой подход реализованы, например, в файловых системах как Bcachefs [3], ZFS [4] и Btrfs [5]. Основным преимуществом этой концепции является то, что пользователь может полностью управлять как слоями так и накопителями в СХД.

Так же важно отметить, что технологии тиринга и кэширования широко используются в решениях программных систем хранения данных, например, в распределенной файловой системе GlusterFS [7], или объектном хранилище данных Ceph [6].

Многоуровневые системы хранения данных.

В основу концепции многоуровневых систем хранения данных положен принцип объединения отдельных хранилищ данных в единую СХД с общим центром управления и мониторинга, основанную на модели программно-определяемых систем хранения данных (software-define storage или SDS). Основным компонентом многоуровневого СХД является контроллер, который осуществляет мониторинг и анализ статистики работы отдельных хранилищ данных, управляем миграцией между уровнями, и обеспечивает подключение новых хранилищ к СХД. Общая структурная схема многоуровневого СХД приведена ниже.



На верхнем уровне, как правило, располагают хранилища данных взаимодействующим с пространством пользователя. На втором

уровне, как правило, располагают хранилища данных основное назначение которых хранить «холодные данные» с хранилищ первого уровня, и поддерживать миграцию данных между ними. На третьем и ниже уровне могут находится хранилища архивных данных. Основным преимуществом такой модели СХД является оптимизация хранения данных на первом уровне и распределение нагрузки между хранилищами.

Для реализации многоуровневых СХД могут быть использованы различные программные решения, например, в качестве контроллера может быть использован open source проект CorpHD [8]. Одним из его преимуществ является интеграция с объектным хранилищем данных Ceph, и гибкая настройка архитектуры.

Программно-определяемые хранилища данных являются перспективным направлением развития систем хранения данных и поэтому Linux Foundation в октябре прошлого года объявил о старте нового проекта по созданию Open Controller for Software-Defined Storage [9], который станет основой для построения открытых решений SDS. Ознакомится с этим новым проектом можно на его официальном ресурсе [10].

Литература

[1] Flash memory summit `https://www.flashmemorysummit.com/`

[2] The official site NVM Express: // `http://www.nvmexpress.org/`

[3] Bcachefs: An advanced new filesystem for Linux // `http://bcachefs.org/`

[4] ZFS: Zettabyte File System (OpenZFS project) // `http://open-zfs.org/`

[5] Btrfs: Btrfs is a modern CoW filesystem for Linux // `https://btrfs.wiki.kernel.org/index.php/Main_Page`

[6] Ceph: The future of storage // `https://ceph.com/`

[7] Gluster: Storage for your Cloud // `https://www.gluster.org/`

[8] CorpHD: An open source software defined storage controller // `https://coprhd.github.io/`

[9] The Linux Foundation creates Open Controller for Software-Defined Storage // `https://www.linuxfoundation.org/announcements/linux-foundation-creates-open-controller-for-software-defined-storage`

[10] OpenSDS Project // `https://www.opensds.io/`

Выкарыстанне вольнага праграмнага забеспячэння ў ЛНУ імя Івана Франка і ЛНМУ імя Данііла Галіцкага *

С. Апуневич, Г. Злобін, Р. Рикалюк, П. Риковський,

Р. Шувар , Ľviv, Ukraine

The report analyzes 20 years of experience in the implementation of free software at Lviv Ivan I. Franko University and Lviv D. Galitsky University

На працягу многіх гадоў львоўская група карыстальнікаў Linux праводзіла агітацыйную працу па ўкараненні вольнага праграмнага забеспячэння ў навучальны працэс навучальных устаноў г. Львова. З 1998 г. пачалося ўкараненне АС Linux у навучальны працэс ЛНМУ імя Данііла Галіцкага. У 2002-2003 гг. было прынята рашэнне дэпутацкай камісіі па пытаннях адукацыі Львоўскага гарадскога савета аб закупцы класаў вучэбнай камп'ютарнай тэхнікі з прадусталяванай АС Linux. Для настаўнікаў школ, якія атрымлівалі камп'ютарную тэхніку з АС Linux, былі праведзены навучальныя курсы па асновах працы ў АС Linux [1]. У 2004 г. пачаўся праект па аптымізацыі дыстрыбутыва Debian для карыстання камп'ютарнай тэхнікі з Еўропы, якая падавалася экалагічным арганізацыям Украіны на бязвыплатнай аснове [2]. У той жа час пачалося карыстанне АС Linux у навучальным працэсе факультэта электронікі ЛНУ імя Івана Франка. У 2013 г. праз цяжкую сітуацыю з фінансаваннем факультэта з аднаго боку і ў сувязі з пачаткам ціску прадстаўніцтва Microsoft ва Украіне па выкарыстанні ліцэнзійнага ПЗ фірмы Microsoft у навучальных установах Украіны кіраўніцтва факультэта электронікі прыняло рашэнне аб пераводзе навучальных лабараторый на АС Linux.

На жаль, з-за разнамаснай камп'ютарнай тэхнікі ў школах (пачынаючы з ПЭВМ з працэсарам Intel 80486) і супраціў настаўнікаў інфарматыкі адбыўся пераход школ да неліцэнзійнай АС Microsoft Windows і праграмнага забеспячэння для яе. Затое ў ЛНМУ імя

*zlobingg@gmail.com, http://lvee.org/ru/abstracts/248

Данііла Галіцкага і ЛНУ імя Івана Франка выкарыстоўваецца па сённяшні дзень. Разгледзім сітуацыю па выкарыстанні ВПЗ ў гэтых навучальных установах.

I. Варта падкрэсліць, што пераход навучальнага працэсу ў ЛНМУ імя Данііла Галіцкага на ВПЗ адбыўся пад пагрозай праверкі ліцэнзій ўсталяванага ПЗ – «добразычліўцы» папярэдзілі рэктара ЛНМУ за паўгода да запланаванай праверкі. Сёння ва універсітэце налічваецца больш за 800 камп'ютарных працоўных месцаў, да некаторых з іх падлучаныя прылады друку, сканеры і шматфункцыянальныя прылады, ёсць 20 камп'ютарных класаў, 10 лекцыйных аўдыторый, якія выкарыстоўваюцца для правядзення ліцэнзійных экзаменаў КРОК і відэаканферэнцый, імітацыйны клас, у якім знаходзяцца 4 праграмуемых манекенаў.

Напрамкі выкарыстання СПО у ЛНМУ:

- **серверныя прымянення** — Linux (Debian, Ubuntu), Unix (FreeBSD);
- **навучальны працэс** — аперацыйная сістэма (Debian, Ubuntu), офісны пакет (LibreOffice), сродкі праграмавання (Bluefish), тэхналогіі тэлемедыцыны, відэаканферэнцсувя- зі, кантроль за напісаннем ліцэнзійных экзаменаў КРОК, імітацыйны медыцынскі навучальны клас, элементы сістэмы дыстанцыйнага навучання;
- **студэнцкая навуковая праца** — аперацыйная сістэма (Debian, Ubuntu), офісны пакет (LibreOffice), сродкі праграмавання (Bluefish), сістэмы кіравання базамі дадзеных (MySQL, Base).

Вынікам сказанага з'яўляецца той факт, што:

1. У цяперашні час універсітэт цалкам функцыянуе на ліцэнзійным ПЗ, праведзена адпаведная дакументацыйная і арганізацыйная праца.
2. Шырокі спектр даступнага вольнага праграмнага забеспячэння паказаў магчымасць поўнага забеспячэння працоўнага, навучальнага і навуковага працэсаў у універсітэце.
3. Пераход на свабоднае праграмнае забеспячэнне дазволіў не толькі пазбегнуць выкарыстання пірацкага праграмнага забеспячэння ў навучальным працэсе, але і паменшыць рызыку нападаў на працоўныя станцыі з боку зламыснікаў і аператараў станцый, паменшылася пагроза віруснай паразы і неабходнасць перманентнай пераўсталёўкі АС.

4. Працягваецца праца па выкарыстанні элементаў сістэмы дыстанцыйнага навучання.

II. У адрозненне ад ЛНМУ ў ЛНУ імя Івана Франка выкарыстанне ВПЗ не мае ўсёабдымнага характару, а адбываецца ў асобных падраздзяленнях універсітэта. Сёння ў ЛНУ імя Івана Франка налічваецца больш за 2000 камп'ютарных працоўных месцаў, некаторыя з падлучанымі прыладамі друку, сканерамі і шматфункцыянальнымі прыладамі, ёсць 37 камп'ютарных класаў, 20 лекцыйных аўдыторый, якія выкарыстоўваюцца для чытання лекцый з відэапраектарам і правядзення відэаканфэрэнцый. З мэтай забеспячэння ліцэнзійнай АС Microsoft Windows у 2013 была праведзена аплата двухгадовай падпіскі Dream Spark для дзесяці натуральных факультэтаў, аднак два факультэты (фізічны і электронікі і камп'ютарных тэхналогій) гэтай падпіскі не атрымалі з-за грубых памылак у аплікацыйных формах. На гуманітарных факультэтах забяспечанасць ліцэнзійным ПЗ з'яўляецца частковай. На факультэце электронікі і камп'ютарных тэхналогій ВПЗ прысутнічае ў пяці навучальных лабараторыях, у трох навучальных лабараторыях працуюць дзве аперацыйныя сістэмы: Linux і Microsoft Windows 7 або 10 дзякуючы бясплатнай падпісцы Dream Spark, якую ўдалося аформіць на кафедру радыёфізікі і камп'ютарных тэхналогій пасля правалу платнай падпісцы Dream Spark.

Напрамкі выкарыстання ВПЗ у ЛНУ імя Івана Франка:

- **серверы** — Linux (Debian, Open SuSE, CentOS), Unix FreeBSD;
- **навучанне** — аперацыйная сістэма (Debian, Open SuSE, CentOS, SlackWare), офісныя пакеты (OpenOffice, LibreOffice, Planner), сістэма дыстанцыйнага навучання Moodle, сродкі праграмавання (gcc, IDE Kuzya, IDE CodeBlocks, Qt Creator), матэматычныя пакеты (Octave, SciLab, SMathStudio, Maxima, Labplot), пакеты мадэлявання (Packettracer, Altera Quartus, KiCAD,
  Proteus), кліент-серверныя тэхналогіі;
- **студэнцкая навуковая праца** — аперацыйная сістэма (Debian, Open SuSE), офісны пакет (OpenOffice), сродкі праграмавання (gcc, Kuzya IDE, Qt Creator), матэматычныя пакеты (Octave, SciLab, Maxima, Labplot), сістэмы кіравання базамі дадзеных (MySQL), эмулятары апаратных сродкаў і аперацыйных сістэм;

– **навуковыя даследаванні** — аперацыйная сістэма (Debian, Open SuSE), офісны пакет (OpenOffice), сродкі праграмавання (gcc, IDE Kuzya, Qt Creator), матэматычныя пакеты (Octave, SciLab, Maxima, Labplot), арганізацыя вылічальных кластараў і кластараў высокай даступнасці (Scientific Linux, CentOS PaceMaker, Kickstarter, Webmin, SGE, Ganglia, OpenMPI, MPICH2, BLAS, FFTW, NorduGrid ARC, Condor, CUDA 5.0 production release).

Высновы:

1. Нягледзячы на працяглы і няпросты працэс пераходу на ВПЗ у ЛНМУ імя Данііла Галіцкага апынуўся ўдалым [3]. Варта адзначыць актыўную падтрымку гэтага працэсу з боку кіраўніцтва ЛНМУ, якая, на думку аўтараў дакладу, мела асабліва матэрыяльны характар. Дзякуючы пераходу на ВПЗ, ЛНМУ імя Данііла Галіцкага не толькі зэканоміў значныя грашовыя сродкі на набыцці ліцэнзій на ПЗ, але і пазбег штрафных санкцый за выкарыстанне неліцэнзійнага ПЗ.
2. Нягледзячы на тое, што ў ЛНУ імя Івана Франка ёсць неабходны кадравы патэнцыял, свабоднае праграмнае забеспячэнне выкарыстоўваецца не ва ўсіх падраздзяленнях універсітэта. Абумоўлена гэта тым, што кіраўніцтва універсітэта не надае належнай увагі ліцэнзійнай чысціні праграм.
3. Прыходзіцца канстатаваць, што выкарыстанне прапрыетарнага (і нават неліцэнзійнага) праграмнага забеспячэння прыводзіць да разрыву паміж навучальным працэсам у ЛНУ імя Івана Франка і вытворчым працэсам у IT-фірмах Украіны, што дае падставы некаторым аўтарам сцвярджаць пра састарэласць навучальнага працэсу ў навучальных установах Украіны і у прыватнасці ў ЛНУ імя Івана Франко [4].

Літаратура

[1] Матеріали науково-методичного семінару «Використання ОС Linux у навчальному процесі середніх і вищих навчальних закладів». Львів-2003, 60 с.

[2] С. Апуневич, В. Бойко, Г. Злобін, В. Семенюк, С. Кудрик «Linux — це просто як Borsch» Львів-2006, 116 с.

[3] Курьянович О.В. Використання вільного програмного забезпечення в ЛНМУ імені Данила Галицького. Збірник науко-

вих праць сьомої науково-практичної конференції FOSS Lviv 2017. – Львів, Львівський національний університет імені Івана Франка, с. 57

[4] http://igate.com.ua/news/13725-top-5-mifov-kotorye-meshayut-zhit-ukrainskim-programmistam

Typical mistakes when submitting a new code to Linux kernel*

Andy Shevchenko, Espoo, Finland

Code review is a crucial part of developing a stable and robust open source project, such as Linux kernel. New comers and surprisingly many active and known contributors have been doing same mistakes, like using non-scalable APIs, over and over. How can we eliminate or avoid them? Looking at the patterns the best practices would be suggested. In addition, relatively new Linux kernel APIs, such as Unified Device Properties, will be discussed.

Introduction

According to the Linux kernel v4.11 development statistics [1] the code base is grown by nearly 300,000 lines. It means that from now on many of new drivers and other modules are in the vanilla kernel. While performing review on some of the drivers I have noticed that even oldbies have been making same mistakes over and over when submitting a new code. Majority of the issues is poor knowledge of the library APIs in the kernel. For drivers it includes, for example, Device Resource Management API [2], [3] or Unified Device Properties API [4], while other modules re-implement simple helper functions, like hex2bin(), from the internal library. Besides that driver developers often are focusing on single CPU architecture or hardware platform which makes the code not so scalable. In the following chapters the most occurred mistakes will be considered. In some cases the non-obvious issues would be discussed, e.g. devm_request_threaded_irq() coexistence together with tasklets and 64-bit hardware registers access and handling.

Device Resource Management

Device Resource Management API [2] [3] is quite old idea (it dates back to 2007!), that is still evolving, though developers don't use or use it in a suboptimal way.

*andy_shev@mail.ru, http://lvee.org/ru/abstracts/240

The most recent example I have met in [5] is the following piece of code:

```
ret = of_address_to_resource(np, 0, &res_xbar);
if (ret) {
    dev_err(dev, "Failed to get xbar resources");
    return ret;
}
if (!devm_request_mem_region(dev, res_xbar.start,
    resource_size(&res_xbar), res_xbar.name)) {
    dev_err(dev, "Failed to get xbar resources");
    return -ENODEV;
}
xbar_membase = devm_ioremap_nocache(dev, res_xbar.
    start, resource_size(&res_xbar));
if (!xbar_membase) {
    dev_err(dev, "Failed to remap xbar resources");
    return -ENODEV;
}
```

Obviously the developer is not familiar with non-device tree platforms and Device Resource Management API. The better approach is just missed, i.e.

```
res_xbar = platform_get_resource(pdev,
    IORESOURCE_MEM, 0);
xbar_membase = devm_ioremap_resource(dev, res_xbar)
    ;
if (IS_ERR(xbar_membase))
    return PTR_ERR(xbar_membase);
```

In the end we got much simpler code, resource provider agnostic — it will work on Device Tree, ACPI, or legacy platform data enabled platforms.

Of course there are some nuances when the driver is converted to new API. One of them is a good understanding when it's safe to use devm_request_threaded_irq(). For example, if driver is using tasklets the IRQ handler should be removed before tasklets. It will guarantee that no more tasks are scheduled on non-existing data.

Nevertheless previously mentioned examples maybe are not the best ones when we speak about hardware IP blocks which would have been using on various CPU architectures and platforms, where Unified Device property API is a big helper.

Unified Device Properties

Back to 2014 Rafael Wysocki had submitted a new common API [4] for drivers to get device properties in resource provider agnostic way. It means legacy platform data, ACPI, or Device Tree resource providers do not require special handling anymore in the drivers.

Besides that few other subsystems, i.e. GPIO, I2C, SPI, were updated accordingly to hide resource provider details in their core parts.

One of the recent and nice example of conversion to new API is the commit 95129b6f0806 ("NFC: pn544: Get rid of code duplication in probe()") /citeShevchenko6 along with few preparations in vanilla kernel.

While the above is related mostly to the drivers, the small parts of the rest of the code might have been reinventing the wheel, such as hex2bin() helper or even more often helpers to dump small buffers in hexadecimal or escaped format.

Special extensions of %p

How often do you need to dump several bytes from the buffer in the log to reveal some details about an error? I'm pretty sure it have been happening sometimes. Linux kernel vsnprintf() and thus its derivatives implements special extensions of %p specifier [7]. In particular, instead of reinventing hexdump functionality the developer may just use something like:

```
u8 data[100};
...
pr_debug("Buf:␣%*phC\n", (int)sizeof(data), data);
    /* only first 64 bytes! */
```

That code will dump first 64 bytes in hexadecimal format with ':' (colon) as a delimiter.

There are also many other extensions, e.g. printing buffer in escaped format, IPv4 and IPv6 addresses, MAC addresses, architecture dependent types, device resources and so on.

Recommendations on how prepare changes to Linux kernel

When developer consider the code tested and ready for submission the last things to be worth doing are:

– **Check the code again for duplication.**

- **Take the material from the above chapters into consideration when doing drivers.**
- **Establish internal mailing list for review and review process if it's not done yet.**
- **If in doubt, feel free to ask.**

References

[1] : 4.11 Kernel development statistics https://lwn.net/Articles/720336/

[2] : Devres – Managed Device Resource https://www.kernel.org/doc/Documentation/driver-model/devres.txt

[3] : The managed resource API https://lwn.net/Articles/222860/

[4] : Unified Device Properties Interface for ACPI and Device Trees http://events.linuxfoundation.org/sites/events/files/slides/unified_properties_API_0.pdf

[5] : MIPS: lantiq: Convert the xbar driver to a platform_driver https://patchwork.ozlabs.org/patch/765088/

[6] : NFC: pn544: Get rid of code duplication in probe https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=95129b6f0806d1ba6109dc1df4d9753ad3d4a94c

[7] : printk() formats https://www.kernel.org/doc/Documentation/printk-formats.txt

Make your own USB gadget*

Andrzej Pietrasiewicz, Warsaw, Poland

A widely accepted standard for adding functionality to computer systems is the USB bus. GNU/Linux systems have long been supporting USB hosts, but a less known fact is that they also support USB devices. The traditional way of preparing support for devices includes creating a kernel module, which makes the process significantly more difficult. A new approach is presented, which requires only shell scripting.

Introduction

Serial communication has been present since very early days of electronic computers in the semiconductor era. In serial communication in principle the data stream is finally decomposed into individual bits which are transmitted over the medium and such a process can be implemented in various ways. The most widely employed standard used to be RS-232-C, which was formalized in 1969 [1]. Since then it has found widespread acceptance and served its purposes very well for the next 20 years. However, with processing speed increasing rapidly a demand for faster transmission appeared. On the other hand computer systems miniaturization progressing stimulated dropping the usage of voltages higher than 5V. These, among others, have been the reasons to introduce a newer standard, better suited for the then current and future needs. Such a standard was USB, the Universal Serial Bus [2], first specified in 1996. The standard was and still is successful, and there have been a number of major updates to it in 1998 [3], 2000 [4], 2008 [5] and 2013 [6].

USB high level architecture

USB is a bus with one host node and many (up to 127) device nodes physically connected in a „tree" topology. According to the standard the logical topology is in fact a „star" topology with the host being the central unit. The roles in the communication are strictly defined.

*andrzejtp2010@gmail.com, http://lvee.org/ru/abstracts/241

In particular, a dedicated piece of hardware is needed at both sides of the cable (host-device) to maintain the physical aspects of data transmission over the medium and to maintain logical interaction between the communicating parties. While the host part is in widespread use in all modern PCs (and other equipment) and is supported by GNU/Linux, the devices tend to be specialized appliances or gadgets like printers, pendrives, broadband modems or digital TV receivers.

USB gadget support in GNU/Linux

GNU/Linux systems have in-kernel support for the device part of USB communication, which means that provided the system running GNU/Linux is equipped with appropriate hardware it can be turned into a full USB device controlled by Linux kernel and userspace running on it [7]. Device (also called gadget) supporting code is organized in a so-called composite framework [8], which factors out code common to all USB functions implemented on the one hand, and delegates function-specific implementation to separate „function" files. USB standard requires that a device provides at least one configuration, and in a configuration there are a number of interfaces providing interesting services to the USB host. There can be more than one configuration, but only one can be active at a time. Composite devices are possible, where there are a number of functions (interfaces) and configurations available, e.g. a mass storage device composed with a broadband modem, where the mass storage device contains driver software for the modem. While Linux kernel provides a rather wide choice of USB functions (interfaces), they need to be somehow combined into a device. The traditional way of composing such USB devices has been to provide dedicated kernel modules, where the choice of USB functions, their composition into configurations and the gadget identity is hardcoded. Such an approach makes it difficult to compose new devices as it requires the skills needed to program Linux kernel. And even provided this difficulty is overcome, the code of the new module must be maintained either in-house or accepted by the community to become a part of officially maintained Linux kernel. Either of the two is not necessarily an easy task.

New interface for USB gadget composition

Around the beginning of 2015 a process of adding support for composing USB gadgets at runtime out of existing USB functions has been completed. The new approach still uses the composite framework, but the added functionality allows the components to be specified and composed into a gadget at runtime, which means that it is the user (the system administrator) who controls the gadget creation process. The interface used for this purpose is based on ConfigFS, which in turn is a pseudo file system similar to sysfs, so all interactions of the user controlling gadget creation use well known file system concepts. These are: directory creation/removal (creating gadget's configuration), symbolic link creation/removal (grouping things together, e.g. assigning functions to configurations), file writing/reading (specifying gadget parameters, querying gadget parameters).

The new interface for runtime composing USB gadgets out of existing functions allows new interesting possibilities. Among them is the helper userspace library, libusbgx [9]. It encapsulates the „bare" ConfigFS interface in an easy-to-use C interface. It also supports so called gadget schemes, which allow specifying gadget's composition and identity in a declarative style configuration file, rather than using an imperative style shell scripting. The former allows the gadget creator to concentrate on what the gadget is going to be composed of, while the latter forces the creator to also know how the ConfigFS interface works. Such declarative gadget schemes enable future use of the framework e.g. for distributing gadget schemes as installable applications.

References

[1] EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange. Washington, USA: Electronic Industries Association, Engineering Department. 1969. OCLC 38637094 / ITU-T V.24

[2] Universal Serial Bus Specification, 1996 `http://fl.hw.cz/docs/usb/usb10doc.pdf`

[3] Universal Serial Bus Specification, 1998 `http://esd.cs.ucr.edu/webres/usb11.pdf`

[4] Universal Serial Bus Specification, 2000,`http://www.usb.org/developers/docs/usb20_docs/usb_20_033017.zip`

[5] Universal Serial Bus Specification, 2008,http://www.usb.org/developers/docs/documents_archive/usb_30_spec_070113.zip
[6] Universal Serial Bus Specification, 2013,http://www.usb.org/developers/docs/usb_31_080416.zip
[7] Andrzej Pietrasiewicz, „Make your own usb gadget", LinuxCon North America 2014, Chicago, USA,http://events.linuxfoundation.org/sites/events/files/slides/LinuxConNA-Make-your-own-USB-gadget-Andrzej.Pietrasiewicz.pdf
[8] Michał Nazarewicz, „The USB composite framework", https://lwn.net/Articles/395712/,2010
[9] https://github.com/libusbgx/libusbgx

О Модели Данных SQL/JSON

Андреев Юрий, Санкт-Петербург, РФ*

An SQL/JSON model is a part of SQL 2016 Standard[1] which describes a JSON data model for SQL. It describes a *jsonpath* data type as a path language and nine functions to deal with JSON data. A new type of queries will be shown on a special branch of PostgreSQL.

Введение

В новою редакцию стандарта SQL[1] была включена SQL/JSON модель. Это означает, что стандарт теперь предполагает работу со слабо-структурированными данными в JSON формате[2]. СУБД PostgreSQL имеет долгую историю поддержки такого типа данных и теперь идет работа по поддержке новых возможностей стандарта[4]. Далее мы покажем, какие преимущества дает способ хранения в JSON формате и как предполагается делать запросы к таким данным.

Описание

Для запросов в стандарте описаны девять новых функций, с префиксом `JSON_ (JSON_{OBJECT, QUERY, ...})`, а также *jsonpath* — язык для навигации по JSON. Это частичный аналог XPath для XML. Однако, в отличии от последнего, его синтаксис был продиктован уже имеющимися конструкциями в javascript. Выражение начинается со знака доллара, переход к дочерним элементам осуществляется оператором 'точка'. (см. подробнее в [3]). Путь передается во втором аргументе функций.

Пример использования

Хорошей иллюстрацией может служить список достопримечательностей. Допустим, мы хотели бы хранить дополнительную информацию об интересных местах. Если мы возьмем архитектурную

*andreev.yurij@gmail.com, https://lvee.org/ru/abstracts/247

достопримечательность, то в формате JSON информация могла бы быть записана следующим образом:

```
{
    "nearest_city": "city_name"
    "creator": "author_name"
    "style": "style_name"
    "date": {
        "start_centery": number
        "end_centery": number
    }
}
```

В тоже время, для природных объектов бессмысленно указывать имя создателя или дату, и могут понадобиться новые ключи: сведения о территории, высоте, глубине. С другой стороны, и те и другие могут содержать имя ближайшего населенного пункта, географическое расположение и прочее.

Записать информацию в базу мы можем с помощью функций-конструкторов `JSON_OBJECT, JSON_ARRAY`.

Если наши данные содержатся в колонке `details`, то запрос на наличие создателя (ключ `creator`) может выглядеть следующим образом:

```
SELECT
    JSON_EXISTS(details, '$.creator')
FROM table_name
WHERE id = object_id;
```

И вернет булево значение (`t/f`). Вывести время начала строительства можно функцией `JSON_VALUE`:

```
SELECT
    JSON_VALUE(details, '$.date.start_centery')
FROM table_name
WHERE id = object_id;
```

Тут заметим, что вообще период строительства может описываться по разному, в зависимости от точности имеющейся информации. Поэтому могут понадобиться разные ключи и, соответственно, разная логика в запросе.

Заключение

Преимущество JSON/SQL модели, помимо прочего, заключается в том, что мы можем эффективно работать с данными, не имеющими четкой структуры. Компактно хранить и лаконично писать запросы к таким данным, добавлять новые ключи, без необходимости изменения схемы таблицы (что затратно).

Литература

[1] Стандарт SQL, ISO/IEC 9075-2:2016, `https://www.iso.org/standard/63556.html`

[2] Стандарт JSON, RFC 7159, `https://www.rfc-editor.org/info/rfc7159`,

[3] Описание jsonpath, `http://goessner.net/articles/JsonPath/`

[4] Исходный код PostgreSQL, sqljson branch, `https://github.com/postgrespro/sqljson/tree/sqljson/`

Электромиографическое распознавание движений пальцев руки человека на базе Arduino*

Вадим Шамонин, Брест, Belarus

An open hardware project for the electromyography measurements is presented. Arduino platform is engaged in getting data from the epidermic sensors and passing them to the receiving software to detect and classify muscle activity. Results of fingers movements recognition are presented for two sensors placement approaches.

Введение

В настоящее время разрабатывается большое количество устройств, позволяющих использовать пальцы рук в качестве источника управляющих сигналов. Все их можно разделить на три категории:

- механические манипуляторы, повторяющие движения человеческой руки;
- взаимодействие с объектами в системах виртуальной и/или дополненной реальности;
- мониторинг биологических параметров организма.

Применения электромиографического метода (ЭМГ) для этого круга задач делает возможным не только распознавание степени сгибания пальцев с высокой точностью, но также позволяет использовать полученные данные для выявления медицинских отклонений пациента. ЭМГ также предоставляет новые возможности в области реабилитации после инсульта. К примеру, антропоморфная робототехника, применяемая в медицине, позволяет регулярно разгибать руку у человека, который в результате болезни потерял такую возможность (это называется нейрореабилитацией, и применяется в том числе после инсультов и нейротравм).

*shadimx@gmail.com, http://lvee.org/ru/abstracts/245

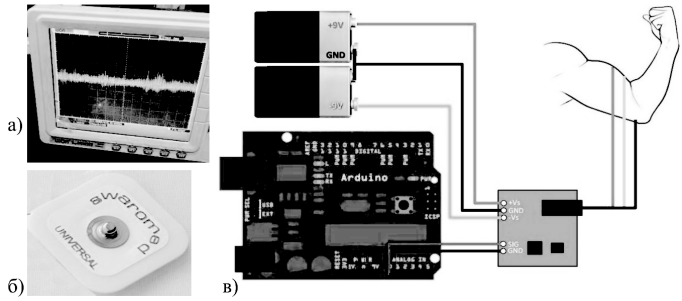


Рис. 6.1. Используемые накожные электроды Ag/AgCl (а), визуализация сигналов с них, снятых без усиления (б) и схема подключения усилителя (в)

Аппаратная платформа и особенности реализации

Разработанное устройство построено на основе платформы Arduino. Проект разрабатывается как open hardware, наработки доступны по адресу `https://github.com/shadimx`.

При помощи неё данные обрабатываются при подключении к ПК. Электрическую активность мышц при помощи накожных Ag-электродов, широко используемых в медицинских целях при записи ЭКГ (рис. 6.1). Для более точного выявления биопотенциалов используется специализированный OpenHardware-модуль цифрового усиления сигналов для Arduino — muscle sencor v3 от sparkfun (рис. 6.1).

Особенности измерений

Для решения задачи распознавания движений пальцами важную роль играет методика регистрации ЭМГ. В работе рассмотрены 2 методики регистрации сигналов при использовании неинвазивной ЭМГ.

В рамках 1-й методики положения №4 и №5, также как и №6 и №7 (рис. 6.2) регистрировались одним электродом. Сигналы записывались относительно канала “Reference”, положение которого выбиралось на участке выше локтя, на котором отсутствуют сокращения мышц при движении пальцами. Электрод канала “Ground” располагался в районе плечевого сустава. В рамках 2-й методики

те же положения регистрировались отдельными электродами, при положении электродов каналов “Reference” и “Ground” таком же, как в 1-й методике. Кроме того, снималось также 5 дифференциальных каналов между положениями 1 и 2; 3 и 4; 5 и 6; 7 и 8; 9 и 10. В результате методика с использованием дифференциальных каналов позволила улучшить отношение сингал/шум, что обеспечило существенно лучшее качество распознавания движений классификатором (рис. 6.2). Отметим, что использование некоторых каналов не улучшает работу классификатора. Для 2-й методики достаточно использовать 5 дифференциальных и 2 отдельных канала.

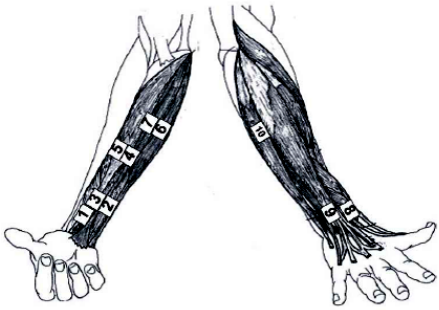


Рис. 6.2. Положение электродов при регистрации ЭМГ

Практическим методом была установлена взаимосвязь амплитуды получаемого сигнала на конкретном участке мышцы от степени сгибания пальцев руки.

В результате проверки выбранных методик измерения были получены 65% и 95% вероятности правильного определения сгибания пальца для 1-й и 2-й методик соответственно (рис. 6.3). Время работы наиболее удовлетворительного алгоритма классификации составляет 0,5 мс, что позволяет применять его в режиме реального времени.

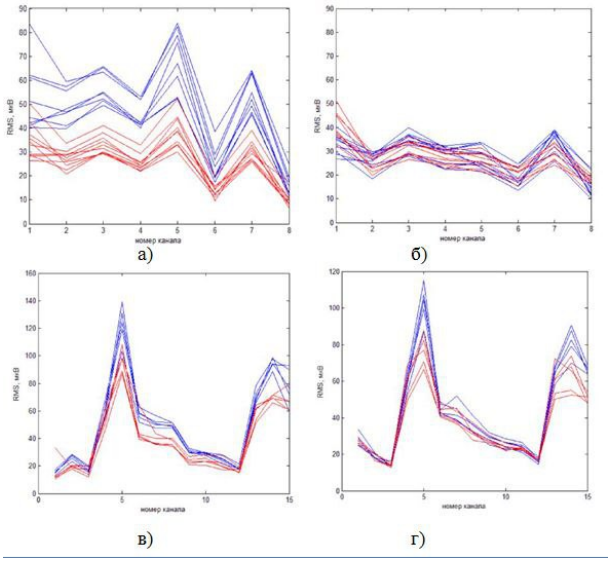


Рис. 6.3. Примеры распознавания движений мизинца (а, в) и указательного пальца (б, г) в рамках 1-й (а, б) и 2-й (в, г) методик

Литература

[1] Muscle Sensor v3 Kit `https://www.sparkfun.com/products/retired/11776`

Hubzilla — введение, возможности, Hubzilla-сообщество *

Gustav Wall, Oldenburg(Oldb), Germany

Hubzilla is a free and open source platform running on a special kind of web server, called a «hub», that can connect to other hubs in a decentralised network called «the grid», providing sophisticated communications, identity, and access control services which work together seamlessly across domains and independent websites. It allows anybody to publicly or privately publish content via «channels».

Что такое Хабзилла?

Hubzilla (Хабзилла, в дальнейшем Hbz) — это платформа в форме децентральной сети, созданная с целью обеспечить возможность

*tech@sprechrun.de, http://lvee.org/ru/abstracts/244

общения без цензуры и с соблюдением приватности. Сама структура Hubzilla сети и Hbz протокол Zot [1] обеспечивают комфортное общение всех участников сети на равных. Hubzilla обладает многими полезными свойствами. Важность, значение каждого из этих свойств в каждом конкретном случае зависит от того, для каких целей вы используете это платформу.

Свойства, важные для всех — вездесущность и доступность

Два замечательных свойства Hbz понятны, важны и привлекательны для всех пользователей этой платформы — вездесущность и доступность.

вездесущность в контексте Hbz это:

- с одной стороны *вездесущность в буквальном глобальном смысле*, т.е. способность и возможность для участников сети Hubzilla получить доступ и пользоваться сервисами этой сети везде, где участник или участница сети имеют возможность войти в свою учётную запись и контактировать своего виртуального *кочующего двойника*. Неисправность одного из узлов не оказывает существенного влияния ни на работоспособность отдельных участников сети, ни на работоспособность сети в целом.
- вездесущность в смысле возможности *с небольшими затратами времени обменивать сообщения с большим количеством контактов*. Эта вездесущность обеспечивается за счёт того, что при вводе сообщения участники Hbz-сети простым вводом символа **@(собака)** с последующим вводом какой-нибудь буквы или нескольких букв могут показать **все контакты**, содержащие введённую последовательность символов. Такой контакт может быть и рассылкой. Каждый подписчик рассылки может получить сообщение без того, что отправитель и получатель до этого общались.

Необычно **высокие значения доступности сервисов** в смысле ITIL [2] являются ещё одним уникальным свойством Hbz сети и достигается благодаря вездесущности его участников. Доступность самой сети и вездесущность её участников — это две стороны одной

[1] `https://hub.libranet.de/help/developer/zot_protocol`

[2] `https://itsm365.ru/blog/upravlenie-dostupnostyu-availability-management-po-itil/`

медали. В сети Hbz доступность, стабильность сети растёт пропорционально с увеличением количества её участников.

Кочующий двойник и другие удобства

Кочующий двойник является одним из центральных понятий в Hbz-сети, которое позволяет реализовать в сети эффект вездесущности. Уникальное свойство вездесущности возможно благодаря тому, что участники могут на своё усмотрение с согласия оператора узла размещать свою учётную запись параллельно на разных узлах сети, удалять эту запись, что приводит к **повышению надёжности доступа к учётной записи практически до 100%**. Если участник изменит данные на одном из узлов, обновляются данные и на узлах-двойниках. Непрерывно в режиме реального времени двойники обменивают данные, благодаря чему любой из двойников постоянно знает то, что знает и его протагонист.

Доступно по умолчанию

Следующими возможностями могут пользоваться владельцы узла после установки Hbz **по умолчанию**:

- адресная книга (контакты)
- сайт, веб-ресурс
- социальные сети
- форумы
- календарь мероприятий
- вики
- чат
- облачное хранилище файлов
- набор приложений (apps), который может расширяться в соответствии с потребностями владельца узла.

Обмен сообщениями в сети производится в соответствии с https-протоколом. Дополнительно имеется возможность закодировать часть или всё сообщение паролем.

Многосторонне одарённый двойник

Общительность — способность обменивать сообщения с другими сетями, например такими как Диаспора[3] или Фриндика[4].

Всеядность — способность переваривать, интегрировать множество форматов — *HTML*, простой текст, *bbcode, markdown, application/x-php*, благодаря чему пользователи могут в своих сообщениях комбинировать содержимое из разных источников с малыми затратами времени.

Универсальность, неприхотливость — в зависимости от нагрузки на конкретном узле Hbz, зависящей от количества зарегистрированных пользователей, от активности, в том числе от количества контактов этих пользователей и от количества посетителей на узле можно установить этот узел на соответствующем железе — Hbz работает на мини-компьютере Raspberry Pi [5] так же надёжно, как на сервере с мощнейшими AMD- или Intel-Xeon- мультипроцессорами.

Виртуальный кабинет — рабочее место двойника

Zot — это [6] специальный протокол, который обеспечивает обмен информацией, управление двойниками и управление доступом в децентральной сети, состоящей из независимых узлов (hubs). Эту сеть в Hbz-контексте часто называют грид (grid). Hbz-узлы выполняют различные функции, важнейшие из которых для владельца узла:

- предоставление **виртуального кабинета** *кочующему двойнику* владельца узла, который имеет в своём распоряжении:
  - контакты владельца узла
  - место встречи с другими двойниками
  - своего рода **передающую станцию** с одним или множеством каналов или **издательство** с одним или множеством изданий
  - систему для **прецизионного управления доступом** к каждому из объектов на узле

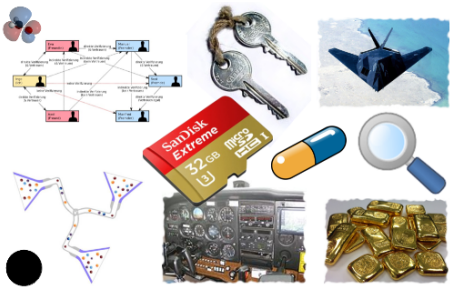
[3] `https://ru.wikipedia.org/wiki/Diaspora`
[4] `https://ru.wikipedia.org/wiki/Friendica`
[5] `https://www.raspberrypi.org/`
[6] `https://hub.libranet.de/help/developer/zot_protocol`

– двойник в соответствии с доверенными ему полномочиями выполняет массу дел, связанных с общением с другими двойниками из сети Hbz или с гостями сети
– практически неограниченное количество принадлежащих владельцу узла **виртуальных кабинетов**, куда он может пустить как *постояльцев* чужих двойников:
  – каждый из чужих двойников имеет в своём распоряжении практически те же возможности, что двойник владельца узла, особенно если постоялец и владелец доверяют друг другу или они заключили соответствующий договор о найме виртуального кабинета.

Таким образом сеть Hbz представляет собой пространство, среду обитания для двойников, объём полномочий которых предопределяется оригиналом.

Возможности применения

Картинки отображают образно или схематично возможные сценарии применения Hbz. MicroSD отображает тот факт, что сервер Hubzilla можно установить на этой карте размером с ноготок.



Литература

[1] Hubzilla помощь — `http://hub2.sprechrun.de/help/`
[2] The history of Hubzilla — `http://www.talkplus.org/blog/2016/the-history-of-hubzilla/`

[3] Аккаунт, двойник автора (Gustav Wall) в сети Хабзилла — https://%hub.libranet.de/channel/nmoplus
[4] Web of Trust Von https://commons.wikimedia.org/wiki/User/Ogmios — https://upload.wikimedia.org/wikipedia/commons — https://upload.wikimedia.org/wikipedia/commons/4/4e/Web_of_Trust.svg,CCBY-SA3.0, https://commons.wikimedia.org/w/index.php?curid=30127548
[5] Zwei gleiche Schlüssel Von Sebastian Hartlaub — Eigenes Werk, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=696513

Бізнэс-мадэлі для супольнай уласнасці або як прадаваць атамы і не біты.**

Mikhail Volchak, Minsk, Belarus

In the presentation different business models based on the digital commons were summarized. The first point is a consideration who controls a modern pool of resources and which characteristics are critical for the resource management. In the second point we discuss how the digital revolution change price-creation of a digital content. Further we elaborate how to make money and foster development based on Creative Commons principles. The last section describes how businesses can operate based on non-market premises and develop the third way in the content creation, distribution and consumption.

Бізнэс мадэлі для супольнай уласнасці або як прадаваць атамы і не біты.

«Кожны збытак стварае новы дыфіцыт» (Свабодны, Андэрсан)

Найбольш частае пытанне, якое задаюць слухачы падчас лекцый і выступаў па тэме распаўсюду ідэй Creative Commons: як творчы чалавек можа зарабіць у Беларусі сабе на жыццё або стаць багатым выкарыстоўваючы Creative Commons ліцэнзіі?

Перад тым як пачаць пералічваць што значыць быць зробленым творчым агульным, звернемся да асноўваў сучаснай палітэканоміі. Тут будзе палягаць частка адказу мета ўзроўню, а менавіта хто кіруе рэсурсамі.

Агульнае супольнае

Гістарычна склалася 3 асноўных падыхода у кіраванні рэсурамі:

1. Дзяржаўная манаполія, значыць, што урад кантралюе вытворчасць, размеркаванне і спажыванне рэсурсаў,
2. Падыход рынкавых адносінаў кантралюе рэсурсы інстытутам прыватнай ўласнасці.
3. Супольныя рэсурсы — гэта значыць, што рэсурсы кантралююцца калектыўна, грамадзянамі.

**fannrm@gmail.com, http://lvee.org/ru/abstracts/253

Доўгі час у Беларусі мы мелі сітуацыю, калі рынкавыя і грамадскія механізмы былі ўлучаны ў адзін — дзяржаўны, і кантраляваліся урадам (або партыяй).

Апошнія 25 год незалежнасці Беларусь абрала рынкавы механізм. У разуменні шляхоў развіцця замацавалася два асноўных падыхода. Першы кажа, пра тое, што дзяржава павінна кантраляваць рэсурсы і эканоміку, а другі прасоўвае свабодны рынак, які павінен выканаць функцыю мадэрнізатара эканомікі і жыцця грамадзян.

Але ў гэтай схеме ёсць недаацэненая кампанента — грамадства. Якое прапаноўвае іншыя варыянт кіравання рэсурсамі у адрозненні ад дзяржаўнага або неаліберальнага капіталізма. Гэты падыход базуецца на — супольным, або супольнай уласнасці.

Эліанор Острам выдзяляе асноўныя 4 кампаненты ў кіраванні рэсурсамі:

- Характарыстыка рэсурсаў. Яны натуральныя або створаны чалавекам, іх дэфіцыт, або лішак, маюць фізічную або лічбавую аснову?
- Нормы і правілы. Што кантралюе рэсурсы: нефармальныя нормы, або фармальныя правілы — законы?
- Людзі і працэсы. Хто мае доступ і можа карыстацца імі, хто мае ўладу над рэсурсамі, прамую або ўскосную?
- Мэты. Як карыстаецца здабытае, рэсурс пастаянна дадаецца (як вікіпедыя) або выймаецца (як нафта)?

У 1980-90х адбылася лічбавая рэвалюцыя і Столман з паплечнікамі аб'явілі 4 свабоды для эканомікі супольнага (або грамадскага набытку на падставе праватных дамоваў):

- карыстацца праграмай;
- змяняць і вывучаць праграму;
- распаўсюджваць яе;
- распаўсюджваць у тым ліку зменены я копіі.

Гэты рух надалей стварыў перадумавы для лічбавага супольнага, у які была уключаны не толькі код, але і кантэнт.

Інтэрнет даў тры асноўныя магчымасці:

- Прыблізіць кошто копіі амаль да нуля, што дае магчымасць неабмежаванага тыражу.
- І аўтарства стала ўніверсальнай з'явай дзе амаль кожны можа быць аўтарам і распаўсбджваць свой кантэнт, у тым ліку бясплатна.

– Што ў сваю чаргу стварыла новыя ўмовы для кіравання рэсурсамі. Дзе стваральнікі кантэнту канкуруюць не толькі з тымі, хто прадае кантэнт, але з аўтарамі, які распаўсюджвае яго бясплатна.

Такім чынам, аўтараў, якія жадаюць прытрымліваца супольнага падыходу да кіравання рэсурсамі стаіць няпростае пытанне: як захоўваючы каштоўнасць адкрытага доступа, да сваёй творчасці зарабіць грошы?

Што рабіць з нулявым коштам?

Ніжэй прапанаваны падыходы сабраныя Пола Стэйсці і Сарай Персан з 24 пачынанняў які выкарысталі наступныя прынцыпы у сваіх бізнэс мадэлях:

– Больш аўдыторыі. Creative Commons не ствараюць вірусны эффект, але яны ствараюць ўмовы для свабоднага распаўсюду. Тым самым правакуючы эффект далучэння да большасці.
– Прызнанне і рэпутацыя. Creative Commons механізмы патрабуюць пазначэнне аўтарства твораў, што з'яўляецца нормай для шматлікіх грамадстваў і супольнасцяў. Таким чынам аўтарства як значны атрыбут для дадання вартасци творчаму прадукту.
– Раскрутка і прасоўванне. Электронная версія прадукт для прасоўвання, а фізічная як тавар, такім чынам эканомяцца грошы на рэкламу.
– Гуляць па сваіх правілах. Creative Commons дае магчымасць вызначаць свае правілы гульні і ўцягнення аўдыторыі, на рынку ў тым ліку.
– Супрацоўніцтва. Creative Commons дае магчымасць перапрацоўваць творы і гэта ўключае спажыўцоў (напрыклад, мастакоў і музыкаў) у сумесную творчасць, тым самым павялічваючы сувязі з творамі.

Як зарабляць грошы?

Галоўнае пытанне ў гэтай часцы, што ж стварае дадатковую вартасць, за якую людзі гатовы плаціць?

Выкарыстоўваючы стандартныя рынкавыя мезанізмы:

Прадастаўленне кастамізаванага сэрвіса. Кантэнту бясплатнага шмат, але, каб атрымаць патрэбны толькі сабе кантэнт, неаб-

ходна заплаціць. Прадастаўленне фізічнай копіі. Электроная копія бясплатна, але фізічная рэалізацыя за грошы. Напрыклад, дызайн мэблі або электронікі.

– Аплата за персанізаваную версію. Персанальны досвед творчасці. Напрыклад, канцэрт, або лекцыя.
– Продаж свэга і мерчэндайза. Цішоткі і т.п.
– Сбор грошай з рэкламадаўца і спонсараў. Традыцыйны падыход.
– Стваральнікі кантэнта аплочваюць месца. Напрыклад, платформа навуковых артыкулаў.
– Аплата транзакцый. Напрыклад, платформа, на якой узаемадзенічаюць кліенты і творцы.
– Аплата прыладаў. Платформы збіраюць грошы за выкарыстанне прасунутых інструментаў.
– Выкарыстанне брэнда. Напрыклад, брэнда, які асацыюецца з пэўнай якасцю, накірункам, даверам.

Але ёсць накірунак, які больш факусуецца на не рынкавых механізмах, а на ўзаемнасці:

– Сяброўскія ўнёскі або інтывідуальныя ахвяраванні. Чым больш тых, хто атрымлівае дадатковую вартасць, тым больш тых, хто верагодна можа падтрымаць праект.
– Свабодны кошт. Карыстальнікі плоцяць столькі колькі пажадаюць, як акт удзячнасці.
– Краўдфандынг. Вырошчванне супольнасці для таго, каб яна не толькі падтрымала, але і хацела поспеха праекту.

Злучаць людзей

Акрамя выкарыстання Creative Commons, як прававога інструмента, з'яўляецца сродкам нагадвання, што за кожным творам стаіць чалавек, тым самым? ствараючы пачуццё абавязку. Далей пералічана шэраг ідэй, якія выкарыстоўваюцца ў новых бізнэс мадэлях:

– Расказаваць гісторыі творчасці і не баяцца крытыкі, каманда не лашчоны брэнд.
– Адкрытасць да камунікацыі і адказнасць. Адказнасць — не значыць канфармізм, а магчымасць аўдыторыі ведаць рашэнні і падставы.

– Далёка не ўсё людзі прымаюць рашэнне на падставе эканамічнай выгады і эгаізма. Напрыклад, шмат хто цэніць магчымасць папрацаваць разам або давер адзін да аднаго.
– Разглядаць людзей як індывідаў. Гэта значыць прамыя кантакты стваральніка і карыстальніка/фаната.
– Публічна заяўляўленне сваіх прынцыпаў. Выражэнне каштоўнасцяў павінна быць яўным, не хаваючы іх.
– Будаваць супольнасць. Ствараць практыку патрэбнасці і прыналежнасці да супольнай працы, і стварэння агульнага адзінадумцамі. Супольнасць забірае сілы, але аб'ядноўвае высілкі.
– Шэрынг — значыць, даваць больш, чым браць. Адкрыты кантэнт — гэта не поле халяўных рэсурсаў, і не проста постынг свайго кантэнта, але дадатковая вартасць.Напрыклад, пэўная культура зносінаў або падтрымкі ўдзельнікаў.
– Уключаць людзей у тое, што робіш. Малыя ўнёскі найбольш значныя.

Статыстыка паказвае, што ў свеце існуе больш за 1.4 міл'ярда твораў пад CC. Гэта насамрэч ураджваючая з'ява сведчыць пра наяўнасць сусветнай супольнасці, якая папаўняе менавіта "лічбавае супольнае", як альтэрнатыўны шлях эканамічнага, культурнага і сацыяльнага развіцця двум крайнасцям: дзяржаўнай або рынкавай манаполіі ў кіраванні рэсурсамі.

Літаратура

[1] Paul Stacey, The new world of digital commons, 2017 (Book. Made Wi creative Commons)
[2] Sarah Pearson, How to be made with creative commons, 2017 (Book. Made Wh creative Commons)
[3] Report. State of the commons, 2016 — `https://stateof.creativecommons.org/`

Адкрытыя графічныя прылады Gimp і InkScape. Плюсы і мінусы*

Святлана Ермаковіч, Minsk, Belarus

Comparison of the basic functions and particular tasks is done for free/libre (Gimp, InkScape) and non-free (Photoshop, CorelDraw) graphic editors from the amateur designer point of view. Things which were successfully performed with use of FLOSS are listed as far as those which weren't. Also disadvantages are classified along their severity.

Свабодныя графічныя прылады Gimp і InkScape адрозніваюцца ад сваіх прапрыетарных аналагаў (возмем, Photoshop і CorelDraw). Параўнаем асноўныя функцыі і частыя задачы свабодных і несвабодных прыладаў на ўзроўні аматару дызайну.

Графічныя заданні і асноўныя функцыі

Gimp vs Photoshop

Карэкцыя фотаздымкаў.

Асноўныя функцыі карэкцыі: яркасць, кантраснасць, баланс колераў, адценне, насычанасць і іншыя есць у дзвюх прыладах. Крыху адрозніваюцца алгарытмы гэтых функцый. І таму рэзультат прымянення адной і тойжа можа быць розны.

Калаж: праца са слаямі, выдаленне фону, змена памераў, абрэзка.

Абрэзка фону. У Gimp ёсць функцыі як выдзялення аднаго і таго ж колеру ўсюды, так і «магічная палачка», якая выдзяляе некае адценне ў інтэрвале колера. Інтэрвал можна задаваць. Зручна для апрацоўцы сфатаграфаваных або адсканаваных малюнкаў. У Photoshop есць «магічная палачка» таксама.

Змена памераў. У Gimp існуе асобная кнопка на панэлі. У Photoshop — гэта «гарачая кнопка» Ctrl+T, узгадаць дзе знаходзіцца аналагічная функцыя ў панэлях прыладаў пакуль неўдалося.

*imfantina@gmail.com, http://lvee.org/ru/abstracts/239

Праца са слаямі. У Gimp трэба крыху прызвычаецца да ўстаўкі малюнка на новы слой, бо ен дадаецца не аўтаматычна (як пры выбары слою ў Photoshop), а на віртуальны слой, які неабходна зафіксаваць асобнай кнопкай на выбраны. Функцыі бачнасці-нябачнасці, перасоўвання слаёў, склейкі, капіравання, рэгулявання празрыстасці таксама існуюць у Gimp. Аднак склейка адбываецца не паміж абранымі слаямі (як у Photoshop), апаміж суседнімі (верхні склеіваецца з ніжнім).

Абрэзка. У Gimp магчыма адразу змяняць памер поля абрэзкі пасля выдзялення. У Photoshop памер выдзялення можна змяняць праз меню правай кнопкі. У абедзвюх прыладах есць «разумная абрэзка», калі малюнак абразаецца па контару.

Маляванне з нуля

Тут па сутнасці часта выкарыстоўваюцца вышэй азначаныя функцыі: праца са слаямі, праца з пэнзлікамі і колерамі. У дзвюх прыладах можна лёгка змяняць як колеры (асноўны і колер фона), памеры і віды пэнзлікаў, празрыстаць слаёў. Таксама часта ўжываюцца кнопкі «размыцця», «асвятлення» і «зацямнення». Апошнія дзве прадстаўлены ў Gimp адной кнопкай з магчымасцю выбару дакладный функцыі, а ў Photoshop — дзвюма асобнымі кнопкамі.

Праца з тэкстам.

У Photoshop праца з тэкстам зручней: ён легка маштабіруецца (як малюнак), захоўваючы свойствы. У Gimp мяняць памер тэкста можна толькі змяняя яго ў адзінках шрыфта (напрыклад у пойнтах, і толькі ў адпаведным для гэтага полі). Маштабаваць тэкст як малюнак можна таксама, але пры гэтым тэкст пераўтвараецца ў малюнак.

Перасоўваць тэкст ў Photoshop можна абраўшы слой, на якім знаходзіцца тэкст. Перасованне ў Gimp тэксту (і іншых малюнкаў, якія складаюцца з контураў) выконваецца толькі калі трапіць курсорам менавіта на патрэбны тэкст (не на прабелы і пустыя месцы), ці з дапамогай клавішы «shift».

Экспарт

У Photoshop функцыя «захаваць як» прапаноўвае выбар фармату захоўвання. У Gimp «захаваць як» — гэта захоўванне толькі

ў фармаце .xcf, а каб захаваць у іншых фарматах неабходна карыстацца функцыяй «экспартаваць як».

InkScape vs CorelDraw

Крывыя Без'е

У дзвюх прыладах прадстаўлены як крывыя Без'е, так і іншыя крывыя. Намаляваныя контуры можна рэдагаваць змяняючы вузлавыя кропкі, з якіх складаецца малюнак.

Трасіроўка

Алгарытмы трасіроўкі ў InkScape і CorelDraw па-змоўчанню розныя. У CorelDraw малюнак разбіваецца на кавалкі, адмалёваныя ў вектары (нібыта пазл). У InkScape малюнак трасіруецца на колькасць слаёў, якую можна задаць. Калі слаеў больш за 5-7 малюнак ужо вельмі цяжка рэдагаваць. І бывае, што лягчэй адмаляваць яго наноў у вектары.

Функцыі логікі.

Union, difference, intersection, exclusion, division — ёсць у дзвюх прыладах. (У CorelDraw яны больш інтуітыўныя). Функцыя абрэзкі сваеасабліва рэалізавана ў кожнай з іх. Пад абрэзкай я маю на ўвазе, што ў вас есць малюнак (вектарны, ці растравы) і вам трэба вырэзаць з яго адмысловую форму. Калі гэта форма — прамавугольнік, то ў CorelDraw такая абрэзка робіцца кнопкай з лязом. А калі форма праізвольная, то праз функцыю змяшчэння вашага малюнка у кантэйнер формы. У InkScape не важна якую форму трэба вырэзаць — выконваецца функцыя абрэзкі (з малюнка знізу выразаецца форма, што зверху).

Праца з тэкстам, вёрстка

Праца з тэкстам у дзвюх прыладах даволі зручная. Галоўнае не забываць пераводзіць шрыфты ў крывыя пры экспарце і друку.

Градзіент.

У CorelDraw праца з градзіентамі больш інтуітыўная, у InkScape трэба прызвычаецца да вакна працы з градзіентам і выбарам колераў для кропак пераходу і задання празрыстасці для іх.

Экспарт

InkScape захоўвае ў папулярных фарматах .eps .svg, якія могуць быць адчыненыя і ў CorelDraw, ці іншых прыладах. InkScape экспартуе толькі ў png, што бывае не зручна (часта png важаць болей за jpg).

Чаго не хапае свабодным прыладам?

1. У Gimp і InkScape адсутнічае пераўтварэнне і праца са CMYK гамай. Гэта не зручна для друку, бо скажаюцца некаторыя колеры.
2. У InkScape адсутнічае праца з некалькімі старонкамі, што вельмі неабходна пры верстцы невялічкіх буклетаў і брашур.
3. У Gimp адсутнічае задаванне сцэнароў (паслядоўнасці каманд, якія неабходна зрабіць з файлам).
4. Імпарт з Photoshop у Gimp магчымы, але знікнуць некаторыя фільтры (функцыі), напрыклад, цень. Таксама пры імпарце губляецца дрэва тэчак для слаеў, слаі ўладкоўваюцца па парадку (Гэта не зручна пры стварэнні дызайна для сайта, дзе звычайна ствараецца дрэва тэчак для слаеў з рознымі элементамі). Адваротны імпарт з Gimp у Photoshop немагчымы.
5. Пры імпарце .svg з CorelDraw у InkScape, ці наадварот, вектарны малюнак можа скажацца. Бывае, што экспарт залежыць ад версіі як CorelDraw, так і InkScape.

Картографический сервис на OpenSource - доступная картография каждому*

Дмитрий Степанов, Минск, Belarus

In the modern world, geoinformation technologies are widely represented in the life of both individuals and companies as a whole. These are navigation systems, transport monitoring systems and many-sided programs and applications that use spatial information mapping on a cartographic basis. Modernity as a period is an era of commercial development of GIS (geoinformation systems), expansion of the field of their application through integration with DBMS, the emergence of network applications - all this has opened the way for systems that support corporate and distributed geodatabases. This pushed development of not only commercial geoinformation systems but also freely
distributed products (QGIS, NextGISWEB, MapServer and others).

— Где карта, Билли? Нам нужна карта...
— Какая карта?! У меня нет никакой карты!
— Где карта?!
(мультфильм «Остров сокровищ», 1988)

Введение.

В современном мире геоинформационные технологии повседневно присутствуют в жизни как персонально каждого человека, так и компаний в целом. Это навигационные системы, системы мониторинга транспорта и многоликие программы и приложения, использующие отображение пространственной информации на картографической основе. Современность как период — это эра коммерческого развития ГИС (геоинформационных систем), расширение области их применения за счет интеграции с СУБД, появление сетевых приложений — и все это открыло путь системам, поддерживающим корпоративные и распределенные базы геоданных. Это стало толчком к развитию не только комерческих геоинформационных систем, но и свободно распространяемых продуктов.

*d.stepanov@itg.by, http://lvee.org/ru/abstracts/242

Постановка задачи.

В одной «мифической» компании как-то заметили, что хотя в названии у них есть слово геоинформационные технологии, но услуг, да и собственно геоинформационных технологий, нет. Руководители компании нашли на стороне инженеров и поставили задачу реализовать собственную геоинформационную систему (ГИС) на свободно-распространяемом программном обеспечении, не ввязываясь в дорогостоящий процесс собственно разработки продукта.

Функционал решения должен включать:

- возможность развития системы как продукта
- возможность интеграции решения в существующие собственные программные комплексы организации
- возможность публиковать собственные карты и картографические основы
- возможность реализовывать слои на картографической основе
- система должна быть проста в эксплуатации, дабы не содержать дорогостоящий персонал, для использования функционала продукта и в случае продажи не услуги, а собственно программного решения, то есть ГИС.

Выбор решения.

Проведя анализ предлагаемых ГИС-платформ на проприетарной основе и определившись с тем, что бы хотелось получить в итоге, мы принялись за работу.

Что понадобиться?

Картографическая основа. По причине отсутствия финансирования, а также из-за запрета эксплуатировать покупные карты, мы остановились на использовании собственного тайлового сервера, на картографической основе OSM (openstreetmap) [7]. Популярность данной картографической подложки обусловлена, во-первых, политикой предоставления тайлов, позволяющей любому желающему свободно использовать тайлы OpenStreetMap или иных картографических основ, разработанных на базе OSM в своих приложениях, а во-вторых, простотой их подключения в современных веб-клиентах, таких как Leaflet и OpenLayers.

Использование собственного сервера карт было обусловлено необходимостью наличия картографической подложки при отсутствии доступа в общедоступным картам OSM, Яндекс Народная и т.п. Это позволяет, в случае необходимости реализовать проект без привязки к стороннему сервису картографической основы. В отдельных случаях вместо картографической основы можно использовать shape-слой, созданный на основе картографической основы.

В качестве решения для создания печатных карт, а так же слоев пространственных данных, был выбран программный продукт QGIS [1] (распространяемый по лицензии GNU General Public License) — как унифицированное решение с большим сообществом и соответственно поддержкой разносторонних модулей и мультиплатформенной архитектурой.

Серверов публикации данных было проанализировано несколько:

- MapServer (на данный момент наверное самый популярный вариант сервера для публикации пространственных данных — карт).
- QGIS Web Server (рассматривался исключительно из за высокого уровня совместимости с продуктом QGIS)
- NEXTGIS Web Server (Данное решение, было найдено случайно и как результат позволило получить максимальный результат)

По результатам изучения данного направления, был сделан выбор в пользу продукта от российской компании NEXTGIS: это обусловлено как распространением их решения по лицензии GNU GPL, так и тем, что они (собрав все описанные выше продукты) реализовали довольно интересный и функциональный продукт в «коробочном» исполнении, доступный для освоения обычным пользователям [4]. Также подкупило наличие мобильных приложений (клиентов) для редактирования существующих слоев на веб-сервере и управления сервером с находясь «в поле».

Что же касается QGIS Web Server [5], данное программное решение довольно простое в эксплуатации, но позволяет публиковать данные как демонстрацию слоя и карты, без конечного взаимодействия с объектами нанесенными на картографическую основу, что само по себе конечно уже много, но недостаточно для реализации поставленной задачи.

Отдельно хочется отметить продукт MapServer, так как он оказался структурообразующим продуктом и по сути присутствует в

каждом из описанных выше решений. Это более чем гибкое решение для публикации пространственных данных на картографической основе, но реализация на нем слоев требует дополнительных знаний от конечного пользователя.

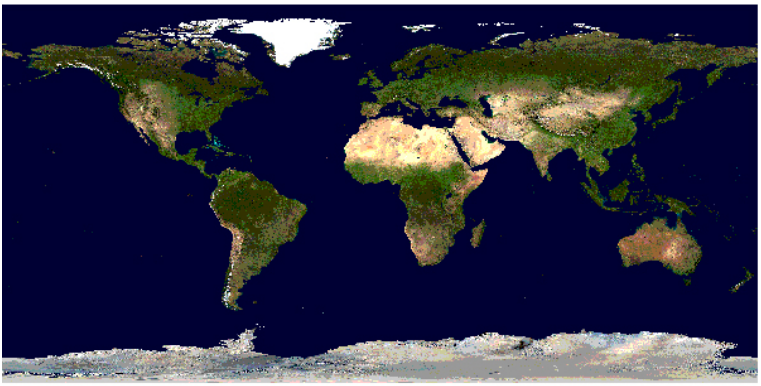
Пример создания mapfile для MapServer и слоя.

```
MAP
    NAME "sample"
    STATUS ON
    SIZE 600 400
    SYMBOLSET "../etc/symbols.txt"
    EXTENT -180 -90 180 90
    UNITS DD
    SHAPEPATH "../data"
    IMAGECOLOR 255 255 255
    FONTSET "../etc/fonts.txt"

    #
    # Start of web interface definition
    #
    WEB
        IMAGEPATH "/ms4w/tmp/ms_tmp/"
        IMAGEURL "/ms_tmp/"
    END # WEB

    #
    # Start of layer definitions
    #
    LAYER
        NAME 'global-raster
        TYPE RASTER
        STATUS DEFAULT
        DATA bluemarble.gif
    END # LAYER
 END # MAP
```

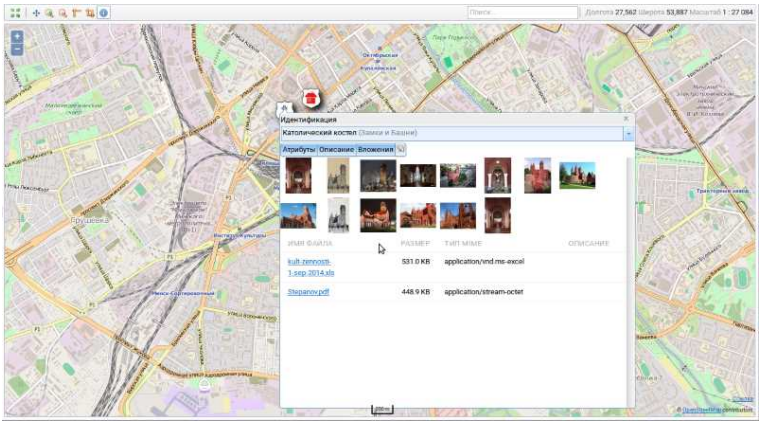
Для хранения пространственных данных большинство рассматриваемых свободно распространяемых продуктов использует СУБД PostgreSQL [8], а точнее, надстройку над СУБД — PostGIS [9],

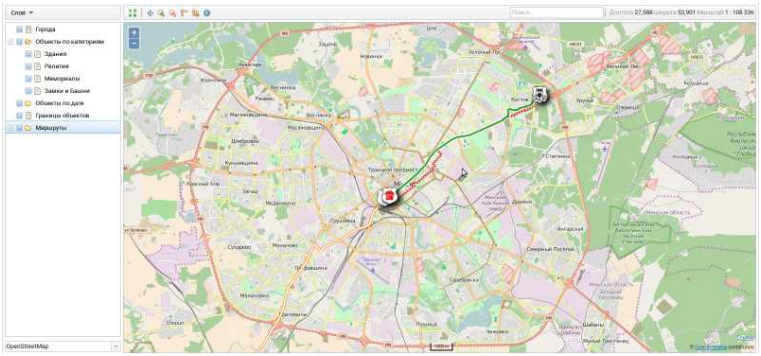


позволяющую хранить в себе пространственные данные (точки, полигоны, линии и некоторые графические объекты).

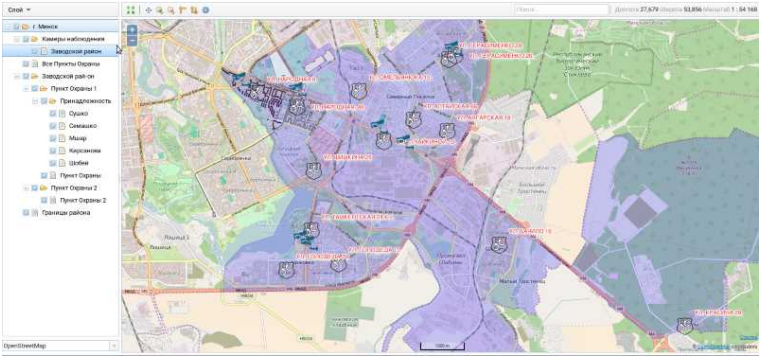
Опыт использования в реальном проекте.

– Реализации проекта по созданию онлайн-ГИС для размещения культурных и туристических объектов Беларуси на картографической основе, доступной всем.





– Отображение на картографической основе информации о работе светофоров с динамическим изменением стилистики отображаемого слоя.
– Отображение на картографической основе зон отношения к определенному гос. учреждению (милиция, поликлиника, ЖЭС и т.д.v1)



Подытожив, хочется воспользоваться цитатой из киносерии про пирата Джека Воробья:

— Да, раньше этот мир был куда больше...
— Нет, мир остался прежним. Стало меньше содержимого.

Литература

[1] Официальный портал продукта QGIS — http://qgis.orgqgis.org
[2] Проект QGIS на гитхабе — https://github.com/qgis/QGIS
[3] Веб-клиент для работы с QGIS — https://github.com/qgis/QGIS-Web-Client
[4] Сайт продукта NEXTGISQGIS — http://nextgis.ru/nextgis-qgis/
[5] Реализация QGIS WEB Server — https://live.osgeo.org/ru/quickstart/qgis_mapserver_quickstart.html
[6] GIS-Lab — неформальное сообщество специалистов в области ГИС и ДЗЗ — http://gis-lab.info/
[7] Ресурс картографической основы — http://www.openstreetmap.org
[8] СУБД — https://www.postgresql.org/
[9] Расширение для базы данных PostgreSQL — http://www.postgis.net/
[10] Проект NEXTGIS на гитхабе — https://github.com/nextgis
[11] Сайт продукта NEXTGIS — http://nextgis.ru/
[12] Проект NEXTGISWEB на гитхабе — https://github.com/nextgis/nextgisweb
[13] Интеграция публикации стилей слоев QGIS на NEXTGISWEB — https://github.com/nextgis/nextgisweb_qgis

USB attacks explained *

Krzysztof Opasiak, Warsaw, Poland

USB is the most common external interface in the world. Even machines which, for security reasons, are disconnected from the Internet often offer USB connectivity. This creates a new attacks surface for skilled hackers.

USB implementation may be exploited on various levels. To effectively protect against such attack, knowledge about already exploited vulnerabilities is required. This paper is a survey of state-of-the-art USB-related attacks.

Introduction

On a very high level, USB is a communication protocol which allows to provide and use some abstract services (functionalities). Machine which provides some additional functionality is called a USB device and machine which uses this functionality is called a USB host. Typically USB host is a computer, DVD player etc. and USB device is a pendrive, web camera or sound card but it can also be a mobile phone or tablet! One of very famous USB features is Plug&Play. It means that USB host is able to automatically detect new USB devices and discover functionalities offered by them. Together with automatic support provided by most of host operating systems it makes USB very user-friendly and easy to use. Unfortunately, the same automation may lead to new security threats ...

USB attacks

Popularity and blind trust in USB security seem to be one of the major reasons why malware started spreading also using this attack vector. From the security perspective we can distinguish three types of attacks toward USB based on their main target:

- USB host focused attacks,
- USB traffic analysis, modification and injection attacks,
- USB device focused attacks.

*kopasiak90@gmail.com, http://lvee.org/ru/abstracts/243

USB host focused attacks

This type of attacks aims at taking over the control of the USB host machine. We can distinguish three subgroups of such attacks. The first group of such attacks uses vulnerabilities in the high level system infrastructure related to support of given USB function. Very good examples of such attacks are Conficker [1] and Stuxnet [2], which used the vulnerabilities in external storage support. The second group of USB host focused attacks tries to exploit vulnerabilities in USB stack and drivers implementation. This is mainly done by creating malicious devices which sends specially prepared payload to exploit some buffer overflow or other vulnerability. Very good repentant of such attacks is Plug & Root device presented during BlackHat Conference in Las Vegas back in 2005 [3]. Thanks to dedicated fuzzers like facedancer [4] and umap [5] and increasing developers' awareness USB stacks (esp. Linux stack) are getting more and more resistant to this group of attacks. Finally, the third group of attacks abuses Plug & Play philosophy and blind user trust in harmlessness of USB devices. This group of attacks became famous in 2014 thanks to BlackHat USA presentation – BadUSB [6]. This attack simply tries to trick user to connect device which offers different functionality than user may assume based on its physical outfit.

USB traffic analysis, modification and injection attacks

This type of attacks usually aims at discovering some secret information like passwords or at modification of USB traffic to abuse functionality expected by user. First group of those attacks are passive listeners usually recording HID protocol which is used by for example keyboards. Those devices are relatively cheap and are being found from time to time in public places like libraries [7] or schools [8]. Second group involves not only listening but also modification of USB traffic and injection of additional messages. Recent example of such attacks may be BadUSB 2.0 introduced by David Kierznowski and described in [9].

USB device focused attacks

Third type of attacks is targeted at USB devices. Usually not those simple tiny device lie pendrive because people try to keep them safe but rather those more complicated like mobile phones and tablets. All those devices carry a lot of sensitive user data which can be accessed via USB. It's worth to mention that the same USB port is often used for both data transfer and battery charging. Short battery life encourages user to connect mobile device to publicly available charging stations. This leads to hazard of unauthorized access to private data like photos or contacts. More advanced attackers may event try to take over the control of device using for example ADB resource exhaustion attack [10].

Summary

USB is a extremely popular external interface. It has been adopted to various use case and incorporated by most devices on the market. Through being extremely usefully, USB should be also considered as a real security thread especially in high security environments.

References

[1] M.Hypponen, «The conficker mystery,» in Black Hat, Las Vegas, NV, USA, July 2009. Online. Available:http://www.blackhat.com/presentations/bh-usa-09/HYPPONEN/BHUSA09-Hypponen-ConfickerMystery-PAPER.pdf

[2] L. O. M. Nicolas Falliere and E. Chien, «W32.stuxnet dossier,» Feb. 2011. Online. Available:http://www.symantec.com/content/en/us/enterprise/media/securityresponse/whitepapers/w32stuxnetdossier.pdf

[3] D. Barral and D. Dewey, ««plug and root,» the usb key to the kingdom,» in Black Hat, Las Vegas, NV, USA, July 2005. Online. Available:http://www.blackhat.com/presentations/bh-usa-05/BHUS05-Barrall-Dewey.pdf

[4] «Facedancer21 (usb emulator/usb fuzzer).» Online. Available:http://int3.cc/products/facedancer21

[5] «umap: The usb host security assessment tool.» Online. Available:https://github.com/nccgroup/umap

[6] S. K. Karsten Nohl and J. Lell, «Badusb – on accessories that turn evil,» in Black Hat, Las Vegas, NV, USA, July 2014. Online. Available:http://srlabs.de/wp-content/uploads/2014/07/SRLabs-BadUSB-BlackHat-v1.pdf

[7] «Hardware keyloggers discovered at public libraries.» Online. Available:http://nakedsecurity.sophos.com/2011/02/14/hardware-keyloggers-discovered-public-libraries/

[8] «Us school expels pupils for using hardware keyloggers to change grades,» Feb. 2004. Online. Available:http://www.techworld.com/news/security/us-school-expels-pupils-for-using-hardware-keyloggers-change-grades-3500558/

[9] D. Kierznowski, «BadUSB 2.0: USB man in the middle attacks,» Royal Holloway University of London, Tech. Rep., 04 2016.

[10] T. Vidas, D. Votipka, and N. Christin, «All your droid are belong to us: A survey of current android attacks,» in Proceedings of the 5th USENIX Conference on Offensive Technologies. Berkeley, CA, USA: USENIX Association, 2011, pp. 10–10. Online. Available: http://dl.acm.org/citation.cfm?id=2028052.2028062

Безопасность в браузерах: Альтернативы SSL*

Алексей Хлебников, Oslo, Norway

Browser makers spend a lot of effort for making SSL right and make PKI harder then ever, but pay too little attention to alternative methods of estimating the web site security. In this talk I will tell about such alternative methods and a little bit about using risk management for security level estimation.

Ожидания и реальность

На первых этапах внедрения SSL в Web, на SSL возлагались большие надежды. Предполагалось, что будет примерно так:

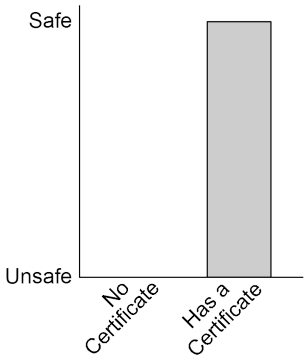


Рис. 12.1. SSL-безопасность в Web: что предполагалось

Однако, реальность оказалась сложнее. Например, пользоваться SSL стали и компьютерные преступники.

Скриншот с форума, где DDOS-еры предлагают свои услуги:

*alexei.khlebnikov@gmail.com, http://lvee.org/ru/abstracts/249

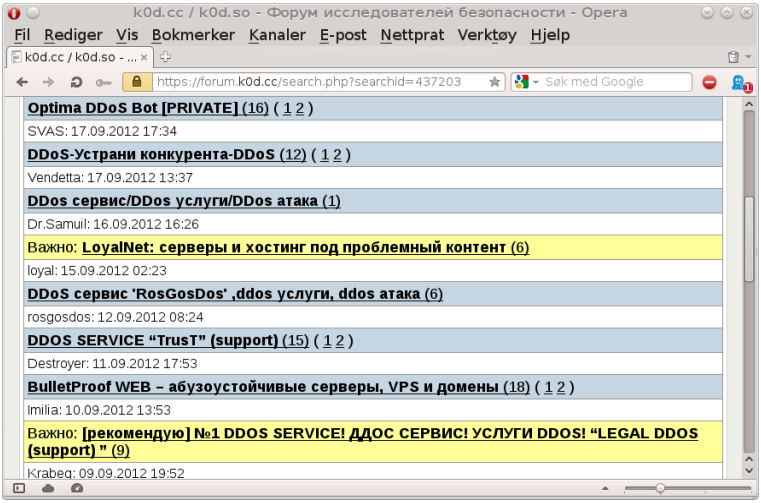


Рис. 12.2. Форум киберпреступников, на котором предлагаются услуги DDOS. SSL-сертификат успешно прошёл проверку

Как видно, SSL-сертификат успешно проверен, замочек показан. Пользователь может быть уверен, с сайтом «всё в порядке».

Видим, что сертификат браузеру определённо не нравится 12.3. Внимательный читатель наверное уже догадался, что сертификат выписан на имя «www.visa.com», а введённый адрес — просто «visa.com», без префикса «www». Но обычный пользователь вряд ли это поймёт, а вполне может подумать, что его атакуют.

Совсем плохо приходится некоммерческим CA. Их сертификаты будут признаны недействительными без ручной установки в браузер, а значит у большинства пользователей. Браузер отвергает даже сам сайт такого CA 12.4:

Заметим, что браузеры совсем не выдают таких предупреждений для сайтов, которые предоставляются не по HTTPS, а по HTTP, то есть вообще без сертификата. Таким образом, упрощённая модель SSL-безопасности в Web выглядит скорее 12.5, что не очень логично, так как «плохой» сертификат всё-таки лучше его отсутствия.

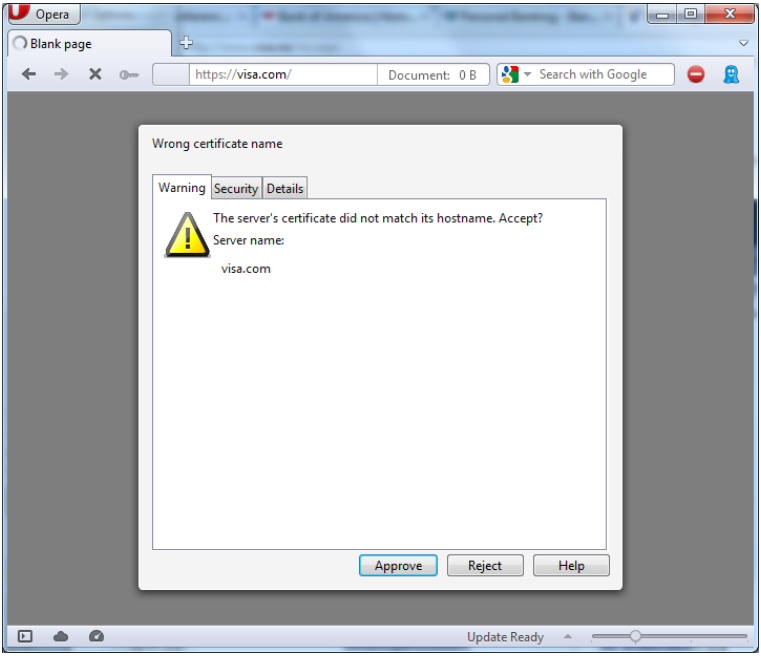


Рис. 12.3. Сайт платёжной системы Visa. SSL-сертификат не прошёл проверку

Логичнее было бы 12.6.

Взломы SSL

На этом проблемы SSL не заканчиваются. За прошедшие годы случались инциденты и взломы CA-инфраструктуры. Вот их неполный список:

– 2001 — Атакующий прикинулся сотрудником Microsoft и получил у Verisign сертификат для Microsoft.
– 2008 — На сервисе live.com зарегистрирован бесплатный e-mail sslcertificates@live.com, затем у Thawte получен сертификат для login.live.com.

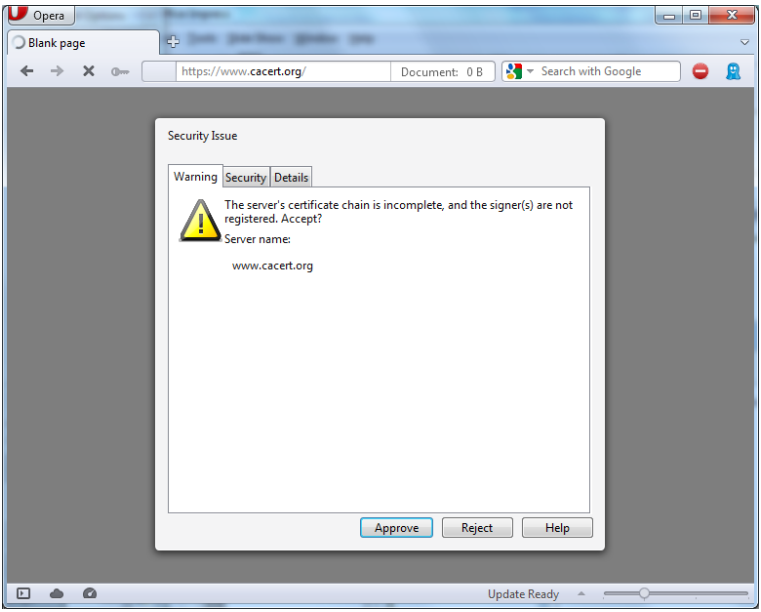


Рис. 12.4. Сайт некоммерчесского CA. SSL-сертификат не прошёл проверку

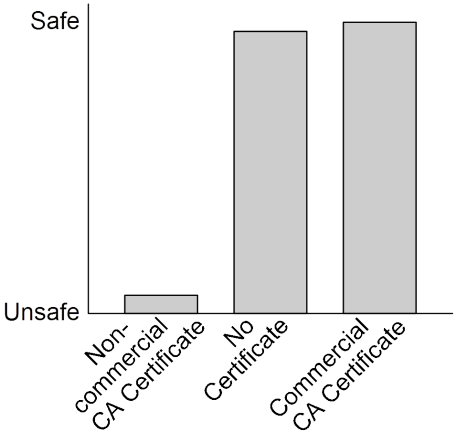


Рис. 12.5. SSL-безопасность в Web: что получилось

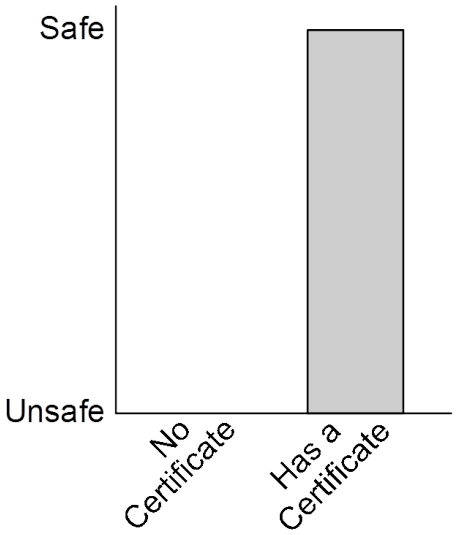


Рис. 12.6. SSL-безопасность в Web: что было бы логичнее

– 2008 — Взломы StartCom и Comodo, у Comodo получен сертификат для www.mozilla.com.
– 2009 — Взлом ipsCA через NULL prefix attack, получен сертификат для www.paypal.com.
– 2011 — Взломы Comodo, Diginotar и TurkTrust. Получены сертификаты для `www.google.com`, `mail.google.com`, `addons.mozilla.org`, `login.live.com`, `login.yahoo.com`, `login.skype.com`. Шуточная атака «Honest Achmed» на Mozilla.
– 2014 — Взлом NICCA, получены сертификаты для доменов Google и Yahoo.
– 2015 — Взломы CNNIC, WoSign, Let's Encrypt и Symantec. Получены сертификаты для доменов Google, GitHub и других.
– 2016 — Найдены уязвимости в инфрастркутурах выдачи сертификатов у StartCom и Comodo. Были ли эти уязвимости использованы и как — пока неизвестно.

А также взломы разной степени эффективности самого протокола SSL/TLS:

– 2009 — Renegotiation attack, SSL stripping.

– 2011 — BEAST, StartTLS injection.
– 2012 — CRIME.
– 2013 — BREACH, TIME, Lucky13.
– 2014 — FREAK, POODLE, Triple Handshake.
– 2016 — DROWN.

А также не будем забывать о периодически находимых уязвимостях в реализациях SSL, например Heartbleed в OpenSSL.

Система CA недостаточно надёжна

До сих пор никто так и смог дать убедительного ответа, почему сайты и пользователи должны доверять тем самым CA, которых большинство пользователей в глаза не видело. Ответ «по умолчанию» на этот вопрос - потому что CA берут деньги за выдачу сертификата, и если они вдруг выдадут неправильный сертификат или будут взломаны — сразу потеряют доверие, клиентов и разорятся. Мне в этот аргумент почему-то не очень-то верится. Особенно в свете известных вышеперечисленных взломов. Кто из CA понёс серьёзное наказание в результате взлома и прекратил выдачу сертификатов? Почти никто. Только маленький Diginotar. А взлом Comodo в том же году, наделавший немало шума - спустили на тормозах. Потому что Comodo — «too big to fail». Работает дальше и продолжает «радовать» нас периодическими уязвимости своей инфраструктуры выдачи сертификатов. А сколько взломов не стало достоянием общественности? А ведь CA довольно неохотно раскрывают факты взломов своей инфраструктуры, и бывали уличены в сокрытии такой информации.

А можем ли мы быть уверены, что если в офис CA придут сотрудники местных спецслужб и попросят выдать им сертификат на не принадлежащий им домен, то CA откажет им? Скорее, мы можем быть уверены в обратном. А знает ли уважаемый читатель, что в США полицией и спецслужбами уже несколько лет используются так называемые MITM boxes? Это устройство, которое в реальном времени осуществляет MITM-атаку и расшифровывает весь SSL-траффик, проходящий через устройство. Причём, пользователь, которого прослушивают, не получает никаких предупреждений от своего браузера, потому что MITM-box на лету выдаёт «правильные» сертификаты для всех серверов, посещаемых пользователем. Как устройчтво это делает? Очень просто — устройство обладает intermediate-CA сертификатом, который любезно выдан

крупным американским CA, которому, конечно, доверяют все браузеры.

Большая проблема здесь — то что абсолютно любой CA может выдать сертификат абсолютно любому сайту. Это ещё хуже, чем единая точка отказа. Доверие CA-центрам — абсолютно, а они его, прямо скажем, не заслужили.

А ещё некоторые CA доверяют выдачу своих сертификатов другим фирмам, эдаким дилерам. Прямо как подключение телефона в салоне связи. А ещё многие CA вполне официально выдают некоторым крупным бизнес-клиентам сертификаты, позволяющие выдавать другие сертификаты, то есть частично делегируют полномочия CA. Всё это увеличивает прибыль CA, но, к сожалению, уменьшает безопасность сайтов и пользователей.

Чем отвечает индустрия

Как видно, в проблем в концепции SSL и CA хватает. Чем же отвечает индустрия? Правильно, индустрия отвечает закручиванием гаек. «PKI them harder». В частности, предлагались или предлагаются следующие решения:

- Короткоживущие сертификаты
- Публичные списки выдаваемых сертификатов
- Обязательное OCSP
- HTTP Strict Transport Security (HSTS)
- HTTP Public Key Pinning (HPKP) и TACK
- Convergence и Mutually Endorsing CA Infrastructure
- The Monkeysphere Project и Web of Trust

Некоторые из этих предложений хороши, некоторые сомнительны, некоторые уже не выдержали испытание временем, но большинство из них пытаются «наложить заплатки» на существующую систему SSL и CA, вместо того, чтобы взглянуть на проблему шире. Ведь наша цель — повысить безопасность пользователя, и закручивание гаек в PKI — далеко не единственный способ.

Диверсификация защиты и управление рисками

PKI — не серебрянная пуля. И не священная корова. Нужна диверсификация средств безопасности. Не стоит складывать все яйца в одну корзину, полагаясь лишь на PKI.

Диверсификация защиты применяется в реальном мире с начала времён. Её преимущество очевидны — нет единой точки отказа. Если механизм защиты вдруг отказал, то

- без диверсификации — отказала вся система,
- с диверсификацией — возрос риск, но не отказала вся система.

Безопасность с помощью диверсифицированной защиты и управления рисками:

- Применяется комплекс защитных мер.
- Ни одна из мер сама по себе не даёт сильной защиты.
- Комбинация защитных мер даёт сильную защиту.
- Возможна детальная оценка безопасности и «обратная связь».

Оценка уровня безопасности сайта

Браузер должен оценивать уровень безопасности сайта исходя из многих критериев, чтобы предоставлять комплексную диверсифицированную защиту. Что же можно использовать в качестве таких критериев?

Прежде всего, несколько предложений относительно самого SSL.

Так повелось, что SSL применяют как абсолютно «беспамятную» технологию. В отличие от, например, SSH. Каждое общение с сайтом — как в первый раз. Сменился сертификат у сервера — никакой реакции у браузера на это. Сменился CA — тоже никакой реакции.

Следующие факторы должны понижать оценку безопасности сайта:

- У сайта другой сертификат
  - И срок действия предыдущего не вышел
- У сертификата другой ключ
  - И предыдущий сертификат не отозван
- Внезапная смена CA
  - Французский CA, бразильский сайт?
- Смена EV на DV

«Память» неплохо бы использовать не только для SSL, но и для IP:

- Кардинально сменился IP
  - Совсем другая подсеть?
  - Совсем другая страна?

– Traceroute стал гораздо короче (MITM?)

Браузер должен помнить о ранее посещённых сайтах, когда сайт просит пароль:

– Сайт посещался много раз, и много раз пароль здесь уже вводился - риск меньше.
– Сайт посещается в первый раз и уже просит пароль — риск больше.

Хостинг. Стоит проверить whois и AS (autonomous system) info. Пример: банк Societe Generale.

– Хостится на AS.
– Имя AS: SOCIETE-GENERALE.
– Подключена к французскому бэкбону.
– Работает с 1995 года.

Такой хостинг заслуживает доверия.

По части DNS можно проверить следующее:

– Reverse lookup: `client321.adsl-pool.isp.com`
– Маленький TTL
– Комбинация записей A, MX, NS
– Регистратор DNS и IP:
  – RIPE: риск меньше
  – GoDaddy: риск больше

Можно проверить и TCP/IP stack fingerprint сервера:

– E-commerce сайт хостится на Windows Home Premium?
– Открыт порт, по которому слушает «популярный» троян?

Проверка URL:

– Смешанный алфавит в именах популярных сайтов.
– Spell-check: `panascanic.com`, `bankoffireland.ie`.
– Подстроки: «members», «adsl-pool», etc.

Мы видим ту же веб-страницу, что и остальные Интернет-пользователи, или нам её подменили?

– Сравнение с веб-архивом.
– Сравнение с кэшем Google.
– Сравнение с User Agent = Googlebot.

Проверка содержания самой веб-страницы:

– Страница пытается задействовать известные уязвимости?
– HTML тэги в «неправильных» местах?
– Несколько тэгов html, head, title, body?
– Много объектов подгружается из других доменов?

Проверка Javascript на странице:

– Длинные строки и их энтропия.
– Вызовы eval(), большое количество substring(), concat(), fromCharCode().

Для оценки безопасности страницы наверняка можно использовать и байесовский фильтр, широко применяемый для оценки безопасности e-mail и борьбы со спамом.

Злоумышленники могут препятствовать анализу страницы с помощью обфускации, но обфускация сама по себе вызывает подозрение, а значит снижает оценку безопасности.

После всех проверок, оценка безопасности сайта может выглядеть, например, 12.7.

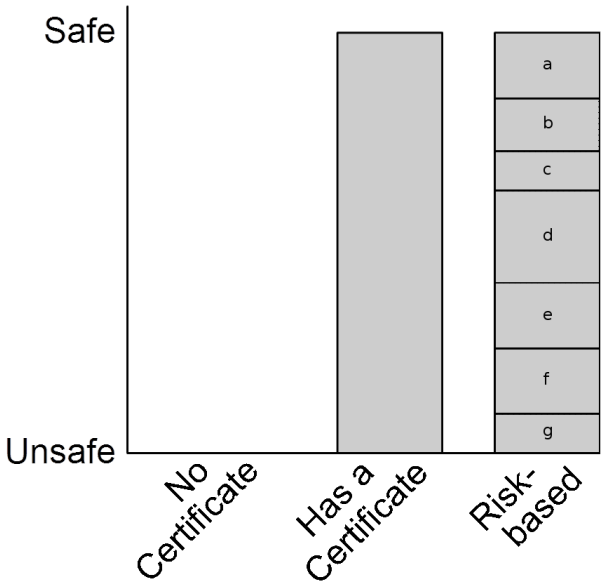


Рис. 12.7. Пример оценки безопасности Web-сайта при использовании управления рисками

Превращение Android-устройств в мобильные серверы при помощи FOSS*

Виталий Сороко, Minsk, Belarus

Modern mobile devices such as smartphones and tablets now have powerful processors and sometime large amount of memory inside. So these devices can be suitable for standard server tasks, for example data warehousing and HTTP request handling. However, since the architecture of mobiles is different from standard PCs and servers, it is necessary to recompile open source server software for further usage. This is not very simple process and here is the description of key points and main ways of that.

Современные мобильные устройства, такие как смартфоны, планшеты и даже «умные» часы, с каждым годом становятся всё мощнее и мощнее. Благодаря увеличивающемуся объему памяти, многоядерным процессорам и наличию встроенной батареи они вполне могут работать в качестве маленького переносного сервера, ведь у них внутри есть все что для этого нужно – вычислительные мощности и резервный источник питания на случай сбоя в энергосети. Однако, процесс превращения мобильного устройства в маленький переносной сервер не так прост как может показаться на первый взгляд. Для того чтобы сделать это, вам придётся пройти через следующие основные шаги:

1. Покупка и подготовка оборудования.
2. Выбор набора необходимого серверного ПО
3. Компиляция, установка и настройка серверного ПО
4. Проверка работоспособности системы
5. Оптимизация

Итак первым этапом является покупка и подготовка оборудования. Как известно, для работы серверного ПО вам понадобиться сам сервер и сетевое соединения, а также оборудование для обеспечения работы сетевого соединения. В качестве сервера мы будем использовать мобильное устройство. Поэтому, вам понадобиться смартфон, планшет или какой либо другой гаджет на базе Android.

*vitalyok@tut.by, http://lvee.org/ru/abstracts/252

Кроме этого, необходимо также определиться будет ли вам достаточно только соединения по WiFi или 3G/4G/LTE. Для того чтобы это сделать, нужно чётко понимать в каких целях будет использоваться ваш сервер. Если это будет простой веб-сервер, предназначенный для выполнения задач, которым не требуется постоянное, надёжное и безопасное соединение, то вполне будет достаточно 3G/4G/LTE соединения со статическим IP-адресом или обмена данными через WiFi-маршрутизатор. Совсем другое дело если вы захотите сделать максимально приближённый аналог обычного сервера, работающий по проводному соединению. Для этого вам понадобится либо устройство изначально оборудованное Ethernet-портом (а таких сейчас продаётся не очень много), либо переходник OTG и USB-сетевой адаптер (перед покупкой вам необходимо проверить поддерживает ли ваше устройство работу в качестве USB-хоста при подключении данного переходника). Кроме этого, в случае OTG-переходника и внешней USB сетевой карты вам необходимо будет также настроить проводную сеть, а это в зависимости от устройства может представлять собой не очень простой процесс.

На втором этапе вам необходимо определиться с серверным программным обеспечением, то есть выбрать подходящий веб-сервер, СУБД, а также другие вспомогательные компоненты. После этого, вам нужно будет загрузить их на ваше устройство. Тут к слову, как и в обычных Linux системах способы могут быть разные. Самый легкий способ это конечно поиск и установка готового набора через Google Play. Главный недостаток данного способа – это зависимость от поставщика набора серверных компонентов, а именно вы будет пользоваться серверными компонентами только тех версий, которые доступны в данном наборе, а обновления будут приходит только вместе с обновлением Android приложения, содержащего данный набор. К слову, из личного опыта, могу с уверенностью сказать что многие приложения содержащие такие наборы очень часто не обновляются. Кроме этого, за Android приложения содержащие внутри себя набор серверных компонентов вам скорее всего придётся заплатить тем или иным образом (либо путём покупки приложения, либо путём просмотра содержащейся в нём рекламы). Поэтому, этим способом я рекомендую пользоваться только если вы совсем не понимаете что такое Linux и как на нём можно устанавливать приложения из исходных кодов. Кроме первого способа на просторах интернета вы можете найти статьи про то, что установить серверные компоненты на Android вам поможет приложение

BotBrew (или аналогичное). В целом могу сказать, что данный способ вполне работоспособен, но есть несколько «но». Во первых эта BotBrew совместим далеко не со всеми устройствами. Во вторых он сильно грузит мобильное устройство. И в третьих эта штука по сути представляет собой что-то среднее, а именно после её установки вы получите некое подобие Debian репозитория из которого можно устанавливать приложения таким же способом как и в Debian. С одной стороны это хорошо, а с другой проблемы не самых свежих версий серверных компонентов никуда не исчезнут. В общем это способ могу рекомендовать только в тех случаях когда вам лень возиться с компиляцией из исходных кодов, вы хорошо знаете что такое Debian и на сто процентов уверены что на вашем устройстве это будет работать, а также если снижение производительности устройства вас не сильно беспокоит. И на всякий случай поделюсь своим опытом использования данной программы: после её установки планшет на базе Android 4 стал медленно работать и регулярно делать оптимизацию установленных приложений. После того как я понял в чём дело, удалить BotBrew позволил только сброс устройства на заводские настройки (Hard reset). Итого, если после прочитанного вы поняли что первых два способа вам не подойдут, то скорее всего мне удалось вас убедить что наиболее правильный и хороший способ установки серверных компонентов – это их сборка, компиляция и установка из исходных кодов. Ведь только так вы получите у себя на устройстве самые свежие версии компонентов, которые к тому же будут относительно быстро работать и безо всякой рекламы.

Итак, теперь чуть больше конкретики про процесс сборки, компиляции и установки приложений из исходных кодов под Android. В целом он полностью аналогичен таковому в обычном линуксе. Вам нужно будет скачать исходные коды с официальных сайтов (репозиториев) выбранных компонентов и приступить к компиляции и установке. Для этого , как и в Linux, вам надо будет использовать команду ./configure и make. Тем не менее есть и отличия. Так, например, в общем случае вам необходимо будет скачать и установить на ваш компьютер Eclipse с CDT, Android SDK, NDK и прочие необходимые компоненты и библиотеки. Сама сборка и компиляция для Android устройства будет осуществляться на вашем персональном компьютере. Далее после сборки и компиляции вам необходимо будет поместить полученный результат на Android устройство и немного «поколдовать» для его настройки и запуска.

Для этого нужно будет проверить сетевые настройки, конфигурационные файла каждого из серверных компонентов, ну и конечно проверить работает всё это или нет при помощи браузера (в случае веб сервера) или клиента (в случае СУБД и других сервисов). По сути это и есть этап проверки работоспособности. После завершения данного этапа можно будет смело попробовать сделать своё собственное приложение с набором серверных компонентов, но это не очень простой процесс, который имеет свои особенности и поэтому выходит за рамки данной темы, так как заслуживает отдельного рассмотрения.

Ну и в завершении, хочу сказать пару слов про оптимизацию. Если честно то без неё можно обойтись, но всё-таки лучше этим заняться. На мой взгляд оптимизировать надо всё, но в меру. Начинать оптимизацию я рекомендую с измерения производительности выбранных вами серверных компонентов и их влияния на систему в целом. Сейчас существует огромное множество различных программ-мониторов на любой вкус и цвет, поэтому мне даже сложно порекомендовать что-то конкретное. Ваша главная задача – это проверить при помощи таких программ не появилось ли у вас на мобильном устройстве какого либо серверного компонента который перегружает ваше устройство. Если случилось именно так, то вам необходимо либо произвести оптимизацию серверного компонента за счёт изменения настроек в конфигурационном файле или файлах данного компонента, либо, если первое не помогло, самое время задуматься о выборе другого компонента выполняющего данные функции. Как известно в мире Opensource обычно существует немалое число различных веб-сереверов, СУБД и прочих серверных приложений с отрытыми исходными кодами и вы всегда можете выбрать какую либо более легковесную замену с аналогичным функционалом. Кроме такого рода оптимизаций, в некоторых случаях могут быть также полезны архитектурные и аппаратные оптимизации. Например если мобильное устройство не тянет нужный вам набор серверных компонентов, то можно либо купить более мощное устройство взамен прежнего, либо разнести серверные компоненты на несколько устройств, тем самым поменяв архитектуру.

И последнее: в качестве подведения итогов я решил написать пару слов про то, зачем это всё нужно и почему вообще я решил этим заняться. Превращение мобильного устройства в сервер даёт целый ряд преимуществ: меньшее энергопотребление по сравнению

со стандартными серверами, высокая мобильность (вы можете всегда взять сервер с собой куда угодно), ну и конечно мобильные устройства занимают гораздо меньше физического пространства чем, например, их братья в модульном исполнении (серверы типоразмера 1U). У мобильных устройств, превращенных в серверы, есть конечно и недостатки, один из которых производительность, но тут всё зависит от того что и с чем сравнивать.

Практическое использование сервисов контейнеризации в облаке Амазон

Andrew Rewoonenco, Minsk, Belarus*

Amazon cloud service (AWS) provides several container-based services (EC2 CS, Lambda, CodeBuild). These services are quite special to Amazon and differ a lot from Open Source analogs however based on them. The goal of this report is to review common problems based on real experience and to show solutions author had find out.

Введение

Современные технологии часто ориентируются на облачные сервисы из-за их высокой надёжности и простой расширяемости. Один из старейших облачных провайдеров это Амазон (обычно сокоащаемый до AWS). Он предоставляет огромный набор различных сервисов, очень хорошо связанных в единое целое.

Здесь мы затронем сервисы, связаные с контейнеризацией. Доклад основан исключительно на эмпирических выводах (реальном использовании), и иногда достаточно сильно расходится с декларируемыми или ожидаемыми особенностями сервисов и их использования.

Из большого набора сервисов Амазона с контейнеризацией мы рассмотрим несколько сервисов, по которым есть достаточно достоверная выборка:

- сервис контейнеризации Амазона (EC2 CS);
- сервис краткосрочных операций (Lambda);
- сервис компиляции/тестирования (CodeBuild).

Общее у них следующее:

1. Все эти сервисы запускаются на виртуальных машинах (инстансах), работающих под управлением Amazon Linux.
2. Они построены на сервисе контейнеризации docker.
3. Возможно контролировать практически все параметры контейнера.

*arewoonenco@gmail.com, https://lvee.org/ru/abstracts/255

Рассмотрим декларируемое назначение, недостатки и способы их исправления для каждого из сервисов.

Сервис контейнеризации Амазона (EC2 CS)

Декларируемое назначение данного сервиса это разнообразные высокодоступные сервисы, обслуживающие входящий трафик, преимущественно веб-сервисы.

Обычный сценарий работы: создаётся кластер контейнеров из нескольких ВМ (инстансов), и сервис запускается на них. Для высокодоступности используется ещё один сервис Амазона — Балансировщик (ELB), перенаправляющий запросы по необходимости в несколько контейнеров и даже умеющий расширять их число при повышении нагрузки.

Но в реальных сценариях использования он имеет ряд недостатков:

1. Фиксированное ограничение CPU/Memory.
2. При размещении нескольких контейнеров на 1 ВМ это не работает, если котнейнеры используют одни и те же порты.
3. Не умеет распределять контейнеры правильно по датацентрам Амазона (Availability Zone), и из-за этого нестабильно работает балансировщик нагрузки.
4. Псегда надо иметь резерв по ВМ, иначе при перезапуске сервиса можно попасть в Deadlock.

Сервис краткосрочных операций (Lambda)

Декларируемое назначение данного сервиса это разнообразные высокодоступные краткосрочные клиент-сервисы, например, системы мониторинга, контроля, сообщений. Имеют жёткий лимит на время действия — не более 5 минут, используют только скрипт для работы (python, java jar, javascript nodejs). Не имеют входящих портов принципиально, но умеют обрабатывать сообщения из очередей Амазона (например SNS).

Сервис очень удобен для крошечных простейших операций (например, проверить веб-сервера и отослать СМС в случае недоступности) но при более-менее серьёзном использовании очень ограничен. В реальных сценариях использования имеет ряд недостатков:

1. Из-за времени использования многие возможности принципиально недоступны.

2. Нет входящих портов или протокола UDP за редкими исключениями.
3. Отладка просто чудовищно сложна из-за использования CloudWatch для хранения логов.
4. Работа с бинарными приложениями страшно неудобна и сложна.
5. Доступ к сервису SSH весьма нетривиален.

Сервис компиляции/тестирования (CodeBuild)

Декларируемое назначение данного сервиса это компиляция/сборка/тестирование разнообразного кода. В идеале позволяет заменить комбайн типа Jenkins на контейнеризованные решения. Стандартное использование — исходный код забирается с хранилища S3 (веб-хранилище Амазона), строится в контейнере, записывается снова на S3.

Сервис очень удобен для стандартный простейших операций (скомпилировать), но для более-менее сложных проектов ограничен. В реальных сценариях использования имеет ряд недостатков:

1. Для сервиса надо вначале подготовить базовый контейнер с системой сборки.
2. Отладка построения очень сложна из-за использования CloudWatch для хранения логов.
3. Сама постройка требует очень много времени из-за частей: развернуть контейнер, забрать файлы, положить файлы.
4. При постройке есть фиксированный набор этапов, он выполняется весь, даже если этап вернул ошибку, и только в конце сообщается о неудаче.
5. Интеграция «из коробки» через CodePipeline с git (например с гитхабом) очень ограничена.

Заключение

Как мы видим, декларируемые и реальные сценарии использования сервисов Амазона весьма отличаются. Несмотря на то, что все сервисы базируются на хорошо известных открытых решениях, стандартные сценарии их использования внутри системы Амазон не подходят. Да и рекомендованные самим Амазоном не особо хороши при использовании, чуть отклоняющемся от стандартного.

Taurus: тесты на любителя

Taras Svirinovski, Minsk, Belarus *

It makes sense to automate anything you repeat 10+ times.
Taurus improves experience of JMeter, Selenium and others.

Статья ставит целью продемонстрировать некоторые примеры практического применения Taurus tool. Материал предназначен в первую очередь разработчикам и другим техническим специалистам, непосредственно не занимающимся тестированием.

Введение

Taurus — это кроссплатформенный проект с открытым исходным кодом, предназначенный, в первую очередь, для упрощения автоматизации нагрузочных и функциональных тестов. Установка весьма проста — наш инструмент, в частности, распространяется через pip, поэтому для установки вам хватит Python'а (установщику под win не нужно и этого). Есть deb- и rpm-пакеты (что не особо актуально), есть образ для докера. Много дополнительного софта мы можем поставить сами, но часть — нет (например, JVM/JDK, TestNG, ...), но мы можем проверять на наличие в системе и в случае остутствия понятно ругаться.

Минимальный скрипт

Мало кто начинает работать с новым инструментом с чистого листа, совершенно не имея каких-либо наработок. Пусть у нас имеется каталог со скриптами для selenium, написанными на Python (а ещё Java, JavaScript,...). Самым популярным способом озадачить Tarus является передача ему скрипта на языке YAML. Рассмотрим как выглядит выполнение готовых скриптов:

```
execution:
- executor: selenium
  scenario:
    script: python_tests/
```

*grey.fenrir@gmail.com, https://lvee.org/ru/abstracts/246

Командная строка:

```
\$ bzt j1.yml
```

После запуска вы увидите псевдографическую консоль. Во многих случая она не обязательна, но гики любят, поэтому мы решили её добавить. Разумеется, возможности визуализации в текстовом режиме ограничены, но мы можем показать немало — текущую нагрузку в хитах/сек, изменение времени отклика, возникающие ошибки и предупреждения, затраченное/оставшееся время теста и многое другое. После окончания работы Taurus'а можно видеть краткий отчет, кроме этого для каждого запуска создается каталог с файлами логов и отчетов (artifacts dir).

Тест с параметрами

Усложним скрипт и разберем некоторые базовые опции и возможности.

```
execution:
- executor: jmeter
  concurrency: 10
  ramp-up: 30s
  hold-for: 1m
  scenario:
    requests:
    - http://blazedemo.com
modules:
  jmeter:
    version: 3.1
```

Здесь мы указали параметры теста (инструмент — jmteter, десять виртуальных пользователей, полминуты линейного роста нагрузки, минута её поддержания), сценарий теста (получить страницу по URL'у) и параметр для инструмента (запрос конкретной версии).

Multi execution

Рассмотрим несколько более развитый сценарий:

```
execution:
```

```
- executor: jmeter
  concurrency: 10
  hold-for: 1m
  ramp-up: 30s
  scenario: shared_scenario
- executor: pbench
  concurrency: 3
  hold-for: 20s
  delay: 10s
  scenario: shared_scenario
scenarios:
  shared_scenario:
    requests:
    - http://blazedemo.com
```

Обратите внимание на описание структуры в языке YAML — иерархия определяется отступами, элементы списка начинаются с дефиса, все остальные элементы, имеющие подчиненную структуру, считают ключами словаря. Здесь мы видим запрос выполнить два теста с разными инструментами, временными и количественными показателями, но с одним выделенным сценарием. Для разнообразия запустим тест с ключом -report и полюбуемся на результаты. Особой пользы пока нету, но что-то явно работает.

Дружелюбный selenium

Следующий тест позволяет зайти и выйти из веб-почты. Оцените краткость, читабельность, легкость составления и визуальный контроль работы.

```
execution:
- executor: selenium
  scenario:
    browser: Chrome
    requests:
    - url: \url{https://mail.ru
      actions:
      - keysByName(Login): strange_user
      - keysByName(Password): secret_password
      - clickByID(mailbox__auth__button)
      - waitByLinkText(выход)
```

```
    - clickByLinkText(выход)
```

Автоматизация

Рассмотрим типичную автоматизацию посредством csv-файлов. Итак, пускай имеются следующие данные:

```
\$ cat urls_and_keywords.csv
URL,KEYWORD
\url{https://java.com,Java
\url{https://www.python.org,Python
\url{https://lvee.org,LVEE 3017
```

Здесь мы видим через запятую адреса и ключевые слова на странице. Последняя строчка должна вызывать ошибку, так как конференция с таким названием в этом году не проводится.

```
execution:
- concurrency: 2
  iterations: 2
  scenario:
    data-sources:
    - urls_and_keywords.csv
    requests:
    - url: \${URL}
      assert:
      - \${KEYWORD}
```

После выполнения этого скрипта мы заглянем в kpi.jtl, размещенный в artifacts dir. Этот файл является csv-отчетом, который вернул jmeter. Обратим внимание на следующую часть таблицы:

label	responseCode	success
`https://java.com`	200	true
`https://www.python.org`	200	true
`https://lvee.org`	200	false

Здесь видно, как виртуальный пользователь по очереди запрашивает адреса из списка, определенного в csv-файле. Ответы получены (RC 200), но последний адрес не прошел проверку — и мы уже знаем, почему. На этом я желаю вам удачи с использованием нашего инструмента. Мы всегда рады помочь вам, выслушать предложения и рассмотреть пулл-реквесты. За контактами и документацией прошу на gettaurus.org.

The Invisible Internet Project *

Andrew Savchenko, Moscow, Russian Federation

This talk discusses why I2P network is needed, provides an insight on how the I2P network works, how it is different from Tor. Some use cases and safety tips will be discussed.

Introduction

The internet was created as a military DARPA project with security model oriented on external threat like infrastructure destruction. But its stack was not designed to withstand internal threats like IP spoofing or malicious routes injections, or to provide any kind on anonymity.

BPG hijacking cases are quite often these days [1], providers can easily spoof IP addresses or DNS names, http traffic is not secure and even https may be hijacked by bogus or broken CA providers. Search engines, markets, providers and other parties are actively collecting private user data.

To remedy this problems the Tor network was created. While it is a great achievement and compensates aforementioned problems up to a large extent, it is not perfect as any pioneer solution and was already compromised by the powerful adversaries [2]. The main Tor drawbacks are:

- highly centralized architecture
- most users use entry points and most resources are accessible using exit nodes, so endpoints are not protected from monitoring and spoofing
- just 1 hop between entry and exit points, this simplifies statistical analysis

To remedy these problems the second generation of overlay networks was created: the I2P (Invisible Internet project).

How I2P works

I2P and Tor have somewhat different goals: Tor is focused on providing an anonymous access to the Internet resources, while providing

*bircoph@gmail.com, http://lvee.org/ru/abstracts/258

hidden services as an additional byproduct. I2P is focused on hidden services only, the network itself does not provide any entrance or exit points to the clear net; of course it is always possible to setup gateways both ways on individual machines.

In I2P each node, every client or server are the network routers by design. Routes in I2P are unidirectional, so request and replies are travelling different paths. Furthermore I2P does not use onion routing scheme like in Tor, instead it uses the garlic routing [3]: it is a multi-layered encryption scheme:

- each transmitted packet is being encrypted and split into chunks which are send via different tunnels
- each chunk is mixed with random chunks from transit traffic to form a new packet
- between routers P2P encryption is used the same way as in onion scheme
- number of hops is not limited and may be changed run-time depending on application needs (usually 2 —3 hops are used)
- multiple different unidirectional tunnels are being used to send and receive data

This way even if adversary will be able to break router-to-router decryption only chunks of random encrypted packets will be available.

All I2P peers are addressed by 516-byte (or longer) keys containing 256 bytes of public key and 128 bytes of signature. For convenience each address can be shortened to 32 bytes using SHA-256 hash, these addresses are known as b32.i2p. The tunnels for these addresses are available using DHT. This way it is not possible to spoof peer's address like it can be done in IP network and with much more effort in Tor network, since Tor uses now insecure SHA-1 for that purposes. Only private key holder can decrypt assembled packet.

There is not DNS service in I2P. Address books are used instead and can be fetched from multiple sources, they are just maps between shorter names and b32 addresses and contain full public key for each b32. Of course each host is reachable using b32 address as well. This way is resilient to an attack on centralized directories like in case of root DNS servers or Tor directory services, because there are not directory servers in I2P.

I2P supports number of access protocols, SOCKS is the most commonly used one. Any user can easily setup a unique b32 address for each services available from his host, any TCP or UDP port may be exported.

Implementations

There are two implementations of I2P available, both are the free software. Original implementation [4] is written in Java.

Due to Java shortcomings in both performance and memory consumption an alternative i2pd [5], [6] implementation in C++ was created, allowing to run I2P daemon even on Raspberri Pi class devices [7].

I2pd is fully compatible with Java I2P, but provides an additional feature of optionally using GOST 34.10 signing with both CryptoPro and TC26 parameter sets and GOST 34.11 hashing [8]. Considering the open nature of TC26 parameter set and fully open algorithms this provides a very good level of data protection, at least from NSA and alike adversaries.

Usage hints

Start your experience from any I2P wiki and/or identity service.

Aside from using http resources there is an interesting application in building your own VPN, since I2P works well behind NAT: just export your own services, you may use b32 white listing for clients as well. Since it is impossible to find unpublished b32 except using full address space brute force, you will have additional privacy.

Even if I2P network may be considered secure itself, do not forget that incautious user will easily lose anonymity and privacy by advanced fingerprinting, so:

- do not allow javascript, flash and other plugins
- do not use persistent cache
- do not use the same browser for clear net, tor and i2p
- use isolated environments for both daemon and browser (or other clients)

References

[1] BGP hijacking `https://bgpmon.net/category/hijack/`

[2] How Did The FBI Break Tor? `https://www.forbes.com/sites/kashmirhill/2014/11/07/how-did-law-enforcement-break-tor/`

[3] Garlic routing `https://geti2p.net/en/docs/how/garlic-routing`

[4] The I2P https://geti2p.net/en
[5] i2pd https://github.com/PurpleI2P/i2pd
[6] i2pd docs https://i2pd.readthedocs.io/en
[7] Cross-Compile static I2PD for Raspberry Pi https://i2p.rocks/blog/cross-compile-static-i2pd-for-raspberry-pi.html
[8] GOST R in I2P https://habrahabr.ru/post/325666/

Основы IPv6

Ivan Semernik, Minsk, Belarus *

2016 is a year when exponential growth of world IPv6 adoption was confirmed. Fifth part of Global IP traffic going over IPv6 now. Per country adoption level of IPv6 is very different (from 0% to almost 50% in Europe region). Belarus is only 0,03%. There are many reasons for that and poor technical knowledge of IPv6 is one of them.

IPv6 — новая версия протокола IP, призванная решить проблемы, с которыми столкнулась предыдущая версия (IPv4) при её использовании в Интернете, за счёт использования 128-битной адресации вместо 32-битной. Протокол был разработан IETF[1].

В настоящее время протокол IPv6 уже используется в нескольких десятках тысячах сетей по всему миру (более 40000 сетей на лето 2017 года), но пока ещё не получил столь широкого распространения в Интернете, как IPv4. 2016 год стал годом подтвержденного экспонетциального роста IPv6-подключений по всему миру [2]. На середину 2017 года доля IPv6 в общемировом сетевом трафике составляет около 18% [3].

После того, как адресное пространство в IPv4 закончится, два стека протоколов — IPv6 и IPv4 — будут использоваться параллельно (IP dual stack), с постепенным увеличением доли трафика IPv6, по сравнению с IPv4.

При разработке нового протокола IPv6 были учтены многие проблемы и узкие места протокола IPv4.

Краткое сравнение IPv4 и IPv6 приведено в таблице.

В IPv6 был значительно упрощен формат заголовков пакета, благодаря чему заголовок пакета удлинился всего лишь до 40 байт (фиксированный размер пакета) несмотря на то, что 32 байта из них занимает адресная информация. Появились метки потоков и классы трафика.

Формат адреса IPv6 отличается от IPv4 и состоит из следующих основных частей (на примере адреса 2001:db8:f3d:1::1):

*ivan.semernik@gmail.com, https://lvee.org/ru/abstracts/250

	IPv4 (с 1981)	IPv6 (с 1998)
Разрядность	32 бита	128 бит
Пример адреса	212.98.163.254	2001:db8:6:56::53
Адресное пространство:	4 294 967 296 адресов	340 282 366 920 938 463 463 374 607 431 768 211 456 адресов
на человека	~ 1	$4{,}7 * 10^{28}$
на $1km^2$ поверхности Земли	$\sim 8{,}5$	$6{,}7 * 10^{26}$
MTU (минимальный, байт)	576, фрагментация опционально	1280, без фрагментации
Фрагментация	Роутеры и хосты	Только хосты
DNS записи	A PTR in-addr.arpa	AAAA PTR ip6.arpa
Настройка адреса	Ручная или DHCP	Ручная, StateLess Address AutoConfiguration (SLAAC) и/или DHCPv6
Определение MAC	Broadcast ARP	Multicast Neighbor Solicitation
Broadcast	Да	Нет
Multicast\Anycast	Да\Да	Да\Да
IPSec заголовки	Опционально	Обязательно, но по факту никто не пользуется
Маска подсети	Да	Нет

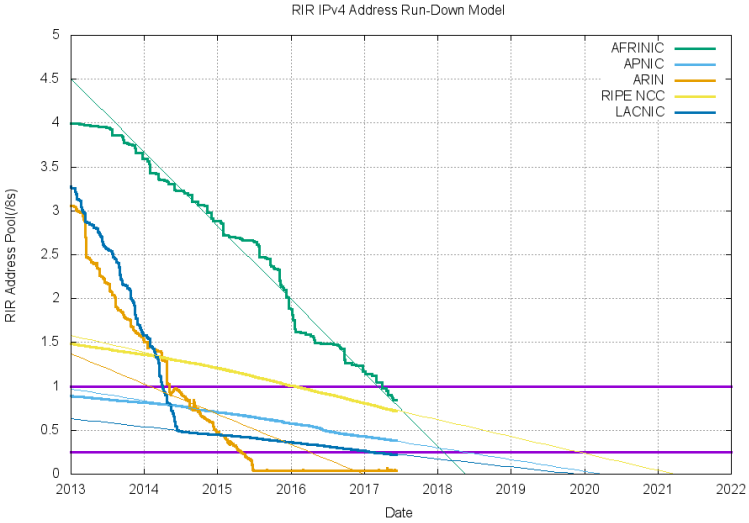


Рис. 17.1. Прогноз исчерпания адресного пространства IPv4 региональными регистратурами, Geoff Huston / ipv4.potaroo.net

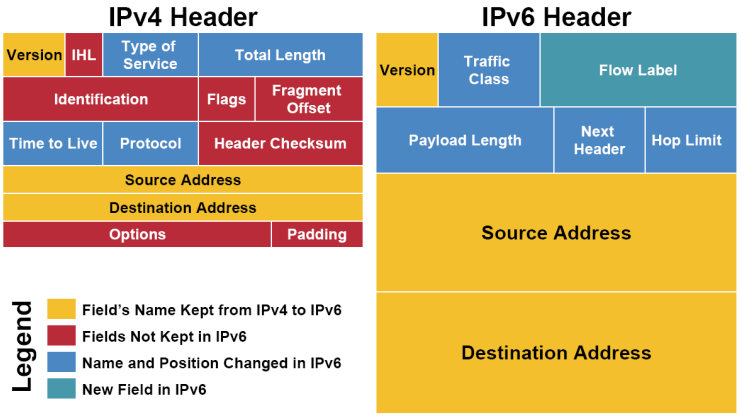


Рис. 17.2. Формат заголовков пакетов IPv4 и IPv6. [4]

Site prefix обычно назначается провайдером или регистратурой используется при маршрутизации. Subnet ID используется в то-

2001:0db8:0f3d	:0001	:0000:0000:0000:0001
Site prefix (48 bit)	Subnet ID (16 bit)	Interface ID (64 bit)

пологии сети клиента, а Interface ID назначается на клиентские устройства автоматически либо вручную.

Различают полную и сжатую (compressed) формы записи IPv6-адресов. При этом существует два основных правила сжатия формы записи:

1. Лидирующие нули (слева) в пределах квартета (4 байта) могут не указываться.
2. Любое количество идущих подряд нулей можно заменить на ::, но только один раз. Если в адресе присутствуют две одинаковые по длине последовательности нулей разделенных любым другим числом, на :: заменяется та, что находится левее.

Примеры:
- 2001:0db8:0ffc:0008:0000:0000:0000:002f — полная форма записи
- 2001:db8:ffc:8::2f — сжатая форма записи
- 2001:0db8:0000:0000:0001:0000:0000:0001 — полная форма записи
- 2001:db8::1:0:0:1 — сжатая форма записи

При доступе к IPv6-адресу через URL необходимо использовать квадратные кавычки. Пример: `http://[2001:db8:ffff::2]:8081`.

В IPv6 появились такие новые понятия как области адресов (address scopes) и специальные адреса. Сегодня активно используется более 16 специальных адресов (например, loopback, unspecified, uniq local addresses, есть даже отдельные адреса для целей документации — 2001:db8::/32) и 7 основных областей адресов (global scope, link-local scope, interface-local и др.)

Адрес IPv6 может быть вручную сконфигурирован на сетевом интерфейсе, либо назначен автоматически с помощью механизма SLAAC, заложенного в сам протокол. Также поддерживается автоматическое назначение IPv6-адресов с помощью DHCPv6-сервера. Для выдачи клиенту не одного адреса, а определённой сети (префикса) широко используется DHCPv6 Prfix Delegation [5]. Любой сетевой интерфейс в современных операционных системах всегда

имеет сконфигурированный автоматически IPv6-адрес с областью действия link-local.

В отличает от протокола IPv4, работа которого базируется на протоколе ARP, в IPv6 ипользуется ICMP unicast и milticast.

Координационный центр европейского сообщества пользователей сети Интернет — RIPE NCC подготовил серию OnLine-курсов, которые являются хорошим началом на пути освоения нового протокола IPv6 [6].

Литература

[1] https://ru.wikipedia.org/wiki/IPv6

[2] https://www.youtube.com/watch?v=s61g_l9NoR8, AlexSemenyaka,RIPENCC

[3] https://www.google.com/intl/en/ipv6/statistics.html

[4] https://343networks.files.wordpress.com/2010/06/ipv4-ipv6-header.gif

[5] https://en.wikipedia.org/wiki/Prefix_delegation

[6] https://academy.ripe.net/course/view.php?id=2 , требуется регистрация на сайте сообщества.

article

Использование открытых исходных кодов для разработки 3D сканера и соответствующего программного обеспечения для 3D реконструкции моделей.

Лина Ширяева, Минск, Belarus *

Nowadays 3D scanning is widely used in different areas of activity. We developed 3D scanner and software using opensource code and open technologies. Using this scanner you can reconstruct models, which size is until 3 metres, with accuracy about 1 mm.

Введение и предпосылки

В настоящее время 3D-сканирование используется в различных сферах деятельности, таких как инженерный анализ, промышленный дизайн, цифровое архивирование, развлечение и игры, медицина и ортопедия и др. Множество конструкций современных 3D-сканеров можно разделить на два типа: лазерные и оптические. На текущий момент существуют свободные аппаратные реализации как лазерных [1], так и оптических [2] 3D-сканеров. Однако нам довелось познакомиться с 3D-сканированием поближе при решении достаточно специфической задачи — в ходе разработки оптического 3D-сканера с возможностью сканирования объектов размерами до 3 метров, в связи с чем существующие свободные решения в их исходном виде были неприменимы.

При реализации проекта ставились следующие основные цели: относительно невысокая стоимость сканера, высокая скорость 3D-сканирования и реконструкции, использование открытых источников (исходные коды, библиотеки, SDK) для создания программного обеспечения управления 3D-сканером и работы с 3D-моделями.

Выбор метода 3D-реконструкции

Задача 3D-реконструкции объекта заключается в определении координат точек объекта в трехмерном пространстве — то есть

*linashiryaeva@gmail.com, https://lvee.org/ru/abstracts/257

в получении облака точек объекта. Существует множество различных технологий построения оптических 3D-сканеров: например, камера и поворотный стол [3], различное количество камер (4 и более, между которыми помещают объект), камера и проектор, и другие. Нами была выбрана открытая технология Structed Light [4] (метод структурной подсветки). В методе структурной подсветки задача реконструкции решается с помощью проектора и камеры: проектор проецирует на объект специальное изображение (структурная подсветка), а фотоаппарат регистрирует объект со специальным изображением (рис. 18.1).

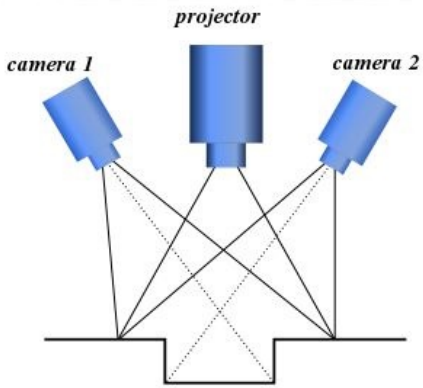


Рис. 18.1. Выбранная конструкция 3D-сканера

Подробнее о разработке

В разработке 3D-сканера, соответствующего ПО к нему и, соответственно, в самом процессе сканирования можно выделить следующие элементы:

1. *Конструкция сканера.* В разработанном сканере используются две камеры Canon серии EOS и LCD-проектор разрешением 1920×1080 пикселей. Выбранные камеры предоставляют возможность удаленного управления через интерфейс USB. Для

написания клиентского приложения, управляющего камерами, использовался родной SDK Canon EOS [5], реализованный на языке C.

2. *Калибровка.* Калибровка 3D-сканера состоит из двух этапов: внутренней калибровки отдельных частей сканера, в результате которой рассчитываются фокусные расстояния и дисторсии камер и проектора, и внешней или геометрической калибровки установки, в результате которой мы получаем матрицы поворота и смещения для сопоставления системы координат проектора и камеры. Самым распространенным и простым способом калибровки камеры в данный момент считается метод Zhengyou Zhang [6], основанный на использовании плоского шаблона в виде шахматной доски. Данный метод реализован в открытой библиотеке OpenCV [7], [8], распространяемой на условиях лицензии BSD (написана на C/C++, а также имеет биндинги для Python, Java, Ruby, Lua и др. популярных языков).
3. *Сканирование.* Основным недостатком метода с использованием нескольких камер без проектора является сложность сопоставления точек двух изображений (т.е. определения координат одной и той же точки на одном и втором изображении). Точка должна обладать характерными особенностями, чтобы ее можно было однозначно идентифицировать. В методе структурной подсветки эта проблема решается за счет правильно подобранного проецируемого шаблона – в итоге мы получаем больше точек (в идеале – все освещенные проектором точки объекта) при меньшем количестве изображений. Т.е. основная задача состоит в выборе такого способа подсветки, который позволит однозначно определить, какая точка изображения проектора освещает точку объекта, зарегистрированную камерой. Мы выбрали для исследования серые коды, как более точные для декодирования. На объект проецируются серые коды для вертикальных и горизонтальных линий. В Structed Light есть реализация двух технологий декодирования: ray-ray (пересечение луча камеры и луча проектора) и ray-plane (пересечение луча камеры с плоскостью проектора). К преимуществам метода ray-ray следует отнести более высокую точность реконструкции в сравнении с методом ray-plane за счет использования как вертикальных, так

и горизонтальных линий паттернов, что приводит к увеличению времени сканирования.

4. *3D-реконструкция.* Построение 3D-облака состоит из следующих этапов: декодирование, расчет оптических лучей камеры и проектора (для метода ray-ray) или расчет оптических лучей камеры и плоскости проектора (для метода ray-plane). Координата точки пересечения лучей камеры и проектора (или плоскости проектора) и является искомой координатой точки 3D-облака. Для работы с 3D-облаками используется Point Cloud Library (PCL) [9] — кроссплатформенная открытая библиотека (Linux, MacOS, Windows и Android/IOS), которая также распространяется в условиях лицензии BSD. В данной библиотеке реализованы различные методы обработки 3D-облаков, такие как сглаживание, фильтрация, совмещения облаков и т.д. Данная библиотека позволяет сохранять 3D-облака в общеизвестных форматах (*.ply, *.obj, *.stl). Начиная с версии 1.8.0 PCL поддерживает распараллеливание расчетов некоторых алгоритмов на GPU. Результат 3D-реконструкции представлен на рис. 18.2.

5. *Построение поверхности.* Облако точек объекта дискретно, и размер точки определяется размером минимально различимого на изображении элемента – пикселя изображения. Чтобы получить данные о всей поверхности объекта, необходимо интерполировать пространство между точками некими поверхностями или функцией. В случае больших и сложных объектов интерполяция одной функцией невозможна. Поэтому обычно применяется интерполяция набором геометрических фигур – «кусочков» плоскостей. Самый простой способ – интерполяция треугольниками (триангуляция). Три точки треугольника однозначно определяют плоскость в пространстве. Для создания триангуляции мы использовали программу с открытым исходным кодом MeshLab [10]. Данная программа распространяется под лицензией GPLv2, имеет реализацию под Linux, Windows и MacOS и предоставляет большой функционал по обработке 3D-облаков и поверхностей.

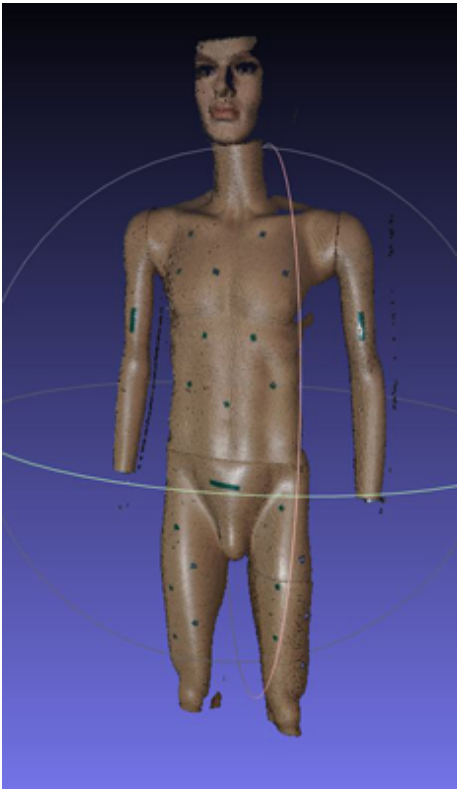


Рис. 18.2. Пример 3D-реконструкции

Заключение

В заключении хотелось отметить, что использованные особенности конструкции и описанные программные компоненты позволилил создать 3D-сканер с высокой высокой точностью реконструкции (относительная погрешность порядка 1 мм), быстрой скоростью съемки и обработки данных (менее 1 минуты от начала процесса сканирования до получения результата).

Литература

[1] `http://www.instructables.com/id/`

DIY-Arduino-3D-Laser-Scanner/
[2] http://www.makerscanner.com/docs/1-makerscanner.html
[3] http://photopizza.ru/
[4] http://mesh.brown.edu/byo3d/source.html
[5] https://www.developersupport.canon.com
[6] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf
[7] http://opencv.org
[8] http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html
[9] http://pointclouds.org/
[10] http://www.meshlab.net/#description

The Invisible Internet Project

Andrew Savchenko, Moscow, Russian Federation *

This talk discusses why I2P network is needed, provides an insight on how the I2P network works, how it is different from Tor. Some use cases and safety tips will be discussed.

Introduction

The internet was created as a military DARPA project with security model oriented on external threat like infrastructure destruction. But its stack was not designed to withstand internal threats like IP spoofing or malicious routes injections, or to provide any kind on anonymity.

BPG hijacking cases are quite often these days [1], providers can easily spoof IP addresses or DNS names, http traffic is not secure and even https may be hijacked by bogus or broken CA providers. Search engines, markets, providers and other parties are actively collecting private user data.

To remedy this problems the Tor network was created. While it is a great achievement and compensates aforementioned problems up to a large extent, it is not perfect as any pioneer solution and was already compromised by the powerful adversaries [2]. The main Tor drawbacks are:

- highly centralized architecture
- most users use entry points and most resources are accessible using exit nodes, so endpoints are not protected from monitoring and spoofing
- just 1 hop between entry and exit points, this simplifies statistical analysis

To remedy these problems the second generation of overlay networks was created: the I2P (Invisible Internet project).

How I2P works

I2P and Tor have somewhat different goals: Tor is focused on providing an anonymous access to the Internet resources, while

*bircoph@gmail.com, https://lvee.org/ru/abstracts/258

providing hidden services as an additional byproduct. I2P is focused on hidden services only, the network itself does not provide any entrance or exit points to the clear net; of course it is always possible to setup gateways both ways on individual machines.

In I2P each node, every client or server are the network routers by design. Routes in I2P are unidirectional, so request and replies are travelling different paths. Furthermore I2P does not use onion routing scheme like in Tor, instead it uses the garlic routing [3]: it is a multi-layered encryption scheme:

- each transmitted packet is being encrypted and split into chunks which are send via different tunnels
- each chunk is mixed with random chunks from transit traffic to form a new packet
- between routers P2P encryption is used the same way as in onion scheme
- number of hops is not limited and may be changed run-time depending on application needs (usually 2-3 hops are used)
- multiple different unidirectional tunnels are being used to send and receive data

This way even if adversary will be able to break router-to-router decryption only chunks of random encrypted packets will be available.

All I2P peers are addressed by 516-byte (or longer) keys containing 256 bytes of public key and 128 bytes of signature. For convenience each address can be shortened to 32 bytes using SHA-256 hash, these addresses are known as b32.i2p. The tunnels for these addresses are available using DHT. This way it is not possible to spoof peer's address like it can be done in IP network and with much more effort in Tor network, since Tor uses now insecure SHA-1 for that purposes. Only private key holder can decrypt assembled packet.

There is not DNS service in I2P. Address books are used instead and can be fetched from multiple sources, they are just maps between shorter names and b32 addresses and contain full public key for each b32. Of course each host is reachable using b32 address as well. This way is resilient to an attack on centralized directories like in case of root DNS servers or Tor directory services, because there are not directory servers in I2P.

I2P supports number of access protocols, SOCKS is the most commonly used one. Any user can easily setup a unique b32 address for each services available from his host, any TCP or UDP port may be exported.

Implementations

There are two implementations of I2P available, both are the free software. Original implementation [4] is written in Java.

Due to Java shortcomings in both performance and memory consumption an alternative i2pd[5,6] implementation in C++ was created, allowing to run I2P daemon even on Raspberri Pi class devices [7].

I2pd is fully compatible with Java I2P, but provides an additional feature of optionally using GOST 34.10 signing with both CryptoPro and TC26 parameter sets and GOST 34.11 hashing [8]. Considering the open nature of TC26 parameter set and fully open algorithms this provides a very good level of data protection, at least from NSA and alike adversaries.

Usage hints

Start your experience from any I2P wiki and/or identity service.

Aside from using http resources there is an interesting application in building your own VPN, since I2P works well behind NAT: just export your own services, you may use b32 white listing for clients as well. Since it is impossible to find unpublished b32 except using full address space brute force, you will have additional privacy.

Even if I2P network may be considered secure itself, do not forget that incautious user will easily lose anonymity and privacy by advanced fingerprinting, so:

- do not allow javascript, flash and other plugins
- do not use persistent cache
- do not use the same browser for clear net, tor and i2p
- use isolated environments for both daemon and browser (or other clients)

References

[1] BGP hijacking `https://bgpmon.net/category/hijack/`

[2] How Did The FBI Break Tor? `https://www.forbes.com/sites/kashmirhill/2014/11/07/how-did-law-enforcement-break-tor/`

[3] Garlic routing `https://geti2p.net/en/docs/how/garlic-routing`

[4] The I2P https://geti2p.net/en
[5] i2pd https://github.com/PurpleI2P/i2pd
[6] i2pd docs https://i2pd.readthedocs.io/en
[7] Cross-Compile static I2PD for Raspberry Pi https://i2p.rocks/blog/cross-compile-static-i2pd-for-raspberry-pi.html
[8] GOST R in I2P https://habrahabr.ru/post/325666/

«Эльбрус» на альте

Михаил Шигорин, Москва, Russian Federation *

The report covers state of progress of porting ALT Linux to Elbrus machines, which is currently self-hosted and targeted at rebuilding the Sisiphus package base.

Как упоминалось в докладе прошлого года [1], в «Базальт СПО» начали эксперименты по переносу своего дистрибутива на новую аппаратную платформу e2k в виде рабочей станции «Эльбрус-401». Ранней весной 2017 года она уже была переведена на загрузку с альтовского корневого раздела, и в основном начальная «раскрутка» была завершена с формированием пакетного репозитория ёмкостью более 1200 пакетов с исходным кодом в основном из Sisyphus. На данный момент число пакетов уже превысило 1500.

Изначально работа велась в chroot под управлением штатной ОС Эльбрус (OSL). После пересборки репозитория в hasher [1] и запуска mkimage вместе с mkimage-profiles [2] получилось изготовить архив корневой файловой системы, каковой и был в итоге развёрнут на отдельном диске. Отдельное спасибо разработчикам удобной в этом плане базовой прошивки, умеющей загружать ядро и образ initrd с файловой системы Ext2, что позволяет отказаться от GRUB.

По мере расширения репозитория менялись и проблемы, с которыми приходилось иметь дело — от жёстко заданного пути самых первых шагов к минимальной возможности подумать о том, куда и как двигаться дальше, а затем — с выходом на «оперативный простор» базовых сборочных зависимостей — скорее «какие подсистемы в каком порядке удобней брать в работу» (Qt? Java?).

Судя по текущему положению дел, в итоге получается первая собранная на нынешнем «Эльбрусе» без применения кросс-компиляции операционная система общего назначения.

На момент подачи тезисов (начало лета 2017 года) сборка переносится на четырёхпроцессорный «Эльбрус 4.4» с практически линейным ростом скорости сборки — и эта машина, разумеется, тоже загружена под альтом с уже собранным нами ядром.

*mike@altlinux.org, https://lvee.org/ru/abstracts/251

Литература

[1] Шигорин М. Альт на «Эльбрусе». Материалы LVEE Winter 2016. http://lvee.org/ru/abstracts/180

[2] Шигорин М. Макраме из дистрибутивов: mkimage-profiles. Материалы LVEE 2012 https://lvee.org/ru/reports/LVEE_2012_11

Голос спонсора: EPAM Systems

Голос спонсора: SaM Solutions

Голос спонсора: mycloud.by

Интервью с участниками

По традиции в сборник материалов входят интервью, в которых активные участники сообщества open source делятся своим мнением о свободном ПО, открытых технологиях, роли и месте свободных лицензий, рассказывают, как видят проблематику свободных проектов. В этот раз, из-за англоязычности всех интервьюируемых, интервью приводятся на двух языках - английском и русском.

Paweł Chojnacki

Warsaw, Poland.

LVEE: Can you briefly introduce yourself?
Paweł Chojnacki: My name is Paweł, I live in Warsaw. I used to be a part of Warsaw hackerspace; right now I'm not engaged in any organizational activity, may be beside the Global Innovation Gathering (`globalinnovationgathering.com`), a community which also involves hackering. I'm a web developer, a freelancer, a frontend professional doing some Open Access Science in the meantime. I deal a lot with open source in my professional activity. And in addition to that I'm giving cybersecurity lectures, an introduction to cybersecurity: what to store, why should you use Firefox and Chrome but not Internet Explorer and Safari. And I work with open source every day.
L: How did you get acquainted with open source software? Do you remember?
P: I've started using Firefox very early in my life. It was the first year of Firefox existence, I don't remember exactly which year it was. But I didn't consider it open source, I knew nothing about it. So my first conscious meeting with open source was related to GIMP graphic editor. I just wanted some program which had more options than Paint, and I didn't want payed Photoshop version. That's why I learned GIMP. I looked through one tutorial, and the second, and started learning to work with GIMP. Than I met some people speaking Polish who were interested in GIMP and helped them to found Polish Gimp Users' Forum (messaging board for Polish-speaking GIMP users). And from that I started reading and got very interested in Linux and the open source philosophy, and that was, I think, in mid school and high school.
L: By the way, did you have some previous experience with Photoshop, before starting GIMP, or not?
P: I didn't have any previous experience. Nothing with Photoshop, or Corel, or anything else.
L: It's an interesting question, because we know a lot of examples, when different previous experience had complicated people using GIMP more or less. And you have also mentioned Hackerspace. It's a hackerspace in Warsaw?
P: Yeah. Warsaw hackerspace.

Paweł Chojnacki

Варшава, Польша.

LVEE: Для начала, несколько слов о себе.

Paweł Chojnacki: Меня зовут Павел, я живу в Варшаве. Раньше активно участвовал в Варшавском Хакерспейсе; в настоящий момент организационной активностью не занимаюсь, ну может быть кроме Global Innovation Gathering (`globalinnovationgathering.com`), сообщество, связанное в том числе с хакерством. Я веб-разработчик, фрилансер, специалист по разработке фронтэнда, а кроме того занимаюсь наукой с открытым доступом (Open Access). Мне много приходится работать со свободным ПО в профессиональной деятельности. А еще я читаю лекции по компьютерной безопасности, такое введение в информационную безопасность: что хранить, почему использовать Firefox и Chrome, а не Internet Explorer или Safari. Ну и каждый день имею дело со свободным ПО.

L: Не вспомнишь, как ты вообще познакомился со свободным ПО?

P: Я начал пользоваться Firefox в очень раннем возрасте. Это был первый год существования Firefox как такового, сейчас не вспомню, в каком именно году. Но я не воспринимал его как свободное ПО, просто ничего об этом не знал. Так что первая сознательная встреча со свободным ПО была связана с графическим редактором GIMP. Мне была нужна какая-нибудь программа с бóльшими чем у Paint возможностями, и мне не хотелось иметь дело с платной версией Photoshop. Поэтому я изучил GIMP. Заглянул в один туториал, потом в другой, и начал учиться с ним работать. Затем я познакомился с польскоговорящими людьми, которые интересовались GIMP, и помог им создать Polish Gimp Users' Forum (площадку для польскоговорящих пользователей GIMP). И начиная с этого момента я стал читать, заинтересовался Linux и философией свободного ПО, думаю, это было в средних или старших классах.

L: А кстати, у тебя уже был какой-то опыт работы в Photoshop до GIMP или нет?

P: Никакого прежнего опыта. Ни с Photoshop, ни с Corel, ни с чем-либо ещё.

L: The topic of hackerspace itself is rather interesting, because we have finally officially registered Hackerspace in Minsk some time ago. How long does it work in Warsaw?

P: Four or five years. I was in the board of hackerspace, and was basically driving it full time as a community manager, or whatever you want to call it, for several months. It was founded basically by some enthusiasts but got wheel speed when Warsaw University of Technology decided to shut down their experimental technology division and all people who wanted to experiment went to the hackerspace. And right now Warsaw hackerspace is a club. It's typical members are people who have quite similar background from the Internet, who chat there... Even with a similar sense of humor. Those who like specific areas of IT, the security, administration, and who want to contribute to some projects or just to use technology for fun.

L: Usually hackerspace communities have strong connections with some open hardware projects: they are either developing their own or using some devices previously developed under such open hardware licenses. Perhaps there is the same situation in Warsaw, isn't it?

P: We don't have any projects that the hackerspace is creating as a hackerspace, but several members have created several hacks, for example way to install OpenWrt on specific devices which where unknown to be able of doing this.

Also there is some tool for gathering money from the members, just to send e-mails with reminders, and show current status. So hackerspace has some infrastructure. It is open but not necessary all of it is open source.

A lot of things are based on Arduino, but I wouldn't call it open hardware with us, as not all this things are documented and shared. But at least few things are.

L: At the very beginning open hardware was exotic, it was even supposed that open source licenses are good for software but not for hardware and other areas, like art. But well, later several really popular projects have appeared published under such licenses. How do you think, what's the reason of their success?

P: I think it is because they bring right now technology to the people who have open hardware which is also standartized.

L: Это интересный вопрос, потому что есть достаточно примеров, когда предыдущий опыт с чем-то другим в той или иной степени осложнял людям изучение GIMP. Итак, ты упоминал хакерспейс. Это хакерспейс в Варшаве?

P: Да. Варшавский хакерспейс.

L: Тема хакерспейса сама по себе интересна, потому что не так давно мы наконец официально зарегистрировали хакерспейс в Минске. Как долго он существует в Варшаве?

P: Четыре или пять лет. Я был в совете хакерспейса, несколько месяцев фактически тянул его на себе на штатной основе в качестве комьюнити-менеджера, или как ещё можно назвать эту должность. Он был сначала основан несколькими энтузиастами, а раскрутился, когда в Варшавском технологическом университете закрылось экспериментальное технологическое отделение, и тога все, кто хотел заниматься экспериментами, отправились в хакерспейс. А сейчас Варшавский хакреспейс — это клуб. Его типичные члены — это люди со схожим опытом в Интернет, которые в нём общаются... Вплоть до схожего чувства юмора. Те, кому нравятся определённые области IT, безопасность, администрирование, кто хотел бы вносить вклад в некоторые проекты или просто возиться с технологиями ради собственного удовольствия.

L: Обычно комьюнити хакерспейсов имеют сильные связи с проектами свободного аппаратного обеспечения: разрабатывают что-то своё, или используют устройства, созданные ранее под свободными лицензиями. В Варшаве наверное такая же ситуация?

P: У нас нет проектов, создаваемых в хакерспейсе именно от имени хакерспейса, но некоторые члены были инициаторами некоторых хаков — например, способ установить прошивку OpenWrt на некоторые устройства, для которых прежде ничего не было известно о такой возможности.

Ещё есть инструмент для сбора денег с участников: напоминания по e-mails и т. д., отображение текущего статуса. Понятно, у хакерспейса есть определенная инфраструктура. Она открыта, но не обязательно является свободной вся целиком.

Многие вещи построены на Arduino, но в нашем случае я не назвал бы это свободным аппаратным обеспечением, потому что не все проекты задокументированы и выложены в общий доступ. Но несколько — да.

Because right now you can take Arduino, take Raspberry Pi, and you can be sure that the most of software written for the one piece will work on any similar kind of chip, anywhere in the world.

For example we have OpenBCI brain amplifier created by a small startup in US, which is becoming a new Arduino for brain amplifying, for just brain signal analysis which is also a very important for the community. There are a lot of other projects...

So they are bringing the costs down, as you don't have any kind of overprotection from company that puts control and keeps the prices so that they are profitable.

This hardware projects are just tools for sharing knowledge and for sharing technical experience using them as building blocks in practice.

L: Thank you. I think that open source is the most powerful in software areas nowadays, but with open hardware and open media like Wikipedia for example, or creative Commons materials online – perhaps together they are three most widely used open directions.

P: I think you forgot about one thing. You've mentioned the media as Creative Commons and Wikipedia, but there's also open access and open notebook at science.

Right now I'm working on the scientific project of my own, creating it with the totally open notebook access, so that I'm commenting every step of my experiment, with use of open source tools. This approach allows people to replicate the experiment in same environment. If I actually had the equipment which would be open hardware, it could be much easier to actually replicate the data gathering.

L: It's really interesting, as it's not very easy to reproduce results of other researchers sometimes. And open source software has some powerful positions in scientific research because it allows to see sources and control how where all the calculations really done.

P: Yes especially, but there also may be problems. I don't know if you have heard, recently we have lost about 40,000 articles regarding neurology and fMRI, because someone put the wrong code into scientific tools used to monitor them.

The package itself was open source, but nobody was actively testing it, nobody was actively looking. So it's not only about saying "Hey this is open", as universities are saying – "This piece of software is open, but please don't look at it because only we understand how it works".

L: В самом начале свободное аппаратное обеспечение считалось экзотикой, считалось даже, что свободные лицензии хороши для программ, но не для аппаратуры и некоторых других областей, как например искусство. Но однако, позже появилось несколько действительно популярных проектов именно под такими лицензиями. Как ты думаешь, в чём причина их успеха?

P: Думаю, в том, что они прямо сейчас несут технологию людям, в распоряжении которых есть аппаратура, свободная и при этом стандартизованная.

Прямо сейчас можно взять Arduino или Raspberry Pi и быть уверенным, что бóльшая часть ПО, написанного для одного устройства, будет работать на таком же чипе в любой точке планеты.

Например, существует усилитель электрической активности мозга OpenBCI, созданный небольшим стартапом в США, который становится новым Arduino эля энцефалографии, для анализа сигналов мозга, и это тоже очень важно для комьюнити. Есть много других проектов...

Они снижают затраты, потому что нет никакой избыточной защиты от компаний которые всё контролируют и должны поддерживать цены на таком уровне, чтобы сохранить прибыльность.

Эти аппаратные проекты — просто инструменты, которые позволяют делиться знаниями и техническим опытом, используя их на практике в качестве готовых блоков.

L: Спасибо. Думаю, на текущий момент свободные технологии наиболее сильны в области ПО, аппаратного обеспечения и свободных медиа, таких как Википедия, например, или Creative Commons — пожалуй, вместе это три наиболее сильные направления.

P: Я думаю, ты забыл одну важную вещь. Ты упомянул свободные медиа, такие как Creative Commons и Википедия, Но есть ведь еще в сфере науки и материалы с открытым доступом, и открытые лабораторные журналы (open notebook).

Прямо сейчас я работаю над собственным научным проектом, создаю его в режиме открытого журнала, комментирую с помощью свободных инструментов каждый шаг эксперимента. Такой подход позволяет людям воспроизвести эксперимент в аналогичных условиях. И будь у меня оборудование, распространяемое на условиях

Yes, you have to document it. You have to put it to open environment and encourage people to test it, to modify, to understand how it works.

If you are the the owner and the creator, and also the only person who understands how it works, it's not really the open source. Somebody has to look at it, somebody has to analyze it. If that would happened to the FMRI tools that where used around the world for the last 15 years, we could have all the knowledge gathered with this tools intact. But now we have to scrape basically the 15 years of neurology, because the software was faulty. It contains different statistical methods than it was supposed to...

L: And what is the role of community in supporting such projects? For example there is the known problem of open source projects with only one person who really knows their internals. If such person goes away for some reasons and abandons the development process, the project is in real trouble. So what about the problem of the community involvement in the development?

P:I believe that in case of some projects community has to be more involved, test something, develop harder, even if sometimes there is no need in development. For example we have that issue with the set of statistical tests that nobody else is going to implement because they are working, they are fast, they are okay. But we still need someone who actually got stuck in the code. Those who got check it from time to time and notice if anything does go wrong.

Basically university workers can be maintainers in some cases. I think that if there's nobody from the community to volunteer, nobody able to make some package of software – then we should start looking at the university's officials, asking universities to maintain that. Actually start paying the university researchers to maintain some software and respond to all the people who requires something or would like to ask some questions.

I am speaking about employees, who would do it instead of working on some of their own projects. Because I know how this is in computational neuroscience in Warsaw, and in Poland. More than 60% of people aren't contributing to science. They are mostly just read that old papers and do not creating anything but useless pieces of software that only they can use, but no one else. It is waste of money that are going to the universities. If they are to be useful, they should become

свободных лицензий, было бы намного легче воспроизвести сбор данных.

L: Это очень интересная тема, результаты других исследователей вообще бывает не так уж легко воспроизвести. А свободное ПО в научных исследованиях имеет серьёзные преимущества, потому что позволяет посмотреть исходный код и проконтролировать, как на самом деле выполнялись все вычисления.

P: Да, именно, но здесь тоже могут быть проблемы. Не знаю, слышал ли ты, недавно мы потеряли около 40 000 статей по нейрологии и функциональной МРТ, потому что кто-то поместил неверный код в программные инструменты, использовавшиеся для мониторинга.

Сам по себе пакет был свободным ПО, но никто его особенно не тестировал, никто не уделял ему особого внимания. Так что дело не только в том, чтобы сказать «Привет, это открыто», как любят делать в университетах: «Это ПО открытое, но даже не пытайтесь заглянуть в его код, всё равно только мы понимаем, как оно работает».

Да, вам нужно заниматься документированием. Нужно разместить его в открытой среде и вовлекать людей в его тестирование, изменение, чтобы они понимали, как там всё работает.

Если вы владелец и создатель проекта, и при этом единственный, кто понимает, как он работает — это не настоящее свободное ПО. Кто-то должен за ним присматривать, анализировать его. Будь так с теми инструментами по функциональной МРТ, которые использовали во всем мире в течение 15 лет, знания, полученные с помощью этих инструментов, не пострадали бы. А сейчас нам надо вычистить примерно 15 лет неврологии из-за дефектного ПО. Статистические методы там оказались совсем не те, что предполагалось...

L: А какую роль вообще играет комьюнити в поддержке таких проектов? Например, есть известная проблема свободных проектов, внутреннее устройство которых знает один единственный человек. Если такой разработчик по какой-то причине отходит от дел и прекращает работать над проектом, то проект оказывается в большой беде. Так как быть с проблемой вовлечения сообщества в разработку?

P:Я верю, что в некоторых проектах сообщество должно больше участвовать, что-то тестировать и больше вкладываться в разра-

maintainers of the scientific packages, they should start analyzing them, they should start being responsible for how they work.

I believe that would be one of the best uses of university money.

Questions and Russian translation by Dmitriy Kostiuk.

ботку, даже если иногда в разработке нет особой нужды. Например, этот наш случай с набором статистических тестов, которые никто не собирается реализовывать заново, потому что они работают, они быстрые, и с ними всё в порядке. И всё же нужен кто-то, кто реально возится с кодом. Кто-то, кто проверяет его время от времени и заметит, если что-то пойдёт не так.

Вообще-то в некоторых случаях сотрудники университетов могли бы быть мэйнтейнерами. Думаю, если не находится волонтёров от комьюнити, и никто не может подготовить некий программный пакет — тогда есть смысл посмотреть в сторону университетской администрации, обратиться к университетам с просьбой о поддержке таких проектов. Фактически, платить исследователям в университетах за поддержку какого-то ПО, и за общение с людьми, интересующимися каким-то функционалом или имеющими какие-то вопросы.

Я говорю о сотрудниках, которые могли бы заниматься этим вместо работы над некоторыми своими проектами. Потому что я знаю как обстоят дела с вычислительной нейробиологией в Варшаве, и вообще в Польше. Более 60% людей не вносят вклад в науку. Они в основном перечитывают старые статьи и не создают ничего кроме бесполезного ПО, которым не может пользоваться никто кроме них. Это пустая трата поступающих в университеты денег. Чтобы приносить пользу, они могли бы стать мэйнтейнерами каких-то научных пакетов, могли бы начать их анализировать, взять на себя ответственность за то, как эти пакеты работают.

Я верю, что это было бы одним из лучших способов тратить университетские бюджеты.

Вопросы и русский перевод Дмитрия Костюка.

Michael Meeks

Cambridge, UK. General Manager, Collabora Productivity

LVEE: Hi, Michael. Can you briefly introduce yourself?
Michael Meeks: I'm Michael Meeks: Christian, Husband, Hacker — I live near Cambridge in the UK, and these days I work primarily on LibreOffice.
L: Tell us something about your first experience with the open source software.
M.: Heh; so — I played with Linux a fair bit quite early on, when I became a Christian.
L: Your interest to GNU/Linux was driven by a Christian ethics? Can you recall (approximately), which age was it? Some more details?
M.: When I was around eighteen, as a somewhat aggressive and inquisitive, agnostic skeptic, lodging in my gap year with a very loving and patient Christian family. I guess I had enough of my questions, and fears addressed that I was confident that the rest could be. I've never regretted building my life on this rock, though I often fail in so many ways He is so satisfyingly good.

I became rather unwillingly convicted that my stolen copy of Windows, compiler tools etc. were wrong. So I installed this (at the time unutterably terrible) GNU/Linux thing, and got hooked, after replacing my hard-disk (presumed bricked by some kernel bug). Anyhow — in the last 20 years, things have improved — and that choice turned out to launch me on an amazing career working with some extraordinarily sharp and interesting people on GNOME, openSUSE, and LibreOffice.
L: And, which GNU/Linux distro was your first one? This one, unutterably terrible.
M.: I started with some ancient version of Slackware, I really can't remember which version; I recall the upgrade to RedHat Linux 4.2 with some pleasure. Now I use openSUSE exclusively across the family's estate; its a great distribution, quite apart from my familiarity gained working on it. My preferred DE is (sadly) these days XFCE — due to its reliable 2D virtual workspaces — but my less expert family use vanilla GNOME. For development I use a mix of emacs, occasionally vim, Visual Studio, and XCode (depending on platform).

Michael Meeks

Кембридж, Великобритания. General Manager, Collabora Productivity

LVEE: Привет, Майкл. Представься, пожалуйста.
Michael Meeks: Меня зовут Майкл Микс, я христианин, муж, хакер, живу недалеко от Кембриджа в Великобритании, и сейчас в основном работаю над LibreOffice.
L: Расскажи нам, пожалуйста, о своём первом опыте со свободным ПО.
M.: Ну... я начал интересоваться Линуксом достаточно давно, когда я стал христианином.
L: Твой интерес к GNU/Linux был вызван христианской этикой? Помнишь (примерно) в каком возрасте это было? Интересны подробности.
M.: Когда мне было около восемнадцати, я был достаточно агрессивным и пытливым агностиком-скептиком. В то время я снимал комнату в доме любящей и терпеливой христианской семьи. Я нашёл ответы на большинство своих вопросов и опасений, и пришёл к убеждению, что всё это имеет смысл. Я никогда не жалел о том, что связал свою жизнь с христианством, хотя я часто терплю неудачи в вещах, в которых Он настолько хорош.

В какой-то момент я нехотя пришёл к выводу, что пользоваться пиратскими Windows, компилятором и т. д. неправильно. Так что я установил эту штуковину под названием GNU/Linux, невероятно ужасную в то время, и «подсел» — после замены жёсткого диска (который я «убил» каким-то багом в ядре). В любом случае, за последние 20 лет всё сильно улучшилось, и в результате этого решения мне открылись двери к замечательной карьере и возможности работать с невероятно умными и интересно людьми в проектах GNOME, openSUSE и LibreOffice.
L: И какой же был твой первый дистрибутив GNU/Linux? Ну тот, невероятно ужасный.
M.: Я начинал с какой-то старой версии Slackware, не помню уже даже какой; что я помню, так это то, что апгрейд на RedHat Linux 4.2 принёс много радости. Сейчас у меня дома везде openSUSE; кроме того, что я к нему привык, я считаю, что это замечательный дистрибутив. Я предпочитаю XFCE (к сожалению) — в основ-

L: How did you become the developer of FLOSS?

M.: So, after some small hacks I found I was missing a warning for undefined behaviour in gcc; so I grabbed the code, read it, added a warning for this behavior (-Wsequence-point). Unfortunately this lingered for years on the gcc mailing list with no response; so I moved on to GNOME, making mahjongg solvable, then lots of work on Gnumeric around file formats, before LibreOffice, openSUSE and more.

L: Patching a compiler is rather serious start. Our readers would be interested to know, how much development experience you had up to this moment?"

M.: Oh; yes — a particularly serious start when you read the gcc code for a bit: not a good place to learn to program. Having said that — it really is just software I suppose, not some kind of magic — I'm convinced that a reasonable, intelligent person can become the world's expert on any one particular (well scoped) issue in FLOSS simply by focused research and code reading in a reasonable time even without a programming background. Having said that, I had a lot of preparation: writing games in BBC Basic, 6502 assembler then x86 assembler, C, and a year's internship writing Pascal at Quantel for real-time embedded video editing systems (68040 based, compiled on a VAX — using a real VT420 green-screen terminal). That certainly helped a lot.

L: Something about your experience with the community?

M.: My experience of the FLOSS community is of amazing diversity of experience and background, a huge richness of ideas and cultures which is exciting, interesting, and challenging. I love the experience of intellectual freedom — of discussing ideas and perspectives intensely without people taking things personally. I have good friends with whom I disagree really profoundly on all sorts of topics, but really enjoy working alongside in various projects.

L: And what about your impressions from contributing to GNOME and OpenOffice/LibreOffice? How is it to be the part of such a large FLOSS project?

M.: Its great. Most newbies to FLOSS think they want to start their own project, and grow it to being huge and successful — so they can take the credit. Inevitably this dooms them to lonely futility. A FLOSS project has a significant fixed-cost for releasing, infrastructure, website, translation, marketing, brand building etc. Even with the great new services springing up around git — its still not trivial to build your own project — and to scale it. Working with others is really stimulating,

ном, из-за надёжно работающих 2D виртуальных рабочих столов, но члены моей семьи пользуются «ванильным» GNOME. Для разработки я пользуюсь emacs, иногда vim, Visual Studio и XCode — в зависимости от платформы.

L: Как ты стал разработчиком свободного ПО?

M.: Однажды я выяснил, что в gcc не хватает предупреждения о неопределённом поведении. Я скачал код, изучил его, добавил предупреждение (-Wsequence-point). К сожалению, патч слишком долго пробыл без ответа в рассылке gcc, так что я устал ждать и занялся GNOME, починил Mahjongg, много работал над поддержкой файловых форматов в Gnumeric, ну а потом LibreOffice, openSUSE и так далее.

L: Патчить компилятор — это серьёзное начало. Нашим читателям было бы интересно узнать, каков был твой опыт разработки на тот момент?

M.: А, ну да, чтение кода gcc — довольно серьёзное начало, там можно многому научиться. С другой стороны, это просто код, никакой магии. Я уверен, что любой разумный человек может стать экспертом в любой области свободного ПО просто усиленно изучая код какое-то продолжительное время, даже если у него не было серьёзного опыта программирования до этого. У меня, правда, опыта было много: написание игр на Бейсике для микрокомпьютера BBC, потом ассемблер 6502 и x86, Си, год практики в Quantel, программируя системы видеоредактирования в реальном времени на Паскале (процессор был 68040, но компилировали мы на VAXe — с зелёным терминалом VT420!) Всё это, конечно, помогло.

L: А как насчёт опыта взаимодействия с сообществом?

M.: Сообщество удивительно разнообразно опытом и интересами отдельных его членов, люди принадлежат к разным культурам, и у них всех разные мнения и идеи. Это одновременно интересно, захватывающе, но и сложно. Мне нравится этот опыт интеллектуальной свободы: горячо обсуждать идеи и точки зрения, при этом не принимая вещи на свой счёт. У меня есть хорошие друзья, с которыми я совершенно несогласен во многом, но вместе с которыми мне в то же время приятно работать.

L: Что скажешь насчёт опыта работы с проектами GNOME и OpenOffice/LibreOffice? Каково это, быть частью такого огромного проекта?

and a great learning experience. I'd strongly recommend contributing to a large existing project where you can gain deep skills before trying to start anything of your own — as it turns out LibreOffice has a great Easy Hacks list of simple first tasks to get involved with.

L: What do you think about serious changes in the vision of the project, how they occur? Something like transition from GNOME 2.x to 3.x.

M.: Software is a very human thing; it is largely a reflection of the people and personalities that are involved and how well they work together. I think bad things happen to projects when they don't listen carefully particularly to their contributors, but also users. I was involved quite heavily in the development and transition from GNOME 0.x to 1.0, 1.2, 1.4, 2.0 and beyond. Over that time there was lots of change eg. the team who decided that 'sawfish' was written in a hard-to-debug scripting language and needed re-writing in C (as Metacity) to make it maintainable to some heart-ache; then some years later the same team decide it needed re-writing into harder to debug and maintain JavaScript (Mutter). Personally I thought the GNOME 3.0 transition pathologies were refreshingly different to anything I'd seen before; so I disagree with the orthodox group-think on that topic that this is somehow normal. The project had different personalities in charge at this transition, but I'm no expert on GNOME anymore sadly. There are some really great guys who started their life in GNOME who now fight at the cutting edge of other big problems in the FLOSS world.

L: It looks like GNOME 3.0 transition was rather uncomfortable for you. What do you think about MATE as a continuation of GNOME 2.x?

M.: Really uncomfortable. With regard to MATE — I was responsible for trying to redeem some of the worse design decisions in GNOME 2.x around the 'Object Model Environment' piece of GNOME, and I'd be rather happy if that code didn't exist anymore. Each to their own though — if maintaining that floats someone's boat. Then again, if there is lots of spare man-power floating around, then we have tons of really valuable work that can be done on LibreOffice and a great team to work with.

L: Let's turn to LibreOffice then :) What do you think about the attempts to supply LibreOffice with an improved tab-based UI, like 'ribbons' appeared in MS Office after 2007?

M.: Это круто. Большинство новичков в свободном ПО хотят начать свой собственный проект, который бы стал известным и успешным, и прославил бы их, но с таким подходом проект совершенно бесперспективен. В проектах по разработке свободного ПО всё имеет определённую фиксированную цену: инфраструктура, веб-сайт, переводы, маркетинг, создание бренда, релизы... Даже несмотря на то, сколько сейчас возникает классных сервисов вокрут git, создать и поддерживать новый проект не легко. Работа с другими людьми очень полезна, приносит новый опыт и по-настоящему стимулирует. Я рекомендую каждому попробовать быть частью большого имеющегося проекта, в котором можно набраться опыта и чему-то научиться, прежде чем начинать что-то своё. Например, в проекте LibreOffice есть список Easy Hacks, простых задач, на которых можно потренироваться, принося пользу проекту.

L: Что ты думаешь о том, когда проект резко меняет курс, как это получается? Например, такие ситуации, как с переходом от GNOME 2.x к 3.x.

M.: Программное обеспечение пишется людьми и с большего отражает характеры и особенности людей, которые его пишут, а также их взаимоотношения. Как мне кажется, изменения к худшему случаются, когда проекты не слушают не только своих контрибуторов, но и пользователей. Я был серьёзно вовлечён в разработку GNOME версий 0.x, 1.0, 1.2, 1.4, 2.0 и далее. В то время в проекте происходило много радикальных изменений. Например, одна команда решила, что т. к. sawfish написан на скриптовом языке, который нелегко отлаживать, его необходимо переписать на Си (результат известен как Metacity), чтобы его хоть как-то можно было поддерживать. Несколько лет спустя та же самая команда решила, что его нужно переписать на Javascript, который тоже нелегко отлаживать и поддерживать (Mutter). Лично я думаю, что процесс перехода на GNOME 3.0 сильно отличались [в худшую сторону] от чего-либо, что я видел ранее, и я несогласен с распространённым мнением, что это нормально. В GNOME за переход были ответственны разные люди, но, к сожалению, я более не слежу за проектом. Есть несколько классных разработчиков, которые начали свою профессиональную жизнь в GNOME, и теперь они на переднем крае, сражаются с другими важными проблемами мира FLOSS.

M.: Personally I really like a tab-based palette of tools to use, LibreOffice now has a Notebookbar which may be more familiar to some newer users. I can't see us ever removing the classic toolbars though, and of course — we have a great sidebar panel which is a far better use of screen real-estate on 16×9 screens I think. Through good design and re-use, they are reasonably easy to maintain (I hope).

Again — I like the power of possibility — of seeing people who firmly disagree on what is the right way to do toolbars, menus (or whatever) work towards a common goal of thrilling each of their constituents with a choice of the functionality they like best. I really don't buy the rationales for: 'We will choose everything for you', and luckily the LibreOffice UX team is interested both in making thing much easier to use (there is lots of scope for that), and also in allowing choice.

L: Похоже, что переход к GNOME 3.0 для тебя не был очень приятным. А что ты думаешь насчёт MATE как продолжения идей GNOME 2.x?

M.: Да, этот переход был для меня очень неприятным. А насчёт MATE скажу, что именно я был ответственным за попытки исправления некоторых плохих решений в области «объектной модели» GNOME, и я был бы более чем счастлив, если бы этим кодом больше никто не пользовался. Но о вкусах не спорят — если этот проект кому-то нужен, то пусть будет. С другой стороны, если кому-то хочется чем-то заняться, у нас в LibreOffice есть много работы и приятный коллектив.

L: Давай поговорим про LibreOffice :) Что ты думаешь насчёт попыток внедрить в LibreOffice интерфейс на основе «вкладок», наподобие «риббонов» из MS Office 2007?

M.: Мне лично нравятся панели инструментов с вкладками, и в LibreOffice теперь есть есть Notebookbar, к которому, вероятно, привыкли новые пользователи. С другой стороны, я не могу представить, что мы когда-нибудь отойдём от классических панелей инструментов. Ну и потом, у нас есть замечательная боковая панель, которая, на мой взгляд, гораздо эффективнее расходует экранное пространство на экранах с соотношением сторон 16:9. При хорошем дизайне и повторном использовании кода, их по-видимому будет несложно поддерживать (надеюсь).

Ну а вообще, мне нравится сама возможность этого, это замечательно, что есть люди, у которых совершенно разные мнения насчёт того, как правильно делать панели инструментов, меню и так далее, и они все работают вместе, чтобы предложить пользователям выбор. Я не согласен с подходом: «Мы всё решим за вас». К счастью, команда UX LibreOffice заинтересована и в том, чтобы сделать LibreOffice проще в использовании (у нас есть что улучшать!), и в то же время предоставлять возможность выбора.

L: Finally, it would be interesting to know how would you compare your impressions on FLOSS with one on proprietary software. Something about good and bad features you are noticing...

M.: Good and bad features?

I guess the bad feature I see in the world of FLOSS is that proprietary software appears extremely easy to sell; whereas services and support around great FLOSS like LibreOffice is hard to close. I regularly suffer customers who refuse to pay even 10% of the price of MS Office for their deployment in return for great services, support and maintenance — which in turn Collabora can re-invest into LibreOffice development. Apparently it is a common hope that *someone else* can pay to make LibreOffice better, and to contribute about the third of commits we write. While the license of course allows this, that's a really bad feature of my experience at least.

On the good side — I'm a huge fan of LibreOffice Online — which Collabora has invested a fortune into creating. We've put a huge amount of time and effort to allow people to host their own collaborative productivity software in conjunction with our many partners in ways which respect their privacy, as well as letting people exchange open file formats without needing to install software at the recipients' end, that's a great 'good' and it excites me.

Questions on behalf of LVEE where asked by Dmitriy Kostiuk and Andrew Shadura. Russian translation by Andrew Shadura.

L: Наконец, было бы интересно узнать твои впечатления о свободном ПО в сравнении в проприетарным. И хорошие, и плохие особенности, которые ты замечаешь...

M.: Хорошие и плохие особенности?

Наверное, плохо то, что судя по всему, что проприетарный софт гораздо легче продать, чем службы и поддержку для свободного ПО (например LibreOffice). Часто случается, что клиенты отказываются платить даже 10% того, что они платят за MS Office, за поддержку LibreOffice — деньги, которые Collabora может пустить на дальнейшую разработку. Видимо, все надеются, что *кто-то другой* может заплатить за улучшение LibreOffice, и внести около трети тех коммитов, которые пишем мы. Хотя, конечно же, лицензия это позволяет, из моего опыта это достаточно неприятно.

Из хорошего — LibreOffice Online. Collabora вложила огромные средства в его создание. Мы потратили много времени и сил, чтобы позволить пользователям размещать на их собственной инфраструктуре офисное ПО с функцией коллективного доступа, общими усилиями с нашими партнёрами, так, чтобы это не нарушало их конфиденциальности, дали людям возможность обмениваться файлами в открытых форматах без необходимости установки клиентского ПО — и это *хорошее* приводит меня в восторг.

Вопросы от имени LVEE задавали Дмитрий Костюк и Андрей Шадура. Русский перевод Андрея Шадуры.

Florian Müllner

Madrid, Spain. Red Hat

LVEE: Can you briefly introduce yourself?
Florian Müllner: I'm Florian Müllner and I work for Red Hat's desktop team from Madrid, Spain, where I live with my husband and cat. Within GNOME, I work mainly on GNOME Shell and Mutter, which I maintain. I also like to put some time each cycle into the Polari IRC client, which started as a fun side project some years ago.
L: Tell us something about your first experience with the open source software? It may be your first understanding that free/libre software is not about 'pay nothing'
F.: Between graduating from high school and beginning my substitute service, I had some months of idle time. It was way too much time to fill with party alone, and as I was interested in computers, I decided to take a look at this Linux-thing people were talking about. I fell in love almost instantly — unlike the Windows 98 of the time, it didn't hide away what's going on behind pretty (rather pixelated by modern standards) images and cryptic errors, but invited you to just dive right in and explore — source included (on a separate CD)! Oh, and its 'Blue Screen of Death' was a screensaver[1] instead of the real thing...

It didn't take long to come into contact with the philosophical underpinnings — not least to find out why software you paid good money for was still calling itself 'free'. So I discovered copyleft — using the same copyright law that was used elsewhere to restrict users' rights to grant and guarantee rights instead didn't just look like a good thing: it was a quite brilliant legal hack that was easy to sympathize with. As much as I've learnt since then, I still think it is a brilliant hack :-)
L: Which GNU/Linux distro was your first one, if it wasn't already mentioned? Which distro (distros) are you using now?
F.: In case the umlaut in my name hasn't given it away already, I'm originally from Germany. Back in the day, SuSE was the dominating distribution there, so SuSE 6.1[2] was my very first Linux distro — purchased in a bookstore as a set of CD-ROMs.

Over the years I have used a couple of different distributions — Debian, Gentoo (yes, really), Arch... I switched to Fedora when I started working for Red Hat (I didn't have to, but then maintaining

Florian Müllner

Мадрид, Испания. Red Hat

LVEE: Представься, пожалуйста

Florian Müllner: Меня зовут Флориан Мюльнер, я работаю в Red Hat в команде разработчиков окружения рабочего стола. Я живу в Мадриде со своим мужем и котом. В GNOME я в основном работаю над GNOME Shell и Mutter, которые я поддерживаю. Ещё я стараюсь уделять какое-то время работе над IRC-клиентом Polari, который я начал разрабатывать в свободное время несколько лет назад.

L: Расскажи нам, пожалуйста, о своём первом опыте со свободным ПО. Например, как ты понял, что свободное ПО — это не о цене?

F.: После окончания школы у меня было несколько месяцев свободного времени — слишком много, чтобы проводить его в одиночку, и поскольку я интересовался компьютерами, то решил посмотреть на этот Линукс, о котором так много говорили. Я влюбился в Linux почти сразу: в отличие от Windows 98, он ничего не прятал за красивыми (по тем временам) картинками и непонятными сообщениями об ошибках, система как будто бы приглашала тебя с головой окунуться в исследования — ведь исходники были здесь же, на отдельном компакт-диске! Ну и конечно же, синий экран смерти был всего лишь заставкой[1], а не всерьёз.

Вскоре я столкнулся и с философской стороной вопроса, в том числе, пытаясь понять, как это ПО может быть «свободным», если ты за него платишь иногда немалые деньги. Так я узнал о понятии copyleft: использовании законов об авторских правах, которые обычно ограничивают пользователей, для того, чтобы дать им права и гарантии. Этот хитрый «хак» показался мне замечательной идеей. И хотя с тех пор прошло много времени и с тех пор я узнал много нового в этой области, мне всё равно нравится этот «хак» :-)

L: Какой был твой первый дистрибутив? И чем ты пользуешься сейчас?

F.: Если умляут в моём имени ещё меня не выдал, я вообще-то немец. В те времена SuSE был самым распространённым дистри-

Fedora packages is part of the job). To tell the truth, I usually build all of GNOME from upstream and use that as my login session[3], so the underlying distribution doesn't matter too much anyway.

L: How did you become the developer of FLOSS?

F.: I was a Linux user that went to university to study CS, so becoming a FLOSS developer seems obvious. Alas, it wasn't that easy at all. For a very long time, I was working *with* FLOSS rather than working *on* FLOSS — partly because I didn't have a particular itch to scratch, but to be honest, partly because putting your code up for everyone's scrutiny can be scary as well (and the idea that people could blame my sexuality for obvious blunders rather than my inexperience didn't help either). And not least, getting yourself to work after working a day job is hard.

However eventually the GNOME project announced plans for a new major version with a revamped user interface (after the initial 'grown-up' plan of only removing deprecated APIs from GTK+). I starting running early versions of GNOME Shell; it was rough and more of a runnable prototype at that stage, but there was a lot to like as well - or rather, there was a lot I knew I would like once it became more fledged out.

At last, I had found my itch. Very soon after contributing a first patch, I was working on GNOME Shell on a daily basis. I was given git commit access and danced around the house, became a GNOME Foundation member, and attended my first GUADEC. GNOME quickly became a second home to me, and it still is to the current day.

L: Something about your experience with the community (anything from first impressions to your current vision)?

F.: There's a popular children's novel in Germany that includes the character of an 'illusory giant' — a normally sized person who appears taller the farther you get away. The community is a bit like that — big and scary from the outside, it gets more approachable the closer you get, until you attend a conference and meet the actual people who make up the community, and they turn out to be just friendly, regular people...

(Well, definitively nerdier than your average person, and as a whole too white and too male, but some effort is put into increasing the community's diversity[4])

бутивом в Германии, так что мой первый дистрибутив был SuSE 6.1[2]. Я его купил в книжном магазине в виде набора компакт-дисков.

С тех пор я пользовался Debian, Gentoo (да, серьёзно!), Arch... перейдя на Fedora, когда я начал работать в Red Hat. Переходить было необязательно, но так проще было поддерживать пакеты в Fedora, чем я занимался по работе. По правде, я обычно компилирую GNOME из апстримных исходников и сам пользуюсь результатом (опыт с Gentoo даёт о себе знать!)[3], так что на каком дистрибутиве я работаю, не так уж и важно.

L: Как ты стал разработчиком свободного ПО?

F.: Я был пользователем Linux, который учился в университете на компьютерной специальности, так что вполне ожидаемо, что я стал разработчиком. К сожалению, это было не так просто, как может показаться. Довольно долго я работал *со* свободным софтом, а не *над* ним — в частности, потому как у меня не было идей, что бы такое конкретно поразрабатывать, но, честно говоря, ещё мне было немного страшно показывать свой код всем на обозрение (кроме того, не хотелось бы, чтобы люди оправдывали мои промахи моей сексуальностью, а не неопытностью). Ну и наконец, заставить себя заняться программированием вечером после рабочего дня не так легко.

Однажды проект GNOME объявил о планах разработки новой версии с переосмысленным пользовательским интерфейсом (после «подросшего» изначального плана всего лишь удалить устаревшие API из GTK+). Я начал пробовать ранние версии GNOME Shell; в то время это был скорее прототип, но уже на этой стадии было много интересного — точнее, такого, про что я понимал, что оно мне понравится в более «отёсанном» виде.

Наконец, я нашёл то, что меня интересовало. Вскоре после отправки своего первого патча я начал ежедневно работать над GNOME Shell. Я получил доступ в git и танцевал по дому от радости, я стал членом GNOME Foundation, в первый раз поучаствовал в конференции GUADEC. Скоро я стал чувствовать себя в проекте GNOME как дома, и так оно и продолжается по сей день.

L: Расскажи что-нибудь о твоем опыте взаимодействия с сообществом (что угодно, от первых впечатлений до своего нынешнего видения)?

F.: В Германии есть такая известная детская книжка, в которой один из персонажей — «мнимый великан», человек нормального

L: Of course it would be great to know something about your impressions from contributing to GNOME. How is it to be the part of such a large project?

F.: It's smaller on the inside!

Joking aside, GNOME consists of lots of more or less independent modules. At least while starting out, it's common to only focus on a single module — so while you are indeed part of a large project, most of your interactions happen with a fairly small group of people working on the same module (or team — let's not forget that there's more to GNOME than code, like translations, documentation, design or engagement).

There's no denying that it's a big project though, when you travel to a GNOME event knowing that besides meeting with old friends, you'll return with a couple of new ones as well.

L: What do you think about serious changes in the vision of the project, how did they occur? Something on transition from GNOME 2.x to 3.x.

F.: This is a hard question for me, considering that I was a GNOME 2.x user who only became a developer during the 3.x transition itself. While development was still in relatively early stages when I joined, in hindsight the most significant shift had already taken place (at least as far as GNOME Shell was concerned): making design and user experience an integral part of the development process. I'm a big fan of that, but can't compare it to what was there before.

Another important change that took place at around the same time was envisioning the desktop as an integrated product, rather than as a grab bag of interchangeable components — it is worth reminding that 3.0 was the first GNOME version that had built-in printer configuration.

Thanks to those two changes, upstream GNOME now provides a level of polish that was previously only possible on a distribution level (for distros that had the resources to do the integration work, and duplicated for each distro).

L: Something about appearance of GNOME Shell, its evolution through the years? May be some pieces of further vision, etc. :)

F.: It's true that there have been many improvements and refinements over the years, for example:

– the overview originally used tabs to switch between window- and app picker, until 3.6 added the show-apps button to the dash;

роста, который кажется тем выше, чем дальше от него отойдёшь. Сообщество в чём-то похоже: большое и ужасное снаружи, оказывается более доступным вблизи, а после посещения конференции и личного знакомства оказывается, что всё это обычные дружелюбные люди...

(Ну, конечно же, более «ботанистые», чем «обычные» люди, и обычно слишком белые и слишком мужчины, хотя есть усилия по увеличению разнородности сообщества[4])

L: Конечно, интересно было бы узнать про твои впечатления от работы с проектом GNOME. Каково это, быть частью такого огромного проекта?

F.: Изнутри он выглядит меньше!

Если серьёзнее, GNOME состоит из множества относительно независимых модулей. Как минимум, новички обычно фокусируются на одном модуле, так что хотя ты — часть большого проекта, в основном ты общаешься с малой группой людей, которые работают над тем же модулем (ну или с командой — не стоит забывать, что GNOME это больше чем код, это ещё и переводы, документация, дизайн, пользовательская активность).

Не буду отрицать, это большой проект, и, например, когда едешь на встречи разработчиков, то знаешь, что не только увидишься со старыми друзьями, но и приобретешь несколько новых.

L: Что ты думаешь о том, когда проект резко меняет курс, как это получается? Например, как с переходом от GNOME 2.x к 3.x.

F.: Сложно сказать, т. к. во времена GNOME 2.x я был пользователем, а разработчиком стал только во время перехода на 3.x. Хотя разработка и была на сравнительно ранних стадиях, когда я присоединился к проекту, но самые существенное изменения к тому моменту уже произошли (по крайней мере, в GNOME Shell): дизайн и UX стали неотъемлемой частью процесса разработки. Мне это очень нравится, но я не могу сравнить ситуацию с той, что была ранее.

Ещё одно важное изменение, которое произошло тогда же — появилось видение окружения рабочего стола как интегрированного продукта, а не набора взаимозаменяемых компонентов. Стоит напомнить, что 3.0 была первой версией GNOME со встроенными средствами настройки принтера.

in 3.8, traditional menu categories where dropped from the app picker in favor of app folders and the separate 'frequent' and 'all' views we have today; scrolling was replaced by pagination in 3.10, and 3.14 finally updated the various transition animations to give the component its current appearance

- notification handling saw two major redesigns - in 3.6 and 3.16 - each followed by further refinements, the last as recent as the current 3.24 release
- most system status menus where combined into a single system menu in 3.10

However as much as GNOME Shell has changed over time, it's equally impressive to see how much of the initial design document[5] is still relevant today, even despite major changes to the overview layout that happened before 3.0. More so when revisiting the actual release[6], which does show how the project has evolved over time, yet is still easily recognizable as GNOME desktop.

As boring as it sounds, you can expect more of the same in the future — improve and refine what we have, while keeping true to a core vision. And of course is anything in need of refinement itches you: please get in touch and start scratching!

Questions on behalf of LVEE where asked by Anastasia Markina. Russian translation by Andrew Shadura.

Благодаря этим двух изменениям «ванильный» GNOME выглядит настолько отшлифованным, как раньше он мог выглядеть только в дистрибутивах (в тех, которые могли себе позволить немалый объём затрат на интеграцию, и занимались ею самостоятельно, дублируя работу друг друга).

L: Что-нибудь про появление GNOME Shell и то как она развивалась за все эти годы? Может быть, какое-то видение будущего проекта и т. д. :)

F.: Да, за эти годы было много улучшений, например:

- в режиме обзора изначально использовались вкладки для переключения между экраном приложений и экраном окон, но в версии 3.6 в док-панель добавили кнопку показа приложений; в 3.8 привычные категории меню были заменены папками с приложениями и отдельными режимами отображения «популярных» и «всех» приложений; вместо прокрутки в 3.10 был применён постраничный просмотр, а в 3.14 наконец поменялись анимации, после чего всё стало выглядеть так, как сейчас
- интерфейс уведомлений был заметно переделан в 3.6 и 3.16, с небольшими улучшениями в последующих версиях, в том числе и в текущей версии 3.24
- большая часть из всех меню статуса были объединены в единое системное меню в версии 3.10

Несмотря на такое число изменений, поразительно, насколько релевантен изначальный проектный документ[5], даже с учётом серьёзных переделок режима обзора, которые произошли перед выпуском версии 3.0. Если взять саму версию 3.0[6], ещё более заметно, как проект, пройдя такой долгий путь развития, всё ещё выглядит узнаваемым, как GNOME desktop.

Как бы скучно это ни звучало, но план на будущее примерно такой же: улучшать и совершенствовать то, что у нас есть, оставаясь верными основной идее. Ну и само собой, если вам интересно что-то улучшить, выходите на связь и присоединяйтесь!

Вопросы от имени LVEE задавала Анастасия Маркина. Русский перевод Андрея Шадуры.

1 https://www.jwz.org/xscreensaver/screenshots/

2 https://en.opensuse.org/Archive:SuSE_Linux_6.1

3 Did I mention that I was using Gentoo in the past?

4 https://www.gnome.org/outreachy/

5 https://people.gnome.org/~mccann/shell/design/GNOME_Shell-20091114.pdf

6 https://help.gnome.org/misc/release-notes/3.0/

Научное издание

ОТКРЫТЫЕ ТЕХНОЛОГИИ

Сборник материалов тринадцатой международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2017

(г. Гродно, 22–25 июня 2017 г.)

Фото на обложке Олега Зинчука

Ответственный редактор Д.А. Костюк
Компьютерная верстка А.А. Маркина

Подписано в печать 13.06.2017.
Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Ризография. Усл. печ. л. 7,6. Уч.-изд. л. 7,3.
Тираж 180 экз. Заказ 2510.

Издатель и полиграфическое исполнение:
частное производственно-торговое унитарное предприятие
«Издательство “Альтернатива”».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий
№ 1/193 от 19.02.2014.
№ 2/47 от 20.02.2014.
Пр. Машерова, 75/1, к. 312, 224013, Брест.