Открытые технологии

Материалы восьмой международной конференции
Linux Vacation / Eastern Europe 2012

ОТКРЫТЫЕ ТЕХНОЛОГИИ

Сборник материалов Восьмой
Международной конференции
разработчиков и пользователей
свободного программного обеспечения
Linux Vacation / Eastern Europe 2012

(г. Гродно, 07–10 июня 2012 г.)

Брест
«Альтернатива»
2012

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:

С.С. Дереченник, кандидат технических наук, доцент, заведующий кафедрой электронных вычислительных машин и систем Брестского государственного технического университета

А.В. Отвагин, кандидат технических наук, доцент кафедры электронных вычислительных машин Белорусского университета информатики и радиоэлектроники

О-83 **Открытые технологии** : сборник материалов Восьмой Международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2012, Гродно, 07-10 июня 2012 г. / под общей ред. Д.А. Костюка. – Брест : Альтернатива. – 160 с.

ISBN 978-985-521-307-0.

В сборник вошли материалы, представленные авторами на Восьмую Международную конференцию разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2012, включая выездную зимнюю сессию конференции LVEE Winter 2012. Материалы докладов представлены на сайте конференции http://lvee.org и распространяются под лицензией Creative Commons Attribution-ShareAlike 3.0. Статьи посвящены новым технологиям и разработкам в сфере свободного (открытого) программного обеспечения и затрагивают широкий спектр платформ – от рабочих станций и серверов, включая решения на базе виртуализации, до встраиваемых систем и мобильных устройств.

Сборник может быть интересен специалистам в области информационных технологий.

УДК 004.45(082)
ББК 32.973.26-018.2я43

ISBN 978-985-521-307-0

Содержание

Heartbeat: построение отказоустойчивого кластера

Юрий Волкович*

The article presents an approach to build a cluster for web server, as well as author's personal experience and some interesting decisions in this matter.

В рамках данной темы под кластером понимается система высокой доступности, т.е. построенная с расчетом того, что она продолжает работать безотказно, даже если какая-то ее часть выходит из строя.

Heartbeat — продукт проекта Linux-HA [1], позволяющий реализовать механизм безотказной работы отдельных частей кластера.

Задача

Решаемая задача заключалась в построении отказоустойчивого кластера из двух web-серверов, которые могут моментально заменить друг друга в случае выхода из строя одного из них. Сервер предназначен для авторизации на таких сайтах, как irr.ru, job.ru и др., аудитория которых насчитывает несколько миллионов пользователей в сутки, поэтому его значимость для компании достаточно велика.

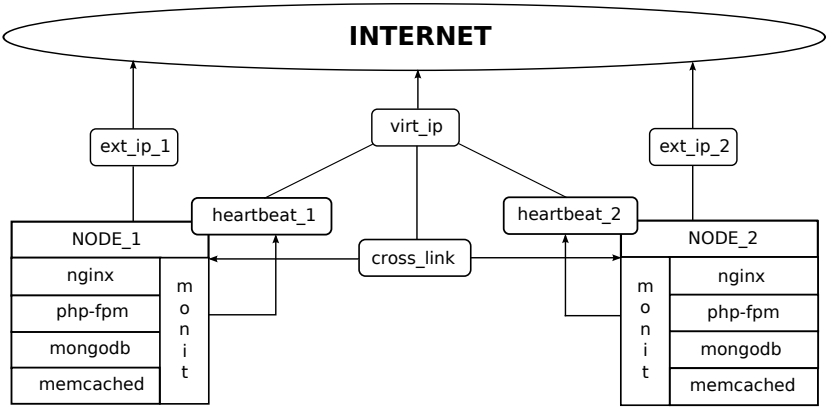
Используемое ПО

Законченное решение было построено на следующем наборе программных продуктов:

- heartbeat (поднимает виртуальный ip и проверяет доступность соседнего сервера);
- monit [2] (следит за правильной работой демонов, обслуживающих сайт);
- web-сервер (связка nginx [3], php-fpm [4], mongodb [5], memcache [6]).

*Минск, Беларусь

Схема взаимодействия демонов



Для начала поднимаются два сервера NODE_1 и NODE_2, на которых запущена работающая версия сайта. Затем на оба сервера подключается heartbeat. Он должен «держать» виртуальный адрес virt_ip и следить за доступностью соседнего сервера, а при сбое на текущем — передавать virt_ip heartbeat'у соседнего сервера. Однако, этого не достаточно для полного отслеживания того, работает ли сайт, поскольку heartbeat не обладает возможностью проверять работоспособность базы данных mongodb либо других используемых сервисов. Эти задачи можно делегировать демону monit, настроенному для проверки того, работают ли все необходимые демоны.

На данном этапе конфигурация, в принципе, является завершенной, и если какой-то из демонов на мастер-сервере «упадет», обслуживание сайта немедленно будет перенаправлено на второй сервер. Но, как показала практика, и этого оказывается недостаточно в случае, когда демон работает, но при этом выдает неверные данные, либо просто зависает, захватив все наличные ресурсы. Для решения данной проблемы нами было написано несколько небольших скриптов, которые запускаются по cron'у и проверяют такие параметры, как репликацию в memcached и mongodb, отдачу http-трафика через php-fpm и др. менее значительные нюансы. И в случае, если проверка не проходит, в скрипте (как и в monit) вы-

полняется команда hb_standby, которая заставляет heartbeat принудительно отдать virt_ip на slave-сервер, если таковой имеется.

Конфигурация демонов

Ниже приведены ключевые параметры, использованные нами при конфигурировании:

heartbeat

- haresources

```
passport1.pronto.ru IPaddr::194.87.222.180/27/eth0
 nginx
```

- ha.cf

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 30
warntime 10
initdead 120
udpport 694
ucast eth1 192.168.0.2
auto_failback off
node passport1.pronto.ru passport2.pronto.ru
ping_group ping_nodes 8.8.8.8
respawn hacluster /usr/lib/heartbeat/ipfail
deadping 30
use_logd yes
debug 1
```

monit

- monitrc

```
check process nginx
with pidfile "/var/run/nginx.pid"
start program = "/etc/init.d/nginx start"
stop program = "/etc/init.d/nginx stop"
group www
```

Тестирование

При проверке работоспособности системы мы использовали следующие способы нарушения работоспособности сайта:

- моментальное отключение питания на мастер-ноде;
- падение интерфейса на мастер-ноде;
- падение любого из демонов, обслуживающего сайт;
- нарушение репликации memcached и mongodb.

Результатом любого из перечисленных событий являлось

- моментальное переключение ip-адреса сайта на соседний сервер;
- отправка администраторам уведомлений обо всех событиях.

Литература

[1] http://www.linux-ha.org
[2] http://mmonit.com/monit
[3] http://sysoev.ru/nginx
[4] http://php-fpm.org
[5] http://www.mongodb.org
[6] http://memcached.org

Kexecboot — Linux как bootloader

Юрий Бушмелев*

The article describes Kexecboot, an implementation of Linux-as-a-Bootloader. It's a C program able to scan the partitions on available devices, offering a graphical framebuffer menu and allowing user to select from which one to boot. Typically kexecboot resides together with kexec in an initramfs, embedded in a custom-tailored kernel compiled with support for initramfs and kexec system call. Both binaries are built static, linked against klibc to optimize size. Kexecboot may be linked against other *libc (glibc, eglibc, uclibc) and may be used as standalone binary as well.

Современные загрузчики операционных систем (grub, u-boot), по сути, сами по себе являются своеобразной операционной системой. В них есть драйвера для дисковых контроллеров, для файловых систем, для устройств ввода/вывода. И все это нужно только для того, чтобы загрузить ядро пользовательской операционной системы и передать ему управление. Причем, если на архитектуре PC есть более-менее стандартные интерфейсы BIOS и EFI, то в сфере embedded hardware ситуация куда более плачевная, что приводит к бесчисленным форкам загрузчиков со всеми вытекающими неприятностями.

Kexecboot — это реализация идеи «Linux как bootloader». Сама идея не нова, она появилась в виде ответа разработчиков на вышеописанную ситуацию. Предполагается иметь простой минимальный загрузчик, который умеет только загружать ядро Linux из определенного фиксированного места и передавать ему управление. Дальнейшие действия по инициализации оборудования и работе с периферией выполняет уже само ядро. Для, собственно, загрузки с других носителей применяется системный вызов kexec, который загружает следующее ядро в память и передает ему управление.

Изначально kexecboot был разработан для КПК Sharp Zaurus. Проблема с данными КПК была в том, что под ядро ОС в штатной схеме разметки NAND был выделен раздел размером всего 1.2Mb.

*Ульяновск, РФ

Прошло несколько лет и ядро Linux со всеми желаемыми функциями перестало входить в этот лимит. Нужно было либо менять загрузчик, либо придумывать пути обхода. Но с другой стороны, в старых моделях Sharp Zaurus объем NAND был всего 16Mb, что накладывало ограничения и на содержимое «прошивки». Требовался загрузчик, который способен запустить систему с SD или CF-карты. Причем, желательно, с возможностью выбора, откуда и что загружать. Так появился kexecboot.

Сам по себе, kexecboot — это программа, которая сканирует устройства и отображает в виде меню список ядер, которые могут быть загружены. Чтобы элемент появился в списке, должны быть выполнены следующие условия:

- на устройстве находится файловая система, которую kexecboot может идентифицировать, а ядро может смонтировать;
- на файловой системе есть файл конфигурации /boot/boot.cfg, который удалось распарсить, либо на файловой системе есть один из файлов /zImage, /boot/zImage (или uImage).

Также в общем меню присутствует системное меню, где можно запустить повторное сканирование устройств, посмотреть некоторую отладочную информацию, а также перезагрузить или выключить устройство. Меню может быть графическим (framebuffer) или текстовым. На данный момент, выбор может быть произведен только аппаратными клавишами устройства.

После выбора элемента в меню, kexecboot формирует строку запуска для утилиты kexec, чтобы сначала загрузить ядро (kexec -l), а потом передать ему управление (kexec -e).

На параметры загружаемого ядра можно влиять при помощи файла конфигурации. В нем можно задать название и иконку, которые будут отображены в меню, а также путь к ядру и параметры его командной строки.

Kexecboot можно использовать как отдельную программу, так и вместо init в составе initramfs. Во втором случае ядро компонуется вместе с образом initramfs, где находятся необходимые каталоги и два исполнимых файла — kexecboot и kexec. На данный момент нам удается держать размер образа ядра с initramfs в пределах 1Mb. Частично это удается за счет более тонкой настройки конфигурации ядра, частично–за счет использования lzma для сжатия образа.

К сожалению, узким местом при использовании kexecboot остается неработоспособность kexec на многих аппаратных платформах. Тем не менее, на данный момент известно о фактах успешного

использования kexecboot на всей линейке КПК Sharp Zaurus, на некоторых моделях КПК iPAQ, на нетбуках Toshiba AC100 и некоторых планшетах Archos. Пакеты для произвольных архитектур можно подготовить с помощью систем сборки OpenEmbedded или OpenWRT.

В будущем планируется добавление альтернативных методов загрузки — switch_root, losetup+switch_root и nfs+switch_root. Это позволит использовать kexecboot даже на системах, где kexec (пока) не работает.

Подробности о проекте можно узнать на сайте `http://kexecboot.org`

Параўнальны аналіз выкарыстання свабоднага праграмнага забеспячэння ў ВНУ Беларусі, Расійскай Федэрацыі і Украіны

Я. Аляксееў*, Г. Злобін†, Дз. Касцюк‡

The article presents a comparative analysis of FOSS usage in higher educational institutions of Belarus, Russia and Ukraine on the basis of the FOSS Lviv-2011 and FOSS Lviv-2012 conferences materials. Authors review workstations' system software as well as auxiliary software used by students and software studied in the academic process.

Стварэнне ў 1981 г. фірмай IBM персанальнай ЭВМ IBM PC з адкрытай архітэктурай справакавала з'яўленне IBM-сумяшчальных ПЭВМ, якія выраблялiся ў многіх краінах свету. Не адсталі ад гэтых краін СССР і краіны Савета эканамічнай узаемадапамогі, якія сталі выпрабляць цэлы спектр такіх ПЭВМ:

– ЕС 1840, ЕС 1841, Іскра 1030, Нейроны (СССР);
– ЕС 1834, ЕС 1835 (ГДР);
– ЕС 1839 (НРБ).

Для ПЭВМ савецкай вытворчасці была створана рускамоўная аперацыйная сістэма АльфаДОС, тэкставы рэдактар Лексікон, тэкставы рэдактар Text tip (Балгарыя), тэкставы працэсар Нейроны-тэкст, таблічны працэсар Нейроны-рахунак, СКБД Нейроны-база. Цяжка сказаць, наколькі ліцэнзійна-чыстымі былі АльфаДОС, Нейрон-тэкст, Нейрон-рахунак, Нейрон-база, бо дзякуючы «жалезнай заслоне» прымяніць да СССР санкцыі з нагоды парушэнняў аўтарскіх правоў уласнікаў праграм было няпроста. Неўзабаве пасля развалу СССР дзякуючы падзенню «жалезнай заслоны» ў многіх краінах СНД пачалася зборка IBM-сумяшчальных ЭВМ з камплектуючых, увезеных у асноўным з краін паўднёва-ўсходняй Азіі. На гэтыя ПЭВМ усталёўваліся пераважна пірацкія версіі як сістэмнага, так і прыкладнога праграмнага забеспячэння. Відавочна, што

*Данецк, Украіна
†Львоў, Украіна
‡Брэст, Беларусь

каштавалі гэтыя ПЭВМ значна танней аналагічных ПЭВМ еўрапейскай і амерыканскай вытворчасці, не кажучы ўжо пра ПЭВМ фірмы Apple. З-за гэтага аперацыйная сістэма MS DOS і офісны пакет Microsoft Office сталі дэ-факта стандартам у ВНУ краін СНД. Ці садзейнічала распаўсюджванню пірацкага ПЗ у ВНУ СНД адсутнасць заканадаўства аб абароне аўтарскіх правоў уласнікаў праграм зараз сказаць цяжка, але разам з тым амаль 10 гадоў мы без абмежаванняў капіявалі і ўсталёўвалі пірацкія копіі прапрыетарнага ПЗ.

У Беларусі, Расійскай Федэрацыі і Украіне законы аб абароне аўтарскіх правоў уласнікаў праграм уведзены з 1996 г. (Беларусь), з 2001 г. (Украіна). У Расійскай Федэрацыі з 1993 г. дзейнічаў закон аб аўтарскім праве і сумежных правах, які страціў моц з 1 студзеня 2008 года ў сувязі з уступленнем у сілу чацвёртай часткі Грамадзянскага кодэкса РФ. Зрэшты гэта мала паўплывала на сітуацыю з пірацкім ПЗ у ВНУ гэтых краін. Выпадкі пераследу ВНУ за парушэнні аўтарскіх правоў у вобласці праграмнага забеспячэння былі малалікімі і не заўсёды яны праводзіліся з мэтай абароны аўтарскіх правоў уласнікаў праграм. Аднак ужыванне законаў аб абароне аўтарскіх правоў уласнікаў праграм да суб'ектаў гаспадарчай дзейнасці стала ствараць ціск на ВНУ: «Вучыце сваіх выпускнікоў таму, з чым яны будуць працаваць на нашых працоўных месцах». Шмат фірмаў спачатку пераходзіць на СПЗ з мэтай памяншэння сумы ліцэнзійных адлічэнняў уласнікам прапрыетарнага ПЗ.

Яшчэ адным аргументам на карысць змены сітуацыі з выкарыстаннем ВПЗ у ВНУ Беларусі, Расійскай Федэрацыі і Украіны стал пачатак эры мабільных працоўных месцаў — цяжка прадбачыць, якая АС і якое прыкладное ПЗ будзе ўстаноўлена на нэтбуке, планшэце ці смартфоне супрацоўніка фірмы. З'яўленне мабільных працоўных месцаў і хуткая змена версій сістэмнага і прыкладнога ПЗ змушае ВНУ да адмовы ад тэхналагічнай накіраванасці лекцыйных курсаў, звязаных з кампутарным тэхналогіямі, на карысць фундаментальнага складніка. А гэта прыводзіць да з'яўлення меркаванняў тыпу «Калі мы павінны навучыць студэнтаў асновам працы з графічным інтэрфейсам ў любой АС, то чаму гэта павінна быць дарагая Microsoft Windows? Мэтазгодней рабіць гэта ў свабоднай і бясплатнай GNU/Linux?» Разам з тым, адмова ад напрацовак метадычнага забеспячэння для выкладчыкаў ВНУ з'яўляецца даволі няпростым працэсам, асабліва ва ўмовах безадказнасці за выкарыстанне пірацкага ПЗ. За час ад падпісання Белавежскага

пагаднення аб спыненні існавання СССР Беларусь, Расійская Федэрацыя і Украіна прайшлі кожная свой шлях развіцця і было б цікава параўнаць стан з выкарыстаннем ВПЗ у ВНУ гэтых краін.

Выкарыстанне ВПЗ ў ВНУ Беларусі

Сёння рынак працы Беларусі патрабуе вывучэння многіх прапрыетарных праграмных прадуктаў, пачынаючы ад платформы Microsoft Windows і скончваючы спецыялізаванымі CAD/CAM-сістэмамі. Гэтая тэндэнцыя суправаджаецца слабай матывацыяй выкарыстання СПЗ, якая абумоўлена тым, што рызыка парушэння ліцэнзіі як і раней застаецца ў вобласці здагадак без рэальных дзеянняў беларускіх кантралюючых органаў. Доля легальна набытага ПЗ узрасла ў апошнія гады за кошт спецыяльных скідак ад пастаўшчыкоў, а таксама праз высокія эканамічныя паказчыкі да 2011 г. Тым не менш, эканамічны фактар не з'яўляецца вырашальным для прыняцця выбару паміж свабодным ПЗ і прапрыетарным. Таму выкарыстанне СПЗ у ВНУ звычайна абумоўлена яго тэхнічнымі перавагамі ў параўнанні з прапрыетарнымі аналагамі або патрабаваннямі рынку працы. Выбар ПЗ сервера можа быць адзіным выключэннем, паколькі ён істотна залежыць ад асабістых густаў сістэмных адміністратараў.

У апошнія гады назіраецца рост цікаўнасці карпаратыўных працадаўцаў да GNU/Linux, пераважна для ўбудавальных і серверных сістэм. У сувязі з гэтым буйныя гульцы рынку актыўна заяўляюць аб сваіх уласных трэнінгах і семінарах, прысвечаных гэтай тэме.

Выкарыстанне СПЗ у ВНУ Беларусі можна падзяліць на тры напрамкі:

1. **ПЗ падтрымкі навучальнага працэсу (пераважна сістэмнае ПЗ на серверах і працоўных станцыях).** У большасці выпадкаў сістэмнае СПЗ на працоўных станцыях прадстаўлена GNU/Linux у рэжыме мультызагрузкі ў якасці альтэрнатыўнай АС у кампутарных класах кафедраў, якія навучаюць праграмаванню студэнтаў інжынерных спецыялізацый. У педагагічных ВНУ GNU/Linux на настольных кампутарах выкарыстоўваецца рэдка ў сувязі з недастатковай распаўсюджанасцю GNU/Linux у школах Беларусі. Разам з тым у некаторых універсітэтах заўважана выкарыстанне Linux у тонкіх кліентах з тэрмінальным Windows-серверам (напрыклад, Гродзенскі дзяржуніверсітэт імя Янкі Купалы);

2. **дадатковае ПЗ, якое выкарыстоўваецца студэнтамі ў самастойнай працы.** Да гэтай групы ПЗ можна залічыць офісны пакет OpenOffice.org і браўзэр Firefox;
3. **ПЗ для выкарыстання ў навучальных курсах.** У гэтым кірунку СПЗ пераважна выкарыстоўваюць у інжынерных ВНУ, асабліва тых, якія вядуць навучанне IT-спецыялістаў, а менавіта СПЗ для навучання праграмавання на мовах Асэмблер, C++, Java і PHP, SciLab для выканання матэматычных разлікаў, QCAD/LibreCAD, Blender, cirquit CAD для вывучэння сістэм аўтаматызаванага праектавання, выкарыстанне свабодных сістэм віртуалізацыі VirtualBox і KVM для вывучэння аперацыйных сістэм, выкарыстанне Moodle і iTest для тэставай праверкі ведаў студэнтаў.

Асобна трэба падкрэсліць выкарыстанне СПЗ для кластараў і нацыянальнай ГРІД-сістэмы Беларусі, у якую ўключаны рэсурсы вядучых універсітэтаў (Беларускі дзяржуніверсітэт, Гродзенскі дзяржуніверсітэт імя Янкі Купалы, Беларускі дзяржуніверсітэт інфарматыкі і радыёэлектронікі, Беларускі нацыянальны тэхнічны універсітэт), навуковых устаноў і прадпрыемстваў краіны ў рамках сумеснай расійска-беларускай праграмы СКІФ-ГРІД.

Выкарыстанне СПЗ у ВНУ Беларусі можна праілюстраваць наступным малюнкам

Іл. 3.1. Выкарыстанне СПЗ у ВНУ Беларусі

Выкарыстанне СПЗ ў ВНУ Расійскай Федэрацыі

У адрозненні ад Беларусі ў Расійскай Федэрацыі ў 2008 г. была прынята канцэпцыя развіцця распрацоўкі і выкарыстання свабоднага праграмнага забеспячэння. У рамках гэтай канцэпцыі ў 2008-2010 гг. была рэалізавана праграма выкарыстання СПЗ у школах Расійскай Федэрацыі (у 35% школ СПЗ устаноўлена больш чым на 50% кампутараў). Трэба адзначыць, што ў адрозненне ад Беларусі і Украіны ў Расійскай Федэрацыі назіраецца некаторая актыўнасць кантралюючых органаў з нагоды ліцэнзійнага праграмнага забеспячэння. Самай рэзанансная справай была справа А.М. Понасава, якая і прывяла да стварэння ў 2008 г. грамадскай арганізацыі «Цэнтр вольных тэхналогій». Як вынікае з [1, 2, 3], у большасці ВНУ Расійскай Федэрацыі назіраецца выкарыстанне як Microsft Windows, так і GNU/Linux. Толькі ў некаторых ВНУ кіраўніцтвам прынята валявое рашэнне аб поўным пераходзе на СПЗ (Санкт-Пецярбургскі гандлёва-эканамічны універсітэт, Томскі дзяржаўны педагагічны універсітэт, Ніжагародскі радыётэхнічны каледж). Як і ў Беларусі выкарыстанне СПЗ у ВНУ Расійскай Федэрацыі можна падзяліць на тры напрамкі [3, 4, 5]:

1. **праграмнае забеспячэнне падтрымкі навучальнага працэсу** (у асноўным сістэмнае праграмнае забеспячэнне на серверах і працоўных станцыях). У большасці выпадкаў сістэмнае СПЗ на працоўных станцыях прадстаўлена GNU/Linux у рэжыме мультызагрузкі як альтэрнатыўная АС у кампутарных класах кафедраў;
2. **дадатковае праграмнае забеспячэнне, якое выкарыстоўваецца студэнтамі пры самастойнай працы** (на жаль аўтары не валодаюць дадзенымі аб гэтай групе СПЗ);
3. **праграмнае забеспячэнне для выкарыстання ў навучальных курсах.** У гэтым кірунку спектр СПЗ значна шырэй, чым у Беларусі. Тут можна згадаць выкарыстанне СПЗ для вывучэння праграмавання на мовах C/C++ (у Яраслаўскім Універсітэце ёсць цікавы вопыт навучання праграмаванню на аснове СПЗ), Pascal (Free Pascal, Lazarus), Java, Haskel, Prolog; SciLab, Octave, Sage для выканання матэматычных разлікаў (шырокі вопыт выкарыстання вольнага матэматычнага праграмнага забеспячэння назапашаны ва універсітэтах Новасібірска, Барнаула, Бійска); арганізацыя сістэм дыстанцыйнага навучання, выкарыстанне свабодных сістэм вірту-

алізацыі для вывучэння аперацыйных сістэм; інструментар для філалагічнага аналізу тэкстаў; выкарыстанне інструментара верыфікацыі ПЗ у падрыхтоўцы магістраў; стварэнне электронных адукацыйных рэсурсаў падтрымкі навучальнага працэсу для завочнай формы навучання (аўтары разумеюць, што рэальны спіс выкарыстанага СПЗ значна шырэй, але ў адкрытым доступе дадзеных пакуль што няма).

У ВНУ Расійскай Федэрацыі даволі актыўна выкарыстоўваюцца вылічальныя кластары з выкарыстаннем СПЗ. Па ініцыятыве рэктараў МДУ ім. М.В. Ламаносава, Ніжагародскага універсітэта ім. М.І. Лабачэўскага, Томскага дзяржаўнага універсітэта, Паўднёва-Уральскага дзяржаўнага універсітэта створаны «Суперкампутарны кансорцыум універсітэтаў Расіі». У спіс TOP500 ад снежня 2011 г. уваходзяць чатыры расейскія суперкампутара (№18, 107, 119, 121).

Варта адзначыць, што ў Расійскай Федэрацыі назапашаны значны вопыт распрацоўкі свабоднага праграмнага забеспячэння. У Расеі распрацоўваюцца дыстрыбутывы GNU/Linux: ALT Linux (`http://altlinux.ru`), Calculate Linux (`http://www.calculate-linux.ru`), ROSA (`http://rosalab.ru`). Наяўнасць кампаній, якія займаюцца распрацоўкай СПЗ, дазваляе распрацоўваць спецыялізаваныя свабодныя праграмы і значна спрашчае рэалізацыю праектаў па ўкараненні GNU/Linux у школы і універсітэты.

Іл. 3.2. Выкарыстанне СПЗ у ВНУ Расійскай Федэрацыі

Выкарыстанне СПЗ у ВНУ Украіны

Ва Украіне «Дзяржаўная мэтавая навукова-тэхнічная праграма выкарыстання ў органах улады праграмнага забеспячэння з адкрытым кодам» была зацверджана ў 2010 г., але да рэальнага яе выканання пакуль што не дайшлі. У адрозненне ад Расійскай Федэрацыі праверкі адпаведнымі дзяржаўнымі органамі выпадкаў парушэння аўтарскіх правоў уласнікаў праграм праводзяцца ў значна меншым аб'ёме і пераважна ў гасразліковых арганізацыях. Вядомыя толькі асобныя выпадкі такіх праверак у ВНУ. Як і ў Беларусі пакуль што адсутнічае эканамічны складнік у выбары праграмнага забеспячэння для ВНУ. Пасля набыцця ПЭВМ пераважна з ліцэнзійнымі Microsoft Windows і Microsoft Office устанаўліваецца вялікая колькасць неліцэнзійнага ПЗ, чым зводзяцца дарэмна фантастычна вялікія выдаткі сродкаў на першаснае набыццё ПА (Львоўскі нацыянальны універсітэт імя Івана Франка да эканамічнага крызісу 2008 г. кожны год купляў прыблізна 1000 ПЭВМ. Сумарній кошт ліцэнзій толькі на Microsoft Windows (ОЭМ-версія) і Microsoft Office складал амаль 300000 у.а. на год–даволі вялікая сума, як для ЛНУ ім.і.Франка!). У большасці выпадкаў выбар менавіта прапрыетарнага ПЗ звязаны нават не са спажывецкімі якасцямі гэтых праграм, а фактам павярхоўнага знаёмства выкладчыка з гэтай праграмай ці нават наяўнасцю ў яго якой-небудзь кніжкі з апісаннем праграмы. Разам з тым, выступленні выкладчыкаў і навуковых супрацоўнікаў на першай і другой канферэнцыі FOSS Lviv [1, 2] сведчаць аб шырокім спектры выкарыстання СПЗ у ВНУ Украіны.

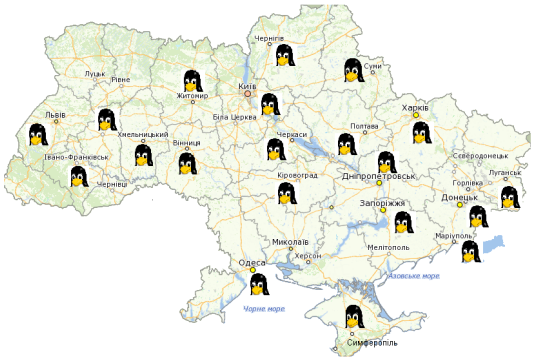
Як і ў Беларусі і Расійскай Федэрацыі выкарыстанне СПЗ у ВНУ Украіны можна падзяліць на тры напрамкі [1, 2]:

1. **праграмнае забеспячэнне падтрымкі навучальнага працэсу (у асноўным сістэмнае праграмнае забеспячэнне на серверах і працоўных станцыях).** У большасці выпадкаў сістэмнае СПЗ на працоўных станцыях прадстаўлена GNU/Linux у рэжыме мультызагрузкі, як альтэрнатыўная АС у кампутарных класах кафедраў;
2. **дадатковае праграмнае забеспячэнне, якое выкарыстоўваецца студэнтамі падчас іх самастойнай працы** (на жаль, аўтары на сённяшні дзень не маюць дадзеных аб гэтай групе СПЗ);
3. **праграмнае забеспячэнне для выкарыстання ў навучальных курсах.** У гэтым кірунку спектр СПЗ значна шы-

рэй, чым у Беларусі. Гэта выкарыстанне сістэм кампутарнай матэматыкі; арганізацыя сістэм дыстанцыйнага навучання, выкарыстанне свабодных сістэм віртуалізацыі для вывучэння аперацыйных сістэм; выкарыстанне СПЗ для тэставання апаратнага забеспячэння ПЭВМ; выкарыстанне офіснага пакета OpenOffice.org.ukr у курсе інфарматыкі ВНУ; выкарыстанне адкрытых сродкаў праграмавання ў навучанні і ў навуковых даследаваннях; (аўтары аддаюць сабе справаздачу ў тым, што рэальны спіс прымяняемага СПЗ значна шырэй, але больш поўныя дадзеныя аб гэтым аўтарам невядомыя).

У ВНУ Украіны эксплуатуюцца вылічальныя кластары с выкарыстаннем СПЗ, нараўне са спецыялізаванымі ўстаноўкамі шырока выкарыстоўваюцца размеркаваныя кластарныя сістэмы і сістэмы з выкананнем вылічэнняў на графічных працэсарах.

У выніку можна канстатаваць як шырокі спектр выкарыстання СПЗ ва ўкраінскіх ВНУ ад дыстанцыйнага навучання да распрацоўкі СПЗ, так і шырокую геаграфію выкарыстання СПЗ ад Луганска на ўсходзе да Львова на захадзе і ад Чарнігава на поўначы да Адэсы на поўдні.



Іл. 3.3. Выкарыстанне СПЗ у ВНУ Украіны

Падводзячы вынікі, можна канстатаваць, што:

1. незалежна ад наяўнасці або адсутнасці канцэпцыі выкарыстання СПЗ, яно выкарыстоўваецца ў ВНУ Беларусі, Расійскай Федэрацыі і Украіны;
2. колькасна аб'ём выкарыстання СПЗ у адукацыі вышэй у Расійскай Федэрацыі;

3. ва ўсіх трох краінах міністэрства адукацыі займаюць адхіленую пазіцыю ў працэсе ўкаранення СПЗ ва універсітэты і школы;
4. ва ўсіх трох краінах узровень выкарыстання СПЗ у ВНУ з'яўляецца недастатковым. Свабода выбару ПЗ, якой карыстаюцца выкладчыкі ВНУ, у большасці выпадкаў не прыводзіць да выбару найлепшага інструментара, а абумоўлена звычкамі альбо стэрэатыпамі (часта памылковымі) выкладчыкаў.

Літаратура

[1] Тези доповідей міжнародної науково-практичної конференції FOSS Lviv-2011. — Львів.: Львівський національний університет імені Івана Франка, 2011. — 196 с.

[2] Друга міжнародна науково-практична конференція FOSS Lviv-2012: Збірник наукових праць/ Львів, 26-28 квітня 2012 р. — 160 с.

[3] Алексеев Е.Р., Брагилевский В.Н. Использование свободного програмного обеспечения в университетах и исследовательских учреждений Российской Федерации. Друга міжнародна науково-практична конференція FOSS Lviv-2012: Збірник наукових праць/ Львів, 26-28 квітня 2012 р.

[4] В.Н. Брагилевский, С.А. Гуда, Г.В. Худолей СПО на мехмате Южного федерального университета. Седьмая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов/ Переславль, 28-29 января 2012 года. М.: Альт Линукс, 2012. 110 с.: ил.

[5] `lists.raspo.ru/Plone/publichnye-drafty-dokumentov/dokumenty-komiteta-po-obrazovaniyu-i-vysshei-shkole/spo-v-rossiiskih-vuzah`

[6] Derechennik S.S., Kostiuk D.A., Pynkin D.A. Free/libre software usage in the belarusian system of higher educational institutions // Друга міжнародна науково-практична конференція FOSS Lviv-2012: Збірник наукових праць/ Львів, 26-28 квітня 2012 р.

Разработка и перспективы развития дистрибутива со вспомогательными технологиями ALT Linux Homeros

Михаил Пожидаев*

ALT Linux Homeros is a GNU/Linux distribution for blind and visually impaired users. It provides speech access based on GNU Emacs environment with emacspeak add-on. The article covers various implementation details as well as propositions for further development.

Дистрибутив ALT Linux Homeros предназначен для использования людьми с различными проблемами зрения, которые не могут обычным способом воспринимать визуальную информацию с экрана компьютера. Работа пользователя осуществляется путём представления вывода программ в текстовом виде при помощи речевого синтезатора, что делает возможным решение многих распространённых задач, хотя и с достаточно серьёзными ограничениями.

В основе дистрибутива лежит популярный текстовый редактор GNU Emacs, оснащённый специальным дополнением emacspeak. Организация рабочего пространства в GNU Emacs предполагает представление всех рабочих объектов в текстовом виде, что значительно упрощает процесс формирования их речевого представления. Последовательность речевых команд, подготовленная emacspeak, передаётся в речевой сервер VoiceMan, разработанный целиком в рамках проекта ALT Linux Homeros. Речевой сервер выполняет автоматическое переключение синтезаторов в зависимости от языка фрагмента текста, а также предотвращает наложение воспроизведения речи от нескольких запущенных экземпляров GNU Emacs. Установка системы на жёсткий диск производится при помощи специального скрипта клонирования дистрибутива с LiveCD. Процесс установки проходит автоматически, все основные параметры пользователь указывает в команде запуска.

В таком виде дистрибутив может оказаться надёжным инструментом при решении многих задач. Он способен быть средой для

*Томский государственный университет, ALT Linux, Томск, РФ

администрирования и восстановления системы после сбоя, поскольку предоставляет возможность запуска речевого интерфейса без установки на жёсткий диск. Среда GNU Emacs загружается в консольном режиме, т. е. исключаются какие-либо проблемы с наличием подходящего видеодрайвера. Она относительно экономна с точки зрения использования системных ресурсов, что позволяет её применять на мобильных устройствах, таких как, например, нетбуки. Круг доступных задач определяется дополнениями, интегрированными непосредственно в GNU Emacs, и набором консольных утилит, вызываемых из командной строки. При этом существует возможность выполнения без визуального контроля некоторых сложных работ, таких как набор больших текстов (статей, дипломных работ, диссертаций, книг) и даже подготовка нотных материалов, при помощи инструментов, предполагающих ввод данных в виде текстовых файлов с использованием некоторой разметки. В случае редактирования текста это позволяет делать latex, а для работы с музыкальными партитурами — lilypond.

Решение этих задач является хорошим результатом разработки, но этого недостаточно для того, чтобы система стала удобным массовым инструментом для повседневной работы. Традиции UNIX, предполагающие хранение конфигурационной информации в текстовых файлах и необходимость написания собственных скриптов на языке Lisp, который широко используется в работе с GNU Emacs, делают систему сложной для новичков. Речевой вывод GNU Emacs производится почти полностью на английском языке, воспринимать который на слух многим пользователям тяжело.

Ситуация усложняется тем, что ряд приложений может запускаться только в графическом режиме, и ориентировка на консольное применение GNU Emacs становится неудобной. Примером такого приложения является веб-браузер. Решить проблему качественной русификации GNU Emacs пока невозможно, поскольку все строковые константы вписаны жёстко в исходный код. Для решения других трудностей в рамках проекта проводятся дополнительные исследования, и подготавливаются новые компоненты. Качественным шагом вперёд может быть широкое задействование сервисов, допускающих взаимодействие через D-Bus. GNU Emacs предоставляет собственные инструменты для работы с объектами D-Bus. Это позволяет управлять важными системными настройками, как, например, конфигурация сети с использованием Network Manager, без редактирования конфигурационных файлов. Получившее распро-

странение в последнее время движение широкого задействования systemd и слияние systemd с udev могут оказаться крайне полезными в повышении удобства работы с дистрибутивом. Помимо этого, речевой сервер VoiceMan в настоящий момент проходит фазу трансформации, после которой он будет способен обрабатывать команды по шине D-Bus, что может повысить гибкость создания новых речевых оповещений.

Рассматривается вопрос запуска GNU Emacs в сессии x.org после установки, но с сохранением консольного режима в LiveCD. Это позволит запускать браузер Mozilla Firefox, который имеет собственные механизмы передачи информации для вспомогательных технологий. Для их задействования необходимо наличие двух компонентов: AT-SPI и экранного чтеца orca. AT-SPI не имеет зависимости от больших сторонних библиотек, но экранный чтец orca является приложением, требующим запуска настольного окружения GNOME, что является достаточно тяжёлым решением. AT-SPI используется в качестве платформы работы со вспомогательными технологиями в мобильной ОС Tizen, при этом, возможно, экранный чтец orca будет заменён на некоторое новое, более лёгкое решение, которое окажется подходящим недостающим звеном для среды ALT Linux Homeros.

Работа по поддержке вспомогательных технологий ведётся в некоторых настольных оболочках, таких как GNOME, KDE и XFCE. Планируется поддержка необходимых для их запуска компонентов в репозиториях ALT Linux, но их применение затруднено на мобильных устройствах, поэтому даже при их наличии существующий подход ALT Linux Homeros сохраняет свою актуальность. Среда на основе GNU Emacs рассматривается как основной инструмент, поскольку предлагает значительно более высокую скорость работы и позволяет решать более сложные задачи.

systemd journal or computer readable logs

Максим Мельников*

Main goal of logs is changing now. Admins don't read logs anymore, but write programs to read logs instead. This is why logs should be readable by computer in first turn, and only in second turn by human. The systemd journal is nice example of how it could be implemented. The article discusses some ideas behind systemd journal and gives examples of its usage.

Проблемы современных логов

Классические системы логирования (syslog, etc) разрабатывались с целью предоставления информации для отладки администратором. В те времена на одного администратора приходилось небольшое количество систем, и он ещё был в состоянии просматривать логи самостоятельно.

Теперь, когда на одного администоратора приходятся десятки, а иногда и сотни/тысячи систем, просматривать логи глазами больше невозможно. Администраторы прибегают к автоматическим системам мониторинга, таким как zabbix и др, а для анализа логов используют средства энтерпрайз-уровня (такие, как syslog-ng), которые занимаются разбором логов, анализом и т.д. Однако классические логи плохо подходят для автоматической обработки, сложны для парсинга, их формат постоянно меняется, а сообщения могут быть локализованны.

Авторы systemd journal предлагают для решения данной проблемы разделить сообщение на 2 части: понятную человеку и удобную для автоматической обработки. Кроме того предлагается дополнить сообщение важной информацией, которая раньше не всегда сохранялась. Предлагается вариант реализации, учитывающий удобство пользования как разработчиками ПО, так и администраторами.

*Минск, Беларусь, World of Tanks Team

Использование библиотеки libsystemd-journal

Пример отправки сообщения в лог:

```
#include <stdlib.h>
#include <systemd/sd-journal.h>
int main() {
    int i=0, j=0;
    while(i<10 && j<10) {
        random()&1 ? ++i : ++j;
        sd_journal_send(
            "MESSAGE=(%d,%d)", i, j,
            "MESSAGE_ID=51141ddad48f4924aef970b1eab2af42",
            "I=%d", i,
            "J=%d", j,
            NULL
        );
    }
    return 0;
}
```

В данном примере значения переменных i и j увеличиваются случайным образом. После каждой итерации значения I и J сохраняются в лог с помощью journal-протокола, функцией sd_journal_send библиотеки libsystemd-journal из пакета systemd.

Сборка может осуществляться через:

```
gcc -O2 -Wall -std=gnu99 -lsystemd-journal example.c
 -o example
```

При сохранении информации используются следующие идентификаторы:

1. MESSAGE — текстовое сообщение, для людей. Записи в логе будут выглядить как пары значений i и j в скобках: (3,4)
2. MESSAGE_ID, I, J — для автоматической обработки. Под MESSAGE_ID обычно подразумевается UUID типа сообщения. Это позволяет легко отделить одни сообщения от других, даже если формат/набор данных совпадает. I и J — это уже бизнес-параметры, которые мы сохраняем как словарь «ключ-значение».

Таким образом мы можем как угодно изменять форматирование сообщения пользователю без изменения данных для автоматической обработки.

Разбор логируемых данных

Для получения данных из лога можно использовать утилиту systemd-journalctl из пакета systemd, а можно написать свою (используя библиотеку libsystemd-journal). Следует учитывать, что данные хранятся в бинарном виде, а формат и подход к хранению логов можно сравнить с некоторыми реляционными и нереляционными базами данных. Формат хранения может изменяться, но разработчки гарантируют возможнлость доступа к данным через библиотеку/утилиты systemd. Выбор бинарного хранилища позволяет хранить информацию более компактно, а также осуществлять быстрый и удобный поиск по любым параметрам сообщения.

systemd-journalctl поддерживает вывод информации в различных форматах: short, verbose, export (сообщение целиком), json (сообщение целиком в json-формате), cat (только «пользовательское» сообщение без дополнительных параметров). Утилита может показывать определённое количество сообщений, а также может быть запущенна в режиме «follow».

В systemd-journal, если вы используете systemd, сохраняется так же все kernel/stdout/stderr/syslog-сообщения сервисов системы и ядра.

Рассмотрим одно сообщение из примера:

```
# systemd-journalctl -o json I=1 -n 1
[
{
   ".cursor" : "...",
   ".realtime" : 1337625703076274,
   ".monotonic" : 95084661530,
   ".boot_id" : "97af7b9ccc9341a78bff939661d1e53b",
   "MESSAGE_ID" : "51141ddad48f4924aef970b1eab2af42",
   "MESSAGE" : "(1,1)",
   "I" : "1",
   "J" : "1",
   "_TRANSPORT" : "journal",
   "_PID" : "6928",
```

```
    "_UID" : "1001",
    "_GID" : "1001",
    "_COMM" : "ex1",
    "_EXE" : "/home/max_posedon/systemd-journald/example",
    "_CMDLINE" : "./ex1 param1",
    "_SYSTEMD_CGROUP" : "/system/kdm@.service/tty7",
    "_SYSTEMD_UNIT" : "kdm@tty7.service",
    "_SOURCE_REALTIME_TIMESTAMP" : "1337625703074362",
    "_BOOT_ID" : "97af7b9ccc9341a78bff939661d1e53b",
    "_MACHINE_ID" : "be1598bd68baa268cb48e33d0000000f",
    "_HOSTNAME" : "m_melnikau-vaio"
}
]
```

Здесь выполнен запрос одного сообщения (`-n 1`) в формате json и фильтрация сообщения по критерию `I=1`. Более подробную информацию можно получить, воспользовавшись командой man systemd-journalctl.

Рассмотрим подробнее сообщение в логе. В нем доступны следующие системные параметры:

1. PID, UID, GID — важные параметры, которые далеко не всегда логируются
2. COMM, EXE, CMDLINE — описание того, какое приложение и как именно запускалась
3. SOURCE_REALTIME_TIMESTAMP — время согласно приложению

Среди параметров, которые сохраняет сам демон journald — следующие:

1. realtime — время, когда сообщение попало в journald (немного позже, чем отметка времени, сделанная приложением)
2. monotonic — системные часы иногда переводятся, и сохраненное значение монотонного таймера может оказаться важным для будущей отладки

И, наконец, бизнес-параметры:

1. MESSAGE — сообщение, понятное пользователю
2. MESSAGE_ID, I, J — то же самое сообщение, но уже в виде, удобном для автоматической обработки
3. по параметрам MESSAGE_ID, I, J, а также по всем остальным удобно искать, фильтроват и агрегировать информацию

В последующих версиях systemd планируется расширить этот списко за счет следующих параметров:

1. файл, строка, функция исходного файла, где вызывалась функция логирования
2. audit и selinux контексты

Таким образом, объём сохраняемых данных многократно возрастает по сравнению с «классическим» подходом к логированию, но это не вызывает значительного роста занимаемого места благодаря бинарному формату и реляционному подходу к хранению данных. Семантический подход к хранению сообщений позволяет легко агрегировать, обрабатывать и находить необходимую информацию.

Заключение

Не факт, что systemd journald станет стандартом для логирования в мире Linux. Однако он демонстрирует принципиально новый подход, решая современные проблемы автоматической обработки логов.

Применение свободного ПО для исследования поведения нелинейных динамических систем

Александра Кононова, Алексей Городилов*

The article discusses a possibility of using free software tools for research of non-linear dynamic systems, arising problems and their resolutions.

В настоящее время существует множество свободных систем, которые могут использоваться при исследовании нелинейных динамических систем. Для выполнения математических вычислений наиболее известны: пакет GNU Octave, созданный как свободная альтернатива известной системы MATLAB, так что язык Octave в основном совместим с MATLAB; пакет Scilab, имеющий схожий, но несовместимый с MATLAB язык программирования; существуют узкоспециализированные программы, такие, например, как Xppaut; а также множество интересных, но плохо документированных или малоизвестных пока пакетов, таких как Euler, model-builder, freemat.

Динамика нелинейной системы чаще всего описывается в виде системы обыкновенных дифференциальных уравнений (ОДУ):

$$\begin{cases} \dot{x}_1 &= F_1(x_1, x_2, \dots x_n) \\ \dot{x}_2 &= F_2(x_1, x_2, \dots x_n) \\ \dots & \\ \dot{x}_n &= F_n(x_1, x_2, \dots x_n) \end{cases}$$

GNU Octave

В Octave существует множество функций для решения систем обыкновенных дифференциальных уравнений [1]. Две из них встроенные — это lsode для решения системы ОДУ и lsode_options для

*Москва, Зеленоград, РФ

задания опций работы lsode. Здесь, как и в MATLAB, существует семейство функций ode* для решения систем ОДУ различными методами. В Debian они доступны после установки пакета octave-odepkg. Наиболее часто используемые среди них — функции ode23 и ode45 (методы Рунге-Кутты). Для решения жёстких ОДУ, в отличие от MATLAB, используются функции ode5r и ode2r, а не семейство ode*s.

Для визуализации фазовой траектории на плоскости используется функция plot, в трёхмерном случае — plot3. Для того, чтобы получившийся график можно было вращать мышкой, перед запуском Octave необходимо выполнить в используемой оболочке команду `GNUTERM=wxt`

Рассмотрим пример — исследование модели развития вуза. Результатом является траектория системы — график её развития. Для расчёта поведения динамической модели развития вуза [2] были заданы уравнения:

```
function xdot=F(t,x);
    Ue = 0.62;
    tR = .1;
    AR = 0.5;
    tS = 1;
    tU = .2;
    AU = 1;
    AS = 2 / (Ue*AR);

xdot=[

(  -x(1) + AR*x(2)  ) / tR;

(  -x(2) + AS*x(1)*x(3)  ) / tS;

(  Ue--x(3)--AU*x(1)*x(2)  ) / tU

];

endfunction;
```

расчёт траектории при интересующих нас начальных условиях (точка $(0.5, 0.1, 0.1)$) на отрезке времени от 0 до 12:

```
[t,x] = ode45(@F,[0 12],[0.5 0.1 0.1]);
```

визуализация:

```
plot3(x(:,1), x(:,2), x(:,3) );
```

На экране появится трёхмерный график — траектория развития системы.

Scilab

В Scilab для решения системы ОДУ предусмотрена функция ode. Метод решения задаётся её первым необязательным параметром [3].

Для визуализации рассчитанной фазовой траектории используются функции plot/plot2d для двумерного случая и param3d/ param3d1 для трёхмерного.

Для расчёта вышеупомянутой динамической модели вуза задаются уравнения (совпадение с GNU Octave вследствие отсутствия матричных операций):

```
function xdot=F(t,x);
    Ue = 0.62;
    tR = .1;
    AR = 0.5;
    tS = 1;
    tU = .2;
    AU = 1;
    AS = 2 / (Ue*AR);

xdot=[

(  -x(1) + AR*x(2)  ) / tR;

(  -x(2) + AS*x(1)*x(3)  ) / tS;

(  Ue--x(3)--AU*x(1)*x(2)  ) / tU

];

endfunction;
```

Для расчёта траектории необходимо: задать вектор-столбец $\mathbf{x_0}$ начальных условий, начальный момент времени t_0 и вектор момен-

тов времени $\mathbf{t}$, в которых требуется рассчитать траекторию (может быть и столбцом, и строкой):

```
x0 = [0.5; 0.1; 0.1];
t0 = 0;
t=t0:0.1:12;
```

рассчитать траекторию функцией ode: `x=ode(x0, t0, t, F);` количество строк вектора $\mathbf{x}$ совпадает с количеством строк $\mathbf{x_0}$, а количество столбцов — с длиной $\mathbf{t}$, т. е., для построения траектории $\mathbf{x}$ делится на строки: `param3d(x(:,1), x(:,2), x(:,3) );`

На экране появится рассчитанная траектория.

Особенности экспорта в текстовый документ

Возможности Scilab и GNU Octave в построении графиков весьма обширны, но не всегда достаточны. Поэтому часто возникает необходимость сохранить траекторию в текстовый файл, который затем может быть вставлен в рисунок TikZ/PGF командой: `\draw plot[smooth] file {plots/sample.trj};`

В GNU Octave сохранить матрицу координат в файл можно командой save: `save sample.trj x;` в Scilab — функцией write: `write('sample.trj',x);`

Xppaut

Xppaut является специализированной программой для исследования нелинейных динамических систем. Она распространяется по лицензии GNU GPL и доступна в репозиториях Debian, начиная с Debian Wheezy (testing). Предназначена для численного исследования различных видов разностных и дифференциальных уравнений. При старте программы необходимо задать исследуемую систему [4]. Это делается с помощью текстового файла, формат которого описан в документации программы по адресу `http://www.math.pitt.edu/bard/xpp/help/xppodes.html`.

Рассмотрим для примера классическую модель Лотки — Вольтерра. Она представляется файлом LotkaVolterra.ode:

```
@# LotkaVolterra.ode@
x'= a*x - b*x*y
y'= d*x*y - g*y
```

```
par a=1,b=2,d=3.7,g=1
init x=1,y=0.1
@ total=200
@ xp=x,yp=y,xlo=-2,xhi=5,ylo=-4,yhi=5
done
```

Кроме вида уравнений, необходимо задать значения параметров, начальные условия и параметры отображения — в данном случае указано, что по оси абсцисс указывается переменная x, по оси ординат — y; заданы границы для каждой из осей. Команда @xppaut LotkaVolterra.ode@ позволит исследовать поведение системы с заданными параметрами. При запуске xppaut без указания системы будет открыт диалог выбора файла.

На основании личного опыта работы в области исследования нелинейных динамических систем как с проприетарным, так и свободным ПО, можно сделать вывод, что возможности СПО в данной области нисколько не уступают возможностям проприетарных программ.

Литература

[1] Алексеев Е. Р., Чеснокова О. В. GNU Octave для студентов и преподавателей. Донецк.: ДонНТУ, Технопарк ДонНТУ УНИТЕХ, 2011. 332 с.

[2] Акулёнок М. В., Кононова А. И., Трояновский В. М. Исследование динамики сложной организационной структуры на примере вуза // Изв. вузов. Электроника (ВАК). 2011. № 1(87). С. 70–77.

[3] Алексеев Е. Р., Чеснокова О. В., Рудченко Е. А. Scilab. Решение инженерных и математических задач М.: ALT Linux; БИНОМ. Лаборатория знаний, 2008. 269 с.

[4] Xppaut online documentation `http://www.math.pitt.edu/bard/xpp/help/xpphelp.html`

Подходы к статическому анализу открытого исходного кода

Зубов М. В., Старцев Е. В., Пустыгин А. Н.*

This article presents contemporary approaches of using program representations for static analysis. This analysis can help to understand and extend open-source software. The most common approaches were implemented in prototypes with open technologies. Also, class-level detailed representation was developed and implemented in prototype to prove the assumption of more effective analysis on less detailed representation.

Статический анализ — анализ кода, проводимый без его исполнения. Это достаточно мощный инструмент, который можно использовать во многих целях: от обнаружения ошибок до получения информации об использованных алгоритмах. Для выполнения статического анализа обычно формируют промежуточные представления исходного кода.

Форма промежуточного представления может определяться целью анализа. В свободном проекте BLAST [1] исходный код программы представляется виде автомата потока управления (CFA), описывающего набор состояний процедур и переходы между ними. Алгоритм анализа называется «абстракция и уточнение по контрпримерам»: создаются абстрактные деревья достижимости (ART), которые описывают возможные пути распространения управления на уровне общности, которая используется на очередной итерации алгоритма.

Наиболее распространенное промежуточное предсталение — абстрактное дерево синтаксического разбора Abstract Syntax Tree (AST) [2]. Известные инструменты генерации парсеров, такие как ANTLR [3] (используется в Checkstyle [4] для языка Java) или специфичные инструменты, например, API компилятора Compass/ROSE (ROSE) [5], поддерживающего языки C, C++ и Fortran, могут быть использованы для генерации AST. Этот подход является простым и проверенным временем. Однако, имеют место и другие подходы.

*ЧелГУ, Челябинск, РФ

Например, в проекте Pylint [6] для языка Python может быть использован отдельный свободно-распространяемый модуль logilab-astng [7]. Авторами были разработаны 2 прототипа генератора промежуточных представлений, основанных на AST: первый, для языка Python — на основе свободного модуля logilab-astng [7], второй, для Java — на основе компилятора javac, входящего в состав открытой реализации средств разработки на Java — OpenJDK [8].

Другой вариант использования промежуточного представления — реляционное представление исходного кода анализируемого ПО. По грамматике исходного языка подготавливается схема реляционной базы данных, далее созданная по этой схеме база данных заполняется в соответствии с конструкциями исходного кода. Анализ сводится к выполнению произвольных запросов к этой базе. В инструментах, использующих такой тип представления, применяются различные языки запросов, как самостоятельные, например, QUEL [9] (в OMEGA [10] для Ada, C и Pascal), так и специально созданные, например, .QL [11] (SemmleCode [12] для Java). В настоящее время ведется создание прототипа построителя реляционного промежуточного представления для языка Java. Применение такого представления будет очень эффективно на больших объемах данных. Идея представления в виде реляционной БД дополняется использованием логических языков в качестве языков запросов (Prolog в ASTLOG[13], Datalog в CodeQuest[14]), более подходящих для анализа структур исходного кода, например, транзитивных замыканий. Запросы к базе знаний транслируются в SQL и выполняются на конкретной базе данных, либо же база данных может быть представлена в виде базы знаний. На практике используется только первый способ.

Из идей реляционного представления, а именно проведения анализа в виде выполнения запросов к исходному коду, выделилось отдельное направление. Оно стоит обособленно, так как не диктует использование конкретного представления. Оно может быть как реляционным в системе CodeQuest [14], так и AST в ASTLOG [13]. Некоторые заявляют о потенциально любом представлении (PQL [15]). В основном, в концепции запросов к исходному коду используют логические языки как наиболее удобные и нацеленные на извлечение знаний. Известны также анализаторы, выполняющие запросы на естественном языке (английском) [16].

Промежуточные представления можно классифицировать по уровню детализации. Так в академическом проекте Bauhaus project

[17] для языков Ada, C, C++, C# и Java имеются 2 представления — низкоуровневое InterMediate Language, описывающее конструкции языка на синтаксическом и семантическом уровнях, и Resource flow graphs, описывающее архитектурные особенности программной системы. В анализаторе, использующем PQL [15], выделяется 3 уровня анализа: модель глобального программного дизайна, модель структуры программы, модель детального программного дизайна. Для создания прототипа такого представления был выбран средний уровень детализации — уровень структуры и связи классов.

Очевидно, что на разных уровнях детализации могут быть эффективны различные представления. Так, например, граф потока управления будет иметь не слишком сложную структуру, но очень большой объем, поэтому для него логично бы было использовать реляционное представление. При работе с любым представлением исходного текста на любом уровне необходимо сохранять возможность соотнести анализируемый объект с местом в исходном коде, в котором он описан. Это достигается сохранением координат объектов в представлениях, как и было реализовано в представлении уровня классов.

Если промежуточное представление может описывать сразу несколько входных языков, то оно называется универсальным или мультиязычным. Оно позволяет производить типовые процедуры анализа для различных языков, упрощая задачу построения анализаторов. Но из его достоинства следует и недостаток, учитывать все детали конкретного языка оказывается невозможным. В качестве прототипа универсального представления было реализовано представление уровня структуры классов для языков Python и Java на основе прототипов генерации частных представлений уровня AST.

За счет универсальности категорий ООП, существующих во всех объектных языках [18], полученное представление является полным для своего уровня анализа и, в то же время, универсальным для объектно-ориентированных языков. Полученное высокоуровневое представление позволяет производить анализ на уровне классов и архитектуры проекта, что позволяет снизить накладные расходы на выполнение анализа по сравнению с более детализированным представлением.

Независимо от форм представлений можно выделить модель анализа с помощью моделей отражения [19]. Она основана на сравнении высокоуровневой модели программы, которую программист мыслит себе в процессе разработки, и фактической, полученной в

результате. Этот подход не требует конкретного представления кода, и иногда может быть выполнен с использованием текстовой обработки. Так, в частности, он был реализован в одном из первых опытов применения подхода для изучения подсистемы виртуальной памяти NetBSD [19]. На основе этого подхода с помощью собственного генератора промежуточного представления для Python, использующего logilab-astng [7], была построена и проанализирована высокоуровневая модель нескольких проектов с открытым исходным кодом на Python.

Литература

[1] BLAST / `mtc.epfl.ch/software-tools/blast/index-epfl.php`

[2] Ахо А.В., Ульман Д.Д. / Компиляторы. Принципы, технологии и инструментарии. / М.: Вильямс, 2008г.

[3] ANTLR / `www.antlr.org`

[4] Checkstyle / `checkstyle.sourceforge.net`

[5] ROSE / `rosecompiler.org`

[6] Pylint / `www.logilab.org/857`

[7] logilab-astng / `www.logilab.org/856`

[8] OpenJDK / `openjdk.java.net`

[9] Ingres 10.0 QUEL Reference guide / Ingres corp. 2010, code.ingres.com/ingres/main/src/tools/techpub/pdf/QUELRef.pdf

[10] M. Linton / Queries and Views of Programs Using a Relational Database System / www.eecs.berkeley.edu/Pubs/TechRpts/1983/5296.html

[11] O. de Moor, E. Hajiyev, M. Verbaere / Object-oriented queries over software systems / ACM Press, 2007.

[12] SemmleCode / semmle.com/solutions/enabling-tools-for-your-projects

[13] R. Crew / ASTLOG: A Language for Examining Abstract Syntax Trees / Microsoft Research, 1997.

[14] E. Hajiyev / CodeQuest — Source Code Querying with Datalog / St. Anne's College, Oxford University, 2005.

[15] S. Jarzabek / Design of Flexible Static Program Analyzers with PQL / IEEE Transactions on software engineering, vol. 24, no. 3, march 1998.

[16] M. Kinmming, M. Monperrus, M. Mezini / Quering Source Code with Natural Language / 26th IEEE/ACM International Conference On Automated Software Engineering (ASE'2011).
[17] Bauhaus project / `www.bauhaus-stuttgart.de/bauhaus/index-english.html`
[18] И. Грэхем / Объектно-ориентированные методы. Принципы и практика / М.: Вильямс, 2004 г.
[19] G.C. Murphy, D. Notkin, K. Sullivan / Software Reflexion models: Bridging the gap between source and high-level models / Third ACM SIGSOFT Symposium on the Foundations of Software Engineering, pages 18–28, New York, ACM Press, 1995.

Проект пиринговой сети, учитывающей особенности сети и требования приложений

Алексей Городилов, Александра Кононова*

Modern network applications are usually oblivious to underlying network equipment and protocols. Some applications like VoIP or streaming video can detect connection speed and latency, but can't share these measurements with other network nodes. This information could be useful to determine optimal paths and connection speeds. Some principles of sharing this information by nodes and ways of using it for more efficient data transmission are given. They can be used to implement free applications and protocols. Advantage of connection information sharing is shown using free ns3 network simulator. The article proposes a project of such an optimised network.

Современные сетевые приложения обычно создаются без учёта свойств сети и оборудования, на котором они работают. Современные протоколы сетевого уровня изолируют приложения от параметров оборудования, что позволяет упростить разработку приложений. Однако различные каналы связи имеют различные значения пропускной способности, латентности и других параметров, а также допускают различные флуктуации этих параметров. Некоторые типы приложений, например приложения для передачи потокового видео или ip-телефонии, измеряют скорость передачи и латентность в процессе обмена данными, и подстраивают производимый ими поток данных под них, но эти параметры почти никогда не передаются другим узлам в сети и не используются повторно.

Сетевые протоколы прикладного уровня являются наиболее быстро эволюционирующей частью сетевых протоколов. Благодаря отсутствию необходимости поддерживать совместимость с протоколами более верхнего уровня отсутствует необходимость стандартизации, что позволяет протоколам и их реализациям развиваться и внедряться в условиях свободной конкуренции. Среди недавно появившихся и получивших широкое распространение можно

*Москва, г. Зеленоград, РФ

выделить такие протоколы, как eD2k, XMPP, а также Kademlia, bitTorrent, Skype.

Преимущество протоколов высокого уровня заключается в том, что они позволяют обойти ограничения низкоуровневых протоколов. Эти ограничения не могут быть устранены в самих протоколах низкого уровня в связи с большой сложностью замены протоколов низких уровней. В настоящее время разрабатываются новые сетевые протоколы прикладного уровня для преодоления еще существующих ограничений протоколов нижних уровней. В их числе Jingle, P-IMAP.

Одним из существенных ограничений современного стека протоколов TCP/IP как версии 4, так и версии 6, является недоступность данных о пропускной способности сети, латентности и других характеристиках сети для приложений. Для решения этой проблемы проводились исследования, направленные на замену или усовершенствование частей стека TCP/IP. Были разработаны и в настоящий момент разрабатываются экспериментальные системы передачи и протоколы, такие как: CANS, Transformer tunnels, NINJA, и другие. Но в них проблема решается ниже прикладного уровня, поэтому их внедрение возможно только в течение нескольких лет и в настоящий момент не рассматривается стандартизирующими организациями.

В настоящий момент не создано, и не находится в стадии реализации протокола, который бы позволил реализовать передачу данных с учетом особенностей сети и потребностей приложений, и при этом был бы легким для внедрения протоколом прикладного уровня. Поэтому разработка высокоуровнего протокола, учитывающего особенности сети и потребности приложений, является весьма актуальной проблемой.

В процессе использования сети для различных целей даже с верхних уровней модели OSI возможно получить информацию о характеристиках каналов связи, например измеряя объем переданных данных за определенное время, время ответа удаленного узла и многие другие параметры. Обмен данными о характеристиках и структуре сети между узлами позволяет каждому узлу использовать эту информацию для построения оптимальных путей передачи с учетом требований приложения и структуры сети.

После включения в сеть и обмена ограниченным объемом служебных данных подключившийся узел имеет сведения о других связанных с ним узлах. Объём этих данных, и соответственно коли-

чество известных узлов, ограничено, оно должно быть достаточно для построения эффективных сетевых путей. Это количество выбрано на основании анализа работы современных p2p-сетей. Подмножество узлов, данные о которых поддерживаются актуальными у одного узла, выбирается исходя из данных о характеристиках каналов связи между узлами, измеренными и полученными из сети. При входе и выходе других узлов из сети и изменении нагрузки возможно изменить множество связанных узлов для более эффективного построения путей. Так как моделью сети является взвешенный граф, для построения пути могут быть использованы алгоритмы поиска кратчайшего пути на графе. При этом каждый узел выполняет только часть алгоритма, для которой у него достаточно данных, что позволяет распределить между узлами вычислительную нагрузку и нагрузку по хранению данных.

Для решения такой задачи традиционно применяется алгоритм Дейкстры. Наличие у узлов сведений о связанных с ними других узлах позволяет применить его распределенную реализацию. Недостатком этой реализации будет необходимость передавать большое количество служебного трафика. Для уменьшения объема передаваемых данных и увеличения быстродействия предложено вместо алгоритма Дейкстры использовать алгоритм A*, достигающий более высокой производительности (по времени) за счет эвристики.

В качестве эвристической функции алгоритма A* используется следующая функция, которая зависит от пропускной способности и загруженности каналов связи узла:

$$h(x) = \alpha_{\text{вых}} \cdot C_{\text{вых}}(x) + \alpha_{\text{вх}} \cdot C_{\text{вх}}(x) + \beta_{\text{вых}} \cdot S_{\text{вых}}(x) + \beta_{\text{вх}} \cdot S_{\text{вх}}(x);$$

где x — текущий узел; $C_{\text{вых}}(x)$ — пропускная способность исходящего канала связи узла x; $C_{\text{вх}}(x)$ — пропускная способность входящего канала связи узла x; $S_{\text{вых}}(x)$ — загруженность исходящего канала связи узла x; $S_{\text{вх}}(x)$ — загруженность входящего канала связи связи узла x.

Был проведен эксперимент с использованием симулятора ns3, из результатов которого можно сделать вывод, что надежность файлообменной сети может быть повышена более чем в два раза за счет учета характеристик каналов связи и требований приложений. Это возможно потому, что файлообменная сеть состоит из множества независимых узлов, особенности взаимодействия которых современные системы практически неспособны учитывать.

Вольныя графічныя праграмы для апрацоўкі выяваў з падвышанай глыбінёй колеру

Антон Літвіненка*

Deep color images (with more than 8 bits per channel) are supposed to be extremely useful in image processing (especially, photographic) due to better quality preservation, but are totally unsupported by GIMP. Author gives a brief overview of typical deep color processing taks and with their solutions using ufraw, ImageMagick, nip2 and Luminance HDR programs.

Фактычным стандартам у вобласці сучаснай кампутарнай графікі з'яўляецца выкарыстанне лічбавых выяваў «True Color»-якасці, якая прадугледжвае выкарыстанне 8 бітаў дадзеных для перадачы значэння кожнага канала для кожнага піксела выявы. Такая колькасць інфармацыі дазваляе перадаць 256 градацый значэння кожнага канала і, ў выпадку каляровае трохканальнае выявы — больш за 16 мільёнаў адценняў, што цалкам дастаткова для стварэння ў чалавечаскага вока ўражання натуральнага колеру. Разам з тым, пад час апрацоўкі выяваў, асабліва фатаздымкаў, інфармацыі ў «True Color» здымку можа апынуцца замала для атрымання настолькі ж натуральнага выніку апрацоўкі (напрыклад, калі зыходная выява фактычна выкарыстоўвае толькі частку магчымага дыяпазону колераў (надта цемная, светлая ці проста малакантрастная)). У такіх выпадках выкарыстоўваюцца разнастайныя стандарты з падвышанай «бітнасцю» (глыбінёй колеру), якія змяшчаюць большую колькасць градацый адценняў. Гэтыя стандарты (якія аб'ядноўваюцца пад агульнай назвай «Deep Color») можна падзяліць на наступныя групы:

1. сямейства фарматаў RAW (12, 14, 22 біта на канал);
2. 16-bit (48-bit) TIFF (16 бітаў на канал);
3. сямейства фарматаў HDR (8, 16, 32 біта на канал).

Агульнавядомай адносна ўніверсальнай вольнай праграмай для інтэрактыўнай апрацоўкі выяваў з'яўляецца GIMP. У той жа час,

*Кіеў, Украіна

праз прынцыповыя абмежаванні ягонай існай архітэктуры падтрымкі deep color у ім няма і бліжэйшым часам не будзе. Праз тое даводзіцца карыстацца альбо праграмамі, адмыслова прызначанымі для пэўных аспектаў deep color апрацоўкі (ufraw, Luminance HDR), альбо прыстасоўваць праграмы, прызначаныя, ў прынцыпе, для іншых мэтаў (ImageMagick, nip2).

Апрацоўка RAW

RAW — нестандартызаванае сямейства фарматаў, прызначаных для захоўвання выяваў з лічбавых фотакамер з мінімальнай апрацоўкай працэсарам камеры. Дазваляе захаваць максімальную колькасць першаснай інфармацыі для ёйнай апрацоўкі па-за межамі сэансу фатаграфавання. Праз вялікую колькасць варыянтаў фармату, незваротнасць апрацоўкі (у RAW не захоўваюць пасля апрацоўкі), а таксама праз спецыфічныя аперацыі (бераўскі фільтр, накладанне балансу белага і г.д.) апрацоўка RAW выконваецца адмысловымі праграмамі альбо адмысловымі плагінамі. FLOSS-прыкладам з'яўляецца **ufraw**, які існуе як у выглядзе плагіна для GIMP, так і ў выглядзе асобнае праграмы (можа ствараць 48-бітныя TIFF-файлы для дадатковай апрацоўкі, калі гэта патрэбна). Праграма цалкам рэалізуе «праяўленне» RAW-выявы разам з шэрагам дадатковых спецыфічных для фотаздымкаў апрацовак (карэкцыя неідэальнасці лінзаў, застасаванне «чорнага кадру» (карэкцыя бітых пікселаў), вэйвлетнае падаўленне шумоў і г.д.).

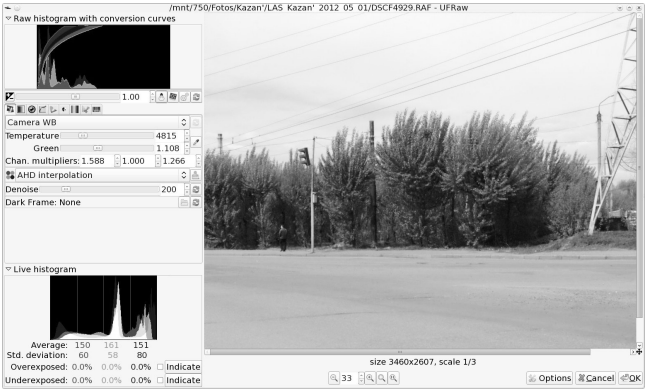
Зручна паядноўваць выкарыстанне ufraw з выкарыстаннем праграмаў, здольных да хуткага папярэдняга прагляду RAW.

Іншыя варыянты: **darktable**, **rawstudio**.

Апрацоўка HDR

HDR-выявы таксама з'ўляюцца спецыфічным выпадкам deep color выяваў, галоўная мэта якога — перадаць вялікі дынамічны дыяпазон адлюстраванае сцэны. Стварэнне такіх выяваў ускладненае абмежаванасцю:

1. Фатаапаратаў (замалы дынамічны дыяпазон сенсара);
2. Фарматаў захоўвання (могуць змяшчаць замалы дынамічны дыяпазон);
3. Тэхнікі для адлюстравання (дысплей, фотапапера).



Іл. 9.1. RAW-файл у вакне праграмы ufraw

Рашэнне — застасаванне спецыфічных спосабаў на ўсіх этапах:

1. Стварэнне: аб'яднанне некалькіх LDR выяваў;
2. Захоўванне: адмысловыя фарматы;
3. Адлюстраванне: пераўтварэнне ў LDR выяву асаблівымі адаптыўнымі алгарытмамі (tonemapping).

Прыклад FLOSS-праграмы: **Luminance HDR**.

Магчыма захаванне 48-бітнай выявы і далейшая апрацоўка (рэзультат працы tonemapping-алгарытмаў не заўсёды аптымальны, і часам мэтазгодна застасаваць далейшую карэкцыю).

Рэдагаванне класічных 48-бітных выяваў

Аб'екты:

1. сканаваныя плёнкі;
2. фатаздымкі у фармаце 48-bit TIFF;
3. фатаздымкі пасля папярэдняй RAW- ці HDR-апрацоўкі.

Тыповыя задачы:

1. Змена баланса белага (паканальная змена яркасці);
2. Рэдагаванне яркасці/кантрасту праз узроўні і гамму (чорны, белы, шэры пункты);
3. Рэдагаванне яркасці/кантрасту праз іншыя функцыі (напрыклад, сігмаідальны кантраст).

Усе задачы зводзяцца да аперацыяў з колерам (застасаванне пэўных функцыяў да значэнняў ўсіх пікселаў выявы). Геаметрычныя аперацыі мэтазгодна рабіць з канчатковай True Color выявай.

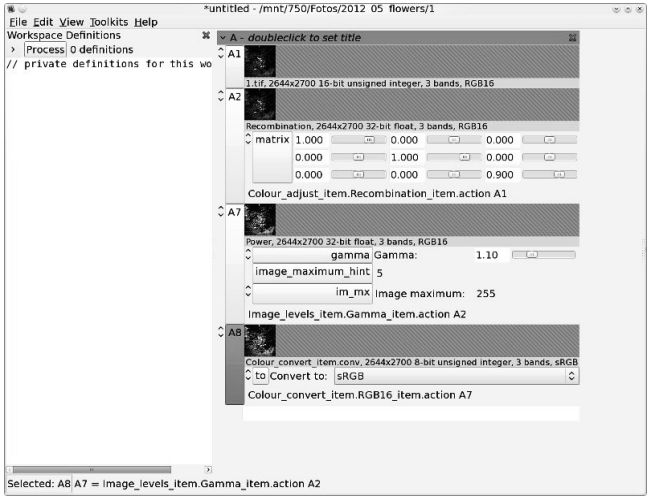
ImageMagick — праграма для пакетнай апрацоўкі выяваў.

Прыклады застасавання для апрацоўкі сканаванае выявы:

– Прыглушэнне сіняга каналу сканаванага слайда:
  `convert 1.tif -channel blue -gamma 0.95 2.tif`
– Павялічэнне яркасці ды кантрастнасці — варыянт 1:
  `convert 2.tif -level 5%,100%,1.2 3.tif`
– Павялічэнне яркасці ды кантрастнасці — варыянт 2:
  `convert 2.tif -sigmoidal-contrast 2.5,35% 4.tif`

Nip2 — праграма, аптымізаваная для апрацоўкі вялікіх выяваў. Прынцып працы нагадвае электронную табліцу (задаецца паслядоўнасць фільтраў, каторыя застасоўваюцца да пікселяў, рыс. 9.2).

Выкарыстанне абодвух праграмаў павязанае са зменай звычнага стылю рэдагавання выяваў і патрабуе некаторага звыкання.



Іл. 9.2. Апрацоўка выявы ў вакне праграмы nip2

Такім чынам, выкарыстанне выяваў з падвышанай глыбінёй колеру з'яўляецца важлівым для пытанняў апрацоўкі выяваў, асабліва фатаздымкаў. З-за праблем з падтрымкай deep color выяваў інтэрактыўнмі рэдактарамі тыпу GIMP, атрымліваецца застасаваць іншыя праграмы, пачаткова не прызначаныя для гэтае мэты.

Особенности возвращения наработок в апстрим

Василий Хоружик*

Presented submitting guide briefly describes an iteration of getting ones code upstream, which includes following steps: code modification, creating patch, submitting patch, getting review.

Введение

Сообщество open source широко известно своим подходом «Если тебе что-то не нравится, возьми и исправь». Но на самом деле испоправить недочеты в программном коде — это только половина задачи. Вторая и главная часть — добится включения патча в апстрим, т. е. в основное дерево исходных кодов проекта.

Зачем?

Включение исправлений в апстрим избавляет их автора от необходимости ручной пересборки пакета с наложенным патчем после выхода каждой новой версии. Не менее утомительно и заниматься постоянной адаптацией патча под изменения, происходящие в коде проекта от версии к версии.

Мейнтейнеры

На самом деле большой проект, даже такой сложный, как ядро Linux — всего лишь программа, а её мейнтейнер(ы) — обычные люди. Всё что требуется от автора патча — это привести патч в приемлимый вид и отправить нужному человеку (людям).

Coding Style

Важными нюансами являются особенности использования табуляций и пробелов. Аналогично обстоит дело с фигурными скобками, которые могут использоваться на той же самой или на следующей строке.

*Минск, Беларусь

Необходимо выяснить, как принято писать код в данном конкретном проекте. Часто у проекта есть соответствующий документ, например linux/Documents/CodingStyle. Есть смысл заранее настроить соответствующим образом свой текстовый редактор, либо переформатировать код в соответствии со стандартом проекта. А в некоторых случаях такие параметры уже подобраны, и даже готов соответствующий скрипт, например Lindent (linux/scripts/Lindent).

В некоторых проектах (u-boot, Linux, qemu) есть скрипт для проверки патча на соответствие стиля кодирования принятым нормам (scripts/checkpatch.pl)

Система контроля версий

Разработчики очень редко используют релизные версии (снапшоты) своих проектов. Пытаться добится включения патча в релизную версию кода — сомнительное предприятие. По всей видимости, для продвижения патча в апстрим понадобится освоить git (svn, hg, или другую систему контроля версий, которую используют в данном проекте).

В случае git всё выглядит просто:

- `git clone git://project.url my_local_project` — создать локальный клон удалённого репозитория
- `git diff [-cached]` — просмотреть текущие изменения
- `git add file_name.1 file_name.2` — добавить изменения для фиксации
- `git commit` — фиксация изменений, для большинства проектов обязательна опция `-s` (т.н. Sign-Off)
- `git format-patch id_коммита` — сформировать патчи для коммитов, начиная с коммита, следующего за указанным.

Куда отсылать

После очередной вычитки патча, прогона через checkpatch.pl наступает момент для отправки его на первый review. Кому именно отсылать — зависит от проекта, причем обычно это написано в README. Для Linux и u-boot есть скрипт get_maintainer.pl (находится в linux/scripts), который анализирует патч и определяет, кому его отправить (выдает список адресов).

Отсылать патчи для проектов, которые используют git, лучше всего с помощью git send-email. Используя какой-либо почтовый клиент, следует убедиться, что он не переносит строки, не заменяет табуляцию пробелами, и вообще ничего не изменяет в патче. Слать патч прикреплённым к письму считается дурным тоном, т. к. его в таком виде неудобно просматривать и комментировать. Поэтому патч нужно «включать» в письмо (inline).

Как правило, приходится отправлять сообщение и в список рассылки (на который лучше всего подписаться), и конкретному человеку (добавляя его адрес в поле CC:).

Для ядра Linux, если вы хотите включения изменения в релизную ветку ядра, также стоит ставить в копию stable@vger.kernel.org.

Review

Как правило, мейнтейнеры отвечают на письмо с патчем, «разбавляя» его комментариями. Очень редко всё получается с первого раза; Как правило, автор получает в ответ пачку рекомендаций по доводке своего кода. Это нормальное явление, поскольку контрибуция в открытый проект является итеративным процессом, а вы в этот момент находитесь на первой итерации. Также не имеет смысла спорить с мейнтейнером по поводу каждой строчки кода. Мейнтейнер практически всегда прав. Несколько рекомендаций:

- Процедура выглядит следующим образом: получить комментарий — поправить патч — отослать снова, с пометкой «v2» (или «vN»). Лучше не затягивать, т.к. при долгих итерациях патч может устареть или мейнтейнер может забыть контекст.
- При долгом отсутствии ответа может потребоваться послать письмо-напоминание. Поскольку мейнтейнер — обычный человек, он может быть занят или загружен работой, или просто забыть о Вас.
- Не имеет смысла ругайться с мейнтейнером. В крайнем случае следует попросить рассудить спор какое-либо третье лицо.
- И, наконец, не следует отчаиваться.

Никогда не следует слать патчи, которые:

- Содержат большие блоки закомментированного кода
- Перемещают файлы без видимых на то причин
- Меняют систему сборки проекта с технологии N на технологию M, без видимых причин и плюсов

– Патчи, реализующие сразу несколько фич либо исправляющие несколько багов. Необходимо следовать правилу «один баг — один патч», «одна фича – один патч».

А что если...

Может случится так, что предложенное изменение расходится с точкой зрения мейнтейнера и/или общей архитектурой проекта. В таком случае существует два выхода:

1. переделать патч
2. инициировать и поддерживать свой собственный форк проекта

Выводы

Итак, работу можно разделить на 4 шага:

1. Правка кода
2. Оформление изменений
3. Отсылка патча
4. Получение review, и при необходимости — возврат к пункту 2.

О преимуществах Erlang

Юрий Жлоба*

The article gives ideas on advantages of a programming language called Erlang. Erlang was created not as a general-purpose language, but for telecom domain where application should serve huge amount of clients, be fault-tolerant, continue working whatever happens and be updated while serving clients. Despite the fact that many new languages were created after Erlang, including such popular ones as Java and .NET, Erlang is still in demand, and today even more than ever. Features making Erlang so good are described, as well as why it is better than any other language for high-load, distributed, fault-tolerant systems.

Немного истории, это важно

Erlang не создавался как язык общего назначения, он создавался для довольно специфичной предметной области — для телекомов. Люди в компании Ericsson, и менеджеры, принимавшие ответственные решения, и инженеры, работавшие над языком, были прагматиками и точно знали, чего хотели. Они хотели:

1. обслуживать очень большое количество клиентов;
2. быть устойчивым к ошибкам, продолжать работать, чтобы ни случилось;
3. обновлять систему не прекращая обслуживать клиентов;

Поэтому в Erlang на уровне дизайна языка заложены:

1. эффективная реализация многопоточности (Concurrency);
2. устойчивость к ошибкам (Fault Tolerance);
3. распределенность (Distributed Computation);
4. горячее обновление (Dynamic Code Loading);

Erlang сразу хорошо показал себя на практике во внутренних проектах Ericsson. Конкурировать ему приходилось в основном с C++, и Erlang легко выигрывал эту конкуренцию. Но тут появилась Java и быстро захватила умы разработчиков и менеджеров. В т.ч. умы менеджеров Ericsson. Они попробовали Java, и оказалось, что для их задач Erlang все равно лучше.

*Минск, Belarus

Примерно до 2006 года Erlang был малоизвестным языком, который за пределами компании Ericsson использовался очень мало. А в 2006 году произошла важная вещь — производители процессоров уперлись в потолок при наращивании тактовых частот и стали наращивать производительность за счет добавления ядер процессоров. Что сделало многопоточное программирование очень актуальным и заставило IT мир обратить внимание на языки, умеющие эффективно распараллеливать выполнение на несколько процессоров. В т.ч. и на Erlang.

Причем реализация многопоточности в Erlang оказалась настолько эффективной, что ее охотно компировали в другие языки, например Scala и Go.

Сравнение виртуальных машин Java и Erlang

Для Scala и Java была создана библиотека AKKA, реализующая многопоточность на основе Message Passing, примерно так же, как в Erlang. Значит ли это, что теперь на этих языках можно создавать такие же эффективные многопоточные проекты, как на Erlang? Увы нет. Ибо есть важные различия на уровне виртуальных машин, которые никакими библиотеками нельзя компенсировать.

JVM использует потоки операционной системы, которые являются довольно сложными и относительно ресурсоемкими сущностями. Создание потока, удаление потока, переключение между потоками — все это происходит относительно долго. Нельзя создать сколько угодно много потоков, их количество лимитировано имеющейся оперативной памятью.

Виртуальная машина Erlang имеет собственную реализацию очень легковестных потоков, работающих поверх потоков операционной системы. Они очень быстро создаются, удаляются, переключаются, и используют очень мало ресурсов. Поэтому их можно создавать очень много — десятки и сотни тысяч в одном приложении.

И из этого следуют важные архитектурные различия JVM и Erlang проектов. Сервер, написанный на Java, не может себе позволить создавать отдельный поток на обслуживание каждого клиентского запроса. Приходится решать сложную архитектурную задачу, распределяя запросы по пулу заранее созданных процессов. Сервер, написанный на Erlang, может создавать поток для каждого запроса. И два потока, если надо.

Кроме этого, в виртуальных машинах сильно отличаются сборщики мусора. Сборка мусора в Java, несмотря на многолетние оптимизации, все-таки существенно замедляет работу приложения в те моменты, когда она работает. В Erlang сборщик мусора свой у каждого потока. Вместо одного большого сборщика, очищающего всю память приложения, работают много мелких сборщиков, очищающих небольшие участки памяти. И этот нюанс опять ведет к важным последствиям:

1. Они не работают все одновременно, а каждый отдельно, в разные моменты времени;
2. Небольшой участок памяти очищается быстрее, чем большой;
3. Среди всех процессов много таких, которые потребляют мало памяти. Их сборщики вообще никогда не запускаются;

В итоге сборка мусора в виртуальной машине Erlang не оказывает заметного влияния на производительность.

Применение Erlang и Perl в ISP

Наим Шафиев*

The article provides a comparison of Erlang and Perl being used by ISPs. In telecommunication we have a couple of tasks which can be resolved by different ways, and also with different languages. The Erlang is a "king"in telecommunication, but the Perl is really applicable to many tasks. An attempt is presented to compare their most applicable features. Language comparing is done along with following criteria: amount and quality of modules (CEAN, CPAN), education aspect, tools for debug, tools for profiling/benchmarking, IDE, development of language by type, tools for ISPs (sniffer, netflow tools, network monitors).

В сфере телекоммуникаций, как и в других сферах ИТ при разработке действует следующее правило. Для решения определенной задачи лучше всего использовать инструмент, который:

1. спроектирован под решение этого рода задач
2. имеет набор готовых библиотек («изобретать велосипед» как правило увлекательно, но недальновидно)

Но данные постулаты хорошо применимы только когда разработка начата с чистого листа, задачи хорошо конкретизированы и не меняются со временем. Как известно, в сфере телекоммуникаций (ISP) сложно заранее закладывать риски по вполне понятным причинам (рост траффика, изменение его качественной структуры), а следовательно, приходиться комбинировать инструменты. Кроме того накладываются как ограничивающий фактор исторические элементы системы (так называемые унаследованные свойства или legacy).

На основе суммированного опыта нами рассматривается вопрос применимости языков Perl и Erlang для различных задач при различных условиях.

Сравнение языков проведено по следующим прикладным моментам:

1. Объем и качество библиотек
2. Вопрос обучения (взаимозаменяемости программистов)

*Баку, Азербайджан

3. Инструменты для отладки
4. Инструменты для профайлинга/бенчмаркинга
5. IDE
6. Схема и основные коммитеры в развитии языка
7. Инструменты специфические для сферы ISP (снифферы, сетевые мониторы)

Суммированно к плюсам Erlang как платформы по отношению к Perl можно отнести её стабильность, высокую адаптацию под задачи отрасли (встроенные механизмы интеркоммуникации по сети, механизмы балансировки и избыточности, библиотека OTP), поддержку со стороны крупных компаний, встроенные механизмы параллелизации. Но есть и минусы по отношению к Perl: размер библиотек в разы меньше чем у CPAN, более высокая сложность обучения, более слабое сообщество, меньше сторонней документации, отчасти сильное влияние крупных компаний на развитие языка.

Обновление Linux с пятиминутным downtime

Николай Маржан*

The article presents and analysis for the operating system update process in the modern Linux distros, covering the options of resolving the related problems. An example of corporate sector requirements for updates is given. Author suggests his own way to resolve the issue and describes its advantages & disadvantages.

Одной из важных проблем современного мира Linux является проблема обновления всего дистрибутива с версии на версию. В дистрибутивах, нацеленных на корпоративный сектор (RHEL, Ubuntu), этими проблемами занимаются и решают их довольно успешно. В некоторых других дистрибутивах мы можем сталкиваться с серьезными трудностями при обновлении, или даже с продолжительным простоем сервисов (downtime). Простота, надежность и скорость процедуры обновления дистрибутива особенно важны на серверах, при простое которых останавливаются бизнес-процессы. Зачастую обновление дистрибутива на таких серверах проводят только в случаях крайней необходимости. Мотивами могут служить такие факторы, как ошибки программного обеспечения (ПО), наличие проблем безопасности в текущем ПО, а также нередки ситуации, когда для работы новой версии используемого проприетарного решения необходима более новая версия дистрибутива или ядра.

Иногда, вследствие обновления дистрибутива, мы можем получить полностью или частично неработающую систему. Например, конечный пользователь (end-user) может жаловаться на проблемы производительности, или может перестать работать какая-то часть функционала того приложения, которое установлено на данном сервере. В этот момент возникает более сложная проблема: вернуться в предыдущее состояние практически невозможно.

Автору было предложено реализовать систему обновления ПО, которая отвечала бы следующим требованиям:

*Киев, Украина

– Возможность возврата к состоянию «до обновления» через произвольный промежуток времени.
– Время простоя сервисов не более 5 минут.
– Возможность проводить все необходимые действия удаленно, без возможности физического доступа к оборудованию.
– Решение должно легко автоматизироваться, чтобы максимально исключить человеческий фактор.

Ни один из современных пакетных менеджеров не позволяет вернуться в гарантированно предыдущее состояние (при условии, что нужно обновить **весь** дистрибутив на две-три версии выше).

Возможен вариант с применением снимков файловой системы (snapshot), но этот подход приводит к большой потере произвольности дисковой подсистемы, а подход с заменой обычных внутренних жестких дисков на более производительную (или внешнюю) дисковую подсистему неприемлем.

Установка новой версии дистрибутива на неиспользуемый раздел HDD была одним из рассмотренных вариантов. Была предложена следующая последовательность действий на конечном сервере:

1. Скачивание образа дистрибутива или готовых для установки пакетов;
2. Установка дистрибутива на свободный раздел;
3. Копирование конфигурационных файлов с рабочего раздела на новый, включение сервисов в автозагрузку.
4. Настройка конфигурационного файла GRUB;
5. Перезагрузка в новый раздел.

Такой вариант соответствует выдвинутым требованиям, но для его реализации необходимо придерживаться следующих рекомендаций:

1. Используемая дисковая подсистема должна иметь следующие логические разделы:
   – Корневой раздел, на который будет установлена операционная система (ОС) со всем необходимым ПО.
   – Раздел для обновлений, идентичный корневому по размеру. На этот раздел будет выполняться установка новой версии.
   – Раздел для данных. Это самый большой раздел, на котором будут храниться данные, которые нельзя терять после обновления. Например, файлы баз данных, отчеты,

лог-файлы бизнес-приложений, которые должны храниться продолжительное время и т.д.

2. Знание списка конфигурационных файлов, которые нужно скопировать на новый раздел.
3. Наличие KVM over IP (Remote KVM). Для случаев, если была допущена ошибка на стадии редактирования конфигурационного файла GRUB.

Плюсами данного метода являются:

– Малое время простоя. Если все операции выполнены правильно, то оно фактически равно времени перезагрузки ОС.
– Возможность перезагрузиться в старый раздел по истечении любого промежутка времени.
– При фиксированном списке установленного ПО процесс легко можно автоматизировать.

Разработка сетевых устройств на базе дистрибутива OpenWRT

Виктор Полстюк*

The article gives an analysis of OpenWRT distribution as a basis for developing network devices. It also highlights limitations that constrain its use for commercial purposes.

Сначала OpenWRT развивался как альтернативный дистрибутив для серийно выпускаемых маршрутизаторов, который позволял получить более гибкую по сравнению с предоставляемой производителем систему, и дополнить ее необходимыми сервисами. Позже в дистрибутиве появилась поддержка X11, XFCE и LXDE, что позволяет использовать его на устройствах с графическими дисплеями. Сейчас проект перешел в категорию дистрибутивов общего применения (подобно OpenEmbedded), обладая наиболее широкими возможностями в области телекоммуникаций.

OpenWRT поддерживает большое число аппаратных платформ, построенных на процессорах ARM, MIPS, x86, и содержит свежее ядро с набором специфических для маршрутизаторов патчей.

Сборка дистрибутива основана на пакетной системе, что позволяет выбрать перечень программ для добавления в прошивку и включает 2000+ пакетов в официальном репозитории. Система конфигурирования и инициализации позволяет получить согласованно функционирующую систему из набора выбранных пакетов.

Веб-интерфейс является обязательным компонентом сетевого оборудования. Для OpenWRT существует несколько фреймворков для пользовательского интерфейса: LuCI, X-wrt. LuCI построен по принципу Model-View-Controller, что предполагает разделение графического представления и логики работы интерфейса управления.

Имеются примеры коммерческого использования дистрибутива производителями процессоров, OEM-модулей и отладочных плат в качестве сопутствующего SDK. Однако использование mainline OpenWRT в коммерческих целях затруднено отсутствием поддержки аппаратных ускорителей сетевых контроллеров процессоров и централизованных протоколов управления (семейство TR-069).

*Минск, Беларусь

Тонкий клиент на базе процессора Marvell Kirkwood

Сергей Зенькевич*

The article describels hardware and software parts of the thin client device based on Marvell Kirkwood processor and Debian GNU/Linux 6.0.

Представленное устройство предназначено для организации терминального доступа к серверам, построенным на решениях фирм Microsoft (на основе протокола RDP), VMware (VMware View), Citrix (Citrix XenDesktop). Устройство в основном ориентировано на доставку к рабочим столам Windows 7, а также для RDP соединений поддерживается доставка отдельных приложений.

Аппаратная платформа построена на базе процессора Marvell Kirkwood 88F6282. Данный процессор реализован на ядре Sheeva, работающем на частотах до 2ГГц (в устройстве используется процессор 1,6 Ггц). Процессор имеет два гигабитных Ethernet, которые подключены к внешнему PHY 88E1121R от Marvell. В процессоре имеется также два порта PCIe: первый порт используется для подключения GPU устройства, а второй выведен на внутренний разъём mini-PCI, к которому возможно подключение дополнительных внешних устройств или модулей WI-FI.

В качестве графического контроллера используется микросхема VOLARI-Z11 от компании SiS. Видеосигнал выведен на DVI-разъем. Устройство поддерживает разрешение до $1600 \times 1200 \times 32$. Звуковая подсистема реализована на отдельном аудио-кодеке, который имеет выход для наушников и микрофонный вход. Устройство оснащено четырьмя портами USB. Опционально на панель устройства может быть выведен разъем COM-порта. В устройство устанавливается 1 Гбайт оперативной памяти и 1 Гбайт NAND Flash (используется файловая система UBIFS). Питание системы осуществляется от внешнего блока питания.

Основой ПО тонкого клиента является дистрибутив Debian 6 (Squeeze). Так же, на начальном этапе, делались попытки построения дистрибутива на базе Open Embedded. Используется ядро Linux версии 2.6.39. В качестве среды рабочего стола выбран XFCE

*Минск, Беларусь, Promwad Engineering, sergei.zenkevich@promwad.com

4.4. Для организации терминального доступа используются следующие клиенты: rdesktop 1.6, FreeRDP 1.0, VMware View Open Client 4.5.0, Citrix Receiver for Linux 11.100.

Также нами было разработано приложение «Менеджер Тонкого Клиента» (на базе Qt) для управления тонким клиентом. Это приложение позволяет создавать, запускать, изменять параметры терминальных соединений различного типа (RDP, Citrix, VMware), редактировать и разграничивать права доступа к терминальным соединениям, обновлять ПО тонкого клиента, получать информацию о текущем состоянии тонкого клиента. Также на тонкий клиент устанавливается дополнительное ПО для работы в автономном режиме, а именно веб-браузер и мультимедиа-проигрыватель.

Сосуществование Linux и RTOS на одном многоядерном процессоре

Дмитрий Горох*

During the last years Linux have finally been recognized as an established embedded OS. The reasons of embed Linux into various devices are obvious: Linux is free, it has tremendous amount of free and useful applications, supports decent hardware, is POSIX compliant, and has both community and commercial support. However, not every user application can be easily moved from RTOS to Linux. The article discusses a possibility of using both RTOS and Linux concurrently within a multicore processor.

С каждым годом ОС Linux набирает популярность как ОС для встраиваемых систем. Причины тому очевидны: обладая свободной лицензией, ядро Linux вкупе с многочисленными свободными приложениями и библиотеками обеспечивает поддержку широкого спектра аппаратуры (включая современные системы-на-кристалле), устоявшийся POSIX-совместимый API и хорошую поддержку как со стороны сообщества, так и от коммерческих организаций. Однако не все приложения могут быть беспрепятственно перенесены с привычных систем реального времени (ОСРВ) на Linux. Помимо очевидных причин, таких как нехватка ОЗУ или ПЗУ для загрузки и полноценной работы Linux, можно отметить следующие причины:

1. От пользовательского приложения требуется работа в режиме жёсткого реального времени
2. Аппаратные прерывания должны обрабатываться с минимальной задержкой и джиттером
3. Система должна быть максимально предсказуемой и надёжной
4. Накладные расходы, связанные с работой ОС, должны быть минимальными

Один из возможных способов удовлетворить всем этим требованиям, при этом не отказываясь от использования Linux — использовать многоядерные процессоры для одновременного запуска ОСРВ

*Promwad, Минск, Беларусь

и Linux. При этом критически важные пользовательские приложения будут работать под ОСРВ на основном ядре процессора, а второстепенные (такие, как пользовательский интерфейс или сетевой стек) — под Linux на дополнительном ядре. Такая архитектура особенно примечательна при использовании гетерогенных мультипроцессорных систем, в которых процессорные ядра существенно отличаются по сложности, вычислительной мощности и энергопотреблению (например ARM Cortex-A9 и ARM Cortex-M4). Имея полный контроль над системным ПО, запускаемым на каждом ядре, можно добиться близкой к максимальной энергоэффективности для заданной аппаратной флатформы.

Совмещение ОСРВ и Linux на одном процессоре требует разделения аппаратных ресурсов и межядерного взаимодействия.

Разграничивание устройств ввода-вывода между несколькими ОС является довольно сложной задачей, которая решается, например, введением в архитектуру гипервизора, виртуализирующего аппаратные ресурсы. Обычно в геторогенных системах аппаратные ресурсы жёстко закреплены за процессорными ядрами, поэтому как правило единственным разделяемым устройством является ОЗУ. Обе ОС необходимо сконфигурировать таким образом, чтобы они использовали разные регионы физической памяти.

Проблема межядерного взаимодействия на низком уровне решается в зависимости от предоставляемых аппаратных средств. В общем случае для этого выделяется небольшой регион разделяемой памяти. Для оповещения ядер обычно существует возможность генерации программного прерывания.

Multicore Assotiation (MCA) предлагает стандартизированный API реализации интерфейсов высокого уровня: MCAPI. Его цель — унификация межпроцессорного взаимодействия между различными ОС, что позволяет максимально отвязать программную реализацию от аппаратуры и ОС-специфичных программных модулей.

MCAPI не предъявляет жёстких требований к аппаратуре и ОС, и спроектирован так, чтобы портирование было максимально простым и понятным. В настоящий момент существует референсная реализация MCAPI от Mentor Graphics, которая называется OpenMCAPI. OpenMCAPI может работать под Linux и Nucleus. Портирование под другие ОС не должно составить больших трудов.

Данная архитектура была проверена на Freescale SoC P1020, на базе которого разрабатывался индустриальный компьютер для контроля электрических подстанций.

Writeat: доступные книги для читателей и писателей

Александр Загацкий*

The article describes Writeat, a free and simple tool aimed at electronic and printed books creation. Books are written in a pod6 format. Pod6 is a simple and concise markup language. To create a file in this format you can use any text editor. Writeat is opensource startup.

Современная экономика направлена на воспитание в нас потребителей. Поэтому с понятием «доступный» свзяывают в первую очередь низкую стоимость какого-либо товара или услуги. Например: доступное жилье, доступные продукты питания, доступные книги.

Однако насколько легко построить дом или написать книгу? Ответ на данный вопрос зависит от доступности орудий производтства и технологий. Если построить дом просто для строителя, то и квартиры в этом доме будут доступны большему количеству жильцов.

Проект Writeat [1] является бесплатным и простым инструментом для создания книг в электронном и печатном виде.

Пишутся книги в формате pod6. Pod6–простой и лаконичный язык разметки. Для создания файлов в этом формате подойдет любой текстовый редактор.

Шаблон книги выглядит следующим образом:

```
=TITLE Моя очередная книга
=SUBTITLE или как просто делиться знаниями
=AUTHOR
firstname:Вася
surname:Пупкин
=DESCRIPTION
В этой книге описаны основные правила, которые позволят
наиболее просто поделиться своими знаниями и опытом.
=CHAPTER Вступление
Начало книги !
```

*Витебск, Беларусь

Специальные директивы начинаются со знака «=» и следующим за ним названия блока. Используются следующие блоки:

=TITLE заголовок книги
=SUBTITLE подзаголовок
=AUTHOR автор
=DESCRIPTION краткое описание книги
=CHAPTER название главы

Основной чертой формата pod6 является его расширяемость. Благодаря этому стала возможной вставка изображений и разбиение книг на части.

Иногда бывает удобно, чтобы главы располагались в отдельных файлах. Для этих целей используется блок =Include :

```
=Include src/preface.pod6
=Include src/basics.pod6
=Include src/operators.pod6
=Include src/subs-n-sigs.pod6
```

Такой прием облегчает совместную работу над книгой нескольких авторов.

Для вставки изображений используется следующий блок:

```
=Image img/bold1.jpg
```

Writeat позволяет облегчить поддержку технической документации, особенно публичную. Дополнительный блок CHANGES позволяет вести журнал изменений документа. Он располагается в самом начале документа и необходим для описания основных изменений. Например, изменения в API сервиса.

```
=begin CHANGES
Jun 6th 2012(v0.2)[zag]   Предисловие
May 27th 2012(v0.1)[zag]   Начальная версия
=end CHANGES
```

Для установки последней версии пакета writeat для Ubuntu необходимо выполнить команды:

```
sudo add-apt-repository ppa:zahatski/ppa
sudo apt-get install writeat
man writeat
```

Writeat на данный момент находится в начале развития. Он бесплатен и открыт [2]. В его основе лежат открытые технологии и форматы. Наличие бесплатного и простого способа создать книгу позволяет решить задачу доступности книг как для читателей, так и для писателей. Это значит, что будущее, в котором учебники бесплатны, вполне реально.

Литература

[1] Сайт проекта с образцами книг.`http://writeat.com`

[2] Репозиторий проекта writeat. `https://github.com/zag/writeat`

Использование клиент-серверной архитектуры MythTV 0.25

Алексей Бутько*

The article discusses specifics of MythTV usage. MythTV turns a computer into a network streaming digital video recorder, a digital multimedia home entertainment system, or home theater personal computer. It has two main logical elements: the backend, which contains the TV capture cards, and stores the recorded video, and the frontend, which is connected to user's TV screen and lets him watch LiveTV and recorded shows. The simplest configuration puts both frontend and backend in same physical box, while advanced setup might separate backend and frontend hardware.

Слияние технологий сегодня играет немаловажную роль в цифровых устройствах, которые мы используем. В целом, пакет MythTV можно рассматривать как центр управления всеми приложениями и устройствами, которые отвечают за наш цифровой досуг.

MythTV в первом приближении состоит из двух логических компонентов:

– backend — работает с картами захвата, хранит и записывает видео.
– frontend — пользовательский интерфейс, получающий данные от backend'а и выводящий изображение непосредственно на экран телевизора.

Обычно под построение HTPC выделяется один физический системный блок, вследствие чего backend и frontend делят его между собой. Однако все функции такого мощного инструмента как MythTV раскрываются как раз в сетевом (клиент-серверном) исполнении. Первое, и главное преимущество — это возможность использования нескольких frontend'ов по всему дому или даже за его пределами. Также появляется возможность управления конкретным frontend'ом с мобильного устройства по сети. Такой метод

*Минск, Беларусь

управления с расширением использования технологий WiFi приобретает всё большую популярность и вскоре обещает вытеснить традиционные инфракрасные пульты. Не так давно возможность использовать MythTV (иметь полнофункциональный frontend) появилась на платформах Android и iPhone, что делает медиа-контент еще более доступным.

Возможности MythTV и впрямь широки: система умеет работать с большинством карт захвата (аналоговых и цифровых), имеется поддержка IPTV (хотя по мнению автора ещё недостаточно зрелая), мощный планировщик может вести запись телепрограмм по расписанию, имеется возможность удаления рекламы.

Из офицальных плагинов можно перечислить:

- MythBrowser — минибраузер, не полнофйункциональный, но может оказаться полезным.
- MythArchive — средство создания DVD из имеющихся TB-записей и любых других видеофайлов.
- MythGallery — инструмент для просмотра изображений.
- MythGame — интерфейс для эмуляторов игровых консолей
- MythVideo — в текущей версии не являестя плагином, а входит в состав frontend'а. Позволяет обозревать и просматривать домашнюю коллекцию фильмов. Примечателен тем, что выискивает подробную информацию о конкретной картине в онлайн-сервисах.
- MythMusic — прослушивание музыки, составление playlist по различным критериям. До последней версии интерфейс обладал низким функционалом и слабой эргономикой
- MythNews — отслеживание лент RSS и отображение новостей на экране.
- MythWeather — онлайновый прогноз погоды с весьма обширным функционалом. Работает с различными источниками данных, может отображать живую карту метеообстановки.
- MythWeb — плагин для удалённого web-конфигурирования backend'а. В условиях вынесения последнего на отдельный сервер может оказаться крайне полезным.

Последние полтора года активно шла работа над версией 0.25, и, наконец, в апреле состоялся релиз. В новую версии вошло более 5200 коммитов. Из ключевых улучшений можно отметить:

- Долгожданная поддержка аппаратного ускорения декодирования видео с использованием VAAPI и поддержка архитектуры акселерации DirectX Video Acceleration 2;

– Поддержка аудио-кодеков E-AC3, TrueHD и DTS-HD;
– Улучшены средства для управления метаданными для записываемых видеоматериалов. Удалена поддержка утилиты для работы с метаданными jamu, вместо которой теперь используется компонент MythMetadataLookup;
– Представлен полнофункциональный сервисный API для обеспечения взаимодействия внешних приложений с MythTV, как с бэкендом, так и с фронтэндом. Новый API можно использовать в том числе для организации потоковой доставки контента поверх HTTP (HTTP Live Streaming). Ранее используемый API MythXML объявлен устаревшим;
– Полностью переписан модуль MythMusic, используемый для обеспечения проигрывания музыки и управления музыкальной коллекцией. Переработана архитектура видеоплеера MythVideo. Функции MythMusic и MythVideo теперь непосредственно интегрированы в MythTV, а не распространяются в виде плагинов;
– Коллекция визуальных тем MythThemes более не рассматривается как внешний репозиторий, все визуальные темы, включая темы от сторонних разработчиков, могут быть загружены непосредственно через интерфейс выбора тем, интегрированный во фронтэнд;
– Проигрыванием контента при помощи MythNetvision, например, при просмотре роликов из YouTube, можно управлять через пульт ДУ. В MythNetvision по возможности используется встроенный базовый плеер MythTV;
– Поддержка 3D-эффектов при выводе горизнтального и вертикального меню; Начальная поддержка анимации в MythUI;
– Прекращена поддержка механизма акселерации XvMC и удалена поддержка libmpeg2 для проигрывания видео;
– Переписана система ведения логов;
– Прекращена поддержка Python 2.5, в качестве минимальной версии рекомендуется Python 2.6. Также для работы требуется Taglib 1.6+ и Qt 4.6+. Из списка зависимостей исключены libvisual, libsdl, libcdaudio, libcdda_paranoia и wget.

Резюмируя, можно сказать, что MythTV развился в серьёзный и универсальный инструмент для построения разветвлённой домашней медиасистемы. Однако его массовому использованию мешают, отпугивая пользователей, черезмерная сложность в установке и рекомендации по конфигурированию в объеме отдельной книги.

Simulation of Grover's algorithm on parallel computers with shared memory and using the Olib library

Łukasz Świerczewski*

The study presents the aspect of the quantum simulation of the Groover's algorithm using contemporary parallel processors with shared memory. Linux operating system, C programming language and Olib library, created by the author for wider set of numerical calculations, were used as a programming environment.

Introduction

Quantum algorithms are very difficult to simulate using contemporary computers because of their computational complexity. Groover's quantum algorithm implemented for contemporary computer is characterised by exponential complexity which is shown in the Fig. 19.1.

During the analysis, the Olib library was used, which supports parallel computing. Its results in comparison with the Python programming language and NumPy module, which support only sequential operations, are shown in the Fig. 19.2

More on Olib

Olib (`http://www.goldbach.pl/olib/`) is written primarily for Linux. In other systems (eg. Windows, BSD, Solaris) minor compatibility issues with the code may occur. Library implements efficient methods, which can be divided into following groups:

- Linear algebra
- Discrete Mathematics
- Cryptography
- Numerical methods
- Artificial intelligence

*Łomża, Poland

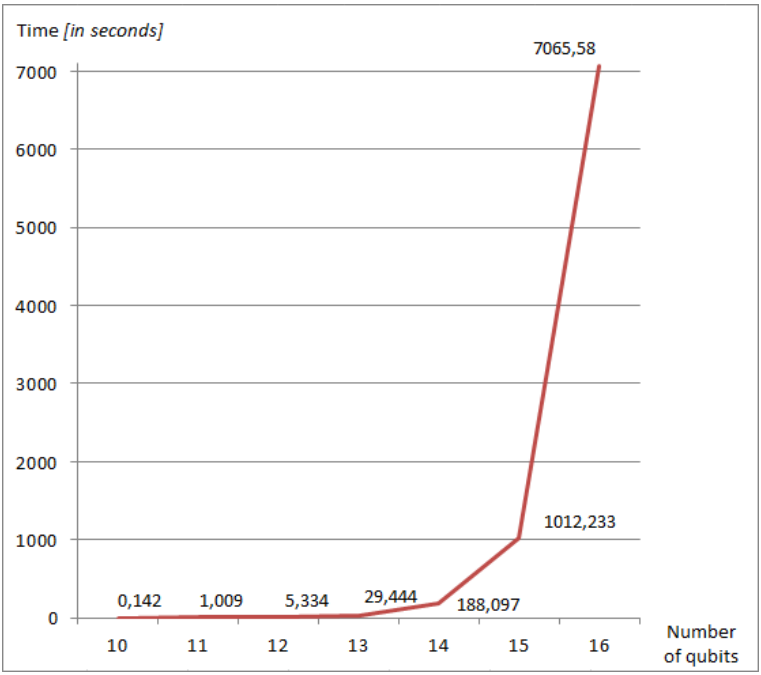


Fig. 19.1. Increase of processing time of sequential algorithm on the Intel Xeon E7-4860 processor during the simulation of the quantum computer depending on size of a quantum register

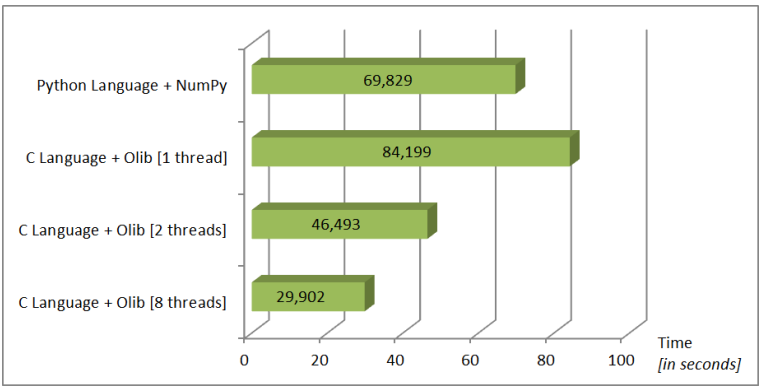


Fig. 19.2. Comparison of Olib library performance with parallel processing and the NumPy (test for 14 qubits and processor Intel Core i7 920)

Olib was optimised for several computing architectures. Within one program, the programmer can combine both functions using multi-core processors and those which support graphic accelerators. The project is rather new and may not have very large set of possibilities in comparison with some other software of this kind, which has been developed for a long time already. However, in many cases very good results can be achieved with it, especially when a programmable graphic accelerator is available, or a multi-core processor, as in our case. Olib is licensed under GPL version 2 license or higher.

Implementation and result

Performance tests were carried out on four different equipment platforms based on Intel Core 2 Quad Q8200 processor, Intel Core i5-2400, Intel Core i7 920 processor and the system which consists of four Intel Xeon E7-4860 processors.

The Fig. 19.3 presents the results of the simulation of the quantum register which consists of 13 qubits and Intel Core i5-2400 processor. The minimum of 1,5 GB of free random access memory (RAM) is necessary to perform this kind of simulation. There were noticeable changes in performance when the single-channel memory was used in comparison to the dual-channel DDR3 memory. Memory in the dual-channel mode reaches 21,2 GB/s maximum throughput thanks to the

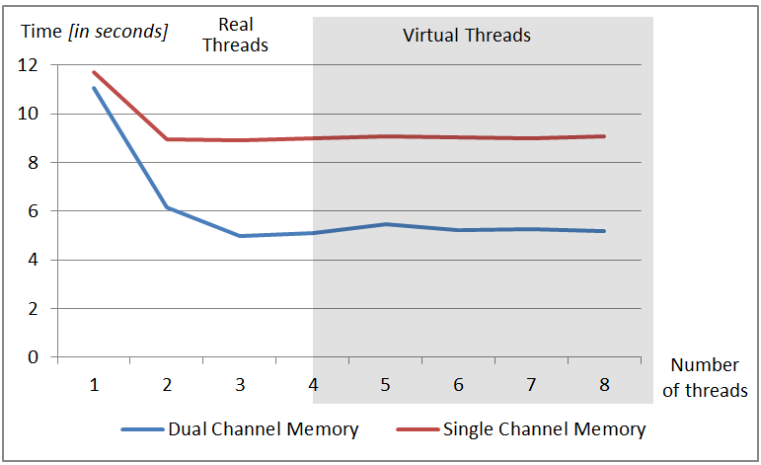


Fig. 19.3. Decrease of time of register simulation which consists of 13 qubits when different threads on the processor Intel Core i5-2400 and single channel memory and dual-channel memory DDR3 are used

frequency of 1333 MHz. When single-channel mode is used maximum reachable throughput is just 10,6 GB/s. Difference in the transfer of the random-access memory has an important impact on the minimum time execution of the algorithm, which for the dual-channel memory is 5,107 seconds and for the single channel memory is 8,916 seconds.

No acceleration was noticed on older Intel Core 2 Quad Q8400 processor despite parallel programming was used. Tested platform used single-channel memory with 10,6 GB/s capacity but processor Core 2 Quad in comparison to the modern processors Core i5/i7 and Xeon doesn't have integrated DDR3 memory controller and that probably significantly slows down the communication process when the algorithm is processed. Results for Intel Core 2 Quad processor are shown in the Fig. 19.4

Another tested platform is based on Intel Core i7 920 processor (fig. 19.5). It uses tri-channel DDR3 1066 MHz memory with total capacity of 25,6 GB/s. In this case, a speedup is noticeable not only for four physical cores, but also after using eight threads supported by Hyper-Threading Technology. By using four threads the execution time of the algorithm is 31,049 seconds, and for eight threads with HT Technology enable it decreases to 29,902 seconds. Some tests were also

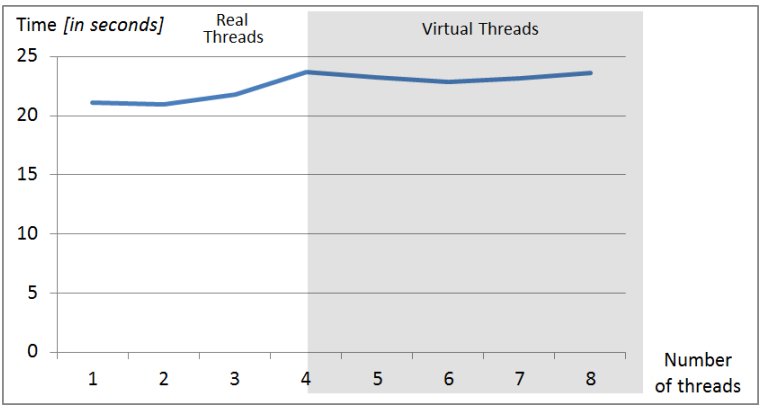


Fig. 19.4. Decrease of time of register simulation which consists of 13 qubits when used different threads on the processor Intel Core 2 Quad Q8400

carried out for eight threads on Intel Core 2 Quad Q8400 (Fig. 19.4) and Intel i5-2400 (Fig. 19.3) processors despite they have only four cores without support of HT Technology. However, it can be noticed that increase of quantity of threads without HT technology has only negative influence on the capacity because system planner must allocate excess threads to less number of processors. Additional threads which exceed physical quantity of arithmetic and logical units and can be operated by HT Technology (if available) were indicated in the diagram as Virtual Threads.

The most advanced tested platform is based on four Intel Xeon E7-4860 processors. These processors support HT Technology which gives total of 80 threads in the system. However the best performance result of 384,308 is reached for 30 threads. It would take 7065,580 seconds, the usual program which is activated in sequential mode to run on this platform and it means a speedup of about 18,385. Reduction of the execution time thanks to applying many threads is show in the Fig. 19.6. This platform had physically 256 GB of DDR3 memory which enabled performing simulation of the system consisted of 16 qubits–in this case the program allocated 96 GB of memory.

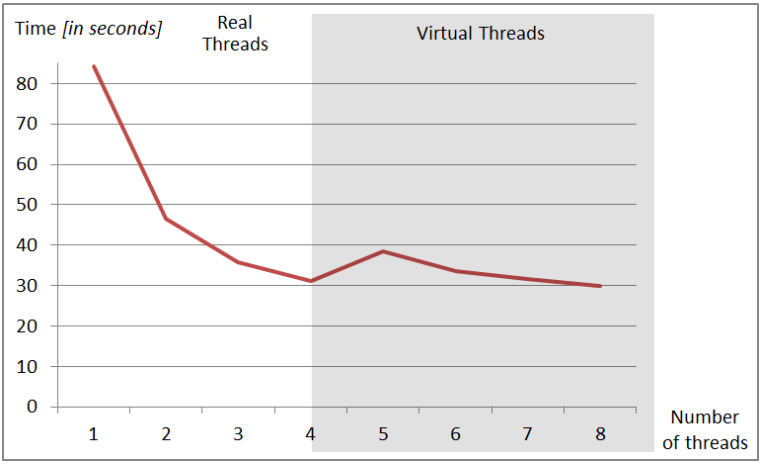


Fig. 19.5. Decrease of time of register simulation which consists of 14 qubits when used different threads on the processor Intel Core i7 920

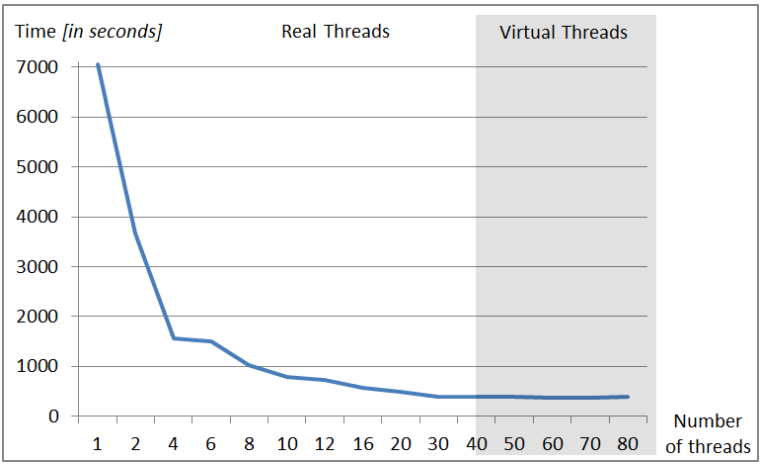


Fig. 19.6. Decrease of time of register simulation consisting of 16 qubits during using different quantity of threads on the platform which consists of four Intel Xeon E7-4860 processors

Conclusion

We've shown the possibility of simulation of the Grover's algorithm, and found out that performance inrease is quite noticeable on the modern parallel architectures. Usage of graphic accelerators for simulation of quantum machines can be also a very promising idea. As graphic processors handle very well typical matrices and vectors operations, this technology has an interesting future.

References

[1] B. W. Kernighan, D. M. Ritchie. The C Programming Language, Prentice Hall, Inc, 1988.

[2] S. Dasgupta, C. Papadimitriou, U. Vazirani. Algorytmy, Wydawnictwo Naukowe PWN, 2010 / Algorithms, McGraw-Hill Higher Education, 2006.

[3] Z. Czech. Wprowadzenie do obliczeń równoległych, Wydawnictwo Naukowe PWN, 2010.

[4] M. Hirvensalo. Algorytmy kwantowe, WSAP, 2010 / Quantum Computing, Springer, 2001

[5] Michel Le Bellac. Wstęp do informatyki kwantowej, Wydawnictwo Naukowe PWN, 2012 / A Short Introduction to Quantum Information and Quantum Computation Science, Cambridge University Press, 2006.

[6] K. Giaro, M. Kamiński. Wprowadzenie do algorytmów kwantowych, Exit, 2003.

Построение частного облака на базе дистрибутива Proxmox Virtual Environment

Дмитрий Ванькевич*

The article discusses Specifics and purposefullness of private cloud usage. Author reviews demands to the cloud, as one one of possible solutions, based on the Proxmox Virtual Environment distributive.

По модели развёртывания облака делятся на частные, публичные и смешанные. Частное облако — это вариант локальной реализации «облачной концепции», когда компания создает ее для себя самой, в рамках одной организации. В отличие от него, публичное облако используется облачными провайдерами для предоставления сервисов внешним заказчикам, а смешанное или гибридное облако — это вариант совместного использования двух вышеперечисленных моделей.

Национальным институтом стандартов и технологий США зафиксированы следующие обязательные характеристики облачных вычислений:

- Самообслуживание по требованию (англ. self service on demand), потребитель самостоятельно определяет и изменяет вычислительные потребности, такие как серверное время, скорости доступа и обработки данных, объём хранимых данных без взаимодействия с представителем поставщика услуг;
- Универсальный доступ по сети, услуги доступны потребителям по сети передачи данных вне зависимости от используемого терминального устройства;
- Объединение ресурсов (англ. resource pooling), поставщик услуг объединяет ресурсы для обслуживания большого числа потребителей в единый пул для динамического перераспределения мощностей между потребителями в условиях постоянного изменения спроса на мощности; при этом потребители контролируют только основные параметры услуги (например,

*Львов, Украина

объём данных, скорость доступа), но фактическое распределение ресурсов, предоставляемых потребителю, осуществляет поставщик (в некоторых случаях потребители всё-таки могут управлять некоторыми физическими параметрами перераспределения, например, указывать желаемый центр обработки данных из соображений географической близости);
- Эластичность, услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме;
- Учёт потребления, поставщик услуг автоматически исчисляет потреблённые ресурсы на определённом уровне абстракции (например, объём хранимых данных, пропускная способность, количество пользователей, количество транзакций), и на основе этих данных оценивает объём предоставленных потребителям услуг [1].

Целесообразность внедрения облачных технологий определяется здравым смыслом, финансовыми возможностями заказчика, недостатками существующей «безоблачной» компьютерной инфраструктуры, осознанием потребности и готовностью внедрять новые технологии. При этом вариант частного облака часто оказывается единственным доступным вариантом — не в последнюю очередь из-за отсутствия «широких» каналов связи. В качестве примера организаций, для которых автору приходилось решать задачу внедрения облачных вычислений по модели частного облака, можно привести:

1. Компьютерные лаборатории общего использования Львовского национального университета имени Ивана Франко, где уже применялись технологии виртуализации в учебном процессе [3].
2. Небольшая фирма по разработке программного обеспечения, обнаружившая потребность в активном использовании технологий виртуализации, и имеющая около тридцати сотрудников, работающих с пятью независимыми серверами виртуальных машин.

В первом случае внедрение облачных технологий представляет академический интерес, во втором, при планируемом увеличении количества серверов виртуальных машин, позволит эффективнее использовать аппаратное обеспечение и даст возможность удобного управления серверами виртуальных машин.

Среди платформ, подходящих для построения частного облака из доступных аппаратных компонентов, автором был выбран дистрибутив Proxmox Virtual Environment [4].

Proxmox Virtual Environment дает возможность использовать следующие технологии виртуализации:

- OpenVZ — виртуализация на уровне операционной системы. Позволяет на одном физическом сервере запускать множество изолированных копий операционной системы.
- KVM (Kernel-based Virtual Machine) — позволяет запускать виртуализованную ОС при аппаратной поддержке процессора (Intel VT, AMD-V и др.) и эмуляции периферии при помощи QUEMU.

К числу достоинств данной платформы можно отнести:

- Удобный инсталлятор (bare metal ISO-installer) позволяющий развернуть сервер виртуальных машин за 10–15 минут;
- Простое управление через веб-интерфейс (с сохранением возможности управления через интерфейс командной строки);
- Наличие библиотеки готовых шаблонов OpenVZ, готовых для промышленного использования («production ready»);
- Встроенная система мониторинга;
- Поддержка разных типов аутентификации: MS ADS, LDAP, Linux PAM, Proxmox VE authentication;
- Гибкое управление правами доступа групп пользователей к консоли управления виртуальными машинами на основе ролей;
- Наличие Proxmox VE API;
- Возможность объединения серверов в кластер и живой миграции виртуальных машин (без остановки гостевой системы);
- Встроенная система автоматического резервного копирования
- Поддержка систем хранения Directory, LVM group, NFS share, iSCSI target.
- Возможность работы на распространённом аппаратном обеспечении (некоторые проприетарные системы виртуализации требуют сертифицированное аппаратное обеспечение). Это особенно
  важно для учебных заведений, которые зачастую не обладают возможностью выбора при закупке оборудования.

По результатам работы с дистрибутивом Proxmox Virtual Environment можно сделать заключение, что он соответствует пе-

речисленным обязательным характеристикам для облачных вычислений.

Литература

[1] http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F

[2] И.П. Клементьев, В.А. Устинов. Введение в облачные вычисления. http://www.intuit.ru/department/se/incloudc/3/3.html

[3] Д. Ванькевич, Г. Злобин Быстрое развёртывание учебного полигона для проведения лабораторных работ в курсе «Системное администрирование ОС Linux» в компьютерных лабораториях общего пользования. http://freeschool.altlinux.ru/wp-content/uploads/2012/01/zlobin.pdf

[4] http://pve.proxmox.com/wiki/Main_Page

OBS — частная практика. Заметки на полях

Денис Пынькин*

Open Build Service (OBS) is an open and complete distribution development platform. It provides the infrastructure to easily create and release open source software for openSUSE and other Linux distributions on different hardware architectures. Article describes some approaches for usage of private OBS instance in development process.

Некоторые проблемы коммерческой разработки

Подавляющее большинство современных дистрибутивов на базе ОС Linux использует бинарные пакеты для распространения программного обеспечения. Таким образом гарантируется, что все пользователи получают программы скомпилированные «правильной» версией компилятора, в «правильном» окружении и с корректно установленными зависимостями на другие программы и библиотеки, необходимые для функционирования.

В общем случае это означает, что между исходным кодом и конечным пакетом находится целый пласт процессов и правил, описывающих как правильно «приготовить» программное обеспечение из исходного кода, причем эти процессы и правила уникальны для различных семейств дистрибутивов.

Такая сегментация часто приводит к тому, что коммерческие разработки все теснее интегрируются с каким-либо единожды и давно выбранным дистрибутивом. Процесс разработки коммерческого продукта предназначенного для bare-metal установки во многом схож с созданием форка выбранного дистрибутива и последующей его поддержкой, так что переход на другую, более подходящую в современных реалиях базу, потребует от производителя больших

*Минск, Беларусь

затрат времени, сил и средств, фактически останавливая разработку своего продукта на время переезда.

Еще одна проблема, с которой сталкиваются разработчики — это смешивание и наслоение правил создания программного обеспечения, сборочной среды и подготовки рабочего образа операционной системы. Даже изначально отлично спроектированный и реализованный процесс, с годами начинает обрастать дополнительными «фичами» и «подпорками», которые сплетают красивую и стройную систему в единый клубок, распутать который становится проблематично.

Сборочные системы призваны упростить и минимизировать работу человека по созданию и сопровождению программного обеспечения, входящего в дистрибутив. Кроме того они стараются отслеживать соблюдение правил создания программного обеспечения и результирующего установочного образа. Одна из лучших по совокупности параметров и возможностей система автоматической сборки общего назначения — это Open Build Service [1], представляющая собой универсальную модульную и расширяемую базу для создания проектов любого уровня.

Проекты в OBS

Проект в OBS — это это организационная единица, представляющая собой единую площадку с общими правилами сборки для всех пакетов.

Проект включает в себя:

- конфигурацию проекта, включая макросы и определения используемые в сборочной среде;
- сборочные цели — дистрибутивы, на базе которых OBS будет пытаться собрать пакеты, входящие в проект.
- пакеты, состоящие из:
  - исходного кода;
  - правил сборки программного обеспечения (spec, dsc) или дискового образа (kiwi).

Конфигурация проекта, сборочные цели и зависимости, прописанные в правилах сборки, описывают, как будет создаваться сборочное окружение для конкретного пакета и какое программное обеспечение туда будет входить. Конфигурации проекта наследуются из базового проекта, который определяется сборочной целью

— так, для сборки одного и того же пакета для разных сборочных целей, например для Fedora и CentOS, будет использоваться различный набор макросов, а зависимость foo, прописанная в spec-файле, преобразуется в foo-fedora и bar-centos соответственно.

Распределенная инфраструктура

Одна из интереснейших возможностей OBS — возможность взаимодействия между серверами. В архитектуру OBS изначально заложена развитая система кэширования, которая позволяет минимизировать сетевой трафик и нагрузку на отдельно взятый сервер — ведь передаются только обновленные пакеты, причем обновления отслеживаются автоматически, как на стороне сервера, так и на стороне клиента.

Такой режим работы особенно актуален для географически распределенных офисов, так как позволяет переложить на плечи сборочной системы отслеживание изменений в зависимостях, их автоматическое скачивание и пересборку локальных проектов с уведомлением о результате. Кроме того, можно создавать локальные сборочные фермы, расположенные близко к разработчикам, если в этом есть необходимость.

Взаимодействие с системами контроля версий исходного кода

OBS поддерживает версионирование исходного кода в стиле Subversion. Однако, по внутренним правилам многих коммерческих компаний для хранения исходного кода требуется использовать какую-либо внутреннюю систему контроля версий.

В таких случаях на помощь приходят сервисы — специальные расширения, позволяющие взаимодействовать с системами контроля версий исходного кода и преобразовать исходный код в нужный вид, например выкачать исходники, находящиеся в определенном бранче git-репозитория, упаковать их в tar архив со строго определенным именем, сжатый gzip. В результате OBS сервер хранит только последнюю версию исходного кода, в остальном полагаясь на систему контроля версий, что, в вырожденном случае, позволяет хранить там весь исходный код, включая правила для сборки.

Частный сервер для проекта

Для организации частной сборочной системы рекомендуется [2] использовать специально подготовленные образы, доступные на `download.opensuse.org/repositories/openSUSE:/Tools/images/`.

В зависимости от того, какие задачи необходимо решать разработчикам, а так же от наличия ресурсов, можно выделить режимы использования локального сервера OBS:

– с подключением к внешним серверам;
– изолированный (сервер без связи с внешним миром).

В режиме подключения локальный сервер зависит от внешних серверов, предоставляющих базу для разработки. Этот режим подходит для компаний, которые хотят протестировать сборку своего программного обеспечения на наборе различных дистрибутивов.

Еще один плюс подключения к внешним серверам — минимизация инфраструктуры, что особенно хорошо подходит для стартапов, так как в OBS интегрировано множество возможностей для совместной работы, начиная от контроля версий исходного кода и заканчивая взаимодействием между разработчиками. Можно подчеркнуть, что отпадает необходимость иметь в штате системного программиста, так как в простейшем случае установить, настроить и использовать OBS сможет даже школьник.

В изолированном режиме придется самостоятельно создавать «базовые» проекты [3] и всю инфраструктуру. Кроме того увеличивается нагрузка на сопровождение, так как приходится самостоятельно отслеживать апдейты системы и изменения в настройках соответствующего проекта. Тем не менее такой режим позволяет создавать полностью автономную и независимую сборочную инфраструктуру, что является определяющим фактором с долговременной точки зрения.

Литература

[1] Пынькин Д.А. Обзор Open Build System. `http://winter.lvee.org/ru/articles/269`

[2] Build Service private installation `http://en.opensuse.org/openSUSE:BuildServiceprivateinstallation`

[3] Build Service private instance boot strapping. `http://en.opensuse.org/openSUSE:BuildServiceprivateinstancebootstrapping`

Использование DRBD в асинхронном режиме

Вячеслав Бочаров*

The effect on the disk subsystem access speed for different communication protocols «A», «B», «C» in construction of a failover cluster based on DRBD is reviewed.

При проектировании ИТ инфраструктуры все чаще возникает задача создания резервного датацентра, чтобы обеспечить непрерывность оказания услуг, даже в случае полного выхода из строя основного датацентра.

Основной и резервный датацентр принято размещать на территориально разнесенных площадках.

Описание протоколов

Технология DRDB использует 3 типа протоколов: **Протокол «A»** — асинхронный. Запись считается завершенной после записи данных на локальный диск и факта отправки данных на удаленный сервер. **Протокол «C»** — синхронный. Запись считается завершенной в момент получения подтверждения от удаленной системы о завершении записи на диск. **Протокол «B»** — промежуточный. Запись считается завершенной после записи данных на локальный диск и получения подтверждения о приеме (но не записи) данных удаленной системой.

Преимущества и недостатки протоколов обмена данными

Основные риски, которые мы рассмотрим, это нарушение целостности данных, снижение скорости обмена с дисковой подсистемой при синхронном протоколе «C» при нахождении второй дисковой подсистемы на удаленной площадке дата-центра. Как видно из описания протоколов, самым медленным является протокол

*Минск, Belarus

«C» , самым быстрым — протокол «A». Практических во всех обзорах указывается использование синхронного протокола, что вполне естественно для обеспечения целостности данных. Что произойдет, если использовать асинхронный протокол «A» либо промежуточный протокол «B», как это повлияет на скорость доступа? Мы задались целью выяснить, действительно ли это снижает надежность файлового кластера.

Результаты тестов кластера при применении различных протоколов

Тесты проводились на системе из двух серверов, находящихся в разнесенных дата-центрах на расстоянии 60 километров. Связь между ними осуществляется по оптоволоконному каналу на скорости 12 Мб.

Для построения кластера использовалась технология Heartbeat DRBD на базе дистрибутива CentOS. Диски SATA Maxtor & Seagate, файловая система ext4. Контрольные результаты тестирования доступа к локальной дисковой подсистеме 127–130 IOPS, что является нормой для таких дисков.

Результаты тестов скорости доступа к дисковой подсистеме при использовании протокола «C» — высокая надежность, невысокая скорость. Скорость доступа примерно равна 30–40 IOPS, проблемы с целостностью данных не наблюдается.

Результаты тестов доступа к дисковой подсистеме при использовании протокола «B» — достаточный уровень надежности, средняя скорость. Скорость повышается до 43–55 IOPS, что в принципе достаточно для не нагруженных файловых сервисов, хранилища виртуальных машин, для критичных сервисов очень даже неплохо.

Результаты тестов доступа к дисковой подсистеме при использовании протокола «A» — низкий уровень надежности, высокая скорость. Скорость доступа еще повышается до 50–57 IOPS, что в принципе тоже неплохо. В боевом режиме сбоев не наблюдается, но при искусственном стресс-тестировании (моделирование отказа обоих серверов одновременно с последовательным восстановлением) понадобилось ручное восстановление данных.

Выводы

Как показывает практика, использование промежуточного протокола «B» дает выигрыш в скорости обращения к дисковой подсистеме и не оказывает существенного влияния на надежность, что позволяет рекомендовать его для организации кластеров высокой доступности.

В тоже время репликация по протоколу «A» хотя и дает выигрыш в производительности, но оставляет высокими риски, связанные с потерями данных, а при использовании низкоскоростных соединений в случае «расщепления разума» («split-brain») может привести к отказу системы. Данная технология отлично подходит для разнесения на основной и резервный дата-центр сервисов, связанных с хранением данных, для повышения отказоустойчивости.

Применение промежуточного протокола оправдано, когда невозможно получить высокоскоростные линии связи.

Макраме из дистрибутивов: mkimage-profiles

Михаил Шигорин*

Once upon a time each distribution image was carefully crafted by hand thus becoming essentially unique. Fast forward to 21th century and we're getting besieged by countless variations of essentially the same thing, some base distro being customized as a desktop, L{A{N,MP},TSP} server, and a myriad of physical and virtual appliances. There's actually no need for full blown configuration forks and we should be able to describe the subtle (or significant) differences while letting the common base to stay, well, common. That's what mkimage-profiles was created for since day one.

Когда дистрибутивов было ещё меньше сотни, а создавались они вручную — вопрос управления конфигурацией особенно не стоял: «большие» универсальные дистрибутивы создавались ровно в одном варианте. Потом начались работы по локализации и поддержке различных архитектур, которые породили немало форков сами по себе. Затем пошли производные «под задачу». А теперь широко доступна ещё и виртуализация, которая сильно подняла интерес к небольшим специализированным образам, сконструированным под конкретную задачу — поскольку появилась возможность под каждую частность выделить отдельный контейнер или VM.

Как мы уже (`http://summer.lvee.org/ru/reports/LVEE_2011_13`) обсуждали ранее, раздвоение чего-либо (форк) может являться мощным средством как развития, так и уничтожения проектов — в зависимости от того, насколько приветствуется и удобно сведение результатов опять воедино (мерж).

На данный момент mkimage-profiles является моим очередным исследовательским проектом по части уменьшения излишнего дублирования общей части конфигурации и вспомогательного кода, необходимого для формирования образов дистрибутивов и виртуальных окружений. Он создан на основе опыта расширения и рефакторинга альтовского mkimage-profiles-desktop и семейства схо-

*Киев, Украина

жих профилей плюс создания набора installer-feature-*, а также более ранних наработок (spt-profiles-*).

Проект стартовал в августе 2010 года по мотивам очередного рефакторинга m-p-d; после первоначальных экспериментов по определению траектории тогда же осенью был опубликован черновик, а ещё через год состояние оформилось в достаточной степени для «официального» представления. Работаю над ним по большей части в две руки (`https://www.ohloh.net/p/mkimage-profiles`), хотя уже появился второй коммитящий и патчи либо кусочки кода приходят от ещё нескольких человек.

Поддерживается:

- наследование конфигурации на всех уровнях — от перечня пакетов до образа
- сборка гибридных ISO с LiveCD, RescueCD, инсталятором или их комбинацией
- сборка шаблонов виртуальных окружений OpenVZ
- архитектура i586/x86_64
- пакетная база ALT Linux 6+ (возможно бэкпортирование для более ранних)

В планах:

- сборка образов VM

Возможны:

- архитектура ARM и при востребованности — PowerPC
- более ранняя пакетная база ALT Linux, как минимум до 5+
- иные пакетные базы (проведены эксперименты с openSUSE 11.4 и CentOS 6)

Основы работы в Украинском Национальном ГРИД

Дмитрий Сподарец, Григорий Драган*

The paper introduces basic concepts and principles of GRID functioning illustrated by the Ukrainian National GRID, describes goals and tasks of High-Performance Computing & Free /Open Source Centre of I.I. Mechinkov Odessa National University, and explains steps required to register and work in GRID.

ГРИД-технологии позволяют объединять информационные и вычислительные ресурсы путём создания единой компьютерной инфраструктуры нового типа, которая обеспечивает глобальную интеграцию этих ресурсов на базе сетевых технологий и специализированного программного обеспечения промежуточного уровня, а также набора стандартизированных служб для обеспечения доступа к географически распределённых информационных и вычислительных ресурсов: компьютеров, кластеров, хранилищ данных.

Для развития и популяризации Украинского национального ГРИД (УНГ) на юге Украины, а также для увеличения собственных вычислительных ресурсов при Одесском национальном университете имени И.И. Мечникова был создан Центра суперкомпьютерных вычислений и свободного программного обеспечения. Сегодня Центр активно участвует в популяризации ГРИД-технологий, на его базе проводятся научные исследования по самоорганизации упорядоченных структур в дымовой плазме, он выполняет роль регионального регистратора УНГ, проводит различные семинары и конференции, в частности, осенью на конференции FOSS Sea 2012 будет сформирована отдельная секция, посвящённая HPC и GRID-технологиям. Кроме того Центр активно поддерживает OpenSource-сообщество, открыт для сотрудничества и воплощения в жизнь новых интересных совместных проектов.

Подключиться в УНГ и начать работать в нём достаточно просто. Рассмотрим основные шаги по регистрации и началу работы.

*Одесский национальный университет имени И.И.Мечникова, Одесса, Украина

Первым делом необходимо получить сертификат пользователя. Полная процедура описана по адресу `http://ung.in.ua/ua/certification/`.

Если вы желаете подключить свои вычислительные ресурсы в УНГ, то вам необходимо пройти ещё регистрацию ГРИД-сайта, которая описана здесь: `http://ung.in.ua/ua/join/`. На кластерах, подключаемых в УНГ, должно быть настроено программное обеспечение ARC. Одно из описаний, которое может помочь в настройке данного ПО, можно найти здесь: `http://clusterui.bitp.kiev.ua/howto/`.

Если вы желаете создать Виртуальную организацию со своими коллегами, то с инструкцией по её регистрации в УНГ можно ознакомиться здесь: `http://ung.in.ua/ua/vo_registration/`.

После того, как получен сертификат и настроено рабочее место, через которое вы будете входить в ГРИД, можно приступить к работе.

Первым делом проверяем наличия всех ключей в директории .globus:

- usercert.pem — цифровой сертификат пользователя (часть с открытым ключом). Ключ обязательно должен быть подписан Центром сертификации и быть добавлен в какую-то виртуальную организацию;
- userkey.pem — секретный ключ сертификата пользователя.

Доступ в среду GRID происходит под именем, содержащемся в сертификате, и контролируется с помощью специальной программы-посредника (электронной «доверенности» – proxy), которая создается на определенный ограниченный срок с помощью персонального ключа (userkey.pem) пользователя. Сервисные службы GRID могут выполнять любые действия, только если располагают копией такой доверенности.

Данная доверенность создаётся при помощи команды grid-proxy-init и действительна на протяжении 12 часов (для увеличения срока можно использовать параметр параметра -hours). Уничтожить доверенность до истечения ее срока можно с помощью команды grid-proxy-destroy. Для получения информации о выданной доверенности используйте команду grid-proxy-info с параметром –all, которая выдает полную информацию о доверенности. Система отправки заданий в среду GRID представляет собой набор команд для направления заданий, проверки их статуса и получения результа-

тов. В отличие от локальных систем управления заданиями (таких как PBS, LSF и др.), система отправки заданий GRID:

- обеспечивает единообразный доступ к ресурсам на различных узлах сети;
- автоматически согласовывает требования, необходимые для выполнения задания, с имеющимися ресурсами.

Как и в кластерных системах, пользовательская команда запуска содержит имя скрипта, запрос ресурсов в котором специфицируется в виде строки XRSL.

Команды управления заданиями имеют следующий вид: ngsub <job.jdl> – команда отправки файла с описанием задания; ngstat <jobID> — запрос статуса задания по его идентификационному номеру jobID (PREPARING — подготовка к выполнению, INLRMS:Q — Ожидание освобождения ресурса в очереди LRMS, INLRMS:R — Выполнение задачи, FINISHING – Завершение задачи, FINISHED – Задание завершено, PURGED – Удалено, FAILED); ngkill <jobID> — отмена задания. Файл с описанием задания создается с помощью языка описания заданий (Extended Resource Specification Language, XRSL) и содержит необходимые входные данные, требования к ресурсам и сведения о том, куда должны быть записаны результаты обработки задания. Типичный xrsl-файл имеет вид:

```
&
(* this is comment *)
(executable=ex.sh)
(executables=example1)
(inputFiles=(example1 ""))
(arguments=’’100000000’’ ‘‘13’’ ‘‘0.324’’)
(stdout="out.txt")
(stderr="err.txt")
(outputFiles=("out.txt" "")("err.txt" "")("sol.ps" "")
("err.ps" "")("data.txt" ""))
(gmlog="gridlog")
(jobname="Example")
(cputime=20)
(middleware>="nordugrid-arc-0.3.24")
```

Больше информации об УНГ, учебные материалы и анонсы можно найти на сайтах `http://ung.in.ua`, `http://infrastructure.kiev.ua`, `http://grid.nas.gov.ua/`, `http://grid.bitp.kiev.ua`, `http://hpcandfosscenter.od.ua`.

Software security

Алексей Чеусов*

The article discusses system techniques and methods for securing the software in UNIX world.

Язык программирования C, на десятилетия определивший успех операционных систем класса UNIX, на протяжении последних лет все чаще становится источником проблем в области безопасности программного обеспечения. Появившись как язык системного программирования, язык C широко применяется также и для разработки прикладного ПО, что, ввиду принципиальной «небезопасности» этого языка, приводит к многочисленным проблемам, таким как получение доступа к системе злоумышленником, повышению привилегий процесса до уровня суперпользователя, rootkit-ам, нестабильности работы ОС и т.п. То же относится и к языку программирования C++.

В ближайшее время вряд ли стоит ожидать значительного падения популярности этих языков в области разработки как системного, так и прикладного ПО, поэтому актуальной становится разработка средств и методов борьбы с перечисленными выше проблемами менее радикальными, чем смена языка программирования, средствами. Таким средствам и технологиям, существующим и развивающимся в различных UNIX-подобных системах, посвящен настоящий краткий обзор.

К числу рассматриваемых технологий защиты можно среди прочих отнести следующие:

- безопасные функции strl{cat,cpy} для работы со строками,
- SSP (stack smashing protection) — защита от переполнения стека,
- ASLR (address space layout randomization) — рандомизация базовых адресов сегментов виртуальной памяти процесса,
- PIE (position independent executable) — позиционно-независимые исполняемые файлы,
- hardened chroot — усиленный chroot,

*Минск, Беларусь

– W^X — защита исполняемых сегментов памяти от записи и записываемых (стек и данные) от исполнения,
– PaX MPROTECT — защита mprotect(2),
– PaX Segvgard — защита от перебора адресов сегментов памяти приложения,
– Information filtering — сокрытие информации, доступной пользователям и процессам,
– per-user /tmp directory — размещение подкаталога временных файлов в домашней директории пользователя,
– SUID/SGIG executables — избавление от исполняемых файлов с установленным битом SUID,
– PAM tcb — замена PAM unix как средство избавления от бита SUID у passwd(8)
– capsicum — расширение POSIX API для обеспечения лучшей безопасности в системе UNIX.
– FUSE, PUFFS — подсистемы для реализации файловых систем в пространстве пользователя
– Микроядерные ОС — класс операционных систем, в которых основные сервисы работают на уровне пользователя, на уровне ядра же работает лишь самое необходимодое, за счет чего достигается надежность и безопасность
– RUMP — подсистема для запуска ядерного кода в пользовательском приложении
– SE Linux — подсистема контроля доступа в Linux
– kauth(9) — подсистема авторизации ядра NetBSD
– jail — система изоляции и виртуализации FreeBSD

Обзор Open Build Service

Денис Пынькин*

Open Build Service (OBS) is an open and complete distribution development platform. It provides an infrastructure to create and release open source software for openSUSE and other Linux distributions easily on different hardware architectures. The article describes architecture and usage of private OBS instances.

Введение

Зачастую коммерческие проекты базируются на открытых платформах, таких, как различные дистрибутивы Linux. Это означает, что для обеспечения поддержки своего продукта вендору часто необходимо самостоятельно заниматься поддержкой открытых проектов, входящих в его продукт.

Даже при условии сокращения количества поддерживаемых пакетов до минимума, необходимого для работы коммерческого программного обеспечения, инженерам приходится заниматься поддержкой сотен сторонних пакетов и приложений.

Системы автоматической сборки значительно облегчают жизнь разработчикам, вынужденным обеспечивать платформу для разработки как открытого, так и коммерческого программного обеспечения. Одна из лучших по совокупности параметров и возможностей система автоматической сборки общего назначения — это Open Build Service, умеющая работать со множеством различных дистрибутивов.

Архитектура

Open Build Service можно разделить на серверную часть и клиентскую [1]. OBS поддерживает сборку следующих целей:

- rpm/spec;
- deb/dsc;
- KIWI image.

*Минск, Беларусь

Серверная часть

Основные задачи, которые решает серверная часть:

- обеспечивает доступ к исходному коду приложений;
- предоставляет ресурсы для сборки пакетов;
- предоставляет инфраструктуру для публикации пакетов;
- обеспечивает взаимодействие между разработчиками.

В свою очередь OBS-сервер состоит из двух частей:

- front-end:
  - публичный API для различных клиентов;
  - доступ к исходникам;
  - доступ к логам и результатам сборки;
  - доступ к собранным пакетам;
  - контроль процесса сборки пакетов;
  - управление пользователями;
- back-end:
  - хранилище исходного кода;
  - сборочная «ферма»;
  - запуск сборки в соответствующем окружении;
  - репозитории собранных пакетов;
  - доступ к логам и результатам сборки.

Клиентская часть

«Официально» поддерживаются 2 клиента: web (входящий в фронтэнд сервера) и утилита **osc** для работы из командной строки. Кроме того существуют и другие — например для Android'а или графический клиент, написанный на **C#**.

Клиент **osc** позволяет произвести сборку пакета или дискового образа на локальной машине.

Как для локальной сборки, так и на сборочной «ферме» используется одинаковый набор скриптов **build**, в задачу которого входит связь с сервером, получение необходимых бинарных и/или пакетов с исходным кодом, организация изолированной сборочной среды (поддерживается изоляция с помощью chroot, kvm, xen и др.) и, при необходимости, отправка результата на сервер.

Проекты

Проекты — это организационная еденица в OBS, представляющая собой единую площадку с общими правилами сборки для всех пакетов.

Проект включает в себя:

- конфигурацию проекта, включая макросы и определения используемые в сборочной среде;
- пакеты — исходный код и правила сборки программного обеспечения и/или дискового образа;
- сборочные цели — дистрибутивы, на базе которых OBS будет пытаться собрать пакеты, входящие в проект.

Литература

[1] Overview about the idea of the Build Service and its architecture. `http://en.opensuse.org/images/8/8e/FOSDEMBuildService.pdf`

[2] Build Service private installation `http://en.opensuse.org/openSUSE:BuildServiceprivateinstallation`

[3] Build Service private instance boot strapping. `http://en.opensuse.org/openSUSE:BuildServiceprivateinstancebootstrapping`

Обзор архитектуры и возможностей GObject Introspection

Антон Васильев*

The article gives an overview of GObject Introspection architecture and use cases. One of GObject Introspection main goal is to share binding infrastructure to make binding-friendly applications and libraries. The introspection project solves this task by putting all metadata inside the GObject library itself, with annotations in comments. This decreases duplicated work for binding authors.

История и цели проекта

GObject Introspection — это проект, который возник для решения проблем с созданием высокоуровневых языковых привязок (bindings) для библиотек GNOME. Долгое время все языковые привязки поддерживались вручную, и зачастую значительно отставали от библиотек [1].

GObject Introspection — это технология, позволяющая создавать автоматические языковые привязки для библиотек, использующих GLib и GObject. Привязки могут создаваться как на этапе исполнения (через FFI) так и на этапе сборки (с использованием компилятора привязок для конкретного языка).

Для извлечения метаданных из библиотек используется система аннотаций в комментариях к функциям/методам.

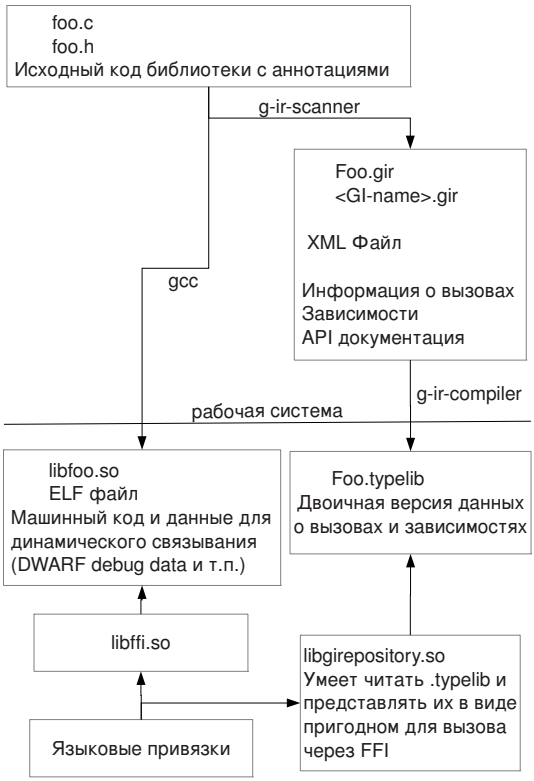
Метаданные хранятся в бинарном формате typelib, обеспечивающем быстрый доступ к библиотечным функциям через FFI. Для создания typelib-файла используется промежуточный формат GIR (GObject Introspection Repository). GIR основан на XML, содержит информацию о вызове функций и документацию API.

Архитектура и примеры использования

Этап сборки:

Я создал репозиторий [2] с примерами использования библиотеки, базирующейся на GObject, через автоматические привязки для разных языков.

*Минск, Беларусь



Рассмотрим пример, использующий язык Vala для создания библиотеки и обращения к этой библиотеке из Ruby:

```
namespace ValaObject {
public void say_hello_to(string lang)
{
print(@"I love You, $lang!!!\n");
print("-- Vala\n\n");
}

public class ValaClass : Object {
public string name = "Vala Class";

public string append_to_name(string suffix) {
```

```
return "%s %s".printf(name, suffix);
}
}
}
```

Ruby использует метаданные из GIR через библиотеку gir_ffi:

```
require 'gir_ffi'

GirFFI.setup(:ValaObject) # Создание объекта

ValaObject.say_hello_to('Ruby')

class MyValaClass < ValaObject::ValaClass
  def append_to_name(suffix)
    super + ' (subclassed) ' + suffix
  end
end

instance = MyValaClass.new
puts instance.append_to_name("called from Ruby")
```

Существующие привязки

Список существующих привязок для GObject Introspection выглядит следующим образом:

- Vala — Родная поддержка GIR.
- Genie — Родная подержка GIR.
- PyGObject — привязки для Python
- Gjs — Javascript (spidermonkey)
- Seed — Javascript (JSCore, WebKit JS engine)
- node-gir — Node.js (V8 Engine)
- ruby-gir-ffi — Ruby
- gobject-for-php — PHP
- lgob — Lua (этап компиляции)
- lgi — Lua (этап исполнения)
- GTK2-Perl/Introspection — привязки для Perl
- guile-gir — Scheme (guile)
- sbank — Scheme (Ikarus, Ypsilon)
- JGIR — Java/JVM (этап компиляции, через typelib)

- GObjectIntrospection/GObjectConsume — C++, Qt (этап компиляции)
- factor-gir — Factor
- gogobject — Go (этап компиляции)
- cl-gobject-introspection — Common Lisp
- cl-gir — GIR for Common Lisp (в процессе)
- haskel-gi — Haskell (в процессе)
- mono-introspect — Mono
- ocaml-gir — Ocaml (этап компиляции)
- gir2pascal — Pascal

Литература

[1] GTK+ Language Bindings. `http://www.gtk.org/language-bindings.php`

[2] GObject for your favorite language. `https://github.com/antono/vala-object`

Managing over 9000 nodes with Puppet

Василий Михаленя*

Puppet is an open source tool to manage configurations. The article describes its features, advantages, scalability, common and best practices. «SSH in a for loop is not a solution», says Luke Kanies, Puppet developer.

Что такое puppet

Puppet — клиент-серверное приложение для удобного распространения конфигураций — состоит из puppetmaster'а, сервера хранящего конфигурации, и puppet agent'а, работающего на конфигурируемом сервере. Puppet написан на ruby, распространяется под лицензией Apache и содержит возможности расширения функциональности с использованием ruby. Проект имеет ruby в качестве единственной зависимости, работает на Linux, Solaris, BSD, Mac OS X, поддерживает Microsoft Windows. Puppet можно использовать как для документирования конфигурации на одном сервере, так и для легкого управления конфигурацией парка серверов в гетерогенной среде. Puppet используют такие проекты как wikimedia, twitter, digg, sugarcrm и многие другие.

Основные примитивы. Puppet или bash

Описание конфигураций происходит на своем собственном декларативном языке. Декларативный язык позволяет описать желаемое состояние системы в зависимости от набора фактов — описание «как именно делать», как правило, не требуется. Если применять некоторую конфигурацию на серверах с разными ОС (или, например, разными дистрибутивами Linux), используя bash скриптинг, получаем трудноподдерживаемый код. Puppet же позволяет быть уверенным не только в том, что изменения были применены единоразово (случай shell-сценариев), но и что текущее состояние

*Минск, Беларусь

соответствует описанному. Простота языка позволяет использовать манифесты puppet как документацию конфигурации систем.

К числу основных понятий языка относятся следующие:

- ресурсы — file, package, user, exec, cron и т.д.
- классы — объединения ресурсов и зависимости между ними
- модули — самостоятельные наборы классов, например, для конфигурации определенного сервиса. Модули независимы и могут распространяться отдельно.
- факты — facts — пары «ключ-значение», которыми оперирует puppet для выбора конфигурации (например, hostname => mars, is_virtual => false, operatingsystem => Ubuntu, processorcount => 8).
- ноды — nodes — соответствие имени сервера (ноды) и набора классов, которые будут к ней применены. Возможно хранение нод в LDAP или использование внешнего классификатора нод — произвольного скрипта.
- манифест — конфигурационный файл puppet, в котором описаны ресурсы, классы, модули или ноды

Управление парком машин от 2 до бесконечности

Очевидно, что преимуществ от автоматизации применения конфигурации тем больше, чем больше систем и сервисов на них настраиваются с помощью puppet. Но некоторые вещи полезно автоматизировать, если имеются хотя бы 2 ноды:

- синхронизация authorized_keys
- синхронизация пользователей, их настроек (bashrc, vimrc ...)
- синхронизация правил брандмауэра
- и т.д.

В гетерогенной среде приходится учитывать «факты» в своих конфигурациях. Деплоймент новой или вышедшей из строя ноды происходит в течение нескольких минут при достаточной степени полноты описания конфигурации. Существует огромное количество написанных модулей для практически любых сервисов (их список можно получить поиском, например, на github). Деплоймент или применение каких-либо изменений для 2 или 2000 нод практически не отличаются по трудоемкости при грамотном подходе.

Как работает puppet

Puppet agent раз в 30 минут запрашивает конфигурацию у puppetmaster:

1. Компиляция манифеста происходит на сервере, результатом являются «базовые хеши» и никакой валидации данных.
2. Инстанциирование — конвертация «базовых хешей» и массивов, полученных при компиляции, в объекты puppet-библиотеки. Валидация входных данных происходит на этом этапе.
3. Конфигурирование — сравнение каждого описанного ресурса с реальным положением дел, и внесение соответствующих изменений при необходимости.

Также существует подход nodeless (masterless). При данном подходе, манифеста с описанием нод не существет — мастер не нужен, вся конфигурация копируется на ноду, а настройка опирается исключительно на «факты».

Как масштабировать puppet

До 100 нод со стандартным периодом в 30 минут могут успешно работать с единственным puppetmaster-сервером (WEBRick Ruby-based HTTP). Дальше начинаются проблемы. Выход – связка mod_passenger + apache + rack. При данной схеме возможно распределение нагрузки на несколько серверов. Часто используют подход со splay time — размазывание по времени нагрузки на мастер — при котором все ноды не должны обращаться к мастеру одновременно. Можно также использовать подход 1 мастер на площадку (сайт), с синхронизацией между площадками, например, через тот же git.

Опыт работы в команде

Опыт работы с puppet подсказывает следующие практики:

– Использование environments, различных наборов манифестов для разработки и продакшна, когда выбор environment происходит на стороне клиента.
– Использование системы контроля версий — например, git. Удобным оказывается использование двух различных веток или репозиториев, для разработки и production соответсвенно.

Применение iSCSI RAID LVM для создания частного облачного хранилища данных

Вячеслав Бочаров*

Creation of a cloud storage system is presented, nodes of which are workstations based on existing technologies (RAID, iSCSI, LVM). Lowcost file storage from nothing and its evolution perspectives are discussed.

Потребность пользователей информационных систем в дисковом пространстве для надежного хранения данных приводит к необходимости развертывания систем хранения данных с достаточно высокой стоимостью приобретения и эксплуатации. В тоже время дисковое пространство рабочих мест пользователей информационной системы остается незадействованным, и не используется в инфраструктуре предприятия. Фактически, такое нужное дисковое пространство «рассыпано» у нас под ногами, и его можно заставить работать на благо нашей инфраструктуры — все нужные технологии у нас уже есть.

Для этого необходимо выполнить следующие шаги:

1. Предоставить часть дискового пространства рабочей станции в общее пользование — для этого мы используем iSCSI taraget. При среднем размере диска рабочей станции в 300 ГБ реально используется не более 150, поэтому еще столько же можно отдать в облако. Если на предприятии 100 рабочих станций — результат будет 15 Тб «сырого» пространства на 100 дисках.
2. Собрать предоставленное пространство в единый массив на сервере.

Реализация перечисленных действий требует решения нескольких проблем. В частности, используя рабочие станции пользователей как поставщики дискового пространства, мы должны учитывать следующее: то, что для СХД является нештатной ситуацией (выход из строя нескольких дисков), для нашего облака — обычное

*Минск, Беларусь

явление. Поэтому избыточность должна быть большой. Я использую конфигурацию RAID-10 через mdadm на CentOS. Это позволяет контролировать состояние RAID-массива, менять вышедшие из эксплуатации узлы ISCSI, оперативно настраивать массив.

Также необходимо большое количество дисков HOT SPARE, и нужна система их активации. Это повышает процент потери пространства (в опробованной конфигурации до 40% уходит на резервирование), но зато это пространство практически взято из воздуха.

Еще один проблемный момент–окончание рабочего дня, когда большая часть узлов выключается. Можно было бы реализовать схему со спящим режимом, но тогда возрастает энергопотребление организации. Проще рассматривать нерабочий период как прерывание работы контроллера системы и выключение контроллера по расписанию либо по достижению определенного порога выхода из сети iSCSI-дисков, что легко реализуется скриптами bash.

1. Вполне естественно будет упомянуть необходимость использования в сети 1Gb и Jambo Frame. В отношении скорости такой системы можно заметить, что данная конфигурация дает показатели IOPS 75/80–значения, сравнимые с показателями SATA 7200 HDD.
2. Оперативное изменение размера облака решаемо штатными средствами–LVM. Благодаря использованию LVM мы имеем возможность расширять существующее дисковое пространство просто добавлением еще одной группы RAID.

Хотя предлагаемое решение не предназначено быть заменой высоконагруженным решениям, оно может вполне успешно использоваться для системы файлового хранения — как известно, самой прожорливой. Стоимость данного решения чрезвычайно низка. Оно позволяет эксплуатировать ресурсы предприятия полностью. Развитие представленного направления в облачных СХД имеет большое будущее. Благодаря таким подходам надежные системы хранения станут не уделом дорогих систем корпоративного уровня, а будут доступны рядовым пользователям. При разработке специализированной системы с резервированием контроллеров, оптимизацией распределение ресурсов между дисками, повышением отказоустойчивости и уменьшением потерь дискового пространства, возможно даже появление сообществ, позволяющих создавать дисковое пространство, разделяемое не только между пользователями одной организации, но и между членами самоорганизующихся сообществ.

NoSQL и Async web фреймворки не нужны

Максим Мельников*

Its a complicated task to write a web-based highload project, and finding best instruments for it is even harder. NoSQL and different async web frameworks becomes widely used. There is a problem with them, as they are promoted as the solution with cost of higher entrance level, offering ability to do everything in one way, in one place. But, in fact, classic, old-fashion way still rocks. While using layering and with the best instruments for each level you can easily get maximum performance and scaleability

Высокоронагруженный web-проект должен не только хорошо держать нагрузку, но и эффективно потреблять ресурсы. Разрабатывая такой проект, необходимо понимать какого рода проблемы и задачи на каком уровне должны решаться. Вот неполный список задач, которые необходимо решить:

1. поддержка огромного количества http подключений одновременно
2. кэширование данных и генерации сложного контента
3. система должна простаивать когда нет нагрузки
4. при росте количества запросов, должно расти потребление ресурсов. а не время ответа
5. потребление ресурсов должно достигнуть 100% при достижении некоторой максимальной нагрузки
6. время обработки запросов пользователей должно быть минимальным
7. время передачи ответа пользователю должно быть минимальным
8. система должна работать стабильно в случае отказа сторонних сервисов, временный отказ должен проходить незаметно для пользователя.

Различные Async(NodeJS, Tornado, Twisted) и NoSQL-решения (Redis, MongoDB, Cassandra) кажутся найлучшим выбором, пока

*Минск, Беларусь

вы делается ваше web как одно большое целое. Если же немного подумать, большинство проектов разбиваются на кучу маленьких, каждый из которых легко и непренуждённо может быть реализован классическим подходом, оставаясь при этом маштабируеймым и высокопроизводительным. А HTTP-протокол, благодоря отсуствию состояний — становится лучшим помошником. Что касается баз данных, ввод грамотно реализованных уровней кэширования невелирует проблему высокий нагрузок на базы данных до момента достижения каких-то фантастических нагрузок, с которыми почти никто на деле и не сталкивается.

Для разработки worldoftanks.ru используется расширенный LAMP (Linux Apache MySQL PHP) подход — LNAMMRP (Linux nginx Apache memcached MySQL RabbitMQ Python).

Основные компоненты используются следующим образом:

– nginx: handling http-сессий от пользователя
– apache: управление подконтрольными рабочими процессами
– memcached: кэширование без проблем
– RabbitMQ: сервер очередей для асинхронной обработки медленных задач
– Python: реализация бизнесс требований (Django)

Это позволяет не тока держать высокие нагрузки, но и использовать накопленный всем миром опыт не оставаясь с проблемой один на один.

Написание консольных утилит и демонов на PHP

Андрей Синицын*

Usage of the PHP framework at developing console applications is reviewed, mainly targeted at creating web application background service. Specific features of PHP CLI programming and input/output redirection are considered as far as ones of multiprocess applications

PHP занял и прочно удерживает пальму первенства. По статистике более 60% сайтов написаны именно на PHP. К сожалению, эта популярность и низкий порог вхождения породили массу некачественного кода, некачественных программистов, а также затмили собой массу достоинств этого, неплохого в общем-то, языка.

Почему PHP?

В наше время javascript используется как серверный язык, perl в стадии затянувшегося угасания, python — в сущности, аналог basic, а ruby известен своей медлительностью. Но вне зависимости от стереотипов, на всех этих языках создаются прекрасные приложения. И в рамках отказа от стереотипов я попробую раскрыть PHP с неожиданной стороны.

На PHP есть не только Wordpress, Joomla и др.: такие гиганты, как facebook или vkontakte, тоже написаны на нем. Помимо web-приложений, на PHP пишутся web-сервера, обвязка для embedded-приложений, и даже была попытка переписать на PHP весь Linux init.

Помимо этого безумия есть также биндинги к GTK и Qt, что, теоретически, позволяет создавать на PHP GUI-приложения. Но на практике все это сыро и работает довольно неустойчиво.

*Тула, РФ

Зачем?

Как правило, в web-приложениях довольно много действий приходится выполнять в фоновом режиме. Это задачи по обслуживанию БД (например, чистка и синхронизация), генерация статистики и много всего другого. Если проект использует как основу какой-то фреймворк, разумно решать с его помощью также и эти задачи. И тут можно вспомнить, что PHP умеет работать в режиме командной строки (CLI), что избавляет от необходимости искать новых разработчиков или просить сисадмина написать нужные скрипты. Вы получаете интеграцию с основным приложением и решаете задачи без лишних усилий. Однако следует помнить об отличиях.

Как?

Нужно помнить о том, что в CLI-режиме у вас больше нет суперглобальных массивов, зато есть аргументы командной строки. Для облегчения работы с ними PHP предлагает свой стандартный набор: $argc, $argv для работы с параметрами командной строки, readline для облегчения интерактивного взаимодействия с пользователем.

Можно использовать нужный she-bang, убрать расширение .php, сделать файл исполняемым и поместить его в директорию, присутствующую в $PATH: в этом случае пользователи вообще не смогут отличить PHP-сценарий от «родных» утилит Linux.

При написании консольных утилит следует помнить о потоках ввода/вывода. В распоряжении программиста есть STDIN для ввода, STDOUT для вывода результатов работы и STDERR для сообщений об ошибках. Также необходимо помнить о кодах возврата (число, которое ставится после команды exit и становится доступно шеллу после окончания работы сценария). Код возврата может пригодиться, например, когда скрипт используется в конвейерах.

Нюансы

Следует помнить о быстродействии. Байт-код в CLI-режиме не кэшируется, и все I/O вызовы (инструкции include) будут постоянно повторяться. Также нужно помнить о ширине терминалов. Хорошим тоном считается форматировать строки с использованием управляющих символов (sprintf(), например) и не делать вывод

длиной более 80 символов. Важен и объем информационных сообщений. Если скрипт работает пару секунд и тщательно докладывает о каждом своем шаге, это раздражает. Используйте ключ -v для такого режима работы, а по умолчанию ведите себя «тихо».

Для разбора аргументов есть функция getopt(), представляющая из себя весьма мощный инструмент.

Не стоит забывать про обработку сигналов ОС. Это весьма пригодится при написании сервера, который работает в фоне, т. к. общаться с ним придется при помощи этих самых сигналов.

standalone-сервер

По сути — это частный случай CLI-утилиты, которая работает в фоне, ждет запросов от пользователя и в ответ выполняет некие действия. Технология «клиент-сервер» хорошо известна. В PHP есть необходимый набор функций для работы с сокетами, что позволяет занять определенный порт и ждать подключений на нем.

Сервер может быть однопоточным и многопоточным. В случае с многопоточностью основной процесс (master) принимает подключение, делает вызов fork(), передает управление порожденному процессу (worker) и продолжает ожидать подключений. Например, в нашем проекте Flirteka.Ru мы с помощью подобного демона, написанного на PHP, выполняем асинхронную загрузку и обработку изображений.

Здесь следует обратить внимание на набор функций pcntl_* (сокращение от Process Control). Эти функции доступны только в среде UNIX. В них реализовано все необходимое для комфортного управления процессами. Работа с POSIX-функциями позволяет управлять сигналами, дочерними и родительским процессом.

Особенности работы с fork()

Действия с ресурсами, такие как, например, коннект к БД и открытие файловых дескрипторов, следует выполнять в дочерних процессах. Открытие коннекта в родительском процессе и передача его в fork приводит к непредсказуемым последствиям. Нужно также помните о кодах возврата и корректном завершении дочерних процессов. И наконец, имеет смысл зарегистрировать свои обработчики сигналов при помощи pcntl_signal.

Как получить практический опыт работы с открытым программным обеспечением

Викентий Лапа*

How to obtain practical experience with FOSS products? It is one of difficult questions that appear in front of a novice when he tries to get a job. The article describes author's personal experience in freelance as one of possible solutions as globalization gives novice or user with little experience an opportunity to improve his skill and to obtain real practice with FOSS products. Freelance tendencies & types of tasks related to FOSS in hosting and system administration are reviewed, as far as some useful tips to get first remote job.

Введение

Источником для данного материала послужил экономический кризис, вынудивший автора искать дополнительные источники доходов. Одним из требований, предъявляемых к новой работе, было использование открытого ПО, и среди возможных вариантов был опробован вариант удаленной работы — фриланс. Результаты позволили проанализировать возможности работы в таких проектах, связанной с открытым ПО.

Классификация проектов

На рынке удаленной работы можно встретить проекты с использованием открытого ПО, но частота их появления варьируется, поэтому проведена попытка классификации по востребованности в промежуток времени. Разделим открытое ПО на три группы по степени востребованности:

1. минимум одно предложение в неделю;
2. одно предложение за месяц;
3. все остальные.

*Минск, Беларусь

Классификация проводилась на основе встроенного в веб-интерфейс поиска по ключевым словам. Охват свободных проектов оказался весьма широк, но следует учитывать, что он ограничен областью работ, в основном связанных с системным администрированием.

К первой группе относятся наиболее востребованные проекты: Linux-дистрибутивы CentOS, Ubuntu, Debian, веб-сервер Apache, Tomcat, базы данных MySQL, PostgresSQL, SQLite, языки программирования PHP, Ruby, Python, Perl, а также другие проекты — Wordpress, Drupal, Mediawiki, Mantis, Redmine, RubyOnRails, Magento, Nagios.

Во вторую группу попали такие ОС, как FreeBSD, OpenBSD, Gentoo, а так в виде исключения исключения представители проприетарного мира UNIX — AIX и HP-UX. Среди веб-серверов — nginx, lighttpd, Varnish, HAProxy, базы данных CouchDB, MongoDB, HBase, языки программирования Shell, Lua, проекты VirtualBox, Xen, OpenVZ Bugzilla, RequestTracker.

К третьей группе относятся редкие проекты, например OpenStack, puppet, Chef, Munin, Monit, Zabbix, Aegir, mod_speed, mod_security, Conky, MenuetOS, язык программирования Lisp и его модификации Scheme и Arc.

Особенности удаленной работы

Для того, чтобы начать работать удаленно, нужно выполнить ряд простых шагов. Это подготовка, поиск интересующего задания, предложение работодателю и переговоры, а также собственно выполнение работы и завершение проекта.

Стадия подготовки включает в себя регистрацию на бирже удаленной работы, заполнение портфолио. Также на этой стадии автору пришлось установить на своей рабочей машине некоторое количество бинарных блобов. В частности, к ним относится специальное приложение для отслеживания времени–например, oDesk Team Manager, которое поддерживает дистрибутивы Ubuntu, Fedora, Suse, Arch, а пользователям Gentoo или ОС из семейства BSD нужно приложить дополнительные усилия по установке. Также для переговоров потребуется Skype. Еще одно из приложений, TeamViewer, позволяет получить доступ к удаленному рабочему столу, и его удобно иметь установленным на всякий случай.

На стадии подготовки следует пройти тесты по своей области компетенции. Это позволяет оценить свои знания и выделить себя из списка других претендентов. Так, например, из 10907 Linux Developers тест прошли только 1300. Тесты могут быть как на самой бирже, так и на независимом сайте тестирования, например Brainbench.

Также некоторые заказчики просят подтвердить свое умение через другие сайты. Это могут быть примеры кода на github либо ответы на сайтах обмена знаниями, таких как stackoverflow или forum.linux.by (давая ответы на вопросы, вы также приобретаете опыт).

На стадии поиска следует запастись терпением и пытаться делать предложения по тем проектам, в которых используется интересующий вас открытый продукт. Есть особое время, когда шанс получить ответ выше, за счет того что количество конкурентов меньше — например в ночное время в Индии, либо в пятницу по религиозным причинам, либо в выходные дни и праздники. Необходимо учитывать разницу во времени с географическим положением заказчика, начало и конец рабочего времени.

На шаге, когда вы делаете предложение заказчику, возникает пауза — период ожидания ответа. Паузу имеет смысл заполнить приобретением знаний: либо чтением документации, либо поиском и составлением плана решения. Но следует учесть, что ожидание может затянуться, либо ответа не последует никогда. Поэтому имеет смысл делать предложения по нескольким проектам, это увеличивает шансы на ответ.

Шаг выполнения работы как раз и добавляет опыт в реальных боевых условиях.

Относительно самого последнего этапа необходимо сделать важное замечание, которое прямо не относится к проблеме получения опыта, но накладывает ограничения на легальность удаленной работы: участникам из Беларуси перед выводом денег следует ознакомиться со статьей 12.7 Кодекса Республики Беларусь об административных правонарушениях «Незаконная предпринимательская деятельность».

Способы тестирования web-ресурсов на уязвимости. Достоинства и недостатки

Дмитрий Никипелов*

Every day world is exposed to new methods of attack on web resources, and therefore technical testing for vulnerabilities becomes very important. The article discusses ways to test for vulnerabilities, including the analysis of their advantages and disadvantages.

Наиболее простым способом воровства конфиденциальной информации является поиск уязвимостей интернет-ресурсов и, в частности, web-приложений. И пользователи и разработчики понимают необходимость борьбы с техническими уязвимостями. Каковы пути решения и как выбрать оптимальный способ обеспечения безопасности?

Прежде всего необходимо позаботиться о безопасности платформы (хостинга), т.к. если будут выявлены уязвимости в структуре платформы, говорить о безопасности и тестировании приложений не имеет смысла. Следует иметь в виду, что некоторые хостинг-провайдеры имеют в ассортименте услуг предоставление защищенной базы для размещения приложений (например, технология «защищенный хостинг» от Hoster.by). Подобные технологии предполагают не только первичную подготовку хостинга, но включают в себя и сервисы по тестированию и анализу уязвимостей размещенных ресурсов.

Если говорить непосредственно о тестировании приложений, имеет смысл выделить следующие способы тестирования:

– *принцип «белого ящика»* — тестировщику заранее известно все, включая исходный код. В этом случае проводятся проверки исходного кода на наличие потенциально опасного кодирования — статически по сигнатурам и динамически на уровне разработчиков (когда модули подвергаются тестированию на ввод некорректных данных). Для открытого ПО этот

*Минск, Беларусь

метод весьма актуален и позволяет выявить большое количество уязвимостей. Недостатком является статичность метода — при отсутствии сигнатуры в базе проверка не будет иметь эффекта. Так же при использовании сложного синтаксиса потенциальная уязвимость может быть просто не замечена. Данный метод эффективен для выявления ошибок обработки данных, но не позволяет взглянуть на ресурс глазами злоумышленника. Такое тестирование проводится средствами среды разработки или модулями анализа кода — например RATS, Yasca (open source).

– *принцип «серого ящика»* — тестеру предоставляются все полномочия кроме непосредственного доступа к серверу. Проверки происходят на предмет повышения полномочий, выявления ошибок обработки данных, реакции на некорректные данные. Данный способ позволяет оценить корректность работы ресурса при вводе разных исходных данных, но, как и предыдущий метод, не позволяет инсценировать атаку на ресурс. Примером свободного инструмента, реализующего данный принцип, можно считать Whisker/libwhisker.
– *принцип «черного ящика»* — тестировщик иммитирует действия потенциального злоумышленника любыми доступными ему способами. При таком способе тестирования проверяется как наличие уязвимостей приложения, так и действия персонала по противодействию атаке и реагированию на инцидент. При проведении подобного тестирования действиям персонала необходимо уделять особое внимание, так как в некоторых случаях способность адекватно реагировать на инцидент имеет решающее значение. Недостатком данного метода является то, что тестер должен иметь весьма высокую квалификацию для достижения эффективных результатов, а тестирование занимает достаточно продолжительное время. В какой-то степени эти недостатки можно компенсировать способами, представленными ниже. Для автоматизации подобного тестирования можно применить мощный open source инструмент Nikto.
– *Honey pot (бочонок с медом)* — так называют web-ресурс, не несущий никакого практического смысла, но вокруг которого раздута шумиха о его важности и значимости. Реально предназначен для сбора информации о возможных средствах и сценариях осуществления атак. Фактически злоумышленники

и являются теми самыми тестерами, причем их количество и квалификация напрямую зависит от того пиара, который создан вокруг web-ресурса. Опыт показывает, что такой способ является весьма эффективным при создании систем противодействия вторжениям и тестировании новых ресурсов.

– *Сбор данных с работающей системы* — метод, при реализации которого осуществляется не технический анализ уязвимостей, а сбор данных на блогах и социальных сетях о реализованных или готовящихся атаках. В данном способе самое важное–это быстро собирать информацию и оперативно реагировать на возникающие ситуации. Способ очень эффективен для игровых ресурсов, где основной целью атак является не нарушение работоспособности ресурса, а получение привилегий во время игры (игровые деньги, ресурсы и т.д.). Наряду с вышеперечисленными способами тестирования позволяет добиться хороших результатов. Недостатком данного метода можно считать то, что он не позволяет выявлять уязвимости до того, как ими воспользовались злоумышленники.

В качестве вывода следует отметить, что только комплексный системный подход позволяет добиваться требуемого уровня безопасности интернет-ресурсов. Причем это должно быть не разовым мероприятием, а осуществляться как периодическая спланированная деятельность, требующая согласованной работы разных специалистов.

Инструменты компании Etersoft для разработчиков

Денис Баранов, Виталий Липатов*

The article describes Korinf and Gitum, two instruments for software developers. Korinf and Gitum initially were created for the internal needs of Etersoft company. Korinf system enables the convertation of software packages for different Linux distros and operating systems. Git Upstream Manager (Gitum) helps to create and maintain the branches of upstream repositories. At the moment both instruments are available for all comers under public licences.

В разработке программного обеспечения большое значение имеет инфраструктура, инструменты, средства для разработчиков, помогающие более эффективно использовать время. В компании Etersoft осуществляется работа над разными проектами: начатыми с нуля, ответвления от upstream и др. Для обеспечения совместимости между существующими дистрибутивами разработана система Korinf, позволяющая собирать индивидуально подходящий для каждой ОС Linux свой пакет. Также разработан дополнительный инструмент для работы с системой контроля версий Git — Git Upstream Manager (Gitum), позволяющий эффективно создавать и сопровождать ответвления от upstream-репозиториев.

Korinf@Etersoft

Korinf — система сборки пакетов под целевые операционные системы на основе единого src.rpm, выполненного согласно правилам ALT Linux (`http://www.altlinux.org/Policy`).

Применение

Сборка пакетов, не являющихся системообразующими (неправильно применять Коринф для сборки glibc или rpm для разных систем). Тестовая пересборка пакета (проекта) «под все системы» (полезно для тестирования разработчиком). Создание дистрибутивоспецифичных репозиториев бинарных пакетов (позволяет не зани-

*Санкт-Петербург, РФ

маться пустой работой по упаковыванию Clip Art для разных систем). Сборка пакетов в автоматическом режиме на основе специального файла задания (робот-сборщик). Полученные репозитории могут быть использованы при сборке специальных версий дистрибутивов (mkimage для ALT Linux).

Единый исходник

Исходной единицей, отправляемой на сборку, является src.rpm со спеком, написанным согласно принятым в ALT Linux правилам. Сборка может осуществляться под различные ОС: Linux, Solaris, Mac OS, Windows. Так система сборки пакетов Korinf уже много лет используется для создания сборок продукта WINE@Etersoft (собственной версии Wine) под различные дистрибутивы. В прошлом году был запущен публичный сервер Korinf, призванный помочь сторонним разработчикам создавать версии своего ПО для разнообразных дистрибутивов Linux. Желающие воспользоваться публичным сервером Korinf, могут обратиться по адресу `korinf@etersoft.ru`.

Сайт проекта: `http://freesource.info/wiki/korinf`.

Git Upstream Manager

В 90% случаев при разработке проекта,на основе свободного ПО берётся стабильный релиз и переделывается, добавляется новый функционал. При попытке смержиться с upstream-веткой происходят конфликты, после исправления которых все наработки размазываются по истории коммитов и уже невозможно легко найти «свои» патчи. При ведении быстроразвивающихся проектов, таких как WINE (`http://winehq.org`), частые мержи и невозможность отделить «свои» от upstream-коммитов делают актуальным вопрос о корректном управлении и разборе кода. В 2011 года была начата разработка проекта, который позволит более легко и быстро ориентироваться в коде. Git Upstream Manager — дополнительный режим работы Git, позволяющий легко вести ветки разработки со своими патчами и обновлять их с upstream-веток, при этом ведя общую общую историю изменений и поддерживая патчи всегда в актуальном состоянии согласно состоянию upstream.

На данный момент выпущена версия gitum-0.4.1 и доступная для свободного использования.

Краткая характеристика рабочего процесса с gitum

Gitum имеет 5 рабочих веток:

1. Ветка апстрим репозитория — upstream.
2. Ветка с патчами наверху — rebased.
3. Ветка с непрерывной историей изменений — рабочая ветка — mainline.
4. Ветка с патчами в виде отдельных файлов — каждый коммит это состояние репозитория — patches.
5. Ветка с конфигурационным файлом, где содержатся имена 4-х предыдущих веток gitum-config.

Таким образом, разработчик всегда имеет актуальную версию upstream-ветки, ветку со всеми «своими» патчами и всей историей изменений в процессе разработке «ответвления».

Разработанная система Git Upstream Manager позволяет разработчикам с меньшем количеством усилий производить обновление своих продуктов до последних версий апстрима и отсылать пачти в основную ветку разработки.

Сайт проекта: http://freesource.info/wiki/GitUM

Сегодня компания Etersoft готова делиться своим опытом и предоставлять доступ к этим решениям всем желающим. На данный момент все инструменты распространяются под свободными лицензиями.

Асаблівасці выкарыстання сістэм кантролю версій для падрыхтоўкі навуковых публікацый

Антон Літвіненка*

The article discusses version control systems (VCS) usage to overcome typical issues at preparing scientific publications, and the peculiarities of this type of VCS usage. Author emphasizes the advantages and disadvantages of centeralised and distributed version control systems with respect to scientific publications preparation features, and gives some proposals on exact VCS types, VCS hosting services, front-end programs etc.

Падрыхтоўка навуковых прац да публікацыі з'яўляецца абавязковым этапам навуковага даследавання. Навуковая праца патрабуе арганізаванай супольнай працы ўсіх суаўтараў (для прац у галіне хіміі характэрна актыўнае выкарыстанне супольных даследаванняў між рознымі аддзеламі, арганізацыямі і краінамі, праз што ў падрыхтоўцы працы часта бярэ ўдзел 5–10 суаўтараў). Для дакладных навук тыповым з'яўляецца выкарыстанне вялікай колькасці дадатковых дадзеных: рысункаў, файлаў з лічбавымі формамі апісання спектраў, крысталічных структур, табліц з вынікамі эксперыментаў, файлаў з вынікамі праграмнай апрацоўкі эксперыментальных дадзеных альбо з вынікамі разлікаў і пад. Мэтазгодным з'яўляецца таксама захоўванне побач калекцый спасылак (напрыклад, базаў BibTeX), некаторых арыгінальных публікацый і пад. Навуковая праца праходзіць праз вялікую колькасць рэдагаванняў цягам пэўнага часу (ад некалькіх тыдняў да гадоў), некаторыя з якіх могуць радыкальна яе змяняць.

Такім чынам, перад аўтарамі паўстаюць наступныя праблемы:

1. Неабходна выконваць рэзервовае капіраванне ўсяго працоўнага каталога. З улікам колькасці рэдагаванняў ды аб'ёмаў дадзеных поўная гісторыя публікацыі можа займаць заўважную колькасць месца на дыску.

*Кіеў, Украіна

2. Рэзервовае капіраванне элементарнымі сродкамі не спрыяе напісанню дакладных каментарыяў наконт сутнасці зробленых зменаў, што робіць складаным пошук па гісторыі рэдагаванняў.
3. Бязладнасць працэсу абмену дадзенымі, парушэнне аднастайнай сістэмы нумарацыі рэдагаванняў вядзе да рассінхранізацыі гісторыі рэдагаванняў на працоўных ды хатніх камп'ютарах розных суаўтараў — такім чынам, частка гісторыі становіцца недаступнай, а гісторыя ўвогуле — хаатычнай і неарганізаванай.
4. Змены файлаў з дадатковымі дадзенымі не заўсёды звязаныя са зменамі асноўнага тэксту публікацыі, і адказ на пытанне «якому рэдагаванню аднаго файла адпавядае гэтая версія другога файла» можа быць нетрывіяльным.

Для рашэння аналагічных праблем у галіне распрацоўкі праграмнага забеспячэння актыўна ўжываюцца сістэмы кантролю версій (СКВ, version control systems, VCS). Мэтазгодным падаецца выкарыстаць аналагічны падыход і для падрыхтоўкі навуковых публікацый. Гэта дазваляе не дубляваць інфармацыю (захоўваючы толькі розніцу між версіямі), зрабіць больш арганізаванай гісторыю рэдагаванняў, цэнтралізаваць абмен рэдагаваннямі.

Разам з тым, у задачы падрыхтоўкі публікацыі ёсць некаторыя асаблівасці ў параўнанні з задачай распрацоўкі ПЗ:

1. Рэдагуюцца пераважна двайковыя дадзеныя (асабліва ў выпадку, калі тэкст публікацыі рыхтуецца ў рэдактарах WYSIWYG).
2. Навуковая праца не прадугледжвае сталага развіцця, яна канчаецца апублікаваннем.
3. Навуковай працы не ўласцівае актыўнае галінаванне, таму большасць магчымасцяў СКВ, звязаных з галінаваннем, не з'яўляюцца істотнымі для гэтай задачы.
4. Роля часткі суаўтараў часта зводзіцца выключна да «плённай дыскусіі» і выказвання заўваг, актыўна рэдагуюць звычайна ўсяго некалькі людзей.
5. Навуковая праца мусіць заставацца закрытай да моманту яе апублікавання.

Практычныя спробы рэалізаваць падрыхтоўку навуковае працы да публікацыі былі выкананыя (і выконваюцца цяпер) з выкарыстаннем СКВ SVN і Mercurial (як прыклады цэнтралізаванай і дэцэнтралізаванай СКВ, адпаведна).

Галоўныя перавагі цэнтралізаванай СКВ у працэсе падрыхтоўкі навуковых прац:

1. Вялікая колькасць двайковых дадзеных схіляе да мадэлі супольнай працы, якая прадугледжвае блакіраванне файлаў (locking) — гэта магчыма толькі ў цэнтралізаванай СКВ.
2. Мадэль працы СКВ прасцей для разумення неадмыслоўцамі.

Галоўныя перавагі дэцэнтралізаванай СКВ у працэсе падрыхтоўкі навуковых прац:

1. Незалежнасць ад наяўнасці і якасці канала сувязі.
2. Большасць аперацый адбываецца лакальна, і, праз тое, хутка.
3. Наяўнасць Інтэрнэт-сэрвісаў, што надаюць паслугі хостынгу дэцэнтралізаваных СКВ для закрытых прац малых груп суаўтараў бясплатна (напрыклад, bitbucket.org) — для SVN знайсці такія сэрвісы цяжэй.

Асаблівасці задачы цалкам дапускаюць працэс выкарыстання СКВ выключна з дапамогай графічнага інтэрфейсу (напрыклад, RapidSVN, TortoiseSVN, TortoiseHg). Для стварэння бясплатных рэпазіторыяў могуць быць выкарыстаныя сэрвісы bitbucket.org, xp-dev.com і г.д.

Такім чынам, сістэмы кантролю версій, выкарыстанне якіх з'яўляецца тыповым для задач распрацоўкі праграмнага забеспячэння, могуць быць плённа выкарыстаныя для задач падрыхтоўкі навуковых прац да публікацыі, хаця такое выкарыстанне СКВ мае шэраг асаблівасцяў, якія ўплываюць як на выбар СКВ, так і на працэс працы з ёй.

Применение свободного ПО при создании и внедрении Системы контроля за выполнением поручений Правительства

Максим Радюк*

One of the main functions of the Council of Ministers of the Republic of Belarus is to control the realization of the government orders. To improve the organizational and technical issues of the order control in the automated information system of the Council of Ministers, the automated System of Control presented here was recently developed. This system is based on free / open source software and therefore does not require additional investment. Being a part of the electronic documents circulation of the Council of Ministers the System of Control is accessible through the infrastructures of both the Council of Ministers and e-mail (mailgov.by) of other executives. The core of the System uses Ubuntu Server with HTTP and XMPP protocols. Its client part requires Sencha Ext JS in the browser of and allows review of orders to execute, including information about the process, and final report in the executive affiliation. Currently the System of Control is in active development being exploited in 67 executives in the Republic of Belarus.

Описание и цели создания

Одной из основных функций Аппарата Совета Министров Республики Беларусь (Аппарат) является контроль за выполнением органами государственного управления поручений, поступающих от Премьер-министра, заместителей Премьер-министра Республики Беларусь. До недавнего времени существующий технический порядок обработки поручений в аппаратно-программной платформе подсистем делопроизводства и контроля исполнения поручений (АПП) автоматизированной системы обработки информации (АСОИ) Аппарата учитывал не все организационные и технические моменты контроля поручений Правительства, что затрудняло оперативное решение поставленных задач.

*Минск, Belarus

Информацию о постановке на контроль и напоминания о приближении сроков исполнения поручений органы государственного управления получали по электронной почте. Ответственному работнику Аппарата приходилось по телефону выяснять, кто является непосредственным исполнителем поручения в органе государственного управления. Также, на стороне госоргана, отсутствовала возможность внесения информации об исполнении того или иного поручения.

Для совершенствования существующей инфраструктуры в инициативном порядке было принято решение о создании Системы контроля за выполнением поручений Правительства (Система контроля).

Система контроля обеспечивает организацию единого информационного процесса контроля за выполнением поручений Правительства независимо от внутриведомственной деятельности пользователей и выполняет следующие функции:

- просмотр списка активных и исполненных поручений
- печать списка поручений
- получение информации об исполнителе поручения в госоргане
- получение информации о выполнении поручения в госоргане
- получение информации о соисполнителях поручения в других госорганах
- поиск поручений по реквизитам
- формирование справочников
- отправка почтовых сообщений работникам Аппарата
- обмен мгновенными сообщениями между работниками Аппарата и работниками других госорганов (в разработке).

Архитектура

Схема на рис. 36.1 представляет положение Системы контроля в структуре АСОИ Аппарата.

Система контроля является надстройкой над подсистемой электронного документооборота АСОИ Аппарата и представляет собой веб-ресурс , доступный как из инфраструктуры АСОИ Аппарата, так и через сетевую инфраструктуру электронной почты государственных органов mailgov.by.

Структурная схема Системы контроля изображена на рис. 36.2 и состоит из следующих компонент:

- основной сервер, доступный по адресу `http://cmcntlsys.cm.by`
- реплика-сервер

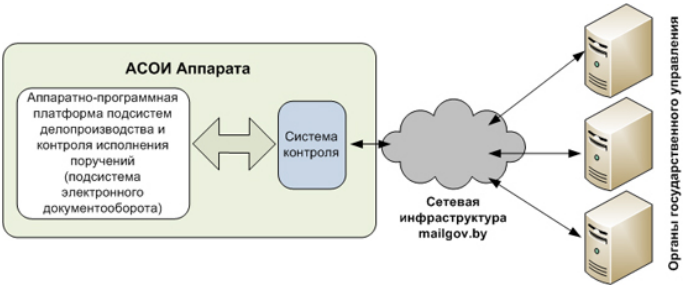


Рис. 36.1. Система контроля внутри АСОИ Аппарата

– ftp-сервер обмена данными
– корпоративный контроллер домена Active Directory
– xmpp (jabber) — сервер обмена мгновенными сообщениями.

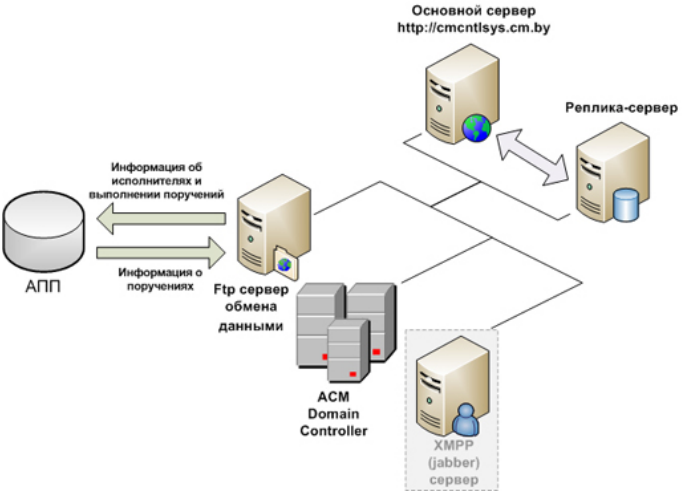


Рис. 36.2. Структурная схема Системы контроля

Серверная часть (ядро) аккумулирует всю информацию о ходе процесса контроля выполнения поручений и обеспечивает предоставление информации пользователям. Обмен данными между Системой контроля и АПП Аппарата обеспечивается по ftp-протоколу

через выделенный сервер. Ядро Системы контроля построено с использованием следующих продуктов и технологий:

- операционная система: Ubuntu Server 10.10
- web-сервер: Apache 2.2.16
- СУБД: MySQL 5.1.49
- язык разработки: PHP 5.3.3
- XMPP-сервер: Openfire 3.7.1.

Клиентская часть Системы контроля представляет собой приложение, разработанное с использованием javascript-библиотек Sencha Ext JS 3.3.0., выполняемое в браузере пользователя. Основное её предназначение — просмотр списка поручений, подлежащих исполнению, а также внесение информации и отчета об исполнении поручения в органе государственного управления.

Выбор свободного программного обеспечения при разработке Системы контроля обусловлен следующими факторами:

- отсутствуют какие-либо дополнительные финансовые затраты;
- нет необходимости осуществления обязательных процедур обоснования, согласования и проведения тендеров на закупку ПО — т.н. «быстрый старт».

В настоящее время ведется активное развитие и опытная эксплуатация Системы контроля в 67 органах государственного управления. В презентации мы представим более подробную информацию о порядке работы в Системе контроля, пользовательский интерфейс и разъясним технические детали реализации проекта.

Обзор одноплатного микрокомпьютера BeagleBone

Дмитрий Горох*

Over the last years the single-board computers based on ARM CPU core such as Raspberry Pi, PandaBoard and SheevaPlug have rapidly gained popularity. Volunteer driven start-up BeagleBoard.org recently added the new device to this list: a single-board hardware hacker oriented computer for only $89.

Последнее время стремительно набирают популярность недорогие одноплатные микрокомпьютеры на базе ARM процессоров, такие как Raspberry Pi, PandaBoard, SheevaPlug и другие. Некоммерческий стартап BeagleBoard.org недавно пополнил этот список своей новой разработкой: одноплатным компьютером BeagleBone ориентированным на DIY энтузиастов ценой всего $89.

Краткое описание

Среди большинства других одноплатных микрокомпьютеров BeagleBone выделяется своей ориентированностью на энтузиастов, желающих иметь расширяемую аппаратную платформу на базе производительного и мало потребляющего процессора под управлением ОС Linux. Вокруг BeagleBone уже образовалось активное сообщество, публикующее новые отчёты об оригинальном применении BeagleBone. На момент написания статьи на сайте было зарегистрировано 244 проекта, среди которых можно найти:

- погодная станция
- сетевая камера слежения
- игровая консоль
- осциллограф
- сетевое хранилище файлов

Кроме того ничто не мешает использовать BeagleBone как традиционный Linux сервер. В пакетном репозитарии поставляемого Linux дистрибутива можно найти большое множество серверов: lighthttpd (HTTP Server), rtorrent (Bittorrent Client), postfix (SMTP Mail Server), git (distributed revision control) и многое другое.

*Минск, Беларусь

Разработка ПО

Для разработчиков с платой поставляется полный набор программного обеспечения, позволяющего собрать из исходников ядро Линукса, загрузчик и корневую файловую систему дистрибутива Ångström. На плате имеется USB-to-UART конвертер, подключённый к UART и JTAG портам процессора. Таким образом вам понадобится только USB кабель чтобы иметь полный контроль над платой в процессе разработки. В качестве IDE поставляется Eclipse и Cloud9. Последняя IDE предлагает интересный способ разработки простых приложений в web браузере. Приложения пишутся на jScript в асинхронном стиле и запускаются из-под сервера Node.js опираясь на библиотеку bonescript. Пример программы, мигающей светодиодом:

```
var bb = require('./bonescript');
var ledPin = bone.P8_3;
var ledPin2 = bone.USR3;
setup = function() {
    pinMode(ledPin, OUTPUT);
    pinMode(ledPin2, OUTPUT);
};
loop = function() {
    digitalWrite(ledPin, HIGH);
    digitalWrite(ledPin2, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    digitalWrite(ledPin2, LOW);
    delay(1000);
};
bb.run();
```

Разработчики библиотеки bonescript ставят своей целью создание простой для освоения и использования платформы, аналогичной Wiring, существующей для Arduino.

Платы расширения

BeagleBoard имеет 2 46-пиновых разъёма, дающие доступ практически ко всем портам процессора. Форм-фактор платы устроен таким образом, что платы расширения можно стыковать друг на

друга «бутербродом». На настоящий момент существует около десятка плат расширения, и этот список растёт за счёт активности со стороны сообщества. Имеются платы реализующие DVI видео выход, LCD дисплей, аудио входы и выходы, батарейное питание. Форм-фактор платы также располагает к быстрому прототипированию простых схем навесным монтажом или на макетной плате.

Заключение

BeagleBoard является мощной основой как для любительского творчества, так и для профессиональной разработки Embedded Linux приложений. Современные Linux дистрибутивы, ориентированные на ARM платформы, позволяют работать с одноплатными компьютерами практически с тем же комфортом, что и на x86 машинах. При этом ARM системы (в особенности Cortex-A8) обеспечивают беспрецедентно низкое удельное энергопотребление на единицу производительности (Watt / MIPS) и лидируют по компактности, позволяя строить широкий спектр портативных устройств с автономным питанием.

Литература

[1] Домашняя страница BeagleBone `http://beagleboard.org/bone`

[2] Руководство по быстрому запуску BeagleBoard `http://beagleboard.org/static/beaglebone/a3/README.htm`

[3] Библиотека bonescript `https://github.com/jadonk/bonescript`

[4] Использование Eclipse для BeagleBone `http://elinux.org/BeagleBoardEclipse`

[5] Дистрибутив Ångström `http://www.angstrom-distribution.org/`

[6] Пример программирования на jScript под Cloud9 и на Python `http://www.gigamegablog.com/2012/01/05/beaglebone-coding-101-blinking-an-led/`

[7] Инструкция по установке Android `http://processors.wiki.ti.com/index.php/BeagleBone-Android-DevKit_Guide`

[8] Платы расширения для BeagleBone `http://beagleboardtoys.com/wiki/index.php5?title=Main_Page`

[9] BeagleBone на Farnell `http://ru.farnell.com/circuitco/bb-bone-000/kit-dev-beaglebone-cortex-a8/dp/2063627`

Экспресс-тест десктопных возможностей BSD дистрибутивов

Дмитрий Ванькевич*

Different flaws of BSD operating systems family are reviewed on the subject of their readiness as a desktop OS. Criteria of desktop readiness covering the ease of configuration and maintenance, as far as needed desktop software are presented, and the easy sequence of steps is proposed to make quick comparison covering large subset of these criteria.

К числу BSD-дистрибутивов, более или менее ориентированных на десктоп-системы, можно отнести следующие:

- DragonFlyBSD
- FuguIta
- PC-BSD
- GhostBSD
- PУС-BSD
- DesktopBSD
- VirtualBSD 9.0
- Frenzy
- Jibbed 5.1
- TrueBSD

В списке сознательно не рассматривается FreeBSD, потому что изначально требует значительных усилий в настройке для применения в качестве десктопа.

Вопрос пригодности дистрибутива для десктоп-систем можно рассматривать исходя из его соответствия достаточно большому списку требований:

- работа на персональных компьютерах, типичных для офисных рабочих мест
- лёгкость в использовании
- лёгкость в установке
- наличие качественной локализации
- тщательность подбора программ (одна задача — одна программа)

*Львов, Украина

– лёгкость в обновлении
– возможность установки дополнительных программ пользователями
– иметь стандартный и приятный внешний вид
– наличие необходимого для офисной работы инструмента: текстовой процессор, электронные таблицы, средства просмотра для разных файловых форматов и утилиты печати
– интеграция с Linux и Windows серверами
– возможность удалённого администрирования
– наличие стандартных программ для работы в Internet
– возможность сетевой установки на бездисковые рабочие станции

Критерии взяты из статьи `nklug.org.ua/lg/lg81/arndt.html`.

На практике была использована следующая последовательность действий для определения, насколько дистрибутивов соответствует критериям пригодности для десктопа:

– установка дистрибутива
– если дистрибутив ставится без значительных усилий и корректно установились драйвера для оборудования — выход в интернет с выполнением обычных задач характерных для пользователя.
– определение недостающих программ и их установка
– если установка и настройка необходимых программ требует «титанических усилий» — переход к следующему дистрибутиву

Бета-версия доклада была прочитана на заседании `linux.lviv.ua/`. Автору были даны ценные замечания по поводу презентации.

Выводы:

PC-BSD несмотря на некоторые недостатки и «шероховатости» — самый дружелюбный дистрибутив. Подойдёт как начинающим пользователям, так и пользователям со стажем. В PC-BSD имеется графическая программа установки и удаления пакетов PBI. В то же время в ней есть и система портов (ports) и пакетов (packages) FreeBSD.

PУС-BSD — Содержит все необходимые для повседневной работы программы. Но базируется на устаревшей версии FreeBSD.

VirtualBSD — в ногу со временем (в свете всеобщей виртуализации и облачных технологий)

Голос спонсора: SaM Solutions

Компания SaM Solutions выступает в роли системо-образующего спонсора конференции Linux Vacation Eastern Europe с момента зарождения LVEE в 2005 году и на протяжении всех лет её проведения.

Сложившаяся корпоративная практика не случайна. Продукты и решения, задействующие Linux и другие Free/Open Source Software проекты, составляют заметную часть пакета разработок SaM Solutions. Кадровая политика компании направлена на поощрение профессионального развития своих сотрудников, организацию их эффективного отдыха и привлечение хорошо мотивированных кандидатов к работе на компанию. Формат конференции LVEE успешно позволяет решать все три задачи.

Одним из подразделений компании является отдел Linux и Embbeded. Специалисты компании на протяжении десятилетий работают с СПО. Компанией реализован ряд проектов по адаптации ОС GNU/Linux для работы в различных устройствах, построенных на таких платформах как ARM, PowerPC, x86, MIPS. В последние годы — на ведущие позиции выходит разработка управляющего ПО для серверов Enterprise-класса, от низкоуровнего BMC Firmware на основе Linux до высокоуровневых систем контроля виртуализации и графических интерфейсов управления, от прошивок устройств хранения данных до BSP интегрированных плат для разработчика. Надёжность, качество и широкий функционал множества свободных проектов позволяет строить нам системы любого уровня и сложности, опираясь на высококачественные готовые компоненты.

В рамках направления Linux и Embedded направления успешно выполнены проекты для таких знаковых заказчиков, как Novell/SUSE, Fujitsu Technology Solutions и осуществляется партнёрство с компаниями IBM и Oracle/Sun в области Open Source решений.

Мы разрабатываем, модифицируем и адаптируем различное свободное программное обеспечение для наших заказчиков, но не забываем и о своих нуждах — наши сотрудники используют в своей работе существующие програмные продукты и вносят вклад в их развитие. Часть внутренней инфраструктуры, а именно интранет-сеть компании, тестовые стенды отдела контроля качества, рабочие места сотрудников профильных подразделений — также рабо-

тает под управлением СПО (серверные и десктопные платформы GNU/Linux и FreeBSD).

В минувшем году, в рамках реорганизации, был разработан долгосрочный план развития направления Linux и Embedded в SaM Solutions. В нём впервые были кодифицированы уже имеющиеся внутренние неофициальные практики по взаимодействию с community-based проектами. В частности разработаны меры и правила по

- возврата изменений в родительские проекты (upstreaming);
- вхождения в состав постоянных разработчиков активно используемых нами FOSS-компонентов;
- публикации сообщений об ошибках (bug reporting);
- участия и помощи в организации community events;
- стимуляции участия и докладов на технических конференциях.

И план немедленно начал претворяться в жизнь.

Силами отдела организовано внутреннее обучение сотрудников на регулярной основе. В частности в этом году уже проведены курсы по Git, Debian/Ubuntu Packaging, современным технологиям тестирования и контроля качества. Полностью готов и запланирован к прочтению курс по прагматичному функциональному и unit-тестированию. Курс Debian/Ubuntu Packaging по согласованию с автором готовится к публикации (видео, презентация и исходные тексты презентации в LaTeX). Для создания и обучения кадрового резерва на ближайшее будущее запланированы постоянно действующие внутренние проекты в области Embedded Linux, результаты которых также запланированы к публикации.

Визиты представительных делегаций на Embedded World 2012 и Linux Con Europe /Embedded LinuxCon Europe 2011 обогатили нас новыми идеями куда можно двигаться дальше и что сейчас актуально. А выступления на Software Engineering Forum for Students, на круглом столе по СПО в рамках TIBO-2012 и LVEE-Winter 2012 позволили поделиться нашим опытом со всеми заинтересованными сторонами.

При поддержке SaM Solutions, с декабря 2011 года возобновились регулярные встречи Minsk Linux Users Groups, под названием «Линуксовка в SaM Solutions». Техническое оснащение линуксовок и открытый формат встреч позволил им практически мгновенно стать заметным дискуссионным клубом по широкому спектру вопросов, прямо или косвенно связанных с СПО. Свободная картография (OpenStreetMap), технологии виртуализации, минский

hackerspace, Linux Mobile, бойкот Голливудской продукции, systemd, загрузчик u-boot, белорусская локализация GNOME — это только часть тем, поднятых за последние линуксовки.

Быстрые и положительные изменения, как внутри компании SaM Solutions, так и в экосфере СПО (и Linux в частности) наполняют нас уверенностью, что направление движения выбрано верно.

Голос спонсора: EPAM Systems

Infoblox в вопросах и ответах, или Шанс вступить на путь хакера и сделать при этом карьеру

Infoblox: Кто это? Что это? Где это?

Infoblox — довольно известная в кремниевой долине компания, производящая стоечные сервера, призванные помочь в управлении крупными сетями со сложной инфраструктурой. Основной программный продукт компании, NIOS, является довольно сложной, с технической точки зрения, системой состоящей из множества взаимодействующих компонент, что автоматически делает его интересным для серьезных разработчиков.

А причем тут хакеры?

Фундаментом всей системы служит разрабатываемый собственными силами дистрибутив Linux, основанный на Fedora 12, и имеющий кодовое название Granville. Infoblox делает Granville 64-битным, чтобы следовать в ногу со временем. В EPAM Systems есть специальная команда, которая занимается разработкой и поддержкой Granville, ведь NIOS имеет в среднем около 4 релизов в год. Работа с Granville — это шанс вступить на путь настоящего хакера.

И это все, что делают инженеры? Не маловато будет?

Infoblox вносит свой вклад в развитие ПО с открытым кодом: все модификации open-source кода публикуются под лицензией GPL. Кроме того высокий профессиональный уровень инженеров EPAM Systems, работающих с Infoblox, позволяет им в свободное время принимать участие в разработке других проектов с открытым кодом, среди которых есть даже ядро Linux.

Сказали «А» — говорите и «Б»!

И если Granville является лишь основой, то ядро системы это, во-первых, распределенное (сервера имеют возможность организации в сетку с центральным управлением) серверное приложение для обработки запросов управления сетью и сетевыми службами, построенное по трехуровневой архитектуре, и во-вторых ряд служб-демонов (например DNS или DHCP службы), обрабатывающих специфические запросы возникающие в инфраструктуре сети.

Огласите весь список... инструментария!

Инженеры EPAM Systems постоянно используют несколько языков программирования и множество различных инструментов для решения каждодневных задач. Кроме богатого инструментария, доступного разработчикам на Unix подобных системах, достойны упоминания такие классические вещи, как C, Perl и Bash, без которых эффективная разработка под Unix системы просто немыслима, современные, перспективные и быстроразвивающиеся Ruby и Python, которые выбрали для себя Google, NASA, CERN и Motorola — и это список лишь наиболее часто используемых инструментов, с которыми приходится иметь дело.

Ключевым компонентом NIOS можно назвать движок управления базами данных oneDB, реализованный поверх Berkeley DB. В Infoblox добавили возможность работать с подмножеством SQL, а также подобие слоя объектно-реляционного отображения, при это не потеряв гибкости и легковесности BDB.

А кому все это нужно? Кто это покупает?

Развитие новых технологий является важной составляющей деятельности Infoblox. Так, совместно с Boeing, IBM, HP, Intel, Microsoft, US National Security Agency и другими заинтересованными сторонами компания участвует в разработке нового протокола под названием IF-MAP, к которому уже сегодня проявляют большой интерес многие компании как изнутри, так и извне IT индустрии. Infoblox, к тому же, поставляет реализацию IF-MAP сервера, как один из своих продуктов. Разработчики EPAM Systems имеют тесное отношение к успешному продвижению этого продукта.

Так а чем вы лучше других?

Больным местом многих отечественных компаний разработчиков ПО является полное, либо частичное, отсутствие четкого процесса изготовления продукта. Это часто приводит к конфликтам, снижению производительности, потере квалифицированного персонала. Работающие с Infoblox инженеры EPAM Systems полностью интегрированы в процессы Infoblox. Здесь практикуют модульное тестирование и рецензирование кода, есть четкое расписание, которое не изменяется от одного только желания менеджмента добавить новую возможность в следующий релиз. Проектная документация исправно ведется и находится в свободном доступе. Специалисты рабочей группы имеют возможность руководить проектами Infoblox, разрабатывая проектную документацию и дизайн само-

стоятельно, тем самым поднимая сотрудничество EPAM Systems и Infoblox на более высокий уровень.

«Выдавали тайны» Infoblox:

Михаил Бойко, Senior Software Engineering Manager в EPAM Systems; Олег Орёл, Lead Software Engineer в EPAM Systems; Даниил Прищепа, Software Engineering Manager в EPAM Systems.

Готова к вашим вопросам служба рекрутинга EPAM Systems

Проекту срочно нужны:

– Perl разработчики.
– C/C++ разработчики.
– Linux Kernel разработчики.

Команда ждет своих новых героев!

Голос спонсора: hoster.by

Hoster.by — один из крупнейших хостинг-провайдеров Беларуси и технический администратор доменной зоны .BY. Компания работает на рынке хостинга с 2000 года и обслуживает более 15 000 клиентов.

Hoster.by предоставляет все виды хостинга для работы любых проектов, а также почтовые сервисы, SSL-сертификаты, аренду лицензий на программное обеспечение и другие сопутствующие услуги. Компания является провайдером, уполномоченным оказывать интернет-услуги государственным органам и предприятиям, использующим в своей деятельности сведения, составляющие государственные секреты.

Собственное оборудование, круглосуточный мониторинг состояния серверов и техподдержка 24/7 — неизменные черты hoster.by.

С февраля 2012 года, когда hoster.by стал техническим администратором национальной доменной зоны .BY, в порядке регистрации доменов произошел ряд качественных изменений:

- отменена возможность 30-дневной блокировки доменного имени;
- отменена проверка заявок в Оперативно-аналитическом центре (ОАЦ), которая ранее могла занимать несколько дней;
- начал работу сервис Whois;
- данные при регистрации домена теперь можно вводить на белорусском языке.

220005, г.Минск, ул. В.Хоружей, 1А, 6 этаж
Тел.: (017) 239-57-02, velcom: (029) 3-4444-83,
МТС: (029) 776-44-83, life: (025) 720-52-66, факс 239-57-20

Голос спонсора: World of Tanks team

Голос спонсора: инновационная компания Promwad

Инновационная компания Promwad реализует полный цикл разработки и сопровождает производство электроники, ежегодно организует форум разработчиков цифровой электроники (DEDF).

Специалисты компании разрабатывают устройства для массового производства, охватывая следующие направления: мультимедиа, автоэлектронику, устройства для датакома/телекома, приборы на базе GPS/ГЛОНАСС.

Promwad является официальным дизайн-центром Texas Instruments, членом международной ассоциации IPC, а также многолетним партнером мировых чип-вендоров: ST, Fujitsu, Marvell и Analog Devices.

С 2004 года команда Promwad успешно реализовала более 140 проектов. Примеры свежих разработок:

– Спроектирован plug-компьютер на базе Linux и C++ — аппаратно-программная платформа и корпус. Новинка предназначена для решения целого спектра задач в IP-сетях, а по размерам сопоставима с зарядным устройством для мобильного телефона.

– Разработан тонкий клиент АКТ-1100 — компьютер на базе процессора Marvell Sheeva (2 ГГц) и ОС Debian Linux 6.0, он переносит процессы обработки данных на удаленный сервер, использует различные методы защиты информации.
– Разработан корпус дозиметра-радиометра «ДО-РА» — миниатюрного устройства для измерения радиации (в виде насадки для мобильных телефонов).

Промышленные дизайнеры и конструкторы Promwad разработали стильный корпус «ДО-РА» для iPhone 4 и 4s, он подключается к смартфону через аудиоджек.

– Разработан процессорный модуль Automotive Jade на базе ОС Linux и системы OpenEmbedded для управления, контроля и диагностики оборудования автотранспорта.

Головной центр разработок Promwad размещен в Минске, но его сотрудники видят результат своей работы в устройствах и программах, которыми пользуются люди по всему миру.

Ценности компании: непрерывное обучение и внедрение инноваций, открытость к сотрудничеству и активный отдых (Promwad является титульным спонсором белорусского кубка приключенческих гонок «Промвад Тур»).

Интервью с участниками

По сложившейся традиции в сборник материалов включены интервью, взятые представителями оргкомитета у участников конференции: как сразу после LVEE 2011, так и во время впервые проведенной в этом году выездной зимней сессии конференции LVEE Winter 2012. Интервью 2011 года было тематическим, посвященным использованию свободного программного обеспечения в сфере образования. Его участники принадлежат к разным странам и в той или иной степени имеют отношение к академическому процессу. Мы попросили их поделиться особенностями использования свободных программ в их вузах и услышали, какова ситуация в академических кругах Украины, Литвы и Венгрии.

1 Григорий Злобин, Львов, Украина (LVEE 2011)

Григорий Злобин: Я доцент Львовского университета, работаю на кафедре радиофизики факультета электроники. Уже шесть лет как у нас ведется подготовка студентов в направлении компьютерных наук. Возникло это благодаря тому, что раньше кафедра занималась разработкой программ машинного анализа электронных цепей, т. е. считалось, что мы имеем достаточный уровень профессиональной подготовки. А специальность, по которой мы готовим студентов — это «Информационные системы».

L: Кем обычно работают выпускники этой специальности?

Г: Если честно, спектр очень широк. Работают и разработчиками, и системными администраторами, и разработчиками встроенного программного обеспечения, правда большая часть разрабатывает ПО для микроконтроллеров фирмы Cypress. Именно здесь наши выпускники имеют очень высокую подготовку, потому что для этих микроконтроллеров мало сделать ПО, нужно еще разработать аппаратную часть, как цифровую так и аналоговую.

L: Учебный процесс по этой специальности как-нибудь задействует свободное ПО?

Г: Используется свободное ПО при чтении как общих курсов, так и при чтении спецкурсов. Общие курсы это «Операционные системы» и «Системное программирование», а спецкурсы — здесь уже

очень зависит от лектора. Поскольку мы по сути дела только в начале пути, то к сожалению наша кафедра не доросла до встраиваемых систем, где нужен Linux; пока мы занимаемся только микроконтроллерами. Правда, этому есть объяснение: очень большой объем заказов у фирм, занимающихся микроконтроллерами, и мы им не в состоянии дать столько студентов. Пока идет замыкание на микроконтроллеры Cypress, но я надеюсь, что мы будем двигаться дальше.

L: Программное обеспечение, используемое в вузе, это еще и средства обеспечения учебного процесса: те же текстовые редакторы и операционные системы...

Г: Да. Поскольку факультет наш принадлежит к естественным наукам, то курса информатики у нас просто нет. Но есть учебные практики после первого, второго, третьего курса, и это чудесная возможность вставить упреждающий по сути курс по ОС Linux, Openoffice, SciLab... Могу похвастаться, что на практике мы используем оболочку «Кузя» — собственную разработку, наших студентов и для студентов.

L: А можно про нее пару слов?

Г: Что было толчком? У нас на кафедре использовалась еще одна оболочка, сделанная под Windows для языка Паскаль. Я ее горячо поддерживал, а другие преподаватели относились с прохладой. Но когда у нас появилась компьютерная специальность, мы получили интересное явление: преподаватель, который читал программирование на Паскале, требовал, чтобы студенты этой оболочкой не пользовались. А студенты дома в ней работали, а потом на лабораторных работах сделанные программы сдавали в Turbo Pascal. И именно это натолкнуло их — студентов — на идею: если оболочка, сделанная именно под учебный процесс, дает возможность делать программы быстрее, давайте напишем ее сами.

Поскольку «Кузя» использует компилятор gcc (и не только) — возможности его очень широкие. Правда, поскольку это работа просто по желанию — не все так хорошо движется, но мы с моим дипломником в следующем году собираемся сделать версию с вебдоступом. Так что ребята работают, это очень приятно... У нас уже поддерживается шесть языков в интерфейсе, один из них белорусский, и есть арабский. Даже благодаря «Кузе» ребята, которые ее сделали, уже больше года работают разработчиками в фирме.

Как ни парадоксально, изменить отношение к СПО помогла фирма Microsoft. Прошлой осенью я получил письмо от Сергея

Байдачного с просьбой срочно, прямо завтра, предоставить резюме студентов, которые могли бы поехать на стажировку в Microsoft в США. Я быстро всех студентов проинформировал, ребята послали свои резюме, в т. ч. и те, кто делали «Кузю», причем один из них признался, что занимается СПО. И именно тот, который признался, прошел первый тур отбора.

L: Это показатель :)

Г: Да. Противники СПО, когда об этом узнали, очень сильно озаботились. Это повлияло на преподавателей, у них начало меняться отношение. Хотя есть и другие примеры. Один из наших преподавателей написал методические указания к Matlab. Я предложил проверить, можно ли эти задания выполнить в Octave, и оказалось, что все выполняется. У нас в этом году открылась научно-исследовательская тема, в рамках которой мы собираемся сделать книгу «Свободные системы компьютерной математики», и там будет раздел по Octave. Показательно, что я с этим же человеком дискутировал, когда он в других методических указаниях вставлял Matlab: «Вот придет наш выпускник на производство и скажет: давайте купим Matlab, и я в нем такие вещи сделаю! А этого выпускника немедленно уволят, потому что Matlab стоит $5000, а библиотека Simulink — $45000. Они скажут, что им такие дорогие работники не нужны». А вот когда мы посмотрели, что то же самое можно сделать в Octave — это уже совсем другой поворот.

L: Какие требования у потенциальных работодателей ваших выпускников? Можно ли говорить о каком-то интересе работодателей к свободному ПО? Эта тема как-то просматривается?

Г: Ну, во-первых, фирмы, занимающиеся программированием во Львове, Softserve, Eleks, ведут разработки как под Windows, так и в СПО, и проявляют к этому большой интерес. Но на контакты с нами, тем не менее, идут не очень охотно, предпочитают напрямую работать со студентами. Потом, есть еще фирмы, нанимающие технический персонал для обслуживания облака в отдаленных от нас регионах, и работающие на открытом программном обеспечении. Но небольшая емкость рынка делает эту политику очень индивидуальной.

L: Как бы вы, коротко, охарактеризовали основные факторы в вузе, мешающие свободному программному обеспечению?

Г: Тотальная безответственность. Основная часть проприетарного программного обеспечения — пиратская, и несмотря на уголовный кодекс, который содержит очень жесткие уголовные нормы за пользование пиратским продуктом, вузы не трогают. И пока это так, вузы живут с позиции «Мне все равно, а если даже будет проверка — проблема будет у ректора». Впрочем, я бы не хотел так уж обвинять своих коллег. Нужно еще учитывать, что уровень оплаты труда преподавателей не слишком соответствует решаемым задачам.

L: И последний вопрос: как вы заинтересовались свободным ПО сами?

Г: Это было лет 15 назад, еще до появления статьи в уголовном кодексе. Я как-то задумался: «Неужели есть только Microsoft Office и больше ничего?» Тогда мне под руки подвернулся Star Office, проприетарный продукт, бесплатный для учебных заведений. Я начал его пропагандировать и постепенно перешел в Linux. С Linux была проблема из-за той компьютерной техники, которую мы тогда имели. Некоторые легковесные дистрибутивы можно было поставить, но они не были локализированы. Это сразу означало, что мы не можем их использовать. А еще для нас было очень важным наличие украинского диалога. Долгое время фирма Microsoft игнорировала этот вопрос, считая, что достаточно русскоязычной версии. Только после появления локализированных дистрибутивов Linux они начали некоторую активность. Для некоторых регионов Украины наличие или отсутствие украиноязычного диалога не имеет никакого значения, но только не во Львове.

2 Миколас Окулич-Казаринас, Вильнюс, Литва (LVEE 2011)

Миколас: Я преподаватель университета им. Миколаса Ромериса в Вильнюсе. Уже три года как у нас создан новый факультет социальной информатики. Я работаю на кафедре информатики программных систем, и мы уже через год будем иметь первых выпускников — студентов по специальности «Бизнес-информатика».

L: Это достаточно новая специальность?

М: Новая, на стыке информатики и менеджмента.

L: Как бы ты охарактеризовал потенциальный рынок труда для ваших выпускников?

M: Мое понимание этого вопроса чуть отличается от понимания некоторых других специалистов. Одна из предусмотренных специальностей — консультант по продажам информационных систем, но я надеюсь, что наши студенты также могут быть руководителями каких-либо отделов или фирм, связанных с IT. Я хотел бы, чтобы высшее образование давало более высокое положение. А с другой стороны, т. к. мы имеем много предметов, связанных с бизнесом, мы не можем сделать из студентов хороших программистов, и я надеюсь сделать их настолько хорошими программистами, чтобы они могли сформулировать задачу для программистов-профессионалов.

L: И как при такой специализации — свободное ПО находит какое-то применение в учебном процессе?

M: Конечно находит. Мы свободное программное обеспечение начали включать еще в 2004 году, т. е. до появления этой специальности и этого факультета. Что мы сделали — это инсталлировали OpenOffice на всех компьютерах кафедры, на которой я тогда работал, не удаляя ничего, что было прежде. Мне приходилось тогда преподавать социальную статистику, все задачи выполнялись на табличном процессоре. И когда я спросил: «Какие табличные процессоры вы знаете?» — около 5% знали про OpenOffice. Из них около половины его раньше включали, и только один студент сказал, что имеет опыт работы. И я объяснил: «Я вам преподаю статистические методы, и приходя на занятие, вы должны уже уметь пользоваться табличным процессором. В свободное время я могу показать вам программу, которую вы не знаете — OpenOffice. Вы можете выполнять задачи на чем угодно, но я вам предлагаю попробовать эту новую программу». После такого предложения один или два студента выполнили работу в Mircosoft Excel, а остальные выполнили их в OpenOffice или использовали обе программы. Причем я внимательно сравнивал качество этих работ — не было никакой корреляции между выбранной программой и качеством выполнения. Если бы студент начал работать с новой программой и столкнулся с проблемами — конечно он перешел бы обратно на уже известную программу. Не у всех хватило храбрости попробовать новую программу с первого занятия, но большинство из тех, кто раньше или позже попробовал, так до конца и делали задания в OpenOffice. Этот курс статистики, который я преподавал не для информатиков, а для совсем другой специальности, позволил мне убедиться,

что в университете для непрограммистов можно без ущерба для учебного процесса использовать OpenOffice. Это был первый шаг.

Кроме того, еще до нового факультета у нас использовалась система Moodle. Мы сейчас развиваем систему сетевого обучения — чуть другой подход, чем дистанционное обучение, и в рамках этой системы у нас сервер работает на Linux. Однако общая инфраструктура создана в университете еще до нас, она работает и нет смысла от нее отказываться. Может быть, если бы она была создана сейчас...

L: То есть свободное программное обеспечение находит применение и в сетевой инфраструктуре?

M: Когда появились информатики — появились задачи программирования. Программировали что-то локально. Я создал сервер, к которому подключается каждый студент (но аутентификацию отдельную мы не делали, через LikwiseOpen наш сервер аутентифицируется в контроллере домена Active Directory), студенту создается автоматически акаунт в этом Linux-сервере, и веб-адрес. Каталог в домашнем каталоге студента отображается на веб-сервере в Apache. Первокурсник, получив задачу запрограммировать что-то на PHP, программирует и может сразу же показать что-то — в т. ч. даже показать друзьям или позвонить домой. Возможно, в будущем — показать своему работодателю. С другой стороны, PHP — это язык, который дает слишком много возможностей, и я не уверен, что это хорошо с педагогической точки зрения.

L: А как у вас преподавательский состав относится к СПО?

M: Очень по-разному. Есть очень много хороших специалистов в области информатики, которые используют СПО, но без политического подтекста. Java, Netbeans, Eclipse... Без разницы, Windows там или Linux. Есть такие, которые используют азартно, кто чувствует, что выучив что-то свободное, студент сможет потом предложить своему работодателю более дешевую и приспособленную систему. Задача университета — скорее подготовить таких специалистов, которые приносили бы деньги своему работодателю, а не указывали, что ему покупать. Есть люди, которые это понимают. Но есть конечно и такие, кто считает: «Я столько версий этого монополиста изучал! Опять переучиваться? Нет уж, мне до пенсии и этого хватит». Самое неожиданное для меня было, когда один преподаватель сказал: «Да, университет сэкономит очень много денег, но это же не я!» Хотя я надеюсь, что это скорее исключения.

Что больше всего радует — понимание на уровне администрации того, что сетевое обучение неотделимо от свободного ПО. Когда студент находится дома, он должен иметь все то же программное обеспечение, которое нужно для обучения, и мы не можем его обеспечить проприетарным ПО. И даже если какой-то поставщик сделает свой продукт для обучения бесплатным — все равно мы хотим, чтобы студент, чему-то научившись, сразу мог использовать этот продукт для своей работы, для частного бизнеса, для чего-то еще... А для этого подходит только открытое программное обеспечение.

Когда преподаватели почувствовали поддержку сверху, от администрации — возмущений стало все меньше и меньше. С другой стороны, свободное ПО совершенствуется быстро, и то что было несколько лет назад — хуже чем есть сейчас. И еще, когда появился compiz, люди не интересовавшиеся Linux, собрались в преподавательской смотреть на поворачивающийся куб...:)

L: Да, надо сказать, что хотя аппаратное ускорение графики придумали не в Linux, зато в Linux его внедрили быстрее всего. И заметнее.

M: Да. С другой стороны, есть люди, которые много лет используют Windows, и они не хотят ничего менять, считая это стабильностью.

L: А как вообще в Литве себя чувствует свободное программное обеспечение?

M: Рост есть. Я — один из учредителей ассоциации открытого кода (АКЛ), мы раньше несколько лет участвовали в региональной выставке информационных технологий InfoBalt. В первом году реакция была: «Что, бесплатно?» Люди не понимали. На следующий год — «Да-да-да, я слышал». Потом — «Я использовал». А на следующий год уже толпа народу: «Я использую Ubuntu». Мы чувствовали, что с каждым годом все больше людей приходило, которые понимали, и что они все больше понимали. Это было очень ярко выраженно. Студенты тоже много используют Ubuntu, даже в тех университетах, которые ее не используют. Но на государственном уровне все пока не так хорошо. Есть очень много хороших специалистов в государственных структурах, которые хотели бы двигаться в этом направлении. У монополистов есть свои методы саботировать эти процессы.

L: А как ты сам заинтересовался свободным ПО?

M: В детстве я программировал. Был Turbo Pascal и Quick Pascal. Более крупная фирма, победившая в университете, использовала не вполне честные методы продвижения своего продукта, заявленные возможности оказались не до конца реализованы, и мне это не понравилось. А потом я хотел иметь легальное программное обеспечение. Купил на какой-то выставке редактор Ami Pro, за $100, и вдруг оказалось, что функциональность идентична редактору от Mircosoft, но я не могу им пользоваться, потому что невозможен ни с кем никакой обмен данными. Я понял, что вся система функционирует против потребителей. И потом, когда я работал в ассоциации потребителей, моя специализация была — защита потребителей от монополистов, главные направления — в IT и в отоплении.

L: Т. е. интерес к свободному ПО родился из соображений защиты потребителя?

M: Да. Причем в начале 2000-х, когда я первый раз инсталлировал себе Linux, я знал, что будут проблемы с драйверами, будет тяжело, но я хотел показать на своем примере, что это возможно. Наверное, полгода прошло, пока у меня начало работать все, что я хочу, а еще через полгода я начал чувствовать, что обратно вернуться не хотелось бы. Сейчас... У нас выпускают дистрибутив Baltix на базе Ubuntu, когда просто ставишь компакт-диск, нажимаешь «Инсталлировать», и имеешь полный набор программ для своего десктопа, в отличие от закрытого программного обеспечения, где инсталлируешь операционную систему, потом офисный пакет, потом антивирусы всякие... Понимаешь, что в Linux на порядок быстрее и проще все сделать. Сейчас уже совсем другая ситуация, все в пользу Linux. А почему люди не переходят?

L: А почему?

M: А потому, что люди консервативны, наверное. Я администрировал компьютеры в одной фирме, учредитель поручил проверить, все ли у нас легально, и оказалось, что Microsoft Office не оплачен. Тогда я предложил директору OpenOffice, бесплатно, и он конечно же был за. Но я опасался, что многие сотрудники будут против, и проинсталлировав его, не сказал им, что это новая программа, а сказал: «Я обновляю наш офис». Все спокойно это восприняли, и даже сказали, что у них на порядок стало лучше управление таблицами. Люди боятся даже не программы, а нового названия.

3 Даниэль Надь, Будапешт, Венгрия (LVEE 2011)

Даниэль Надь: Каждый второй семестр я веду курсы по криптографии в Будапештском университете имени Лоранда Этвёша на факультете естественных наук, на кафедре операционных исследований.

L: Каких специалистов готовит кафедра ?

Д: В основном математиков, и кроме этого есть еще такая специальность, «математик-программист». Ожидаемое место их последующей работы — это либо академическая сфера, либо индустрия информационных технологий.

L: Как в учебном процессе используется свободное ПО?

Д: Достаточно широко. В общем-то, во всех сферах деятельности. Терминалы, которыми пользуются студенты, на Linux. Сейчас это тонкие клиенты. Очень много свободного софта типа Octave, но в любом случае есть конечно и коммерческие приложения, которые в академической сфере очень распространены.

L: В том числе и потому, что приходится ориентироваться на рынок труда?

Д: Частично да, а частично потому, что некоторым приложениям, типа Mathematica, пока еще полноценной свободной замены нет. Но там где можно, совободное ПО широко используется. Большинство статей пишется в TEX, графики рисуются в gnuplot — это две программы, которые знает каждый, кто прошел через кафедру математики.

Unix-подобные системы, еще до того, как они стали свободными, получили достаточно широкое распространение в венгерских вузах, потому что еще в начале девяностых или даже в конце восьмидесятых по некоторым западным грантам было получено оборудование с Solaris от Sun, AIX от IBM, поэтому Unix в венгерской системе образования уже очень давно, и менять его по частям на свободное ПО было легко и безболезненно для преподавательского состава и для научных работников, и естественно, для студентов также. Резкого перехода в духе «Раньше мы использовали что-то другое, а теперь будем использовать Linux» — не было. Постепенно, терминал за терминалом это старое железо от Sun менялось на Linux-станции.

L: А как сам ты заинтересовался свободным ПО, помнишь?

Д: Это было в другом университете. Я сам учился в техническом университете, и там я впервые увидел Интернет, как таковой, старые Unix'ы, которые тогда еще были новыми. Мне очень нравилась система Silicon Graphics. И когда в одной из лабораторий поставили на клиенты Linux, я понял, что на дешевом домашнем компьютере, который я могу иметь, тоже может быть что-то похожее. Тогда я начал интересоваться Linux, и... пошло-поехало.

L: Как вообще свободное ПО чувствует себя в Венгрии?

Д: Достаточно комфортно. В Венгрии есть люди с большим энтузиазмом и большой энергией, которые его распространяют, так что венгерская локализация достаточно качественная. Потом, по причине восточно-европейского безденежья очень многие принимают решение пользоваться свободным ПО. Единственное, где у свободного ПО серьезные трудности — это крупные компании и государственные учреждения. С бесплатного — и откатов нет :)

4 Виталий Липатов, Санкт-Петербург, Россия (LVEE Winter 2012)

Незадолго до конца LVEE Winter 2012 мы взяли интервью у Виталия Липатова — члена команды разработчиков дистрибутива ALT Linux и генерального директора компании Etersoft, известной в первую очередь своей собственной коммерческой версией проекта wine. Вопросы от имени LVEE задавали Дмитрий Костюк и Инна Рыкунина.

LVEE: Вопрос, который всегда легко задавать: а как получилось, что ты заинтересовался свободным ПО?

Виталий Липатов: Где-то году в 92-м или 93-м папа купил мне книжку «UNIX — универсальная среда программирования». Тогда я её прочитал и ничего не понял, большие системы в начавшуюся эпоху персональных компьютеров казались далёкими, как динозавры. А лет через шесть, когда я заканчивал ВУЗ, меня пригласили работать в центральный НИИ судовой электротехники в лабораторию, где разрабатывалось программное обеспечение для пригородной электрички. Я попал в проект не с самого начала, и когда пришёл, выбор операционной системы уже был сделан, а выбирали между QNX и Linux.

L: А что вообще операционная система делает в электричке?

B: Она стоит на контроллерах, которые управляют тормозами, дверями, скоростью движения, пультом машиниста (там монитор, который выводит все показатели, лампочки зажигаются предупредительные)... в поезде множество подсистем, они традиционно разрабатываются разными организациями, как, например, система безопасности электропоезда. Интересно, что тогда мы одни из первых начали применять Ethernet, у нас был проложен кабель сквозь весь поезд. Ранее обычно применялись разные стандартные и нестандартные протоколы на базе последовательного интерфейса.

L: Вагоны в локальной сети...

B: Да, это был 2000 год. И QNX не выбрали — потому, что если умножить стоимость лицензии QNX на количество вагонов, которые предполагалось выпускать серийно... Выбрали Linux, тогда это был Slackware 3.5 с ядром 2.0.34, по-моему. Тогда уже, конечно, существовали и другие дистрибутивы...

L: Но люди любили Slackware, поэтому электричка работала на ней.

B: Это была первая Linux-система, которую я увидел, и она меня очень впечатлила. Так я познакомился с Linux. Немного поработав с ним, почувствовал, что освоить это невозможно. Бесчисленное количество команд с разными параметрами, и совершенно нереально запомнить, что у «cp» есть параметр «-a», а у «tar» нужно писать «xvfz», чтобы что-нибудь распаковать... Но потом мозг как-то со всем этим справился, и я понял, что это та самая система, о которой я всегда подсознательно мечтал. Недостатков даже не было заметно. И вообще, тогда в Linux не было недостатков, они появились намного позже :)

L: Тогда их ещё не успели написать :)

B: И вот, мы для 386 процессоров сделали систему, и всё очень быстро работало.

L: Проект был закончен?

B: Да, электричку мы сделали, но её не пустили в серию, к сожалению. Был изготовлен поезд в шесть вагонов, он прошёл все испытания.

L: В общем, интерес к Linux заметно пережил этот проект. А собственно интерес к свободному ПО?

B: Он появился немного позже. Эта мысль достаточно долго входила в сознание. Я перечитывал ту книжку про Unix, а ещё как раз недавно появился Интернет, и я узнал, что есть другие люди,

которые под это что-то пишут, и втянулся. А потом попал в ALT Linux Team.

L: А кстати, как это произошло?

B: Для дальнейшей деятельности нам потребовался новый дистрибутив, а выбирали мы почему-то между только появившимся Mandriva RE Spring 2001 и ASP Linux. Я долго выбирал. ASP Linux выглядел красиво и профессионально. Но интуитивно меня тянуло к собранному в ALT Linux дистрибутиву. Сейчас я уже не помню подробностей. Возможно, уже тогда я заметил наличие сообщества разработчиков.

Параллельно с НИИ я работал в компании, где писал конфигурации для 1С, причем с нуля. Работал один, написал их три штуки для разных направлений деятельности, заодно что-то администрировал — скучная такая деятельность для творческого человека. Там у меня был сервер на Linux, на котором лежали файлы баз данных для 1С:Предприятия. А и одновременно это был мой рабочий компьютер и мне нужно было на нём писать код, поэтому я в итоге озаботился, а как бы запустить 1С:Предприятие в Linux. Это был год, наверное, 2002 или 2003.

L: Вот, оказывается, откуда растёт WINE@Etersoft!

B: Это уже позже, в 2004 году на выставке Softool в Москве мы от имени свежеорганизованного Etersoft показали, что на WINE@Etersoft запускается 1С:Предприятие. Причём показывали, подключаясь к Linux-серверу с тонкого клиента, то есть в режиме терминального доступа. А в то время я только знал, что есть такой пакет wine, позволяющий запускать Windows-программы, попробовал его... Долго подбирал настройки и библиотеки, и в итоге 1С:Предприятие запустилось. Работать оказалось не совсем возможным, слишком много багов... Тогда я уже был пользователем ALT Linux. В то время wine там собирал генеральный директор ALT Linux Алексей Евгеньевич Новодворский. Пару раз он исправлял пакет по моим замечаниям, а потом прислал письмо в рассылку разработчиков ALT Linux: «Уважаемые коллеги, представляю вам нового сопровождающего пакет wine Виталия Липатова, прошу любить и жаловать». То есть меня как бы поставили перед фактом. А я был очень благодарен за оказанное доверие...

Но Etersoft стал заниматься wine позже: когда мы наконец открылись и думали внедрять ALT Linux в офисах, оказалось, что никому это вообще-то не нужно, потому что для Linux нет необходимых прикладных программ. Тут мы поняли, что в офисах долж-

но работать 1С:Предприятие, причем не просто запускаться, а реально работать, в сетевом режиме, когда есть много пользователей. Именно это востребовано, а особенно на терминальном сервере, т. е. в режиме, который позволяет получить от 1С:Предприятия версии 7.7 максимум производительности. Мы на это нацелились, сделали, и в конце 2005 года начали продавать первые версии. Собственно, первую версию продали уже чуть ли не под новый год.

L: Новогодний подарок. Оно работало?

B: Весьма относительно... Но у нас были принципы, которых мы придерживались: бухгалтерия в нашей компании использовала 1С:Бухгалтерию, и мы запускали её под Linux.

L: А кстати, как по поводу изменений в wine? Этот проект ведь прославился тем, что периодически происходят изменения в апстриме, когда что-то внезапно может сломаться. Насколько часто и насколько болезненно такая ситуация бьёт по коммерческой версии?

B: Раньше, много лет назад, когда релизов wine почти не было, просто забирали из cvs какой-то срез кода и пытались им пользоваться. Стабильность действительно оставляла желать лучшего. Но потом разработчики wine перешли к концепции, что после любого изменения код не должен деградировать, стали требовать этого от разработчиков, появилась система тестирования. Требования ужесточили, в итоге сейчас wine перешёл на чёткие релизы. Каждый день новые коммиты, и каждую неделю выходит новый релиз. Причем невозможно прислать изменения, если ты перед этим не написал тест, который подтвердит, что проблема есть, а после твоего исправления она исчезает. Эти тесты требуют не просто так — в проекте работает автомат, который прогоняет их на десяти разных конфигурациях Windows и Linux и проверяет, нет ли деградации.

Из-за достаточно серьёзного тестирования новых патчей сейчас так сильно всё не ломается. Но до последнего времени мы свою текущую ветку разработки синхронизировали очень редко. У нас мало ресурсов, поэтому мы выбирали некоторую стабильную для нас версию оригинального wine. Дальше мы уменьшали количество багов в ней, не синхронизируясь с основной веткой. Но за последние два года wine дошел наконец до версии 2.0. Мы синхронизировались и написали робота, который периодически переносит изменения из апстрима в нашу ветку разработки. После сборки прогоняются тесты, проверяется, не появилось ли регрессий. Не так идеально, как в оригинальном wine, но всё-таки тесты мы запускаем.

L: Вы передаёте свои исправления в апстрим?

B: Конечно. Раньше — больше, апстрим wine был менее строгим. И сейчас иногда принимают что-то. Теоретически мы должны от этого выигрывать, но на практике этого не замечаем. Количество хаков, которые мы не можем послать, просто потому они не будут приняты, больше, чем то, что у нас приняли. Мы специально для патчей, которые у нас должны бы принять, ведём отдельную ветку. Она свободная, и wine в ALT Linux собирается именно из неё. На нашем ftp-сервере доступны сборки и под другие системы.

L: Насколько широко расходится WINE@Etersoft как коммерческий продукт?

B: Я так понимаю, что это зависит от роста популярности Linux. Процесс взаимосвязанный. Иногда я слышу, что без нашей деятельности по помощи в запуске Windows-программ в таком объёме движения не было бы. Раньше, когда компанию 1С просили сделать версию под Linux, они отвечали: «У нас там нет пользователей». Мы этот порочный круг разомкнули, и появились пользователи под Linux. Теперь в компании 1С сделали Linux-версию, пока только серверную часть, их самый дорогой продукт. Клиент может быть запущен либо под Windows, либо можно подключиться через браузер, но это доступно не для всех конфигураций, и не вся функциональность доступна, поэтому наше решение по запуску Windows-клиента остаётся востребованным.

На этом проекте нам не удаётся много зарабатывать, и, соответственно, быстро его развивать. Специфика такова, что готовых разработчиков не существует, и новых сотрудников нам приходится обучать практически с нуля — как писать код, как отлаживать, какие приёмы использовать для поиска ошибок в бинарном коде. Деятельность не очень благодарная и мало кому нравится.

L: А вы не эксплуатируете практикантов?

B: Вообще-то студенты проходят к нам на практику, но в основном те, кто уже у нас работает. А так... В городе множество компаний, предлагающих более высокие зарплаты, но у нас есть другие преимущества: свободный график, особая атмосфера, интересная работа... Для сотрудников разработана целая система, интегрированная Bugzilla, которая высчитывает рабочее время, учитывает решённые задачи.

L: Если в двух словах коснуться ваших продуктов, менее известных широкой публике, чем wine?

B: Наши другие продукты — во многом эксперимент, что же ещё будет востребовано по теме миграции на Linux. Есть SQL-транслятор SELTA@Etersoft, который позволяет отказаться от Microsoft SQL Server. Версию 2.0 хотим сделать в виде прокси-сервера, со стороны сети выглядящего точно как MS SQL. Там много сложностей: хранимые процедуры, встроенные функции... Это пример полностью нашего коммерческого продукта, построенного, правда, на нашей собственной сборке PostgreSQL, которая конечно же свободно доступна. А пример нашего свободного продукта — UniOffice, который подменяет Microsoft Office на OpenOffice для VBA-приложений. У нас есть и решение для организации терминального доступа, основанное на NX — RX@Etersoft. Ещё у нас есть совершенно другая деятельность: мы делаем промышленные системы управления.

L: Например?

B: Например, мы разрабатываем ПО для электростанций в России. К нам обратилась компания, по заказу которой производились преобразователи частоты в Китае. Там делали железо и программу управления к нему под Windows. Она закрывалась каждые несколько часов из-за различных ошибок... А тут требовалась бесперебойная работа в течение года. Там всё очень серьёзно, преобразователь управляет нагнетателем, подающим воздух для горения газа, и если пропорция смеси нарушается, всё чревато аварийной ситуацией. Нас нашли, уговорили, мы в очень сжатые сроки сделали этот заказ, последние штрихи наводили уже на электростанции. И вот система уже более 5 лет работает, без сбоев. Мы делаем и другие подобные системы управления для судов:. Не системы навигации, там свои поставщики, а мониторинг, контроль, системы сигнализации, пульты в рубку и др. И в том и том случае на базе ALT Linux. **L: На нём плавают корабли и работают электростанции. А как твои сегодняшние впечатления? На что похоже LVEE?**

B: Достаточно необычно. Я участвовал в двух видах мероприятий: тех, что проводятся ALT Linux — в основном с людьми, близкими к ALT Linux Team, в них ещё сильна академическая составляющая. И другого типа — те, что проводятся крупными спонсорами, такими как IBM или Sun, для своих целей, в фешенебельных отелях, на тысячи людей.

У вас отличается формат и люди по глубине задействованности: совершенно другой пласт, с которым на тех мероприятиях не сталкиваешься. Мне очень понравилось мероприятие, рад был встретить такое количество небезразличных к свободному ПО людей.

май 2012

ГЕОГРАФИЯ КЛАСТЕРОВ

ЗАВОЮЙ МИР ВМЕСТЕ С НАМИ!

WARGAMING.NET

LET'S BATTLE

Научное издание

ОТКРЫТЫЕ ТЕХНОЛОГИИ

Сборник материалов
Восьмой Международной конференции
разработчиков и пользователей
свободного программного обеспечения
Linux Vacation / Eastern Europe 2012

(Гродно, 07–10 июня 2012 г.)

Фото на обложке Арсения Случевского

Ответственный редактор Д.А. Костюк
Компьютерная верстка А.О. Шадура

Подписано в печать 25.05.2012.
Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Ризография. Усл. печ. л. 9,3. Уч.-изд. л. 6,02.
Тираж 235 экз. Заказ 3947.

Издатель и полиграфическое исполнение:
частное производственно-торговое унитарное предприятие
«Издательство “Альтернатива”».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
ЛИ N 02330/0494051 от 03.02.2009.
ЛИ N 02330/0150460 от 25.02.2009.
Пр. Машерова, 75/1, к. 312, 224013, Брест.