

Веб-Аналитик.ИНФО

www.web-analitik.info

ХОСТИНГ

Роутинг IPv4 и IPv6 в KVM в Hetzner
Система инициализации Systemd
Обзор панели ISPmanager
MySQL в tmpfs

ВЕБ-РАЗРАБОТКИ

Что мешает сайтам стоять дорого?
SharpDevelop AddIns:
расширяем кругозор
Получаем Object из формы
Функции наносят ответный удар
jQuery без рамок

ИНТЕРНЕТ

Как заработать на поддержке
интернет-проектов?
Тест популярных браузеров
на совместимость с HTML5

ЖЕЛЕЗО

Полтора месяца с iPad'ом
Планшет Toshiba FOLIO 100
Медиаплеер с начинкой

АВТОРСКАЯ РУБРИКА

Мой DockBar
OpenSource Java Math Library
Почему Git

РЕАЛИЗ
UBUNTU 10.10
УЖЕ В СЕТИ

Интервью с
Юрием Азовцевым
Координатором
Нижегородской Группы
Пользователей Linux





Цифровая реальность

Приветствую всех читателей! Это седьмой номер журнала Веб-Аналитик.ИНФО, то есть седьмой месяц подряд, начиная с апреля этого года, мы выпускаем регулярно каждый месяц новый номер. Как вы все знаете, наш журнал абсолютно бесплатен, он распространяется в электронном виде в формате PDF. В связи с чем хочу поделиться с вами одним интересным наблюдением, которое касается электронных/цифровых/виртуальных/интернет средств массовой информации и нашего журнала в частности. Замечу, что это наш не единственный проект, еще мы более двух лет издаем электронный бесплатный игровой журнал X-CONSOLE.COM, к которому все ниже описанное имеет также непосредственное отношение.

На протяжении этих семи номеров Веб-Аналитик.ИНФО мы постоянно сталкиваемся с недопониманием текущей цифровой реальности со стороны потенциальных рекламодателей. Рекламодатели до сих пор ни не могут перестроиться или не понимают, что электронные СМИ, в нашем конкретном случае бесплатный журнал в PDF, имеют куда более широкую аудиторию и распространение среди читателей. Ведь очевидно, что журнал в PDF можно скачать бесплатно из любой точки земного шара. Его проникновение в интернете, в отличие от ларьков и киосков, торгующих бумажной прессой, в тысячи раз шире. Более того, цифровой вариант журнала не накладывает абсолютно никаких ограничений на его количество, то есть тираж издания не фиксированный, он просто безразмерный. Журнал есть всегда, он есть везде и постоянно будет иметься в наличии! Вот эту простую и понятную истину, как оказалось, весьма сложно донести до потенциального рекламодателя. Давайте для интереса, как говорится «на пальцах», разберем конкретный вариант с электронным вариантом Веб-Аналитик.ИНФО.

При переговорах всегда следует первый вопрос: какой тираж вашего издания, и в каких городах вы его распространяете? Возьмем последний сентябрьский номер журнала. По нашим счетчикам этот номер скачан уже более 38 тысяч раз. И это подсчеты только с наших трех зеркал, которые мы даем в качестве источника для скачивания журнала. А в общей сложности все шесть номеров скачало порядка 320 тысяч человек, только вдумайтесь в эту цифру! То есть в среднем — это около 53 тысяч читателей на один номер! При этом мы не учитываем всевозможные торреты, варьезники, интернет-развалы плана «качки все у нас» и так далее, где можно запросто найти Веб-Аналитик.ИНФО (о чем в конце этого текста). Скажите, какой узкоспециализированный журнал по данной тематике может похвастаться таким тиражом? Никакой! И не только потому, что в принципе у нас в нише всего два конкурента, но и потому что ни одно издательство не рискнет выпустить такой огромный тираж в узконаправленной специфике. Например, тираж у одних наших коллег 5 тысяч экземпляров, а у других 7 тысяч экземпляров и распространение у журналов весьма и весьма ограниченное. Да, у них есть еще подписка, но не думаю, что она больше или хотя бы даже равна указанному номиналу тиража, при этом кто знает, сколько идет в возврат. Разве есть сравнение? Думаю, что нет.

Напоследок предлагаю интересный тест. Откройте поисковик Google, Яндекс, да любой нормальный поисковик и наберите в нем фразу «Скачать журнал Веб-Аналитик.ИНФО». Набрали? Вот теперь почувствуйте степень распространения журнала в интернете, а точнее его проникновение на бесплатном виртуальном рынке киосков и ларьков. Друзья, верьте в электронные СМИ и цифровые носители, именно за ними наше будущее. В конце концов, мы же не собираемся вечно вырубать лесные массивы, которые, между прочим, дают нам жизнь.

Станислав Горнаков
Руководитель проекта Веб-Аналитик.ИНФО

Хочешь быть в курсе всех событий?
Подпишись на нашу RSS-ленту новостей или Twitter
Мы вещаем круглые сутки семь дней в неделю



Свежие новости
в Twitter'e



RSS-лента
новостей



Путеводитель в мир
Интернет-Технологий

Веб-Аналитик.ИНФО

Журнал выходит ежемесячно
и распространяется бесплатно

Издательская группа

ООО «Издательство «Стангор»
www.stangor.ru
Журнал Веб-Аналитик.ИНФО
www.web-analitik.info

Издатель и руководитель проекта
Станислав Горнаков sg@stangor.ru

Зам. главного редактора
Игорь Редько ir@stangor.ru

Арт-директор
Светлана Петрова
art-dis@stangor.ru

Редакторы
Сергей Рубан
Ирина Войкова
Андрей Коваль
Василий Сенянский
Марат Ягудин
Игорь Войков
Игорь Периодов

Корректоры
Ксения Рубова
Максим Веров
Роман Ветрин
Максим Злобин

Авторы
Станислав Горнаков
Владимир Пузанов
Виталий Родненко
Ольга Рунова
Игорь Сычев
Андрей Росляков
Антон Чернышов
Ксения Бобкова
Дмитрий БездыХанный
Алексей Сигов
Александр Федора
Василий Чуранов
Антон Регада
Илья Рабинович
Максим Васильев
Павел Канахистов
Вадим Марковцев
Алексей Шаферов
Сергей Кузнецов
Игорь Снытко
Максим Олейник

Отдел маркетинга и рекламы

Руководитель отдела
Виктор Прудников pr@stangor.ru

Реклама в журнале
siteweb@web-analitik.info

Реклама на сайте журнала
siteweb@web-analitik.info

Издание зарегистрировано в Комитете
Российской Федерации по Печати
Свидетельство № ФС 77 - 39005

Журнал издает
ООО «Издательство «Стангор»

Для пресс-релизов и информации о
пресс-конференциях
siteweb@web-analitik.info

Авторам
www.web-analitik.info/authors/

За достоверность рекламной информации
ответственность несут рекламодатели. Ре-
кламные материалы не редактируются и
не корректируются. Редакция ждет ваших
откликов и писем читателей. Фотографии,
рукописи и другие печатные материалы
не редактируются и не корректируются.
При цитировании или перепечатывании
материалов журнала ссылка на сайт www.
web-analitik.info и название журнала Веб-
Аналитик.ИНФО обязательна. Полное или
частичное воспроизведение материала
журналов возможно только с письмен-
ного разрешения Издательства Стангор.
Мнение редакции журнала может не со-
впадать с мнением авторов статей публи-
куемых в журнале. Все товарные знаки
принадлежат их владельцам.

© ООО «Издательство «Стангор»
© WEB-ANALITIK.INFO
© ВЕБ-АНАЛИТИК.ИНФО



ХОСТИНГ

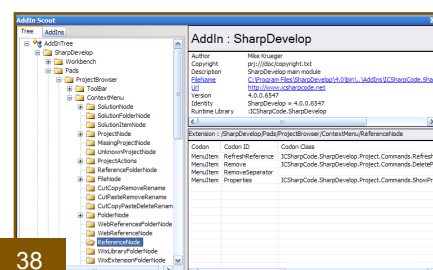
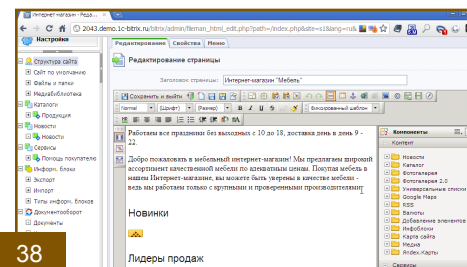
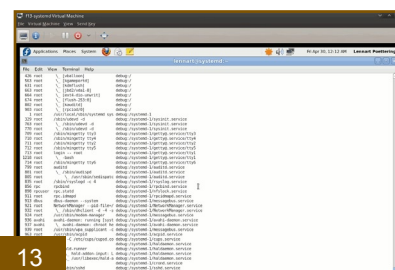
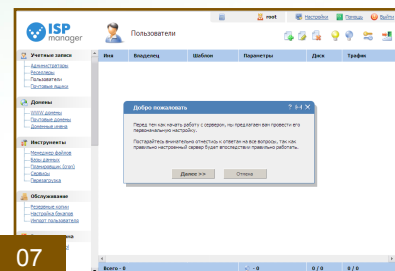
- Роутинг IPv4 в KVM на примере Hetzner..... **05**
- Обзор панели управления хостингом ISPmanager..... **07**
- Система инициализации Systemd..... **13**
- MySQL в tmpfs..... **23**
- Репликация в PostgreSQL 8.x: упрощаем работу со Slony..... **25**

КАТАЛОГ

- Хостинг-компании..... **26**

ВЕБ-РАЗРАБОТКИ

- Что мешает сайтам стоить дорого, а исполнителям быть честнее?..... **28**
- Разработка приложения на основе протокола OAuth для Twitter API на PHP..... **29**
- Получаем Object из формы..... **31**
- Калининградская Аврора в полотнах Эрмитажа 1С-Битрикс..... **32**
- Функции наносят ответный удар..... **34**
- jQuery без рамок..... **36**
- SharpDevelop AddIns: расширяем кругозор..... **38**
- 7-Zip из .NET или как я делал open source проект на CodePlex..... **42**



КАТАЛОГ

Веб-студии..... 44

ИНТЕРНЕТ

Интервью с Юрием Азовцевым
[Координатор Нижегородской Группы Пользователей Linux (NNLUG)]... 46

Как заработать на поддержке интернет-проектов?..... 50

Где найти покупателей в Рунете?..... 51

Ложь, большая ложь, и антивирусы..... 53

ЖЕЛЕЗО

Мультимедийный планшет Toshiba FOLIO 100..... 57

Медиаплеер с начинкой..... 59

Полтора месяца с iPad'ом..... 62

АВТОРСКАЯ КОЛОНКА

DockBar или почему я стал строить свой велосипед..... 66

OpenSource Java Math Library..... 69

Почему Git..... 71

Тест популярных браузеров на совместимость с HTML5..... 73



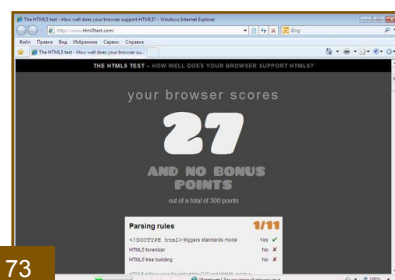
46



57



01



73



ХОСТИНГ

**Роутим IPv4 в KVM
на примере Hetzner**

**Обзор панели управления
хостингом ISPmanager**

**Система инициализации
Systemd**

MySQL в tmpfs

**Репликация в PostgreSQL 8.x:
упрощаем работу со Slony**

Роутим IPv4 в KVM на примере Hetzner

Автор

Владимир Пузанов
Национальная академия
статистики, учета и аудита
Ассистент кафедры
информационных систем
и технологий
farcaller.net

Данная статья посвящена вопросам правильной настройки IPv4 в сетевых конфигурациях, аналогичных тем, что применяются в Hetzner на базе KVM (так же потенциально подходит для любых других HVM, и для Xen). Примеры конфигурации интерфейсов основаны на ifup, так как на хосте и большинстве виртуалок у меня Ubuntu.

Стартовые условия

У данного хостера вы сходу получаете один IPv4 адрес. Дополнительно к нему вы можете попросить/приобрести до трех IPv4-адресов и блоки подсетей. В данном примере будет рассматриваться следующая конфигурация:

IP хоста: 123.45.12.48

Шлюз: 123.45.12.1

Дополнительные IP: 123.45.53.11, 123.45.53.12, 123.45.53.13

Дополнительная подсеть: 123.45.90.112/29

Hetzner шлет все пакеты на MAC-адрес первого (обычно единственного) интерфейса eth0, и ожидает, что все исходящие пакеты будут уходить с этого же MAC-адреса. Через саппорт можно попросить поставить другие MAC-адреса для дополнительных «поштучных» IP-адресов, что позволяет закинуть vnet* интерфейсы KVM-ок в один мост с eth0, но эту конфигурацию я рассматривать не буду.

Итак, что необходимо получить в итоговой конфигурации:

- отсутствие NAT, каждая VM получает минимум один публичный IP-адрес;
- хост занимается всем роутингом:
 - трафик между разными VM;
 - трифик между VM и интернетом;
 - централизованно фильтрует пакеты в iptables (можно ограничивать взаимодействие VM друг с другом и интернетом)
- IP-адреса из выделенных подсетей не теряются (в демонстрационной подсети адреса 123.45.90.112 и 123.45.90.119 обычным методом использовать нельзя – это адреса сети и бродкаста).

Для настройки сети понадобятся следующие пакеты: bridge-utils, dhcp3-server, iptables, iproute2.

Настраиваем основной интерфейс хоста

В /etc/network/interfaces необходимо описать eth0 следующим образом:

```
auto eth0
iface eth0 inet static
address 123.45.12.48
netmask 255.255.255.255
gateway 123.45.12.1
pointopoint 123.45.12.1
```

Маска 255.255.255.255 означает, что все исходящие пакеты будут уходить на шлюз, через опцию pointopoint ifup определяем, что гейтвей технически находится на этом же

интерфейсе (куда он, естественно, иначе не попадает из-за ограниченной маски). Это эквивалентно следующим правилам роутинга:

```
123.45.12.1 dev eth0 proto kernel scope link src
123.45.12.48
default via 123.45.12.1 dev eth0 metric 100
```

Настраиваем мосты для VM

Поскольку в задачу входит ограничение трафика между VM, их нельзя втиснуть в один мост. Для каждого дополнительного IP-адреса и каждого адреса из подсети /29 необходимо описать отдельный мост:

```
auto br112
br112 iface inet static
address 172.30.112.1
netmask 255.255.255.0
pre-up brctl addbr br112
post-up route add -host 123.45.90.112 br112
post-down brctl delbr br112
```

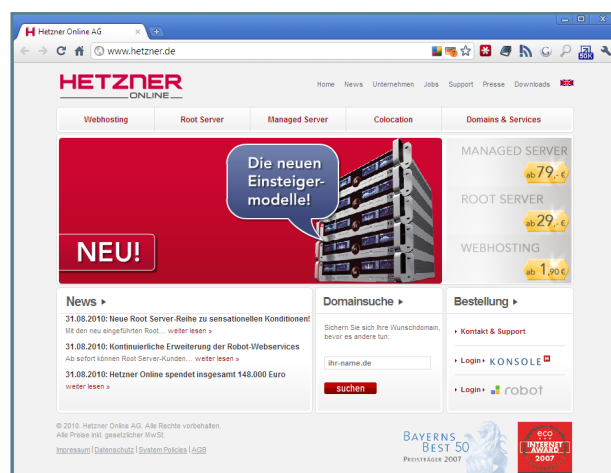
Каждый мост будет называться br<xxx>, где xxx – последний байт IPv4 адреса. На нем поднимается приватный IPv4 адрес из 172.30.xx.0/24 (зачем это надо описано далее), тут же в описании интерфейса привязан brctl (создание моста в pre-up, уничтожение в post-down), и настраивается роутинг (публичный IP-адрес определяется на этом мосту).

Абсолютно идентично описываются br11, br12, br13, и br113-br119.

Настраиваем DHCP

Так как pointopoint конфигурация затруднена в процессе инсталляции некоторых OS, на каждом мосту для VM будет работать DHCP-сервер, который даст базовую конфигурацию IPv4, достаточную для завершения установки.

Модифицируем конфиг dhcp3 в /etc/dhcp3/dhcpd.conf следующим образом: ►



```

authoritative;
default-lease-time 3600;
max-lease-time 3600;
ddns-update-style ad-hoc;
log-facility local7;
use-host-decl-names on;

option subnet-mask 255.255.255.0;
option domain-name «lan»;
option domain-name-servers xx.yy.100.100, xx.yy.99.99, xx.yy.98.98; <-- заменяем на актуальные DNS-серверы

subnet 172.30.11.0 netmask 255.255.255.0 {
    option routers 172.30.11.1;
    range 172.30.11.10 172.30.11.200;
}

subnet 172.30.12.0 netmask 255.255.255.0 {
    option routers 172.30.12.1;
    range 172.30.12.10 172.30.12.200;
}

subnet 172.30.13.0 netmask 255.255.255.0 {
    option routers 172.30.13.1;
    range 172.30.13.10 172.30.13.200;
}

subnet 172.30.112.0 netmask 255.255.255.0 {
    option routers 172.30.112.1;
    range 172.30.112.10 172.30.112.200;
}

...

subnet 172.30.119.0 netmask 255.255.255.0 {
    option routers 172.30.119.1;
    range 172.30.119.10 172.30.119.200;
}

```

Также объясняем, на каких интерфейсах dhcp3 должен работать: в файле /etc/default/dhcp3-server изменяем INTERFACES:

```
INTERFACES="br11 br12 br13 br112 br113 br114 br115 br116 br117 br118 br119"
```

После чего перезапускаем dhcp3: /etc/init.d/dhcp3-server restart

Настройка iptables

```

#!/bin/sh
it="/sbin/iptables"

MY_NET="123.45.90.112/29 123.45.53.11/32 123.45.53.12/32 123.45.53.13/32"
MY_NET_DHCP="172.30.0.0/16"
HOST_IP="123.45.12.48"
MAIN_IF="eth0"

# INPUT
$it -F INPUT

$it -A INPUT -p udp --dport 67 -i br+ -j ACCEPT

```

```

# настраиваем другие политики INPUT
$it -A INPUT -j DROP

# FORWARD
$it -F FORWARD

$it -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

for NET in $MY_NET; do
    $it -A FORWARD -i br+ -o $MAIN_IF -s $NET -j ACCEPT
# VM[fixed_ip] --> net
done

for NET in $MY_NET_DHCP; do
    $it -A FORWARD -i br+ -o $MAIN_IF -s $NET -j ACCEPT
# VM[dhcp] --> net
done

for SOURCE_NET in $MY_NET $MY_NET_DHCP; do
    for DEST_NET in $MY_NET; do
        $it -A FORWARD -i br+ -o br+ -s $SOURCE_NET -d $DEST_NET -j ACCEPT # VM <--> VM
    done
done

for NET in $MY_NET; do
    $it -A FORWARD -i $MAIN_IF -o br+ -d $NET -j ACCEPT # net --> VM
done

$it -A FORWARD -i $MAIN_IF -o gbr1 -d $MY_NET_PVT -j ACCEPT # net --> PVT

$it -P FORWARD DROP

# POSTROUTING
$it -t nat -F POSTROUTING
for NET in $MY_NET_DHCP; do
    $it -t nat -A POSTROUTING -o $MAIN_IF -s $NET -j SNAT --to-source $HOST_IP # nat the dhcp
# NAT для частных IP-адресов от DHCP
done

echo 1 > /proc/sys/net/ipv4/ip_forward

```

Настройка VM

Ну и последний этап – непосредственно настроить VM. Для libvirt/KVM описываем сеть через мост:

```

<interface type='bridge'>
    <source bridge='brXX' />
    <model type='virtio' />
</interface>

При этом на первичной загрузке виртуалка получит приватный адрес от DHCP, а уже после установки необходимо зафиксировать на ней публичный IP:

auto eth0
iface eth0 inet static
    address 123.45.90.116
    netmask 255.255.255.255
    gateway 123.45.12.48
    pointopoint 123.45.12.48

```

HETZNER

ONLINE

Автор

Станислав Горнаков
Microsoft MVP
www.gornakov.ru

Обзор панели управления хостингом ISPmanager

В чистом виде работать с веб-сервером человеку не искушенному в администрировании серверов достаточно сложно, поэтому для управления хостингом используются так называемые панели управления. Панель управления — это обычная графическая надстройка или точнее программа, которая позволяет работать с веб-сервером в визуальном режиме через Интернет браузер вашего компьютера. По большому счету если сравнивать панель управления хостингом с компьютером, то графическая и визуальная подсистема Windows — это и есть та самая панель управления вашим компьютером. Без такой панели вы бы имели дело с черными окошками и «кучей» команд введенных с клавиатуры.

На рынке хостинг-панелей есть несколько заслуживающих внимания программных решений. Это панели управления ISPmanager, Cpanel, DirectAdmin, Plesk и другие. Все провайдеры услуг, так или иначе, используют одну из вышеперечисленных панелей, либо делают свою собственную разработку, и внедряют ее на хостинг. Удобство работы и функциональность каждой панели естественно разнится, но направленность всех панелей одинакова. Это предоставление пользователю графического и визуального интерфейса для управления работой веб-сервера.

Панели управления представляют пользователю несколько уровней взаимодействия с сервером как администратор, реселлер или пользователь. В каждом конкретном случае, уровень доступа определяет набор элементов управления доступных для работы с сервером. Так администратор сервера будет иметь в своих руках максимальное количество «рычагов» управления сервером. Реселлер будет довольствоваться только теми функциями, которые отвел ему администратор сервера, а простой пользователь сможет осуществлять взаимодействие с сервером на

уровне клиента площадки, то есть без доступа к настройкам сервера. Сама панель в зависимости от пользователя системы будет просто переключаться из одного режима в другой, предоставляя каждой группе пользователей свои элементы управления сервером. Такая многоуровневая система прав в панелях управления — это основа всей работы с веб-сервером.

Панель ISPmanager

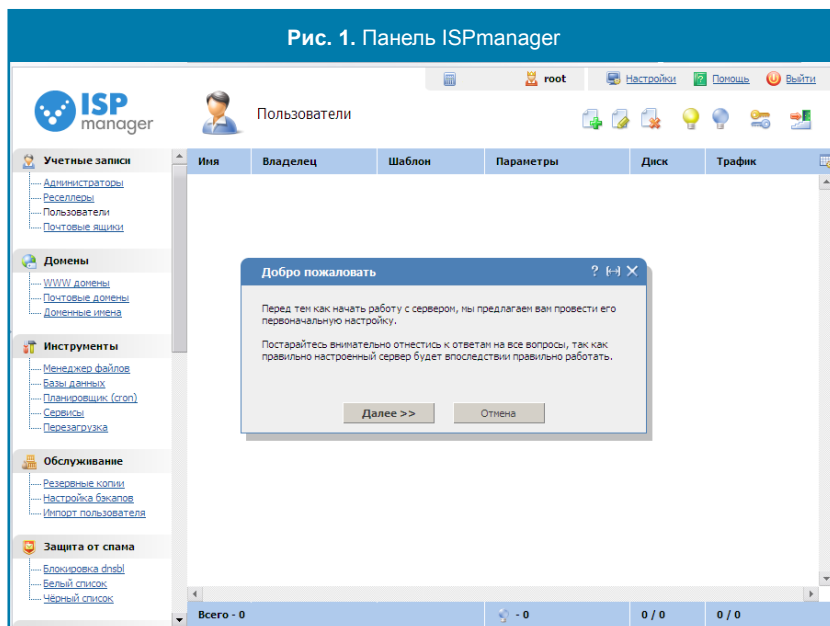
Это самая популярная в России панель управления, которая разработана компанией ISPsystem. Компания имеет целый комплекс программного обеспечения для работы с веб-сервером: панель управления ISPmanager в трех версиях ISPmanager Lite, ISPmanager Pro, ISPmanager Cluster, панель VDSmanager для управления выделенным сервером или виртуальным выделенным сервером, панели DSmanger, DNSmanager,

IPmanager. Полная информация о панелях управления компании ISPsystem, а также форум поддержки вы найдете на сайте компании по адресу в Интернете <http://ispsystem.com/ru/index.html>. На этом сайте также находится масса различной документации и большой набор отличных видеороликов, рассказывающих об основах работы с панелями управления. Далее я предлагаю на конкретном примере, шаг за шагом познакомиться с ISPmanager Pro на базе обычного виртуального выделенного сервера, подробно изучив все аспекты работы с данной панелью.

Администрирование виртуального выделенного сервера

После того как вы приобретете виртуальный выделенный сервер у хо-

Рис. 1. Панель ISPmanager



стинг-провайдера, вы сможете приступить к работе на площадке. Для доступа к панели управления сервером наберите в адресной строке браузера адрес:

`https://XX.XXX.XX.XX/manager/ispmgr` или `https://ваш_сайт/manager/ispmgr`

Появится форма авторизации, где вам необходимо будет ввести логин и пароль, выбрать тему оформления и язык интерфейса. После ввода всех данных нажмите кнопку **ОК** и вы попадете в панель управления своей площадкой (рис. 1). В зависимости от того с какими правами доступа вы вошли на площадку (администратор, реселлер или пользователь), количество и подбор категорий будет отличаться.

Первичная настройка площадки

Входя первый раз на площадку, система автоматически предложит вам пройти первоначальный процесс настройки некоторых параметров сервера (рис. 1). В первом диалоговом окне **Настройки сервера** вам предстоит выбрать часовой пояс. Откройте список **Часовой пояс** и найдите в перечне стран и городов ваше место жительства. Следующее окно **Журнал работы с Web-сервером** предложит настроить обработку статистических данных площадки и включить ведение журнала ошибок. Выберите значение по умолчанию и нажмите кнопку **Далее**. В последнем диалоговом окне **Пароли доступа к серверу** (рис. 2) вам будет предложено изменить пароли к площадке и базе данных. Если вы получили пароль к площадке от провайдера услуг, то обязательно придумайте более надежный пароль. Если вы создавали сами виртуальный выделенный сервер, то оставьте поля для пароля площадки пустыми, а вот пароль для базы данных нужно назначать в любом случае новый.

Обзор возможностей ISPmanager

После первичных настроек вы попадете на площадку своего виртуального выделенного сервера. Прежде чем переходить к работе с сервером предлагаю познакомиться с основными категориями панели управления, которые сосредоточены с левой стороны окна в меню навигации.

- **Учетные записи** — эта категория позволяет работать с учетными записями пользователей площадки. В ней вы как администратор системы можете создавать новых пользователей площадки и редактировать имеющиеся записи. Каждый новый пользователь системы должен принадлежать к одной из групп (администратор, реселлер или пользователь), что в дальнейшем определит его уровень доступа к настройкам веб-

Рис. 2. Смена паролей для доступа к площадке и базе данных

сервера.

- **Домены** — категория для работы с доменными именами и почтовыми доменами. Именно в этой категории происходит создание на площадке новых доменов и редакция уже имеющихся.

- **Инструменты** — большая тематическая категория, где сосредоточены основные инструменты площадки. Такие как файловый менеджер, создание и редактирование баз данных, брандмауэр и планировщик заданий Сгop, мониторинг включенных сервисов сервера (ссылка **Сервисы**), а также механизм перезагрузки сервера.

- **Обслуживание** — эта категория позволяет настроить создание резервной копии или восстановить площадку из ранее сделанной резервной копии. При создании задания для резервной копии, нужно учитывать ваш уровень доступа к системе. Если создается резервная копия от лица администратора сервера, то будет создана копия всего виртуального выделенного сервера, а если от лица пользователя, то будет сделана резервная копия только принадлежащих пользователю данных (сайты, почтовые ящики, базы данных).

- **Защита от спама** — черный и белый списки для настройки фильтра нежелательной и желательной почты. В отличие от всевозможных автоматических фильтров, черный список как мне кажется, имеет более гибкое устройство.

- **Статистика** — большая подборка статистических данных площадки, а именно: статистика использования ресурсов системы, подсчет трафика, текущая активность и журнал событий.

- **Настройка сервера** — очень важная категория, содержащая основные настройки сервера. В этой категории кроме статической информации о конфигурации сервера, выделенных для вас IP-адресах, настроек баз данных, располагаются ссылки, с помощью которых можно включить или выключить, а также установить расширения

PHP и модули PERL. Для некоторых систем управления сайтом необходимы отдельные расширения PHP и вы с помощью раздела **Расширения PHP** можете включить необходимые расширения. Например, для работы CMS 1С-Битрикс требуются дополнительные расширения `posix.co`, `zlib.co`, для UMI. CMS расширения `zlib.co`, `xml.so`, для DataLife Engine — `zlib.co`, `xml.so` и так далее. В документациях к CMS обычно перечисляются все необходимые расширения PHP. Чтобы включить одно из расширений PHP, необходимо перейти к категории **Настройки сервера**, выбрать ссылку **Расширения PHP** и найти в таблице нужное расширение (рис. 3). Возле названия расширения с помощью графической иконки с изображением лампочки будет указан статус данного расширения. Если лампочка желтого цвета, то расширение включено, а если голубая, то отключено. Для включения расширения выделите курсором мыши строку с названием расширения и нажмите в панели инструментов кнопку с изображением желтой лампочки. Сервер включит расширение, и перезагрузится. Таким способом вы можете включать любое из установленных на сервере расширений. В том случае, если расширение не установлено, можно воспользоваться кнопкой панели инструментов **Установить** с изображением компакт-диска в лотке. По выбору этой кнопки откроется диалоговое окно со списком не установленных, но имеющихся на сервере расширений. Выберите в списке необходимое расширение и нажмите кнопку **ОК**. Аналогичная схема работы применяется и для модулей PERL.

- **Настройки** — эта категория содержит перечень ряда общих настроек сервера. Наиболее важный раздел в этой категории это **Настройки DNS**. Именно здесь можно узнать, какие DNS серверы прописаны для вашей площадки. Выберите ссылку **Настройки DNS** и вам откроется диалоговое окно **Настрой-**

ки доменов по умолчанию (рис. 4). В этом окне в поле **Серверы имен** перечислены DNS серверы, которые необходимо использовать у своего регистратора домена. Кроме этого в поле **Почтовые серверы** прописаны серверы входящей и исходящей почты, которые нужно использовать в почтовых клиентах. Записи могут вам показаться несколько необычными, так, например, привычные всем `pop.ваш_домен.ru` (сервер входящей почты) и `smtp. ваш_домен.ru` (сервер исходящей почты) имеют запись `mail.ваш_домен.ru` для обоих серверов.

- **Дополнительные приложения** – доступ к веб-интерфейсу почтового сервера Web-Mail, а также доступ к phpMyAdmin.

- **Справка** – большая подборка всевозможной справочной информации по панели, а также online доступ к обучающим видеороликам. К слову, в самой панели в некоторых местах встроены ссылки на видеоролики по теме, так если вы, например, затрудняетесь создать домен в категории **Домен**, то вам будет предложено просмотреть соответствующее интерактивное руководство с набором исчерпывающей информации.

Создаем нового пользователя

Перед тем как создавать `www`-домены, нужно обязательно создать хотя бы одного пользователя в системе. Администратор сервера в целях безопасности не должен обращаться к сайтам напрямую. Все действия по управлению сайтами необходимо производить от лица пользователя, собственно благодаря этому правилу администратор также не имеет доступа к площадке по FTP. То есть вы как администратор виртуального выделенного сервера должны вначале создать пользователя площадки и уже от его лица работать с сайтами. При этом самого пользователя можно наделять различными правами, вплоть до разрешения установки дополнительных расширений PHP и модулей PERL, и так далее.

Чтобы создать нового пользователя, перейдите к категории **Учетные записи** и выберите ссылку **Пользователи**. Далее щелкните на панели инструментов кнопку **Создать нового пользователя** (листок бумаги с зеленым крестиком). На экране монитора появится диалоговое окно **Новый пользователь** с четырьмя вкладками **Пользователь**, **Права**, **Ограничения**, **Ресурсы** и **Заметки** (рис. 5). На первой вкладке **Пользователь** вам предстоит указать имя и пароль для будущего пользователя системы. Поле **Домен** сейчас лучше оставить пустым, поскольку вслед за этим разделом мы рассмотрим создание домена более подробно. В списке **IP-адрес** будет прописан один из ваших адресов площадки, а в поле **E-mail** впи-

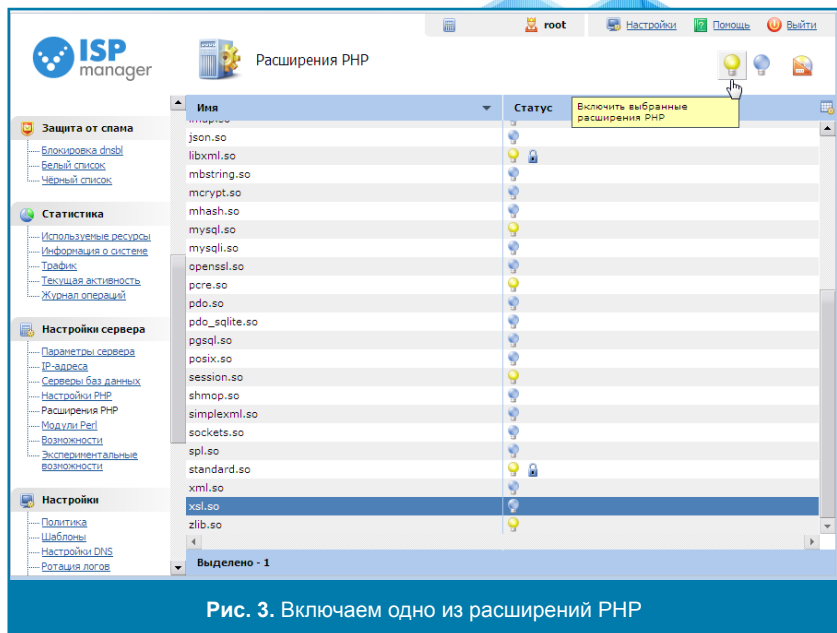


Рис. 3. Включаем одно из расширений PHP

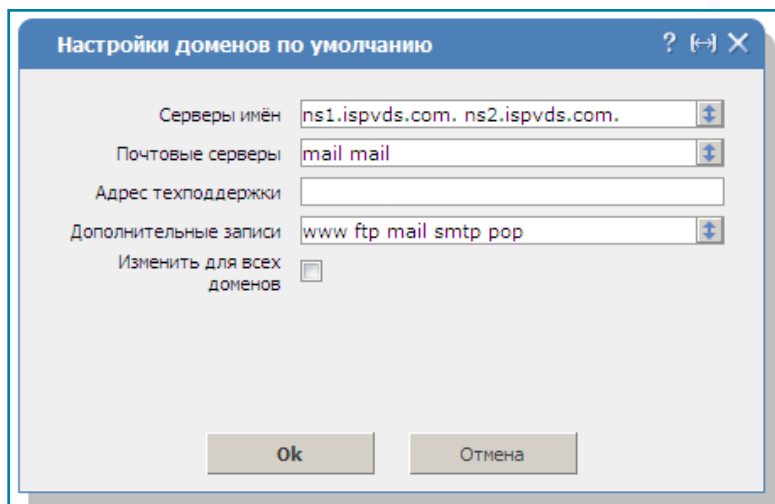


Рис. 4. Настройки DNS сервера домена

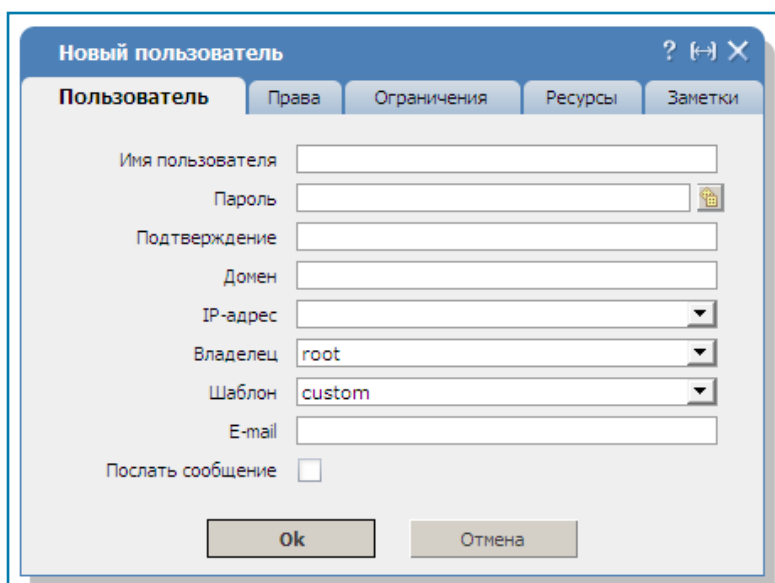


Рис. 5. Первая вкладка Пользователь

шите адрес электронной почты пользователя.

Переходим ко второй вкладке **Права** (рис. 6). Эта вкладка имеет набор судьбоносных для пользователя флажков, с помощью которых вы определите, какие функции сервера будут доступны данному пользователю. Следующая вкладка **Ограничения** призвана помочь вам ограничить некоторые технические характеристики площадки для конкретно взятого пользователя (рис. 7). Последняя вкладка **Ресурсы**, позволит указать ограничения на ресурсы площадки для будущего пользователя веб-сервера (рис. 8).

Создаем на площадке WWW домен

После создания нового пользователя можно переходить к созданию доменов и почтовых доменов. Сам пользователь тоже может создавать домены, если при создании нового пользователя вы на вкладке **Ограничения** в поле **WWW домены** указали больше чем один домен. То есть и вы и пользователь имеете возможность самостоятельно создавать новые домены на своей территории или на своем тарифном плане. Процесс создания домена и для пользователя и для администратора одинаков.

Чтобы создать новый WWW домен, перейдите к категории **Домены** и выберите ссылку **WWW домены**. Затем нажмите на панели инструментов кнопку **Создать новый домен**. Откроется диалоговое окно **Новый WWW домен** (рис. 9). В поле окна **Доменное имя** необходимо прописать доменное имя без префиксов `http://` и `www`, но с доменной зоной. В этом поле можно указать только одно доменное имя, а вот в следующем поле **Псевдонимы**, вы можете указать любое количество дополнительных доменов, которые впоследствии будут ссылаться на основной домен указанный в поле **Доменное имя**. В этом поле можно уже прописать адресацию к домену через буквы `www`. Ниже в списке представлен пример того как могут выглядеть оба поля этого окна.

Доменное имя
gornakov.ru
Псевдонимы
www.gornakov.ru
gornakov.com
www.gornakov.com

Поле **Корневая папка** оставляем со значением по умолчанию **auto**. Далее в списке **Владелец** выбираем из списка пользователя, которому будет принадлежать создаваемый домен. Список **IP-адрес** и поля **Кодировка**, **Индексная страница** оставляем в значениях по умолчанию. В поле **Администратор** нужно прописать адрес электронной почты пользователя. Список **Автоподдомены** позволяет либо включить, либо выключить автоматическое создание поддомена на площадке. Послед-

Рис. 6. Установка прав для пользователя

Рис. 7. Задаем ограничение площадки

Рис. 8. Задаем ограничение ресурсов веб-сервера

ний список **PHP** позволяет определить есть ли у пользователя возможность использовать PHP скрипты. Очевидно, что такая возможность у него должна быть, поэтому включаем пользователю возможность работы с PHP. И в конце вы можете включить возможность работы с Cgi-bin, а также открыть доступ по SSI и SSL, но это возможно только в том случае если на этапе создания нового пользователя вы избрали эти опции на вкладке **Права**. После заполнения всех полей нажмите кнопку **ОК**. Через некоторое время сервер создаст домены.

Создаем почтовый домен

Почтовый домен создается в категории **Домены** в разделе **Почтовые домены**. Открыв этот раздел, нажмите кнопку с графическим изображением листа бумаги и зеленого крестика. На экране появится диалоговое окно **Новый почтовый домен** (рис. 10). Все что вам нужно – это указать доменное имя и пользователя, которому будет принадлежать новый почтовый домен. Так в поле **Доменное имя** пропишите будущий домен и далее выберите пользователя в поле **Владелец**, а затем нажмите кнопку **ОК**. Через считанные мгновения почтовый домен будет создан. На этом вы, как администратор, завершили все необходимые действия по организации минимальных сервисов для площадки и можно переходить на уровень пользователя.

Уровень пользователя

Панель управления ISPmanager для простого пользователя будет выглядеть несколько иначе, чем для администратора сервера, а точнее ряд управляющих функций будет недоступен. Если вы администратор сервера, то перейти на уровень пользователя можно прямо из панели ISPmanager. Чтобы это сделать проследуйте к категории **Учетные записи** и к разделу **Пользователи**. Затем выделите курсором мыши пользователя системы и нажмите на панели инструментов кнопку **Вход в систему на уровне пользователя**. Возвратиться на уровень администратора можно по выбору ссылки **goot** в строке приветствия.

Простой пользователь системы может заходить в панель управления площадкой, набрав в строке браузера адрес:

https://адрес_сайта/manager/ispmgr

После входа в систему вы откроете панель управления ISPmanager в том виде, в котором она будет доступна пользователю сервера. Заметьте, что в навигационном меню некоторые команды (в отличие от администратора) исчезли, но и появилось несколько

Рис. 9. Окно создание нового WWW домена

Рис. 10. Окно создания почтового домена

новых записей. В целом все записи интуитивно понятны, и проблем в работе с панелью обычно не возникает ни у одного из пользователей. Чтобы установить на сайт, например одну из CMS, необходимо выполнить несколько важных действий на площадке, а именно создать базу данных, разобраться с файловым менеджером и FTP аккаунтом. Кроме этого не помешает создать несколько почтовых ящиков, поэтому предлагаю в следующих подразделах изучить поочередно ряд выше озвученных действий.

Создаем базу данных

Для того чтобы работать с одной из CMS вам потребуется создать на площадке базу данных. Некоторые хостинг-провайдеры делают это за вас, а некоторые предоставляют такую возможность пользователю. Здесь все зависит от тарифного плана и схемы работы самого провайдера услуг. Мы изучаем ISPmanager на базе владельца вирту-

ального выделенного сервера, поэтому создание баз данных целиком и полностью лежит на плечах администратора. Чтобы создать базу данных, перейдите к категории **Инструменты** и далее откройте раздел **Базы данных**. Нажмите на панели инструментов кнопку **Создание базы данных**. Откроется диалоговое окно **Новая база данных** (рис. 11).

В поле **Имя базы данных** необходимо указать имя будущей базы. Тип и кодировку базы данных нужно указать исходя из ваших потребностей. В списке **Пользователь**, если вы создаете свою первую базу данных, необходимо выбрать пункт **Создать нового пользователя**, и далее в поле **Новый пользователь** указать имя пользователя базы данных отличное от имени пользователя площадки. То есть сейчас на площадке создается и база данных и пользователь этой базы данных. Впоследствии при создании новой базы данных вы можете использовать уже созданного пользователя базы данных либо создать нового. В оставшихся ▶

двух полях придумайте сложный пароль и нажмите кнопку **ОК**. На основе введенных данных, на площадке будет создана новая база данных. Сами данные базы (имя пользователя, название базы данных и пароль) вы будете использовать при установке на хостинг системы управления сайтом или других необходимых вам целей.

Файловый менеджер и атрибуты файлов

В панель управления ISPmanager встроен неплохой файловый менеджер со всеми атрибутами присущему этому виду программ (рис. 12). С помощью встроенного менеджера вы можете удалять, копировать, вставлять, вырезать любые файлы и папки. Закачивать на площадку и разархивировать архивы, создавать новые файлы или папки, редактировать содержимое любых файлов и, что самое главное, изменять атрибуты файлов (права на запись и чтение).

Изменение атрибутов файлов или папок иногда требуется для корректной установки и работы с CMS. Если одна из CMS требует выставления прав на запись или чтение для некоторых папок или файлов, то можно воспользоваться для этих целей файловым менеджером. Для этого выделите один из файлов или папку и выберите на панели инструментов кнопку **Просмотреть свойства выделенного элемента**. Откроется диалоговое окно **Атрибуты файла**. Чтобы установить необходимый атрибут файлу или папке, просто укажите цифровое значение в поле **Права доступа**. Необходимые значения атрибутов для папок или файлов обычно указываются в документации по CMS. Выше перечисленные действия по изменению атрибутов файла вы также можете сделать из своего FTP клиента или по SSH.

Доступ по FTP

По умолчанию в ISPmanager созданный пользователь площадки получает сразу один FTP аккаунт. Доступ по FTP осуществляется по IP-адресу, имени и паролю пользователя площадки. Если вам необходимо создать еще один FTP аккаунт на определенную директорию, то перейдите к категории **Главное** и разделу **FTP аккаунт**. Нажмите на панели инструментов кнопку **Создать новый FTP аккаунт?** и в появившемся диалоговом окне **Новый FTP аккаунт** укажите имя и пароль, а также раздел сайта (список **Домашняя директория**) к которому новый пользователь будет иметь доступ (рис. 13).

Создаем почтовый ящик

Для создания нового почтового ящика перейдите к категории **E-mail** и к разделу **Почтовые ящики**. Нажмите в этом

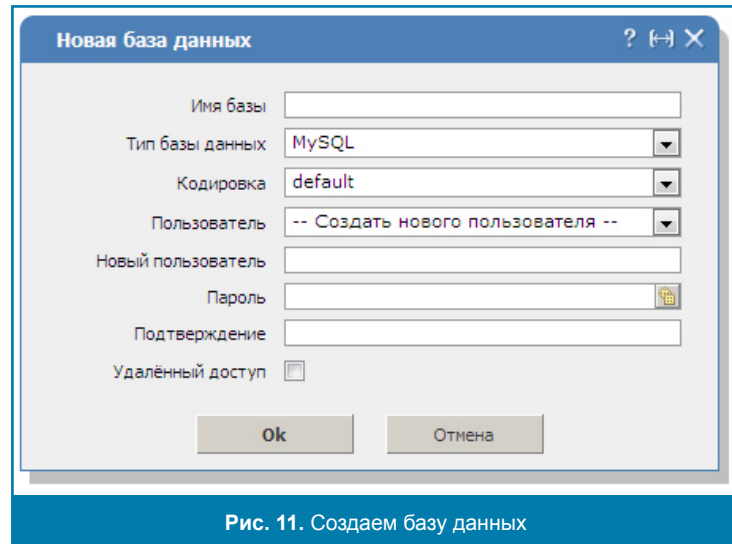


Рис. 11. Создаем базу данных

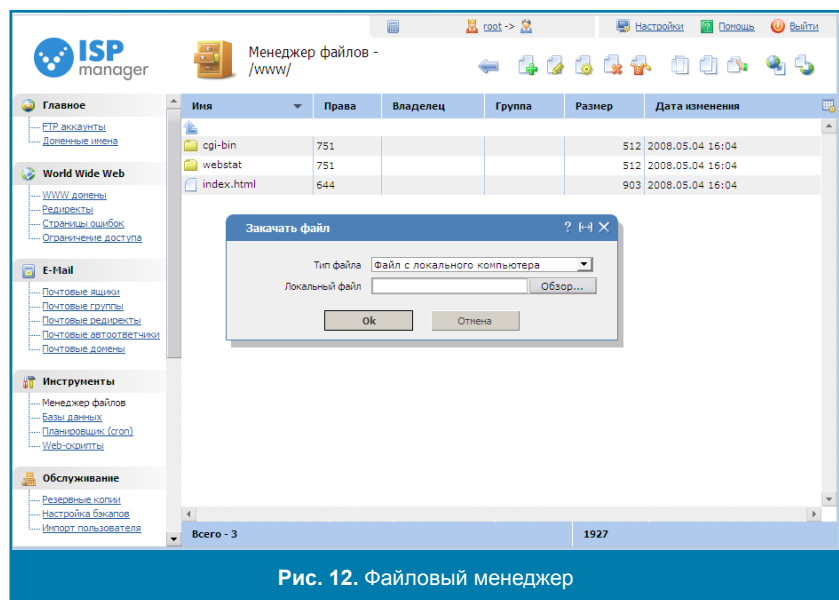


Рис. 12. Файловый менеджер

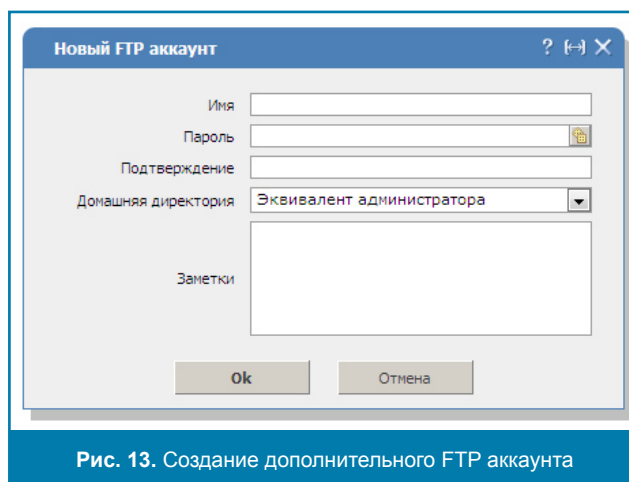


Рис. 13. Создание дополнительного FTP аккаунта

разделе на панели инструментов кнопку **Создать новый почтовый ящик**. Откроется диалоговое окно **Новый почтовый ящик** (рис. 15). В этом окне вам предстоит в поле **Имя** указать имя будущего почтового ящика без указания домена, то есть слово до «собачки» (@). Затем в списке **Домен** избрать домен,

для которого создается ящик, далее ввести пароль, и указать максимально возможный объем ящика в мегабайтах. Дополнительно можно указать e-mail на который будут пересылаться копии писем. По окончании всех настроек нажмите кнопку **ОК** и ящик с указанными настройками будет создан. ■

Система инициализации Systemd

Автор: Lennart Poettering, Red Hat Inc.

Перевод: Антон Чернышов
Novell Certified Linux Engineer, Ubuntu
Certified Trainer
Компания НОУ УЦ R-Style,
Преподаватель

tux-the-penguin.blogspot.com

E-mail: vmlinuz@pisem.net

Каждая Unix-система имеет процесс со своим специальным идентификатором, равным 1. Этот процесс запускается ядром прежде всех остальных и является родительским для всех процессов, которые не имеют собственного родителя. Благодаря этому он может делать много чего такого, что не могут остальные. Например, отвечает за такие вещи, как инициализация и поддержка работы пользовательского пространства в процессе загрузки системы.

Исторически, в Linux в качестве такого процесса выступает старый добрый `sysvinit`, преклонный возраст которого сказывается все чаще и чаще. Многие пытались предложить ему замену, но реально прижился только Upstart (что интересно, один из вариантов перевода слова «upstart» — «выскачка». Прим. перев.), который уже нашел свое место во всех основных дистрибутивах.

Как уже отмечалось выше, главная обязанность `init` — максимально быстро инициализировать и сделать доступным пользовательское пространство. К сожалению, традиционная схема инициализации SysV делает это не так быстро как хотелось бы.

Для организации быстрого и эффективного процесса загрузки системы решающее значение имеет следующее:

- Запустить **минимум необходимого**
- Попытаться запустить параллельно **все остальное**

Что это означает на практике? Запустить минимум необходимого означает запустить лишь самые необходимые сервисы либо отложить запуск каких-либо из них до тех пор, пока они реально не понадобятся. Иногда мы заранее знаем, какие сервисы нам понадобятся (`syslog`, системная шина D-Bus и т. п.), но так бывает не всегда. К примеру, демон `bluetoothd` не должен быть запущен до тех пор, пока не будет подключен Bluetooth-адаптер либо пока какое-то из приложений не захочет установить с ним соединение через интерфейс D-Bus. Аналогично для систе-

мы печати: если машина физически не подключена к принтеру, и/или никакие приложения не пытаются напечатать что-нибудь, то необходимости запускать демон печати, такой как CUPS, нет. Так же и для Avahi: если машина не подключена к сети, или никакое из приложений не захочет использовать его API, ее также нет необходимости запускать. Это справедливо даже для SSH: пока кто-либо не захочет зайти на машину, в нем нет необходимости, его нужно запустить лишь тогда, когда кто-то захочет установить первое соединение (и это же относится ко многим машинам, где может быть запущен `ssh`, в тех случаях, когда к нему присоединяются раз в месяц или реже).

Параллельный запуск всего остального означает, что если нужно что-либо запустить, то мы должны делать это не последовательно, а пытаться запустить все одновременно, чтобы максимально нагрузить имеющиеся ресурсы системы, сократив время ее запуска.

Динамическое изменение аппаратного и программного обеспечения

Современные системы (в частности операционные системы общего назначения) являются весьма динамичными в плане своей конфигурации и использования: они мобильны, запускают и останавливают различные приложения, к ним подключают всевозможное оборудование. Система инициализации, отвечающая за работу сервисов, должна обнаруживать изменения в аппаратном и программном обеспечении, запускать (а иногда и останавливать) сервисы, поскольку они необходимы для запуска программ и/или для обеспечения поддержки определенного оборудования.

Большинство современных систем, которые пытаются распараллелить процесс запуска, по-прежнему синхронизируют запуск различных демонов: напри-

мер, Avahi для работы требуется D-Bus, поэтому сначала запускается D-Bus и только потом Avahi. Аналогично для других служб: для `libvirt` и X11 необходим HAL (да, я знаю, что на Fedora 13 службы игнорируют устаревший HAL), поэтому сначала запускается HAL, а уж потом `libvirt` и X11. А поскольку `libvirt` нужен Avahi, он ждет еще и запуска Avahi. К тому же всем упомянутым службам нужен `Syslog`, они все ждут его запуска. И так далее.

Параллелизация запуска сервисов, зависящих от сокетов (socket service)

Такого рода синхронизация запуска служб приводит к тому, что значительная часть запуска системы проводится последовательно. Правда было бы здорово избавиться от описанных требований такой синхронизации? На самом деле нет ничего невозможного! Для этого мы должны понять, что именно демоны требуют друг от друга и почему их запуск откладывается.

Что касается традиционных демонов Unix, на этот вопрос есть только один ответ: они ждут, пока сокет другого демона станет доступен для соединения. Обычно это сокет `AF_UNIX` в файловой системе, либо это может быть `AF_INET` [6]. К примеру, клиенты D-Bus ждут возможности подключения к `/var/run/dbus/system_bus_socket`, клиенты `syslog` — `/dev/log`, клиенты CUPS — `/var/run/cups/cups.sock`, NFS-клиенты ожидают `/var/run/rpcbind.sock` и открытия порта `portmapper` и т. д. Только подумайте — это единственное, чего они ждут!

Таким образом, если причина задержки только в этом, если мы сможем добиться того, чтобы перечисленные сокеты стали доступны ранее запуска соответствующих демонов, мы сможем серьезно ускорить процесс запуска системы и запустить намного больше процессов одновременно. Как же этого

добиться?

На самом деле в Unix-подобных операционных системах все довольно просто: мы можем создать слушающие сокеты, **зادолго** до реального запуска демона, а затем (когда он запустится) просто передать ему сокет посредством `exec()`. Таким образом, мы можем создать **все** нужные нам сокеты для **всех** демонов на первом шаге системы инициализации, а затем на втором шаге просто запустить все демоны одновременно. Если одна служба нужна другой, то ничего страшного не случится, даже если требуемая служба еще не запущена. Единственное что произойдет – требуемое соединение встанет в очередь на ожидание, и клиент будет ждать этого соединения (будет блокирован запросом). Но заблокирован только один клиент и только одним запросом. Кроме того, чтобы обеспечить необходимую параллелизацию запуска теперь совсем не обязательно конфигурировать зависимости между службами, ведь мы запускаем все сокеты одновременно и запущенный сервис будет уверен в том, что он сможет присоединиться к нужному ему сокету.

Это ключевой момент, и без него все остальное понять будет нельзя, поэтому я попробую пояснить все то же самое, но другими словами и с примерами. Допустим, вы запускаете демон `syslog` и его клиенты одновременно. При этом все сообщения его клиентов будут добавлены в буфер сокета `/dev/log`. Пока этот буфер не заполнится, клиентам не придется ждать, и они смогут запуститься. Как только `syslog` завершит процесс своего запуска, он обработает накопившуюся очередь сообщений. Другой пример: мы запускаем D-Bus и ее несколько клиентов одновременно. Если отправляется синхронный запрос (запрос, требующий мгновенного ответа) и поскольку ожидается ответ, клиент будет блокирован ожиданием, но только этот конкретный клиент и только до тех пор, пока D-Bus поймает этот запрос и обработает его.

По сути, буферы сокетов ядра работают на нас, помогая нам максимально распараллелить запуск служб, причем обработка последовательности запросов и их синхронизация выполняется ядром системы без какой-либо надобности вмешиваться в этот процесс из пользовательского пространства! И если все сокеты становятся доступными прежде реального запуска демонов, управление зависимостями также становится излишним (или, по крайней мере, вторичным). Если демону нужен другой демон, он просто подключится к нему. И если целевой демон уже запущен, то первый сразу же обратится к последнему. А если нет, то обратившийся демон не будет ждать его запуска, если только он не отправил синхронного запроса. И даже если этот другой демон вообще не запущен, это может быть сделано автоматически. С точки зрения первого демона, в описанных случаях

нет никакой разницы, следовательно, управление зависимостями становится не нужно или, по крайней мере, вторично. Процесс запуска происходит с оптимальной параллелизацией запуска демонов либо с их запуском по требованию. Кроме того, такой путь обеспечивает большую надежность, так как сокеты остаются доступными независимо от фактического состояния демона, который может иногда быть недоступным (например, из-за своего падения). В самом деле, вы можете легко написать демон, который будет запускаться, останавливаться (или падать), запускаться снова и т. п., причем клиенты не почувствуют остановок демона и ни один запрос не потеряется.

Ну что ж, теперь самое время для паузы, налейте себе еще кофе и будьте уверены, дальше будет еще интереснее.

Только сначала давайте проясним несколько вещей: это какая-то новая идея? Нет, определенно нет. Я назову вам одну из самых известных систем, работающих, как описано – это `launchd` от Apple (на MacOS прослушивание сокетов всех демонов осуществляется `launchd`). Сами сервисы могут все запускаться одновременно, и для них не надо настраивать никаких зависимостей. Это на самом деле действительно интересная идея, и именно в ней кроется причина того, почему MacOS удается обеспечить фантастическую скорость загрузки. Рекомендую посмотреть видео, где объясняется, как она работает.

К сожалению, предложенная идея так и не получила распространения вне лагеря поклонников Apple. Хотя она, на самом деле, даже старше, чем `launchd`. До нее хорошо известный почтовый `inetd` работал аналогичным образом: сокеты централизованно создавались демоном, который запускал необходимый сервис, передавая ему дескриптор файла сокета посредством `exec()`. Однако центральной идеей `inetd`, конечно же, были не локальные, а интернет-сервисы (хотя более поздняя реализация поддерживала и сокеты AF_UNIX). Также он не являлся инструментом для распараллеливания процесса загрузки системы или какого-либо управления зависимостями между сервисами. Для TCP-сокетов, `inetd`, в первую очередь, использовался таким образом, что для каждого входящего соединения новый экземпляр порождался новый экземпляр демона. Это означает, что для каждого соединения порождался и инициализировался новый процесс, что не очень хорошо для высокопроизводительных серверов. Тем не менее, с самого своего рождения `inetd` также поддерживает и другой режим, где демон порождался первым соединением и потом тот же экземпляр принимал последующие соединения (опция `wait/noawait` в файле `inetd.conf`, которая, к сожалению, плохо документирована). Запуск демона для каждого соединения, вероятно, создал для `inetd` репутацию

медленного сервера, что не совсем справедливо.

Параллелизация запуска служб, зависящих от D-Bus (d-bus services)

Современные демоны на Linux, как правило, предоставляют услуги посредством D-Bus, а не простого сокета AF_UNIX. Таким образом, появляется вопрос – а можем ли мы применить для них ту же логику распараллеливания запуска сервисов, что и для традиционных сокетов? Да, можем! В D-Bus уже есть все необходимое для этого: с помощью активации посредством D-Bus, служба может быть запущена при первом же обращении к ней. Такой способ запуска дает нам минимальную возможность синхронизации на запрос, которая нам нужна для одновременного запуска служб-потребителей и служб-поставщиков. Если мы хотим запустить Avahi одновременно с CUPS (CUPS использует Avahi для поиска mDNS/DNS-SD принтеров), то так и следует сделать, и если CUPS запустится раньше, то с помощью логики активации сервисов посредством шины, мы заставляем D-Bus создать очередь запросов, пока Avahi запускается.

Активация демонов посредством сокетов и шины D-Bus позволяет нам запустить **все** демоны параллельно, без какой-либо синхронизации. Это позволяет нам получить «ленивые» сервисы: если сервис используется редко, можно просто загрузить его по потребности, вместо того, чтобы запускать его во время загрузки

И если это не круто, тогда я не знаю, что круто!

Параллелизация заданий, имеющих отношение к файловой системе

Если посмотреть на графики визуализации загрузочного процесса текущих дистрибутивов, то можно увидеть множество точек, требующих синхронных операций, самое важное – это задания, связанные с файловыми системами. Обычно при загрузке системы много времени тратится на ожидание инициализации всех устройств, перечисленных в файле `/etc/fstab`, затем на проверку файловых систем, инициализацию квот. Только после того как это все закончится, мы сможем продолжить процесс загрузки.

Можем ли мы как-то решить эту проблему? Оказывается, можем! Harald ▶

Ноуег выдвинул идею использования для этого старой доброй `autofs`. Системный вызов `connect()` показывает, что одна служба заинтересована в другой. Аналогичным образом вызов `open()` (или подобный ему) показывает, что служба заинтересована в конкретном файле или файловой системе. Таким образом, чтобы улучшить параллелизацию запуска, можем заставить программы ждать, пока файловая система будет готова (смонтирована). Добиваемся мы этого следующим образом: создаем точки монтирования `autofs`, а затем, когда завершатся разные проверки целостности, инициализируются квоты и т. п., мы заменим ее реальной файловой системой. Когда работает `autofs`, попытки обращения к ней будут поставлены в очередь ядром системы и обратившийся процесс будет заблокирован (но только этот демон, и только эта конкретная попытка доступа). И таким образом мы сможем начинать запускать наших демонов задолго до того, как наши файловые системы станут доступны, без каких-либо потерь файлов и с максимальной параллелизацией.

Распараллеливание заданий, связанных с файловой системой и с запуском сервисов, не имеет смысла для корневой системы, поскольку именно там хранятся бинарники всех сервисов. Однако по отношению к файловым системам, таким как, например, `/home`, которые обычно намного больше, а иногда даже зашифрованы и возможно даже находятся на удаленном компьютере, такой подход может серьезно оптимизировать время загрузки системы. И уж конечно стоит отметить, что такие виртуальные файловые системы, как `procfs` или `sysfs`, никогда не должны монтироваться с помощью `autofs`.

Я не удивлюсь, если кто-то из читателей может посчитать, что интеграция `autofs` с системой инициализации ухудшит стабильность и что это решение вообще какое-то странное, страшное и уродливое. Однако неоднократно опробовав это решение, я могу вам сказать, что оно совершенно правильное. Использование `autofs` позволяет нам создать точку монтирования без необходимости обеспечить наличие самой файловой системы. На практике это влечет за собой только задержку доступа к ФС. Если приложение пытается получить доступ к файловой системе, находящейся под опекой `autofs` и мы очень долго не заменяем ее реальной файловой системой, оно будет находиться в состоянии `interruptible sleep`. Отметим также, что в любой момент, если точка монтирования так и не становится доступной к концу запуска (например, если `fsck` не удалось успешно проверить файловую систему), мы можем просто попросить `autofs` вернуть приложению код ошибки (например, `ENOENT`). Поэтому я полагаю, что хотя на первый взгляд включение `autofs` в систему инициализации может

показаться чересчур смелым, наш экспериментальный код показал, что эта идея работает на удивление хорошо на практике.

Сокращение количества скриптов, вызываемых во время запуска системы

Еще один момент, который вытекает из логики загрузки MacOS: shell-скрипты — зло.

Shell-скрипты имеют свои преимущества и недостатки. Их можно быстро писать и отлаживать, но они медленно работают. Классическая логика загрузки `sysvinit` построена вокруг скриптов. Будь то `/bin/bash` или любая другая оболочка (написанная, чтобы сделать работу скриптов быстрой), в конце концов этот подход все равно упирается в медлительность работы. В моей системе скрипты, находящиеся в `/etc/init.d/`, вызывают `grep` по крайней мере 77 раз, `awk` — 92 раза, `cut` — 23 и `sed` — 74. Каждый раз, когда вызываются эти команды (или какие-то другие), порождаются новые процессы, производится поиск библиотек и т. д. А затем, когда запущенный процесс выполняет какую-то простейшую операцию с текстовыми строками, он завершается. Естественно, что это все работает ужасно медленно. Ни один другой язык кроме оболочки не работает как описано. Кроме того, работа скриптов сильно зависит от различных переменных среды и тому подобного, причем это все трудно контролировать.

Итак, давайте избавимся от скриптов в процессе загрузки! Прежде чем мы сможем это сделать, нам нужно выяснить, как они используются сейчас. По большому счету картина такова — большую часть времени они заняты рутинными операциями. Большинство сценариев производят элементарные операции и вызов сервисов, поэтому эти операции можно переписать на C, либо отдельные исполняемые файлы, либо сами демоны, либо сделать конкретную операцию частью системы инициализации.

Маловероятно, что мы в ближайшее время полностью сможем избавиться от скриптов при старте системы. Переписывание их на C занимает много времени, в ряде случаев игра вообще не стоит свеч, а иногда наоборот, без скриптов вообще трудно обойтись. Но мы, безусловно, можем уменьшить их влияние на процесс загрузки.

Хороший показатель для измерения количества вызова скриптов в течение запуска системы — это PID процесса, который он получает сразу после запуска системы. Загрузитесь, войдите в систему, откройте терминал и введите `echo $$`. Попробуйте сделать это на вашей Linux-системе, а затем сравните результат с MacOS! (Подсказка:

на Linux PID будет порядка 1823; на MacOS примерный PID — 154).

Слежение за процессами

Центральная часть системы инициализации — это выполнение обязанностей по отношению к сервисам: она должна следить за ними. Перезапускать их, если они завершают свою работу. Если они падают, нужно собирать всю необходимую информацию о них и сохранить ее для администратора, предоставлять ее аварийным системам сбора информации (`crash dump systems`) и системам ведения журнала (например `syslog`) и/или системе аудита.

Также она должна позволять полностью завершать сервис со всеми своими потомками. Это, наверное, проще сказать, чем сделать. Традиционно в Unix, процесс, который дважды вызывает `fork()`, может избежать контроля своего родителя и родитель ничего не узнает об отношении нового процесса к тому, что он запустил. Например: неправильно написанный CGI скрипт, который вызван двойным `fork()` не прерывает свою работу при завершении Apache. Кроме того, вы даже не сможете выяснить его отношение к Apache, если не знаете его имя и назначение.

Итак, как же мы сможем усилить за процессами, чтобы они не убежали от нашей няни, и контролировать их как единое целое, даже если они форкаются огромное количество раз (в оригинале *gazillion times* :). Прим. перев.) ?

Разные люди предлагают различные решения. Скажу вкратце, что есть подходы, основанные на использовании интерфейса `ptrace` или `netlink` (интерфейс ядра, который позволяет получить сообщение `netlink` каждый раз, когда любой процесс в системе вызывает `fork()` или `exit()`), которые подвергались критике за топорность, неповоротливость и плохую масштабируемость.

А что же мы можем предложить? В ядре уже достаточно долго существует такая вещь, как `Control Groups` (aka «`cgroups`»). В основном они позволяют создавать иерархию групп процессов, которая непосредственно доступна через виртуальную файловую систему. Названия групп — это обычно названия директорий в этой файловой системе. Если процесс, принадлежащий к определенной группе вызывает `fork()`, его потомок становится членом той же группы. Если он не имеет привилегированного статуса, он никак не сможет этого избежать. Первоначально `cgroups` были добавлены в ядро для организации контейнеров: различные подсистемы ядра могут устанавливать лимиты на ресурсы (ограничение использования процессора или памяти) для определенных групп. Традиционные ограничения ресурсов (как реализовано ►

`setrlimit()` устанавливаются (в основном) только для каждого процесса. `Sgroups`, с другой стороны, позволяют устанавливать ограничение на целые группы процессов. `Sgroups` полезны также для обеспечения соблюдения ограничений в других случаях. Вы можете использовать его, например, для ограничений общего объема памяти или ресурсов процессора для Apache и всех его дочерних процессов. Поэтому неправильный CGI-скрипт не сможет сбежать от установленных ограничений ресурсов путем вызова `fork()`.

В дополнение к своим функциям по созданию контейнеров и использованию для ограничения ресурсов, `sgroups` являются очень полезными как средство для слежения за демонами: членство в `sgroup` надежно наследуется дочерними процессами и избежать этого невозможно. Существует система уведомлений, которая ставит в известность главный процесс, когда `sgroup` становится пустой. Вы можете найти `sgroups` для процесса путем чтения файла `/proc/$PID/cgroup`, следовательно, это очень хороший выбор средства для отслеживания процессов.

Контроль за средой исполнения процесса

Хорошая няня должна не только следить за демонами и контролировать их, но и создавать для них хорошее и безопасное окружение. Это означает, что мы не ограничиваемся очевидными параметрами, такими как настройка ограничений ресурсов для процесса посредством `setrlimit()`, идентификаторы пользователя и групп. Ядро Linux дает пользователям и администраторам достаточно контроля над процессами (некоторые из них в настоящее время используются редко).

Для каждого процесса нам нужно устанавливать контроль над использованием процессора, планировщика ввода-вывода, набор функциональных ограничений, привязку к процессору и, конечно же, дополнительные ограничения посредством `sgroup`. Самое главное здесь — это высокоуровневый контроль, такой как, например, монтирование файловой системы в режиме только чтения при вызове `mount` с опцией `bind`. Таким образом, можно запускать отдельные демоны так, что все (или некоторые) файловые системы будут ему доступны только для чтения. Это можно использовать для контроля за тем, что делают отдельные демоны, нечто вроде бюджетного варианта SELinux (хотя это, конечно, не призвано заменить SELinux).

Наконец, ведение логов является важной составной частью запуска сервисов: в идеале каждый бит информации, выдаваемый сервисом, должен записываться в журнал. Следовательно, система инициализации должна обеспечить процесс ведения логов для де-

монов, с которыми она работает, подключая стандартный вывод (`stdout`) и вывод ошибок (`stderr`) к демону `syslog`. А иногда даже к `/dev/kmsg`, который во многих случаях очень полезная замена `syslog` (те, кто делает встроенные системы, прислушайтесь, пожалуйста, к этому!).

Upstart, внимание, марш!

Во-первых, я бы хотел подчеркнуть, что мне на самом деле нравится код Upstart, он очень хорошо комментирован и его легко изучать. Другим проектам в этом отношении еще учиться и учиться (в том числе и моим собственным). В то же время я не могу согласиться с общим подходом, который реализован в Upstart. Но сначала несколько слов о проекте.

Upstart не разделяет код с `sysvinit`, а его функциональные возможности перекрывают возможности `sysvinit`, и обеспечивает совместимость с какой-то частью хорошо известных скриптов SysV. Его основная особенность — управление сервисами на основе событий. Запуск и остановка процессов связаны с «событием», которое происходит в системе, где «событием» может быть все что угодно: доступность сетевых интерфейсов, запуск какого-то ПО и т.п.

Upstart обрабатывает последовательность запуска служб посредством событий: если срабатывает событие `syslog-started` — оно используется как индикатор для запуска D-Bus, т. к. шина теперь сможет использовать `syslog`. А затем, когда срабатывает событие `dbus-started`, запускается NetworkManager, после чего он может использовать D-Bus, и так далее.

Можно сказать, что логическое дерево зависимостей, которое существует и хорошо понимается администратором или разработчиком, транслируется в систему событий и набор правил, описывающих реакции на события. Каждое логическое «для а нужно b» становится «запустить а, когда нужно b» плюс «остановить а, когда останавливается b». Это своего рода упрощение, особенно для кода самого Upstart. Тем не менее, я утверждаю, что такое упрощение на самом деле вредно. Прежде всего, система логических зависимостей никуда не девается, и тот, кто пишет файлы для Upstart, теперь должен транслировать эти правила в формате «события/действия» (два правила для каждой зависимости). Таким образом, вместо того, чтобы позволить компьютеру выяснить, что нужно делать на основе зависимостей, пользователь должен вручную перевести зависимости в простое правило «событие — реакция на него». Кроме того, поскольку информация о зависимостях не может быть декодирована, она не доступна во время работы системы, фактически это означает, что у администратора,

который пытается разобраться, почему что-то произошло, нет на это никаких шансов.

Кроме того, в случае Upstart логика работы с зависимостями переворачивается с ног на голову. Вместо того чтобы *свести к минимуму* объем работы (что является признаком и целью хорошей системы инициализации), на самом деле происходит увеличение количества работы. Другими словами, вместо того, чтобы имея четко поставленную цель, продумать все необходимые шаги сразу, она делает сначала один шаг, а потом делает все остальные шаги, которые к ней ведут. Или еще проще: то, что пользователь запустил D-Bus, ни в коей мере не означает, что NetworkManager должен быть также запущен (но это то, что будет делать Upstart). А должно быть с точностью до наоборот: когда пользователь обращается к NetworkManager, это признак того, что D-Bus также должна быть запущена (это, безусловно, именно то, что ожидают большинство пользователей, не так ли?). Хорошая система инициализации должна запускать только то, что нужно, и по требованию. Либо по потребности, либо максимально параллелизуя запуск. Однако она не должна запускать больше, чем необходимо, в частности не все из установленного, для чего может понадобиться этот сервис

(Честно говоря, и это только мое мнение, без претензии на истинность, что тут автор пытается притянуть за уши аргументы промис Upstart. Тем более что приведенные автором примеры относительно D-Bus и NetworkManager не соответствуют реальности. Прим. перев.)

Наконец я не вижу реальной ценности от логики, основанной на событиях. Мне кажется, что большинство событий, обрабатываемых Upstart, фактически не мгновенны, а имеют некоторую продолжительность: сервис запускается, работает и останавливается. Устройство «подключается-подключено-отключается». Точка монтирования находится в состоянии «монтирования-полностью смонтирована-или же она размонтирована». Кабель питания подключен, система работает от сети переменного тока, кабель питания отключен. Лишь немногие из событий системы инициализации должны обрабатываться точно в срок, большая часть из них — это старт, запуск и остановка. Информации об этом также нет в Upstart.

Мне известно, что некоторые проблемы, на которые я указал выше, в некоторой степени решаются последними изменениями в конфигурационных правилах Upstart, такими как, например, `start on (local-filesystems and net-device-up IFACE=lo)`. Тем не менее, я считаю, что это выглядит как попытка исправить систему (автор считает, что такое решение — «костыль»). Прим. перев.), основной дизайн которой является некорректной.

Кроме того, Upstart вполне подходит на роль няни для демонов, хотя что-то ▶

она делает сомнительными способами (см. выше).

Есть и другие системы инициализации, кроме `sysvinit`, `Upstart` и `launchd`. В большинстве из них подходы намного серьезнее, чем в `Upstart` или `sysvinit`. Наиболее интересной является `Solaris SMF`, которая поддерживает надлежащую обработку зависимостей между службами. Тем не менее, во многих отношениях она чересчур усложнена и, скажем так, несколько академична, чрезмерно использует XML и новые термины для известных вещей. Она также сильно завязана на специфичных для `Solaris` функциях, таких как `contract system`.

Собираем все вместе - systemd

Выше я объяснил, что должен делать хороший процесс с PID 1 и как работают существующие системы инициализации. Перед тем как перейти к самому главному, давайте сделаем еще паузу. Сходите налейте себе еще кружечку кофе. Это того стоит.

Как вы, наверное, уже догадались, те требования и возможности для идеальной системы инициализации, что я предложил выше, существуют уже сейчас в системе инициализации, названной нами `systemd`, которую я и хочу тут представить. Ниже приведен краткий список ее особенностей и их обоснование.

Поскольку `systemd` запускает и контролирует всю систему, откуда и ее название. Она реализует все возможности, указанные выше, и еще кое-что. Система построена на концепции модулей (*units*). Модули имеют имя и тип. Поскольку их конфигурация обычно загружается из файловой системы — названия модулей на самом деле представляют собой имена файлов. Например: модуль `avahi.service` считывается из конфигурационного файла с тем же именем и естественно, что он реализует работу с демоном `Avahi`. Существует несколько видов модулей:

1. service/сервис — наиболее очевидный тип модуля — это демоны, которые могут быть запущены, остановлены, перезапущены или перезагружены. Для совместимости с `SysV` в `systemd` помимо собственных файлов конфигурации для различных сервисов имеется возможность чтения классических скриптов инициализации `SysV`, а также она умеет разбирать заголовок `LSB`, если он существует. `/etc/init.d` является, следовательно, не более чем просто еще одним источником конфигурации.

2. socket/сокет — данный модуль реализует сокет, расположенный в файловой системе или в Интернете. В настоящее время поддерживаются сокеты `AF_INET`, `AF_INET6`, `AF_UNIX` типов `stream`, `datagram` и последовательных

пакетов (`sequential packet`). Также поддерживаются классические буферы `FIFO`. Каждый модуль типа «сокет» имеет соответствующий ему модуль «сервис», который запускается при попытке установки соединения с сокетом или буфером `FIFO`. Пример: `nscd.socket` при установке соединения запускает `nscd.service`.

3. device/устройство — этот модуль реализует устройство в дереве устройств `Linux`. Если устройство описано через правила `udev`, оно будет представлено в `systemd` как модуль `device`. Набор параметров устройства, установленный `udev`, будет использоваться `systemd` как исходный в определении зависимостей для этого типа модулей.

4. mount/точка монтирования — модуль реализует точку монтирования в файловой системе. `Systemd` контролирует все точки монтирования (их подключение и отключение), а также может быть использован для монтирования и размонтирования отдельных файловых систем. Файл `/etc/fstab` используется как дополнительный источник конфигурации для них, подобно тому, как сценарии инициализации `SysV` могут быть использованы в качестве дополнительного источника конфигурации для модулей `service`.

5. automount/точка монтирования с автоматическим подключением — модуль реализует точку монтирования с автоматическим подключением файловой системы. Каждый такой модуль имеет соответствующий ему модуль типа `mount`, который запускается (т. е. подключается), как только монтируемая файловая система становится доступной.

6. target/указатель — данный тип модулей используется для логической группировки других модулей: на самом деле сам по себе он ничего не делает, он просто указывает на другие модули, которыми таким способом можно управлять вместе. В качестве примера можно привести модули `multi-user.target`, который играет роль 5-го уровня запуска в классической схеме `SysV`, и `bluetooth.target`, активируемый сразу, как только становится доступен `Bluetooth`-адаптер, и запускающий сервисы, имеющие отношение к `Bluetooth` (которые обычно не запущены — `bluetoothd` и `obexd`) (т. е. по сути это придет на смену традиционным уровням запуска `SysV`. Прим. перев.).

7. snapshot/снимок — подобно предыдущему типу модулей снимки также ничего не делают сами, и их единственное предназначение заключается в ссылке на другие модули. Снимки могут быть использованы для сохранения состояния и возможности отката назад состояния всех служб и модулей системы инициализации. Он главным образом предназначен для двух случаев. Первый — чтобы позволить пользователю

временно перевести систему в какое-то специфичное состояние, например, однопользовательский режим с остановкой всех работающих сервисов, а затем легко вернуться в предыдущее состояние с одновременным запуском тех сервисов, которые были перед этим запущены. Второй вариант его использования — поддержка режима `suspend`: достаточно много сервисов не могут корректно работать с этой системой, и зачастую их лучше остановить перед засыпанием, а потом просто запустить.

Приведенные модули могут иметь зависимости друг от друга (как положительные, так и отрицательные, т. е. бывает, что одни без других не могут обойтись, а другие, наоборот, не могут терпеть друг друга). Например, модуль *устройство* может зависеть от какого-то модуля *сервис*, то есть как только устройство становится доступным — запускается определенный сервис. Модули *mount* имеют неявную зависимость от устройства, которое они пытаются смонтировать. Также они наследуют неявные зависимости от префиксов путей к точкам монтирования (например, модуль, подключающий `/home/lennart`, неявно зависит от модуля, подключающего `/home`) и так далее.



Краткий перечень остальных функциональных возможностей:

- У каждого порожденного процесса вы можете контролировать: среду исполнения, ограничения ресурсов, рабочую и корневую директорию, `umask`, настройки `OOM killer`, параметр `nice`, класс и приоритет операций ввода-вывода, политику и приоритеты использования процессора, привязку к процессору, таймер, идентификаторы пользователя основной и дополнительных групп, списки директорий, доступных для чтения/записи, список директорий, доступ к которым запрещен, флаги монтирования, биты безопасности, параметры, относящиеся к планировщику процессора `CPU`, области видимости `/tmp`, привязку к `sgroup` для различных подсистем. Также можно присоединить `stdin/stdout/stderr` сервисов к `syslog`, `/dev/kmsg`, произвольным `TTY`. Если вы присоединяете `TTY` ко входу `system`, то удостоверяйтесь в том, что процесс получает эксклюзивный доступ.

- Каждый запущенный процесс получает собственную `sgroup` (в текущем состоянии разработки, по умолчанию они создаются в подсистеме `debug`, поскольку она все равно не используется). С помощью `systemd` также легко помещать отдельные сервисы в различные `sgroups`, причем это можно сделать из пользовательского пространства, например, посредством утилит `libcgroups`.

- Файлы конфигурации `systemd` име-

ют синтаксис, аналогичный используемому в файлах `.desktop`, который прекрасно разбирается (`parse`) многими имеющимися утилитами и имеет все необходимое для интернационализации. Поэтому администраторам и разработчикам не нужно учить новый синтаксис.

● Как упоминалось выше, мы (здесь и далее под «мы» понимаются разработчики и сама `systemd`, надо смотреть по контексту. Обычно это нормально, когда автор осознает свое единство со своим творением :). Прим. перев.) обеспечиваем совместимость со скриптами `SysV`. Дополнительно к этому обрабатываются заголовки `LSB` и утилиты `chkconfig` `RedHat`. Если их нет, просто используется любая доступная информация (такая, как приоритеты запуска сервисов) из `/etc/rc.d`. Поскольку эти скрипты начинают просто читать другой источник конфигурации, обновиться с `SysV` на `systemd` достаточно легко. Дополнительно мы можем читать классические `PID`-файлы сервисов, чтобы определить `PID` главного демона. Также мы можем использовать информацию о зависимостях из `LSB`-заголовка скрипта и транслировать ее в «родной» формат описания зависимостей для `systemd`. Важное замечание: `Upstart` не может использовать эту информацию. Во время запуска, `Upstart`, в отличие от `systemd`, не распараллеливает запуск большей части скриптов `LSB` и/или `SysV`. Фактически, для `Upstart` все скрипты `SysV` — это одно исполняемое задание.

(Тут опять автор немного лукавит. В `Upstart` просто оставлен слой совместимости со скриптами `SysV`, который действительно работает, как описано. Но это именно что слой совместимости с теми службами, управляющие скрипты которых по каким-то причинам пока не отконвертированы в «родной» формат для `Upstart`, а не «злостная недоработка» разработчиков `Upstart`. Прим. перев.)

● Аналогичным образом, существующий файл `/etc/fstab` читается и используется как еще один источник конфигурации. А с использованием опции `comment=fstab` вы даже можете отметить отдельные записи в этом файле, чтобы передать их под контроль `systemd` (для обеспечения работы функционала автоматического монтирования).

● Если для одного сервиса существует несколько файлов конфигурации (например, есть два файла `/etc/systemd/system/avahi.service` и `/etc/init.d/avahi`), тогда «родной» формат файлов имеет преимущество. Устаревший формат файлов игнорируется, позволяя легко провести обновление.

● Мы также поддерживаем механизм использования шаблонов. Например, вместо того, чтобы иметь шесть одинаковых конфигурационных файлов для шести `getty`, у нас есть только один файл `getty@.service`, который используется сервисом `getty@tty2.service` и ана-

логичными ему. Интерфейсная часть также может наследоваться выражениями, описывающими зависимости, т. е. легко понять, что сервис `dhcpcd@eth0.service` запускается сервисом `avahi-autoipd@eth0.service`.

● Для активации сервисов посредством сокетов, мы поддерживаем полную совместимость с традиционной моделью `inetd`, а также простой режим, имитирующий работу `launchd`, который рекомендуется к использованию для вновь создаваемых сервисов. Режим совместимости с `inetd` позволяет передавать запускаемому демону только один сокет, в то время как «родной» режим работы позволяет передавать произвольное количество дескрипторов файлов. Мы поддерживаем как режим с одним экземпляром сервиса на одно соединение, так и с одним экземпляром на все соединения. Например: `sshd.socket` может запускать сервис `sshd@192.168.0.1-4711-192.168.0.2-22.service` с `cgroup sshd@.service/192.168.0.1-4711-192.168.0.2-22` (т. е. IP-адрес и номера портов используются в качестве имен). Для сокетов `AF_UNIX`, используется `PID` и идентификатор пользователя присоединяющегося клиента. Это предоставляет администратору прекрасный инструмент для определения различных экземпляров одного и того же демона и контроля за ним. «Родной режим» передачи сокета легко реализовать в приложениях: переменная `$LISTEN_FDS`, если она установлена, содержит количество передаваемых сокетов, и демон может найти их в файле `.service`, начиная с файлового дескриптора 3 (хорошо написанный демон также может использовать `fstat()` и `getsockname()` для идентификации сокетов в случае, если их больше одного). В дополнение мы устанавливаем переменную `$LISTEN_PID`, содержащую `PID` главного демона, который получает сокеты, потому что переменные среды обычно наследуются дочерними процессами, что может несколько запутать процессы, находящиеся далее в цепочке. Поскольку логика передачи сокетов очень проста, мы предоставляем примерную реализацию этого процесса под лицензией `BSD`. Также мы портировали множество существующих демонов на эту схему.

● Мы предоставляем совместимость с интерфейсом `/dev/initctl`. Эта совместимость фактически реализована с помощью сервиса, активируемого посредством `FIFO`, который просто транслирует устаревшие запросы в запросы `D-Bus`. На практике это означает, что старые команды `shutdown`, `poweroff` и аналогичные им из `Upstart` и `sysvinit`, будут работать с `systemd`.

● Мы предоставляем совместимость с `utmp` и `wtmp`. Код, который это делает, выглядит гораздо более жизнеспособным, чем сами эти файлы :).

● `Systemd` поддерживает несколько

типов зависимостей между модулями. `After/Before` могут использоваться для определения последовательности запуска. `Requires/Wants` определяет статус зависимости, является она обязательной или опциональной. И наконец, `Conflicts` определяет отрицательный характер зависимости (т. е. две и более службы, у которых в зависимостях указана `Conflicts`, не смогут быть запущены одновременно. Прим. перев.). Есть также еще три, менее часто используемые типы зависимостей.

● `Systemd` построена как система с минимальным транзакций. Это означает: если модуль запросил запуск или остановку, мы добавляем его и все его зависимости во временную транзакцию. Затем проверяем целостность этой транзакции, т. е. последовательности обработки зависимостей `After/Before` для всех модулей, на возможность появления циклических зависимостей. Если они есть, `systemd` пытается исправить ситуацию путем удаления из транзакции «несущественных» (`non-essential`) заданий. Также `systemd` пытается убрать из транзакции такие из «несущественных» заданий, которые могут остановить какой-либо другой сервис (не имеющий отношение к останавливаемому модулю). «Несущественными» являются такие задания, которые не относятся к оригинальному запросу, но при этом помещены в очередь на основе зависимостей `Wants`. В заключение проверяется, могут ли задания в транзакции противоречить заданиям, которые уже находятся в очереди и, если возникла такая ситуация, транзакция отменяется. Если все сработало корректно, транзакция целостна и минимизирована по количеству операций, она ставится в очередь на исполнение. В сухом остатке это означает, что перед запуском запрошенной операции мы проверяем, имеет ли смысл ее вообще выполнять, если возможно, исправляем возникшие ошибки, и затем ничего не делаем, если операцию выполнить невозможно.

● Мы записываем время запуска-остановки, `PID` и статус завершения для каждого из порождаемых и/или контролируемых процессов. Эти данные позволяют увязать демоны с их данными в `abrt`, `auditd` и `syslog` и создать интерфейс, который выделял бы упавшие демоны и предоставлял бы всю необходимую о них информацию. ►



- Мы также реализовали самостоятельный перезапуск процесса `init` в любое время по требованию. Состояние демона замораживается перед этим, и размораживается после. Таким образом, мы упрощаем онлайн-обновление с `SysV init` на `systemd`, а также передачу полномочий от остановленного к перезапущенному демону. Запросы к открытым сокетами и точкам монтирования `autofs`, как уже отмечалось выше, будут корректно упорядочены и поставлены в очередь ядром, поэтому клиенты даже не почувствуют, что что-то вообще произошло. Поскольку большая часть информации о состоянии обслуживаемых `systemd` процессов хранится в виртуальной файловой системе `cgroup`, это позволяет нам не прерывать их исполнение из-за невозможности доступа к данным `init`. Код, реализующий перечитывание конфигурации `init`, является общим с кодом его перезапуска.

- Избавляясь от `shell`-скриптов в процессе запуска системы, мы переписали основную их часть на `C` и поместили непосредственно в `systemd`. В частности, это код таких операций, как монтирование виртуальных файловых систем `/proc`, `/sys` и `/dev` и установка имени хоста.

- Состояние сервиса доступно и может контролироваться через `D-Bus` (за это `Upstart` не пинал только самый ленивый. Прим. перев.). Правда, эта возможность пока не реализована и находится в стадии активной разработки.

- Мы придаем особое значение активации сервисов посредством сокетов либо через `D-Bus` (и, следовательно, поддерживаем обработку соответствующих зависимостей). В то же время мы поддерживаем традиционные зависимости только между сервисами. Также поддерживается несколько способов, которыми сервис может сообщить о своей готовности: путем вызова `fork()` и завершением родительского процесса (традиционное поведение `daemonize()`) и посредством публикации своего имени на `D-Bus`.

- Существует интерактивный режим, который запрашивает подтверждение для каждого из процессов, порождаемого `systemd`. Вы можете включить это, передав параметр `systemd.confirm_spawn=1` в строке параметров ядра.

- С помощью параметра `systemd.default=` в строке параметров ядра вы можете указывать, какой из модулей `systemd` нужно запускать при загрузке системы. Обычно здесь находится что-то вроде `multi-user.target`, но можно указать и какой-то один сервис вместо модулей типа «указатель». К примеру, «из коробки» мы поставляем сервис `emergency.service`, который по своему назначению подобен параметру `init=/bin/bash`, но по сравнению с ним имеет то преимущество, что мож-

но запустить полнофункциональную систему прямо из оболочки восстановления от сбоев (`emergency shell`).

- Также есть минимальный пользовательский интерфейс, позволяющий запускать/останавливать сервисы и наблюдать за ними. Правда, он еще далек от завершения, но вполне может использоваться как утилита для поиска ошибок. Он написан на `Vala` и называется `systemadm`.

Следует также заметить, что `systemd` использует много специфичных для `Linux` возможностей, не ограничиваясь стандартами `POSIX`. Это позволяет использовать огромное количество возможностей `Linux`, разработанных для портируемости, которые остальные системы не предоставляют.

Состояние разработки

Значительная часть перечисленных выше возможности уже реализована. В настоящее время `systemd` уже может использоваться как полноценная замена для `Upstart` и `sysvinit` (по крайней мере, пока не все сервисы еще отконвертированы в формат `Upstart`, за что спасибо разработчикам основных дистрибутивов).

Однако объем тестирования `systemd` минимальный, поэтому номер версии, которую мы имеем на сегодняшний день — это абсолютный и великоколепный НОЛЬ. Так что, если решите использовать ее в таком виде, как она есть, ждите появления некоторого количества неожиданных проблем (автор намекает на необходимость обязательного наличия бубна при использовании `systemd` на рабочих машинах. Прим. перев.). Другими словами, все должно работать почти стабильно. Кое-кто из нас был настолько смел, что загружал наши рабочие машины с помощью `systemd` (не только виртуальные, но и те, которые мы обычно используем для разработки). В общем, тут как повезет, особенно если пробовать `systemd` на дистрибутивах, которые мы, разработчики, не используем.

Предполагаемые направления развития

С набором возможностей, описанным выше, все понятно. Однако у нас в планах есть еще кое-что. Я на самом деле не люблю много говорить о больших планах, но ниже приведу небольшой обзор того, что мы еще планируем сделать.

Мы хотим добавить, по крайней мере, два типа модулей. Первый — это `swap`, который будет контролировать разделы подкачки теми же способами, которыми мы контролируем точки монтирования, т. е. автоматическое разрешение зависимостей для тех устройств, на которых они расположены. Второй — это `timer`, обеспечивающий функцио-

нал, подобный `cron`, т. е. возможность запускать сервисы по определенным временным событиям, используя как обычные временные интервалы, так и календарные даты (т. е. «запустить через 5 часов после последнего запуска» и «запускать каждый понедельник в 5 часов»).

Более важная часть наших планов — это не только экспериментировать с `systemd` для оптимизации времени запуска, но также сделать из него полноценный и идеальный менеджер сессий, чтобы заменить (или, по крайней мере, попытаться) `gnome-session`, `kdeinit` и подобные демоны. Набор проблем и задач, которые надо решить для менеджера сессий и системы инициализации по большому счету один и тот же: максимально быстрый запуск жизненно важных частей и выполнение функций няньки по отношению к запущенным процессам. Для этого можно использовать тот же код. В `Apple` похожим образом используется `launchd`. Поэтому мы хотим использовать описанные выше способы активации сервисов через сокеты, `D-Bus`, максимальную параллелизацию запуска процессов и для менеджера сессий и для системных сервисов.

Надо отметить, что все эти возможности уже частично доступны (но реализованы не до конца) в существующей кодовой базе. Например, запускать `systemd` из-под обычного пользователя уже можно; она даже определяет, что запущен именно таким способом. Поддержка этого режима доступна с самого начала его разработки (Этот режим полезно использовать для отладки! Для его работы нет необходимости полностью конвертировать систему в формат `systemd`).

Однако есть некоторые вещи, которые мы должны предварительно исправить в ядре и в пользовательском пространстве, прежде чем закончить работу над `systemd`: нам нужны извещения об изменениях состояния подкачки, способом подобным тому, как сейчас работает монтирование файловых систем; мы также хотим реализовать извещения когда `CLOCK_REALTIME` перескакивает на `CLOCK_MONOTONIC`; мы хотим позволить обычным процессам получить часть возможностей `init`; нам нужно хорошо документированное и общепризнанное место, куда нам нужно положить сокеты. Ничего из этого не является жизненно необходимым для `systemd`, но может здорово ее улучшить.

Вы хотите увидеть systemd в действии?

Мы пока еще не выпускали релизов, поэтому у нас нет готовых тарболов. Но если вам очень хочется, вы всегда можете сделать снимок с нашего текущего репозитория (`git`). У нас пока нет готовых `RPM`, чтобы их предложить вам. ►

Самый легкий путь – это скачать образ Fedora 13 для qemu, который специально подготовлен для systemd. В меню grub вы сможете выбрать будете ли вы грузиться с помощью Upstart или с помощью systemd. Эта система имеет минимальный набор модификаций. Информация о сервисах читается исключительно из существующих скриптов SysV. «Благодаря» этому мы не получаем всех преимуществ от активации сервисов посредством сокетов и D-Bus. Но особенности systemd позволяют параллелизовать запуск сервисов только на основе чтения заголовков LSB, и уже только это позволяет нам грузиться быстрее, чем через Upstart. Образ сконфигурирован таким образом, чтобы выдавать информацию для отладки на последовательную консоль и в буфер ядра для ведения логов (просматриваемый с помощью dmesg). Поэтому необходимо сконфигурировать для qemu поддержку виртуального последовательного терминала. В качестве всех паролей установлен systemd.

Еще более простой путь – это посмотреть на эти прекрасные скриншоты. На первом приведен интерфейс утилиты systemdadm (рис.1). Здесь показан список загруженных модулей и вывод детальной информации об одном из экземпляров getty.

На рис.2 вывод команды `ps xaf -eo pid,user,args,cgroup`, отражающей насколько аккуратно все процессы отсортированы по своим cgroups (это показывает четвертая колонка с префиксом debug:, такой префикс используется по той причине, что мы пока используем специальный интерфейс cgroup для отладки. Это описывалось выше.)

К сожалению, у нас пока еще нет графиков загрузки или еще каких-то данных по времени запуска системы. Мы их опубликуем, как только сможем полностью параллелизовать запуск всех сервисов, поставленных в Fedora. В то же время, мы приветствуем проведение ваших собственных тестов и публикацию их результатов.

У меня пока есть только две цифры, которые я могу вам привести. Но они еще не заслуживают доверия, поскольку измерялись по времени загрузки виртуальной машины (один процессор). Fedora 13 грузится с помощью Upstart 27 секунд, с помощью systemd – 24 (от grub до gdm, с одними и теми же настройками, цифры измерены один раз, один запуск следовал за другим). Следует помнить, что цифры показывают только преимущество от использования параллелизации запуска скриптов на основе информации из их LSB заголовка. Поскольку не использовалась активация сервисов посредством сокетов или D-Bus, эти цифры, по сути, ничего не показывают.

Написание демонов

Идеальный демон, полноценно использующий возможности systemd должен

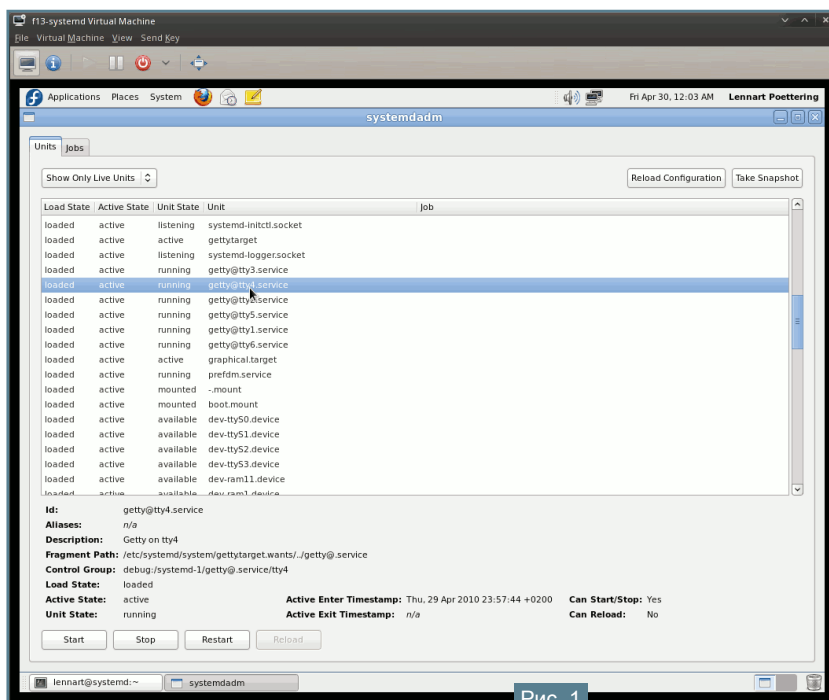


Рис. 1

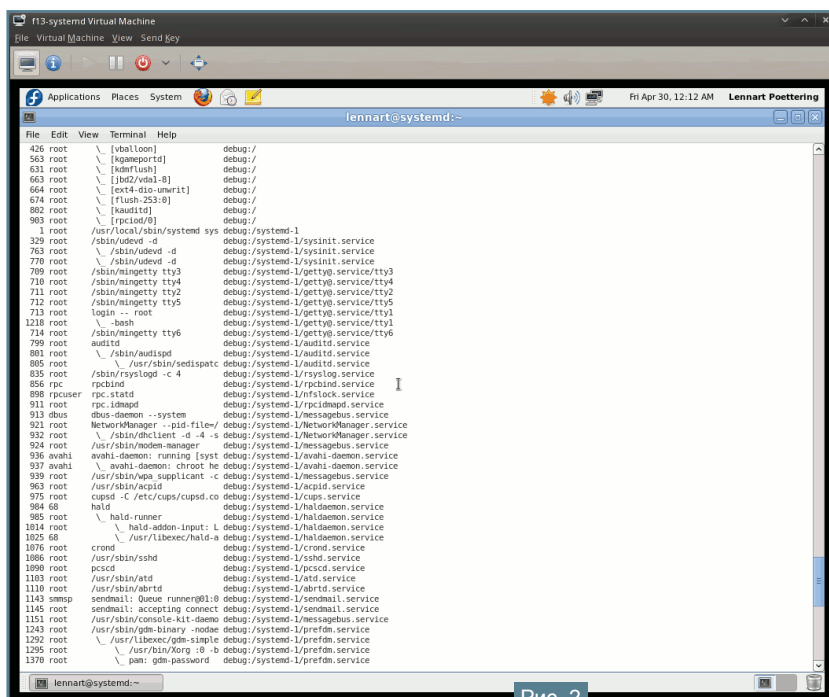


Рис. 2

делать некоторые вещи способами, отличными от традиционного поведения. Позже, мы опубликуем подробное руководство по написанию демона для использования с systemd. Ниже приведено краткое описание того, что нужно для разработчиков демонов:

- Мы просим разработчиков не вызывать `fork()` (или даже двойной `fork()`) в своих процессах, используя цикл событий основного процесса, который systemd вызывает для вас. Также не вызывайте `setsid()`.
- Не стоит сбрасывать привилегии

(Имеется в виду, что демон не должен быть запущен с root-привилегиями. Прим. перев.) с помощью самого демона, предоставьте сделать это systemd и настраивайте это в ее конфигурационных файлах (Тут есть несколько исключений. К примеру, для некоторых демонов нужно сбрасывать привилегии только посредством самого демона после стадии инициализации, которая требует повышенных полномочий).

- Не надо создавать PID-файлы.
- Имя демона следует получать с D-Bus. ▶

- Если вы хотите использовать возможности `systemd` для ведения логов, сбрасывайте все, что нужно включить в логи, на `stderr`.

- Предоставьте `systemd` создать и обслуживать сокет для вас, благодаря чему будет работать активация посредством сокетов. Для этого нужно использовать `$LISTEN_FDS` и `$LISTEN_PID` как описывалось выше.

- Используйте `SIGTERM` для остановки своего демона.

Все, что приведено выше, аналогично тому, что Apple рекомендует для демонов, совместимых с `launchd`. В то же время демоны, поддерживающие `launchd` легко портировать на `systemd`. Заметьте, что `systemd` (для совместимости) поддерживает и демоны, которые не поддерживают того, что описано выше. Демоны, совместимые с `inetd`, для активации посредством сокетов можно использовать без каких-либо модификаций. И, конечно же, как только `systemd` подтвердит свою жизнеспособность в наших экспериментах и начнет адаптироваться дистрибутивами — будет необходимо портировать на нее, по крайней мере, те сервисы, которые должны быть запущены по умолчанию. Мы уже пишем концепцию патчей и тогда процесс портирования станет очень легким. Кроме того, в определенной степени, мы можем использовать ту работу, которая уже была проделана для `launchd`. И более того, демон, поддерживающий активацию посредством сокетов, не становится от этого несовместимым с системами, отличными от `systemd`.

Часто задаваемые вопросы

Кто основные разработчики?

Большая часть кодовой базы моя собственная работа, Lennart Poettering (Red Hat). Однако, общий дизайн и его отдельные детали — это результат моего взаимодействия с Kay Sievers (Novell). Также в проекте участвуют Harald Hoyer (Red Hat), Dhaval Giani (бывший сотрудник IBM), и многие другие из таких компаний как Intel, SUSE and Nokia.

Это проект Red Hat?

Нет, это мой персональный проект. И позвольте мне подчеркнуть: мнение, отраженное в данной статье — это только мое мнение. Это ни мнение моего работодателя, ни Рональда Макдональда, ни кого бы то ни было еще.

Увидим ли мы это в Fedora?

Если наши эксперименты покажут

что наши подходы работают, если сообщество Fedora высказается в поддержку, тогда да, мы увидим `systemd` в Fedora.

Увидим ли мы это в OpenSUSE?

Этим занимается Kay, но думаю, что все будет также же, как и в Fedora.

Увидим ли мы это в Debian/Gentoo/Mandriva/MeeGo/Ubuntu/[Моем любимом дистрибутиве]?

Это вопрос к их разработчикам. Мы можем только приветствовать их интерес и помочь с интеграцией.

Почему бы не добавить все вышеперечисленное в Upstart, зачем было изобретать что-то новое?

Недостатки Upstart приведены выше, так же было показано, что ее основной дизайн, по нашему мнению, ущербен. Этот проект выглядит, как имеющий кучу недостатков в своей основе. Однако имейте в виду, что это не помешало нам найти много идей для вдохновения в кодовой базе Upstart.

Если вам так нравится launchd, почему было бы просто не адаптировать ее?

Launchd прекрасное изобретение, но я не уверен, что она хорошо впишется в Linux, более того, она вообще никак не вяжется по своему дизайну ни с его (Linux) масштабируемостью, ни с его гибкостью.

Это проект NIH?

(Тонкий укол, намекающий на то, что разработчики пошли по собственному пути только по причине того, чтобы начать свой собственный проект. Проект ради проекта. Прим. перев.) Я надеюсь, что хорошо объяснил текстом выше, почему мы пошли по своему собственному пути, вместо того, чтобы использовать как основу Upstart или `launchd`. Мы начали разработку по техническим, а не политическим причинам. И не забывайте, что это Upstart, а не `systemd`, включает библиотеку NIH (которая является, своего рода, новой релизацией `glib`).

Будет ли systemd работать на [моей любимой операционной системе, отличной от Linux]?

Маловероятно. Выше мы отметили `systemd` использует много специфичных для Linux API (таких как `as_epoll`, `signalfd`, `libudev`, `cgroups` и т. п.), поэтому портирование его на другую операционную систему (по нашему мнению) не стоит своих затрат. Мы, разработчики проекта, не заинтересованы в работе, предполагающей портирование на

другие платформы. Для тех, кто заинтересован в этом, мы можем только сказать, что `git` хорошо поддерживает концепцию ветвей (branches).

Скажу больше, портируемость ограничивается не только окружением, в котором работает `systemd`: нам нужны самые последние версии ядра Linux, `glibc`, `libcgroup` и `libudev`.

Если кто-либо хочет сделать нечто подобное для других операционных систем, можем предложить приемлемый режим взаимодействия: мы поможем вам определить какие интерфейсы являются общими с вашей системой, сделаем немножко легче жизнь тех, кто будет писать демоны для `systemd` и ее аналога. Мы можем помочь идеями, но не кодом.

Я слышал, что [система загрузки Gentoo, initng, Solaris SMF, runit, xlaunch, что-то еще] — это реально классная система инициализации, которая также позволяет параллелизовать запуск, почему бы не использовать ее как основу?

Прежде чем начать разрабатывать `systemd` мы изучили имеющиеся системы, и ни одна из них не зацепила нас (естественно, за исключением `launchd`). Если вы этого никак не можете заметить — прочтите приведенный выше текст еще раз.

Помощь проекту

Мы очень заинтересованы в патчах и другой помощи. В этом смысле наш проект ничем не отличается от любых других проектов свободного ПО, основные преимущества которых заключаются в как можно более широком взаимодействии с сообществом. И это, действительно, большей частью правда для такой фундаментальной части системы, как система инициализации. Нам ценно взаимодействие с вами, поэтому мы не требуем передачи прав (в отличие от Canonical/Upstart!). Мы также используем `git`, как всеми любимую систему контроля версий!

Мы сильно заинтересованы в том, чтобы `systemd` заработал на других дистрибутивах помимо Fedora и `openSUSE` (эй, кто-нибудь из Debian, Gentoo, Mandriva, MeeGo, не хотите ли заняться этим?). Ну и помимо перечисленного, нам нужны коллеги любого уровня: мы приглашаем программистов на C, майнтейнеров пакетов, заинтересованных в написании документации или разработке логотипа.

Сообщество

В настоящее время, все что у нас есть — это репозиторий с исходным кодом и IRC-канал (#systemd на Freenode). Нет списков рассылки, веб-сайта или системы отслеживания ошибок. Возможно, что скоро у нас будет нечто подобное на `freedesktop.org`. ■

FastVPS.ru

VPS от 129 рублей

- 100 MB RAM
- 300 MHz CPU
- 6 GB HDD

**Dedicated от
2199 рублей**

- Intel® Core™ i7-920
Quad-Core incl
- 8 GB DDR3 RAM
- 2 x 750 GB HDD



- *один из лучших
дата-центров Германии*
- *служба поддержки
на русском языке*
- *уникальная система
восстановления*
- *удаленный reboot/reinstall*
- *установка сервера
в течение 24 часов*

MySQL в tmpfs

Автор

Александр Федора
Evolution
alexanderfedora.blogspot.com
E-mail: alexander.fedora@gmail.com

Хотелось бы поделиться опытом по использованию MySQL с хранением данных в памяти, а не на диске. Это позволило нам сократить load average сервера, который из-за операций с диском стал сильно расти.

В одном проекте мы используем MySQL с движком MyISAM под Debian Lenny. Общие данные загружаются в оперативку демона на C++, а пользовательские данные один раз загружаются при логине и периодически сохраняются во время работы пользователя. Также сохраняются при логауте (либо по таймауту). Ввиду описанной схемы, select-ов намного меньше, чем update-ов (select: 24%, delete: 4%, update: 61%, insert: 11%).

В принципе данная проблема не админская, а вызвана не совсем удачным выбором инструмента. Скорее всего, нам бы подошел InnoDB, который использует row-level locking (блокировка по записям), а не table-level locking (блокировка всей таблицы при записи) как у MyISAM. Хотя объем данных записываемых на диск это вряд ли сократило бы. С другой стороны, нам SQL особо не нужен и мы склоняемся к портированию проекта на NoSQL (к некоторым из представителей давно приглядываемся (Cassandra), а некоторые (tokyo tyrant, Berkeley DB) активно используем для логирования пользовательских действий). Но выделить время на перевод хранилища данных на другой движок/базу не было, поэтому решили использовать админ. ресурс.

С ростом количества пользователей и введением нового контента мы столкнулись с проблемой роста load average на сервере базы данных (2-3 Ia на 8-ми ядерном сервере наблюдалось всего при 400 запросах в секунду). Причем память была недогружена, как и процессор. Проблема была с большим объемом записи данных на диск (увели-

чение iowait). В некоторые моменты система начинала заикаться и отдельные примитивные запросы занимали 2 секунды, а то и больше. В нормальном режиме такие запросы выполняются за миллисекунды.

Мы немного оптимизировали конфигу MySQL (используя как автоматические скрипты типа mysqltuner, так и ручную настройку), но это дало лишь небольшое снижение нагрузки (процентов на 10-15). По большей части параметры, отвечающие за кеширование данных, нам не подходят, так как у нас основная нагрузка это обновление данных, а не чтение. База находится на SAS диске, но скорости все равно не хватает. Бинарные логи находятся на другом диске (используются для бекапа базы со слейва).

Ускорить работу дисковой системы купив полноценный RAID не подходит из-за его большой цены. Но сам сервер почти простаивает, если не учитывать диски. Возможности сжатия данных в памяти до записи на диск в MyISAM нет, но и переходить на другой движок пока нет возможности.

База у нас занимает около 2-2.5 Gb (в tar.gz 700mb) и мы уже давно пробовали использовать MySQL в tmpfs (методом основанном на mylvmbackup), что позволяло не нагружать диск и упростить создание бекапа. К счастью недавно провели очистку базы путем добавления крона, который удаляет

старых пользователей, почти не пользующихся проектом и ни разу не заплывших. Кроме этого очистили старые данные, которые собирали для статистики и поставили им срок жизни порядка двух месяцев.

В качестве быстрого решения данной проблемы возникла идея перевести главную базу на tmpfs. Что у нас получилось.

Мы имели 3 сервера:

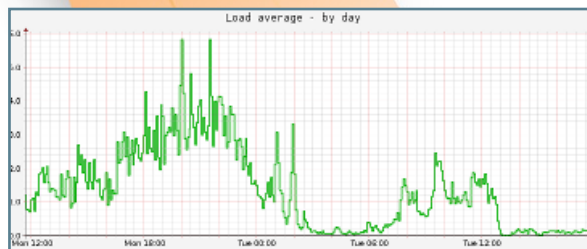
1. сервер А (боевая база куда стучатся демоны (на нем 8Г оперативки))
2. сервер Б (любой сервер на той же площадке со свободной оперативкой (у нас 8Г из них 5Г свободно))
3. сервер В (на другой площадке заточен под бекапы всех проектов с максимум оперативки (у нас 16Г))

Итак на Сервере А поднимаем tmpfs для файлов базы

```
mount -t tmpfs -o size=5G
tmpfs /var/lib/mysql
```

(с запасом в 2 раза больше чем весит база)

Так же не забываем в конфигах базы написать, чтоб бин логи писались на отдельный свободный диск (мастер без них не работает), а под нагрузкой база может писать в эти логи до 1М в сек. Мы храним эти файлы не более 7 дней (более и не надо т. к. для восстановления базы, если что есть слайвы). ►



Соответственно стартуем базу как мастер. Сервер под самими жесткими нагрузками не выходит из 1ЛА и по памяти не более 6Г. Средняя скорость выполнения запроса возросла в десятки раз, если раньше было 0,1-0,3 сек то стало 0.1-3 мсек.

Далее на сервере Б так же поднимаем первый слайв тоже в tmpfs, тут не стоит забывать о настройках relay-log, т. к. если в слайве будет ошибка, а мастер будет доступен в эти логики, то он будет писать запросы которые были на мастере... В итоге это может измеряться десятками гигабайт за ночь, прежде чем вы почините слайв (у нас это около 8-10 гигабайт в час). Бекапим эту базу снапшотами ночью и 1-2 раза в день, когда нагрузка на демоны минимальна. Такой слайв потребляет максимум 0.2-0.3 ЛА и чуть более того что весит база.

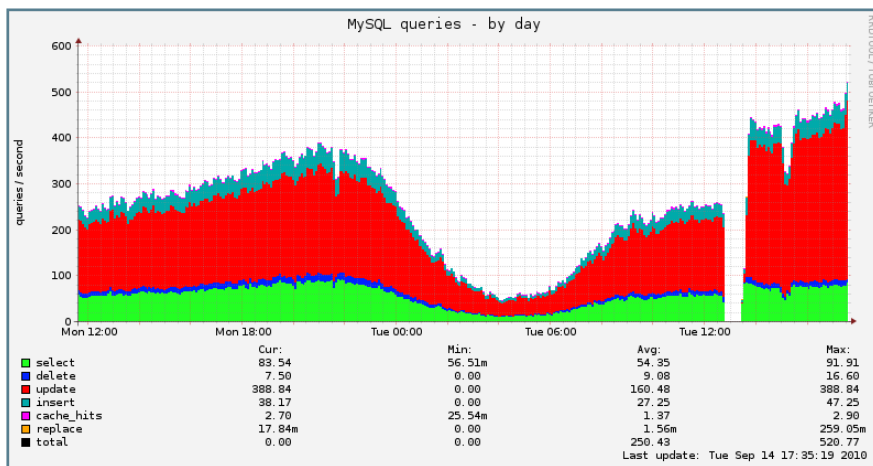
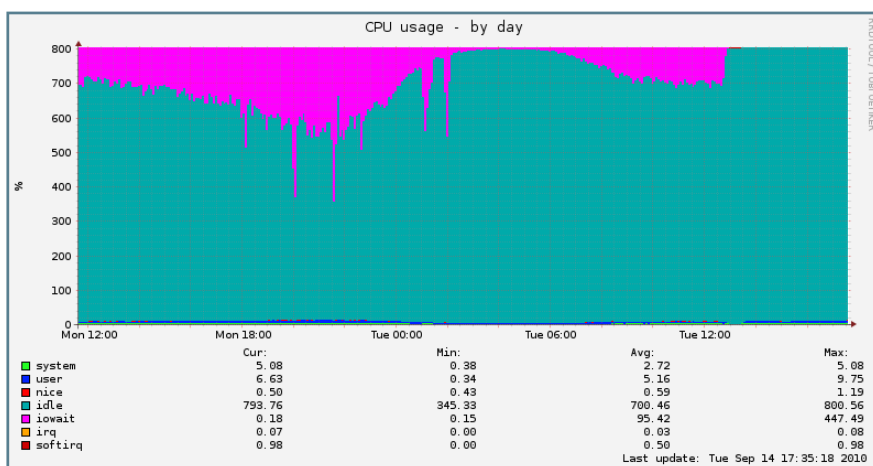
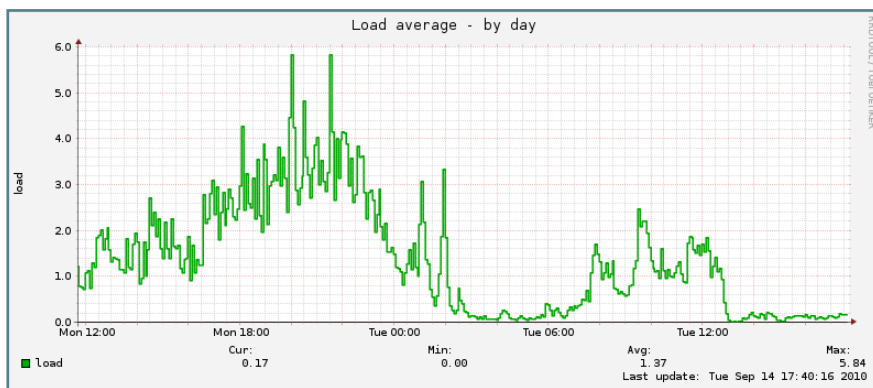
Также на сервере В у нас подняты сразу несколько слайвов на tmpfs под наши проекты. Они спокойно живут вместе особо не нагружая сервер, при этом бекапятся снапшотом раз в 10 минут (если это время разложить, то снапшот делается около 3 сек (базы, вес которой 2,5 гига), ну и сжатие в архив около 6-7 минут) и да же при этом на сервере ЛА не превышает 2, даже когда бекапы пересекаются по времени. Судя по всему, количество бекап-слайвов на одном сервере определяется размером баз и количеством оперативки. Бекапы храним по 10 минут – последние 24 часа, каждые 6 часов – храним 30 дней, каждые 30 дней – вечно.

Далее если вдруг падает слайв Б или В, его можно поднять с любым из бекапов с другого слайва не трогая мастер, если даже оба падают, то поднять не трогая мастер 10 минут – скорость переписывания бекапов.

Если падает мастер, то есть на это два слайва с отставанием максимум 1-2 апдейта от мастера (ни разу не видели чтоб Slave Delay (отставание от мастера) превышал 0 сек), при этом срочное решение проблемы – это в течении 5 минут из слайва Б сделать мастер... и перестроить демоны. При тестах они спокойно живут на одном сервере и не мешают друг другу.

В итоге чтоб одновременно упали 3 машины на 2 разных площадках – это очень мало вероятно. Поднять базу при любом падении сервера или площадки займет не более 10 минут.

Настраиваем оповещалки (mi-



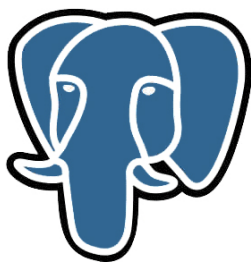
nin, nagios, zabbix – кому что больше по душе) на 2 площадках с измерением отставания слайва, можно по смс или по почте. У нас за этим следит нагиос, при этом мунин его страхует и рисует графики, по которым все очень наглядно видно.

Load average на серверах снижается довольно сильно, теперь основную нагрузку берет на себя процессор и память, а не диск. Можете посмотреть на самые свежие графики, которые мы сняли при еще одном переводе сервера.

Стоит заметить что после перевода

базы, были так же уменьшены интервалы сохранения пользовательских данных в C++, поэтому количество апдейтов возросло.

В общем, единственная потенциальная проблема – это рост базы. Но сейчас память в серверах достаточно дешевая и увеличить ее не составит проблем (на части машин у нас 16Gb, на части 8Gb, но, насколько мне известно, можно поставить в потолок 128Gb). В любом случае кластеризацию базы никто не отменял и мы сможем распределить базу на несколько серверов со схожей конфигурацией. ■



Репликация в PostgreSQL 8.x: упрощаем работу со Slony

Автор

Андрей Росляков
г. Москва
www.mixersk.ru
mixersk@yandex.ru

Так уж исторически сложилось, что из коробки PostgreSQL 8-й ветки (а стабильное светлое будущее с 9-й еще не наступило) не имеет системы репликации. Поэтому для этих целей написан внешний инструмент — Slony1. Работает он довольно просто: демон slon подключается к PostgreSQL-серверу, цепляется за триггеры и ждет, когда произойдет какое-нибудь событие, изменяющее содержимое базы. Так же есть slonik — конфигурационная утилита, употребляющаяся в STDIN конфиги, и еще несколько управляющих утилит для различных целей.

В рамках репликации, когда речь заходит о PostgreSQL, обычно используют следующие термины:

- **Нода (node)** — пара «сервер-база». То есть, например, если у вас на одном сервере несколько баз, участвующих в репликации, то это не одна нода, а несколько.
- **Кластер (cluster)** — группа нод. Тут все просто: мастер и слейв(ы).
- **Таблицы (tables)** — тут еще проще.
- **Секвенсы (sequences)** — счетчики для таблиц.
- **Сеты (sets)** — набор таблиц и счетчиков. Для каждой базы создается отдельный сет.

После удачной конфигурации, Slony создает в базе, которая подлежит репликации схему `_cluster` (где «cluster» — название кластера, указанное в конфиге), а в ней несколько своих таблиц. С назначением таблиц можно ознакомиться в официальной документации, или применив `psql`. Так что если вдруг в процессе настройки вы столкнулись с трудностями, а после перезапуска получили сообщение вида «эта нода/кластер/таблица уже есть в списке», возможно вас спасет дроп схемы.

Так же вам нужно знать, что репликации подлежат только те таблицы, которые имеют `primary key`.

Главной неприятной особенностью слонов, для меня стало то, что оно не умеет делать реплику всех таблиц в базе — обязательно нужно перечислить одну за другой. Когда у вас в базе >50 таблиц, написание конфигов становится крайне утомительным занятием. Для этих целей мной был написан скрипт (о нем ниже).

Самые любознательные наверняка видели подобные маны и возможно считают, что эта статья очередной дубль. Нет, это не так. Все маны, которые я видел, предлагали скамливать

конфиг непосредственно в `slonik` через `STDIN`. При таком подходе система работает, но (по крайней мере в случае с Debian/Ubuntu) будут две проблемы:

1. Не будет работать `init`-скрипт
2. Не будут работать вспомогательные утилиты слонов.

Правильный же подход заключается в том, чтобы хранить `perl`-овидный конфиг в `/etc/slony1/slony_tools.conf` (пример вы можете найти в `ls /usr/share/doc/slony1-bin/examples/slony_tools.conf-sample.gz`), а в `/etc/default/slony1` — номера нод, которые живут на данной машине и будут запущены `init`-скриптом. Как я уже сказал, первый конфиг является `perl`-скриптом и включается в скрипт `slon_start`.

Подготовка

Будем считать, что база, равно как и настроенный PostgreSQL, у вас уже есть, в ней есть таблицы с `primary key` и вы готовы приступить к настройке. Заранее оговорюсь, что все делалось на различных версиях серверной Ubuntu, в Debian будет примерно то же самое.

Итак, нам нужны пакеты `postgresql-8.3-slony1` и `slony1-bin`. Желательно, чтобы оба пакета были второй ветки (она стабильнее).

Настройка серверов

Сначала в `postgresql.conf` нужно раскомментировать строку, содержащую `listen_addresses` и вместо `localhost` указать `ip-адрес интерфейса`, смотрящего в сторону слейва. Так же можно вместо адреса поставить `'*'`, тогда сервер будет ожидать подключения по всем интерфейсам.

Далее в `pg_hba.conf` (тоже лежит в `/etc/postgresql`) нужно добавить строку вида

```
host all all 192.168.1.0/24 md5
```

где вместо `192.168.0.0/24` нужно указать сеть или `ip` слейва. Делает то же самое на слейве, указывая `ip` мастера.

Далее рестартируем PostgreSQL на мастере и слейве, дабы оно подхватило новые настройки.

Еще нам нужно будет создать `pg-суперпользователя` на обеих машинах:

```
createuser -sP slony
```

Скрипт

Сам скрипт можно посмотреть, перейдя по этой [ссылке](#). Заранее скажу, что я

не претендую на приз за красоту кода. Напротив, он писался на коленке в свободное время. Важно другое — он работает, и я расскажу о том, что он делает.

Для нормальной работы вам потребуется запускать его от `root'a` или через `sudo` (т. к. там используется `su` пользователя в `postgres`, а так же производится запись в `/etc` и выполняются `init`-скрипты). Так же желательно для выполнения последних двух стадий, чтобы `slave` пускал к себе `root'a` по `ssh` (лучше ключом, т. к. пароль вводить быстро надоест). А еще необходим пакет `libdbd-pg-perl`.

Текущая версия не принимает параметры из командной строки (хотя следовало бы), поэтому после запуска вы вступите в диалог со скриптом. Итак, алгоритм работы пошагово:

1. Опрос пользователя на предмет названия кластера и реквизитов к мастеру/слейву

Если ошиблись — не беда, в случае проблем с подключением к базе скрипт предложит ввести данные еще раз. Название кластера произвольно.

2. Диалог выбора баз

Тут все просто. Скрипт опрашивает сервер на предмет имеющихся баз и предлагает выбрать те, которые нужно реплицировать. Для выбора или отмены выбора базы нужно ввести ее номер. Звездочкой можно выбрать все. По окончании нужно нажать **Enter**.

3. Добавление таблиц/последовательностей

Имея список выбранных баз, скрипт собирает из них список таблиц. В список добавляются все таблицы и последовательности, находящиеся в выбранных базах.

4. Перенос структуры базы с мастера на слейв

Последовательно выполняются удаленные `createdb`, `createlang`. Далее локальный `pg_dump -s` после чего структура передается по `ssh` на `slave`.

5. Непосредственная генерация конфига и его пересылка

Все необходимые данные есть, теперь генерируется конфиг и, если вы `root`, то кладется в `/etc/slony1` (в том числе на слейв), редактируется `/etc/default/slony1` (указывается, какие ноды следует запускать).

6. Запуск демона

Ну, вот собственно и все. Осталось запустить. ■



Каталог Хостинг-компаний

Superhost.com.ua

Организация: SARBASH Lab.
Страна: Украина
Город: Донецк
Дата-центр: McLean data center, DCA2
Сайт: <http://superhost.com.ua>
Телефон: 380622521461
E-mail: info@superhost.com.ua

Инстант Хостинг

Организация: Inst-Host.ru
Страна: Germany
Город: Nuremberg
Дата-центр: HETZNER Online
Сайт: <http://inst-host.ru/>
E-mail: reg@inst-host.ru

xTremeHost.ru

Организация: xTremeHost.ru
Страна: Казахстан
Город: Кокшетау
Дата-центр: США
Сайт: <http://xtremehost.ru>
Телефон: +77770397355
E-mail: support@xtremehost.ru

КОМТЕТ

Организация: ООО «КОМТЕТ»
Страна: Россия
Город: Москва, Пенза
Дата-центр: Москва, М10
Сайт: <http://komtet.ru>
Телефон: 8-800-200-2511
E-mail: support@komtet.ru

Мирекс

Организация: Мирекс
Страна: Россия
Город: Красноярск, Москва
Дата-центр: Квиклайн, М9, Веб Дата-центр -
Дата Центры в Москве.
Сайт: <http://mirex.su>
Телефон: 8 (391) 215-29-35
E-mail: hosting@mirex.su

Партизанск Телеком

Организация: ИП Нецадим А.С.
Страна: Россия
Город: Партизанск
Дата-центр: Burst, Netdirect, SuperB
Сайт: www.p-telecom.ru
Телефон: +7 42363 69663
E-mail: company@p-telecom.ru

Хостинг Хутор

Организация: HostingHutor.com
Страна: Украина
Город: Одесса
Дата-центр: Hetzner, Utel, Wnet
Сайт: <http://hostinghutor.com>
Телефон: +3 048 7017698
E-mail: [info\[at\]hostinghutor.com](mailto:info[at]hostinghutor.com)

HOST-FOOD

Организация: ИП Седов А.Н.
Страна: Россия
Город: Саратов
Дата-центр: «Инфобокс» в СПб и
«Datacheap» в Мск
Сайт: <http://www.host-food.ru>
Телефон: +7 927-279-3758
E-mail: support@host-food.ru

Казахстанский Хостинг

Организация: Уральск Web
Страна: Казахстан
Город: Уральск
Дата-центр: АО «Казахтелеком», г.Уральск
Сайт: www.solidhost.kz
E-mail: support@solidhost.kz

InstantHost

Организация: Instant
Страна: Россия
Город: Братск
Дата-центр: Майами
Сайт: <http://instanhost.biz>
Телефон: +79501171041
E-mail: iskander019@ya.ru

Iphosters

Организация: Iphosters.com
Страна: USA
Город: New York
Дата-центр: New York
Сайт: <http://iphosters.com>
Телефон: +1-877-745-6763
E-mail: support@iphosters.com

KILOBYTE

Организация: KILOBYTE.COM.UA
Страна: Украина
Город: Киев
Дата-центр: собственная площадка
Сайт: <http://kilobyte.com.ua>
Телефон: +380443600206
E-mail: [sales\[at\]kilobyte.com.ua](mailto:sales[at]kilobyte.com.ua)

StoreHost.ru

Организация: СторХост Лтд
Страна: Россия
Город: Москва
Дата-центр: «Оверсан-Меркурий»
Сайт: <http://storehost.ru>
Телефон: +7 495 9164790
E-mail: support@storehost.ru

SVAI

Организация: ФОП Полудень В.М.
Страна: Украина
Город: Черкасы
Дата-центр: Mhost, Megastyle, Ukrtelecom
Сайт: svai.net
Телефон: +38 0472 569026
E-mail: billing@svai.net

Amberhost.net

Организация: Amberhost.net
Страна: Украина
Город: Киев
Дата-центр: Мхост
Сайт: <http://amberhost.net>
E-mail: support@amberhost.net

KirHost

Организация: СПД Кунев Д.В.
Страна: Украина
Город: Кировоград
Дата-центр: Воля, SteepHost
Сайт: www.kirhost.com
E-mail: roman@kirhost.com

Luckhost.net

Организация: ТОО «КУЛЬСЫН»
Страна: Казахстан
Город: Тараз
Дата-центр: Hetzner
Сайт: <http://luckhost.net/>
E-mail: info@luckhost.net

ProGoldHost.Net

Организация: ProGoldHost.Net
Страна: Россия
Город: Ставрополь
Дата-центр: США - DataJersey / Германия -
Hetzner
Сайт: <http://progoldhost.net>
E-mail: info@progoldhost.net

ISPserver

Организация: ISPserver
Страна: Россия
Город: Москва, Иркутск
Дата-центр: Веб-дата центр (WebDC.ru)
Сайт: www.ispserver.com
E-mail: sales@ispserver.com

Компания iHead

Организация: Компания iHead
Страна: Россия
Город: Киров
Дата-центр: Multinex, Agava
Сайт: www.ihead.ru
E-mail: info@ihead.ru

FirstVDS

Организация: ЗАО «Первый»
Страна: Россия
Город: Москва, Иркутск
Дата-центр: Веб-дата центр (WebDC.ru)
Сайт: www.firstvds.ru
E-mail: sales@firstvds.ru

Добавить хостинг-компанию в каталог бесплатно

ДОБАВИТЬ

Публикуемые в журнале компании взяты из каталога сайта издания. Данные о компаниях публикуются по мере их добавления в каталог. Добавить в каталог данные о компании может любая организация. Добавление данных в каталог сайта осуществляется бесплатно на добровольной основе самой организацией. Редакция журнала Веб-Аналитик.ИНФО не занимается добавлением данных в каталог сайта и не несет ответственности за предоставляемую информацию от организаций.

```

titleBar = false"
backgroundColor = "none"
scriptFile = "salambo.js"
onload = "first();"
onmouseover = "check();"
onload = "JScript:myvolume.value = player.sett

<PLAYER
OpenState onchange="back();"
>

<CONTROLS
currentPosition onchange = "myslider.value =
player.controls.currentPosition;"
/>

</PLAYER>

<SUBVIEW
id = "center"
left = "240"
top = "0"
zindex = "20"
backgroundImage = "center-base.bap"
>

<BUTTONGROUP
mappingImage = "center-map.bap"
hoverImage = "center-hover.bap"
>

<PREVELEMENT
mappingColor = "#00CCFF"
upToolTip = "Previous" />

<PLAYELEMENT
mappingColor = "#00FF00"
upToolTip = "Play" />

<STOPELEMENT
mappingColor = "#FF0000"
upToolTip = "Stop" />

<NEXTELEMENT
mappingColor = "#3300FF"
upToolTip = "Next" />

</BUTTONGROUP>

<pausebutton
left = "190"
top = "1"
visible = "whenenabled:player-controls-pause"
upToolTip = "Pause"
image = "pause-base.bap"
hoverImage = "pause-hover.bap"
disabledImage = "pause-disabled.bap"
transparencyColor = "#040404" />

<CURRENTPOSITIONTEXT
left = "190"
top = "1"
fontface = "ms ui gothic"
fontstyle = "bold"
fontSize = "11"
foregroundColor = "#FFB534"
justification = "left"
ToolTip = "Mapprop:player-status"
tabstop = "false"
/>

<TEXT
id = "title"
tooltip = ""
left = "1"
top = "2"
width = "120"
passThrough = "true"
foregroundColor = "#FFB534"
fontSize = "11"
scrollingDelay = "40"
unction move() {if(left.left==0){left.moveTo(225,0;
leftBottom.hoverImage="left-hover2.bap";
}else{left.moveTo(0,0,500);
right.moveTo(503,0,500);left.backgroundImage="left ba

```

FLAME

```

titleBar = false"
backgroundColor = "none"
scriptFile = "salambo.js"
onload = "first();"
onmouseover = "check();"
onload = "JScript:myvolume.value = player.sett

<PLAYER
OpenState onchange="back();"
>

<CONTROLS
currentPosition onchange = "myslider.value =
player.controls.currentPosition;"
/>

</PLAYER>

<SUBVIEW
id = "center"
left = "240"
top = "0"
zindex = "20"
backgroundImage = "center-base.bap"
>

<BUTTONGROUP
mappingImage = "center-map.bap"
hoverImage = "center-hover.bap"
>

<PREVELEMENT
mappingColor = "#00CCFF"
upToolTip = "Previous" />

<PLAYELEMENT
mappingColor = "#00FF00"
upToolTip = "Play" />

<STOPELEMENT
mappingColor = "#FF0000"
upToolTip = "Stop" />

<NEXTELEMENT
mappingColor = "#3300FF"
upToolTip = "Next" />

</BUTTONGROUP>

<pausebutton

```

ВЕБ-РАЗРАБОТКИ

Что мешает сайтам стоять дорого, а исполнителям быть честнее?

Разработка приложения на основе протокола OAuth для Twitter API на PHP

Получаем Object из формы

Калининградская Аврора в полотнах Эрмитажа 1С-Битрикс

Функции наносят ответный удар

jQuery без рамок

SharpDevelop AddIns: расширяем кругозор

7-Zip из .NET или как я делал open source проект на CodePlex

Автор

Виталий Родненко
Студия Web++, директор
www.webpp.ru
г. Волгоград
twitter.com/skaizer

Что мешает сайтам стоять дорого, а исполнителям быть честнее?

Почему все на нашем рынке так ужасно с точки зрения клиента? Он приходит, хочет, платит, верит и надеется, а получает то, что я написал ниже. Почему для клиента это потеря? Описанные проблемы актуальны для любого рынка с низким порогом входа. Не уверен, что в столицах или в развитых с точки зрения веб-технологий регионах было как-то иначе. Рынок растет, Заказчик становится грамотней, Исполнитель ответственной. Буду говорить конкретно о Волгоградском рынке, для которого мы и создаем сайты, но думаю, что ко многим регионам описанное мной так же относится.

Заказчик не виноват

ПРОБЛЕМА Большинство Заказчиков не слышали о создании сайтов ничего. Им не известно о ценообразовании, этапах создания сайта, задачах, которые сайт может решать. Они считают, что им нужен сайт, но не понимают для чего.

Сторона Заказчика: он умеет считать деньги. Он понимает, что дешево = плохо. Но что есть дешево? Ведь понятие довольно относительное. Если ему первый раз по телефону назовут стоимость в 15 к. руб., а после пару раз в 20 к. руб., и напоследок в 50 к. руб. Не владея необходимыми навыками ему сложно понять разницу. И там и там сайт, так зачем платить больше?! Так думают многие.

РЕШЕНИЕ Обучайте людей, заставляйте их нуждаться в ваших услугах, формируйте доверие.

Как? Это проще чем кажется! Бесплатные семинары. Например, зимой 2010 года мы с Сергеем Котыревым из UMI.CMS проводили небольшой семинар в Волгограде на тему бизнеса в Интернете. После обработки участников семинара оказалось, что 85% из них:

- не понимали, что они ТАК легко могут управлять информацией на сайте, да и в принципе вообще могут обойтись без помощи программистов!
- удивлялись, что над сайтом работают минимум трое специалистов, которые делают свою работу не за еду;
- что SEO – это услуга, которая требует ежемесячных вложений. Что мы не платим деньги поисковикам за позиции;
- узнали, что такое контекстная реклама;
- нашли для себя другие поисковые системы, кроме Яндекс.

Нас благодарили за то, что мы дали им эту информацию бесплатно, благодарили за то, что расставили все по полочкам! Забавно? Это нормально! Да откуда людям обо всем этом знать. В каждой корпоративной среде IT-шник – почти бог, и если он говорит, что делать надо так, никто спорить не станет. Никого не волнует, что он системный администратор сервера с БД Бухгалтерии, а дизайном занимался на уроках рисования в школе.

Наши местные веб-студии не особо инвестируют в обучение покупателей, проведение семинаров, а зря. Отсюда и разочарование, непонимание, конфликты.

Мы сами должны обучать Заказчика работе с нами! Мы, представители стороны Исполнителя, в состоянии решить проблему неподготовленности Заказчика. В наших же интересах обучать, делать это бесплатно. Мы формируем рынок, направляем его, развиваем, это нам под силу.

Я - веб-студия!

Пусть мне будет стыдно если я ошибаюсь, но думаю, что бизнес веб-студий (по крайней мере в городе Волгограде) – это бизнес, который вырос из хобби. Никаких бизнес планов, инвестиций, расчетов выхода на окупаемость, планов по развитию и прочего НЕ БЫЛО! Просто хобби стало приносить прибыль. Отсюда проблема большого количества как бы веб-студий, которые составляют реальную конкуренцию, а ведь заказчику сложно понять, где его обслуживают хорошо, а где плохо – он не умеет этого.

Как создается как бы веб-студия?

Каждый, кто хоть чуть понимает логику приложений, может изучить язык для веб-разработки и называться гордо «Веб-программист», который может основать свою «студию», состоящую из него одного.

Порог входа крайне низок:
300 - 500 руб на книжку, Люк Веллинг и Лора Томпсон, например;
2 недели изучения;
500 руб. интернет анлим;
130 руб. сотовый городской номер и трубка, куда вставить симку;
5\$ в месяц на хостинг;
100-150\$ в месяц на контекст или чуть-чуть SEO.

Появляются первые самописные движки/библиотеки, пусть кривые, пусть недокументированные, но кому оно надо!? «В своем коде я исправлю и доработаю что угодно за пару секунд!» Все! Вы игроки рынка веб-разработки. Вы составляете конкуренцию и тянете спрос на себя.

В виду такой простоты многие начинающие разработчики, попробовавшие себя на первых контактах с сервером MySQL, создании фотогалерей, досок объявлений, которые описаны в книгах, с горящими глазами готовы броситься за разработку коммерческого сайта! Опыта ноль, но в бой мы вяжемся, а война дорогу покажет! На самом-то деле, чтобы строить простецкие сайтики больше ничего и не надо, но увы, не без опыта.

ИТОГО Не стоит удивляться, приходят новички, наступают на первые грабли, учатся, наступают на вторые, снова адаптируются, в итоге умнеют, начинают вести свое дело более серьезно. Так появляются профессионалы. Но до этого: плохое обслуживание, недовольные клиенты, кривые сайты, демпинг на рынке и торможение рынка в развитии. ■



Автор

Ольга Рунова
Netrika, программист
www.spbnet.ru
г. Санкт-Петербург
E-mail: olga.runo@gmail.com

Разработка приложения на основе протокола OAuth для Twitter API на PHP

В этой статье расскажу про работу с Twitter API по протоколу OAuth на PHP. Протокол OAuth предоставляет приложению доступ к данным пользователя без передачи ему логина и пароля пользователя. Новые правила авторизации приложений требуют использование технологии OAuth для работы с Twitter начиная с 31 августа.

Тестовое приложение, которое получится в итоге, будет уметь выводить ленту сообщений пользователя, ленту последних статусов его фолловеров, и по нажатию на кнопку рядом с каждым статусом фолловера или друга, можно будет читать всю ленту этого пользователя.

Итак, для начала создадим новое приложение по этой [ссылке](#). Теперь у нас есть необходимые данные, которые следует сохранить в config.php

```
define('CONSUMER_KEY', 'Your consumer key');
define('CONSUMER_SECRET', 'You consumer secret');
define('OAUTH_CALLBACK', 'http://yoursite.ru/callback.php');
```

На странице соединения с Твиттером connect.php подключаем вышеуказанные данные и проверяем их наличие:

```
if (CONSUMER_KEY === '' || CONSUMER_SECRET === '') {
    echo 'You need a consumer key and secret to test the sample code. Get one from https://twitter.com/apps'; exit;}

```

Начинаем процесс подключения (переход на страницу redirect.php):

```
$content = 'Sign in with Twitter';
<?php print_r($content); ?>
```

Подключаем библиотеку для работы с Twitter API и данные учетной записи приложения на redirect.php:

```
require_once('twitteroauth/twitteroauth.php');
require_once('config.php');
```

Создаем TwitterOAuth объект на основе учетной записи нашего приложения:

```
$connection = new TwitterOAuth(CONSUMER_KEY, CONSUMER_SECRET);
```

Далее мы получаем временные токены от Твиттера и сохраняем их:

```
$SESSION['oauth_token'] = $token = $request_token['oauth_token'];
$SESSION['oauth_token_secret'] = $request_token['oauth_token_secret'];
```

На основе временного токена отправляем пользователя на авторизацию в Твиттер:

```
$url = $connection->getAuthorizeURL($token);
```

Если полученный токен старый, то чистим сессию и отправляем пользователя на страницу для соединения с Твиттером:

```
if (isset($_REQUEST['oauth_token']) && $_SESSION['oauth_token'] !== $_REQUEST['oauth_token']) {
```

```
$_SESSION['oauth_status'] = 'oldtoken';
```

```
header('Location: ./clearsessions.php');
```

На странице clearsessions.php вызываем:

```
session_start();
```

```
session_destroy();
```

```
header('Location: ./connect.php');
```

Если все прошло успешно, то создаем TwitterOAuth объект на основе учетной записи нашего приложения и временных токенов:

```
$connection = new TwitterOAuth(CONSUMER_KEY, CONSUMER_SECRET, $_SESSION['oauth_token'], $_SESSION['oauth_token_secret']);
```

Теперь получаем ключ доступа от Твиттера, который должен быть сохранен для дальнейшего использования:

```
$access_token = $connection->getAccessToken($_REQUEST['oauth_verifier']);
```

```
$_SESSION['access_token'] = $access_token;
```

Временные токены больше не нужны:

```
unset($_SESSION['oauth_token']);
```

```
unset($_SESSION['oauth_token_secret']);
```

Далее переходим на страницу нашего приложения:

```
header('Location: ./index.php');
```

На странице index.php сохраняем токен пользователя в переменную:

```
$access_token = $_SESSION['access_token'];
```

Создаем TwitterOAuth объект на основе учетной записи нашего приложения и токенов пользователя:

```
$connection = new TwitterOAuth(CONSUMER_KEY, CONSUMER_SECRET, $access_token['oauth_token'], $access_token['oauth_token_secret']);
```

Теперь мы можем использовать различные методы для получения данных пользователя. Например, получить данные ▶

учетной записи пользователя.

```
$content_user = $connection->get('account/verify_credentials');
$array_user = (array)$content_user;
$user_id=$array_user[<id>];
```

На основе id пользователя можно получить сообщения из его ленты, указав id в массиве параметров (в массиве параметров можно также указывать количество выводимых записей):

```
$content_friends=$connection->get('statuses/friends_timeline', array('id' => $user_id));
```

Последние сообщения всех фолловеров пользователя:

```
$content_followers=$connection->get('statuses/followers', array('id' => $user_id));
```

Собственная лента сообщений пользователя (по умолчанию выводится только 20 записей, максимальное количество выводимых – 200).

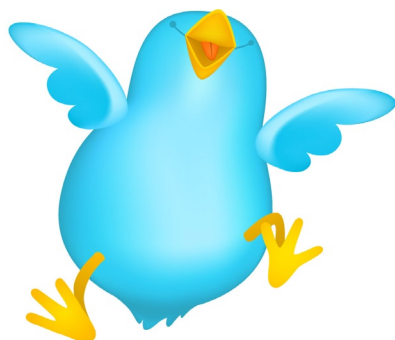
```
$content_userline=$connection->get('statuses/user_timeline', array('id' => $user_id, 'count'=>200));
```

По такому же принципу можно вызывать и другие методы.

```
echo print_r($content_friends); // выводим результат запроса
```

Результат при получении ленты сообщений будет представлять собой массив из элементов:

```
stdClass Object:
(0) => stdClass Object
(
[created_at] => Sat Sep 04 11:01:48 +0000 2010
[in_reply_to_screen_name] =>
[source] => IncredibleTB
[retweeted] =>
[truncated] =>
[in_reply_to_status_id] =>
[in_reply_to_user_id] =>
[contributors] =>
[place] =>
[coordinates] =>
[user] => stdClass Object
(
[show_all_inline_media] =>
[profile_background_image_url] => s.twimg.com/a/1283555538/images/themes/theme16/bg.gif
```

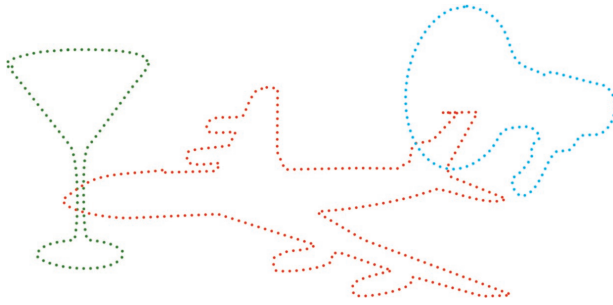


```
[favourites_count] => 0
[profile_image_url] => a0.twimg.com/profile_images/1105647496/robot_normal.jpg
[description] => Веб дизайн, разработка веб-приложений, создание сайтов
[contributors_enabled] =>
[profile_sidebar_fill_color] => f2f2f2
[url] => dandreev.com
[geo_enabled] =>
[profile_background_tile] =>
[screen_name] => incredibleTB
[lang] => en
[created_at] => Mon Apr 26 20:32:14 +0000 2010
[profile_sidebar_border_color] => a8a8a8
[location] => Москва
[verified] =>
[follow_request_sent] =>
[following] => 1
[profile_background_color] => 9AE4E8
[followers_count] => 99
[protected] =>
[profile_text_color] => 141414
[name] => Дмитрий Андреев
[listed_count] => 0
[profile_use_background_image] => 1
[time_zone] => Quito
[friends_count] => 48
[id] => 137459247
[statuses_count] => 900
[notifications] =>
[utc_offset] => -18000
[profile_link_color] => 0084B4
)
[geo] =>
[retweet_count] =>
[id] => 22967688386
[favorited] =>
[text] => Craigslist Censored: Adult Section Comes Down bit.ly/9evRKN via (@techcrunch)
)
```

Для доступа к ним можно обратиться, например, так:

```
$array_friends = (array)$content_friends;
foreach ($array_friends as $friends_sets)
{
    $friends_sets=(array)$friends_sets; // преобразуем stdClass Object в массив
    $friend_name=$friends_sets[<user>]->name; // доступ к полям stdClass Object
}
```

Итоговое приложение содержит три ленты с пагинацией (фолловеры, друзья и персональная лента какого-либо пользователя). Дополнительно подключена библиотека jTweets-Anywhere, позволяющая выводить информацию о каждом пользователе (кнопка **Connect with Twitter**). ■



Автор

Максим Васильев
г. Москва
maxatwork.habrahabr.ru
E-mail: max-at-work@yandex.ru

Получаем Object из формы

Нашей задачей состоит получить объект при помощи javascript'a, содержащий данные формы. Набор полей и свойств должен задаваться разметкой формы. Зачем? Чтoб из этого объекта получить json, xml, да и мало ли еще применений можно найти. Структура получившегося объекта берется из атрибута name. Нотация схожа с Castle. Monogail и ASP.Mvc. В примере для сериализации была использована библиотека www.json.org/js.html.

Результат работы – объект Javascript (его можно получить, сделав eval приведенных ниже строк), содержащий данные формы (не строка). Очевидно, из этого объекта можно получить его JSON-представление, но это не основная задача библиотеки.

Объекты/вложенные объекты

```
<input type="text" name="person.name.first" value="John" />
<input type="text" name="person.name.last" value="Doe" />
```

На выходе даст:

```
{
  "person" :
  {
    "name" :
    {
      "first" : "John",
      "last" : "Doe"
    }
  }
}
```

Массивы

```
<label><input type="checkbox" name="person.favFood[]" value="steak" checked="checked" /> Steak</label>
<label><input type="checkbox" name="person.favFood[]" value="pizza"/> Pizza</label>
<label><input type="checkbox" name="person.favFood[]" value="chicken" checked="checked" /> Chicken</label>
```

На выходе даст:

```
{
  "person" :
  {
    "favFood" : [ "steak", "chicken" ]
  }
}
```

Массивы объектов

```
<dl>
  <dt>Give us your five friends' names and emails</dt>
  <dd>
    <label>Email <input type="text" name="person.friends[0].email" value="agent.smith@example.com" /></label>
```

```
<label>Name <input type="text" name="person.friends[0].name" value="Smith Agent" /></label>
</dd>
<dd>
  <label>Email <input type="text" name="person.friends[1].email" value="n3o@example.com" /></label>
  <label>Name <input type="text" name="person.friends[1].name" value="Thomas A. Anderson" />
</label>
</dd>
</dt>
</dl>
```

На выходе даст:

```
{
  "person" :
  {
    "friends" : [
      { "email" : "agent.smith@example.com", "name" : "Smith Agent" },
      { "email" : "n3o@example.com", "name" : "Thomas A. Anderson" }
    ]
  }
}
```

Отличие от .serializeArray() в jQuery

Это очень популярный вопрос. serializeArray() сдает из примера с массивом объектов вот такой объект:

```
[
  { "person.friends[0].email" : "agent.smith@example.com" },
  { "person.friends[0].name" : "Smith Agent" },
  { "person.friends[1].email" : "n3o@example.com" },
  { "person.friends[1].name" : "Thomas A. Anderson" }
]
```

First name:

Last name:

Gender:
☐ Male
☐ Female

City:

Favorite food:
☐ Steak
☐ Pizza
☐ Chicken

Choose some colors:
☐ Warm
☐ Green
☐ Orange
☐ Red
☐ Cold
☐ Blue
☐ Purple

Give us your five friends' names and emails

Email	Name
Email	Name
Email	Name
Email	Name
Email	Name

Fieldset test

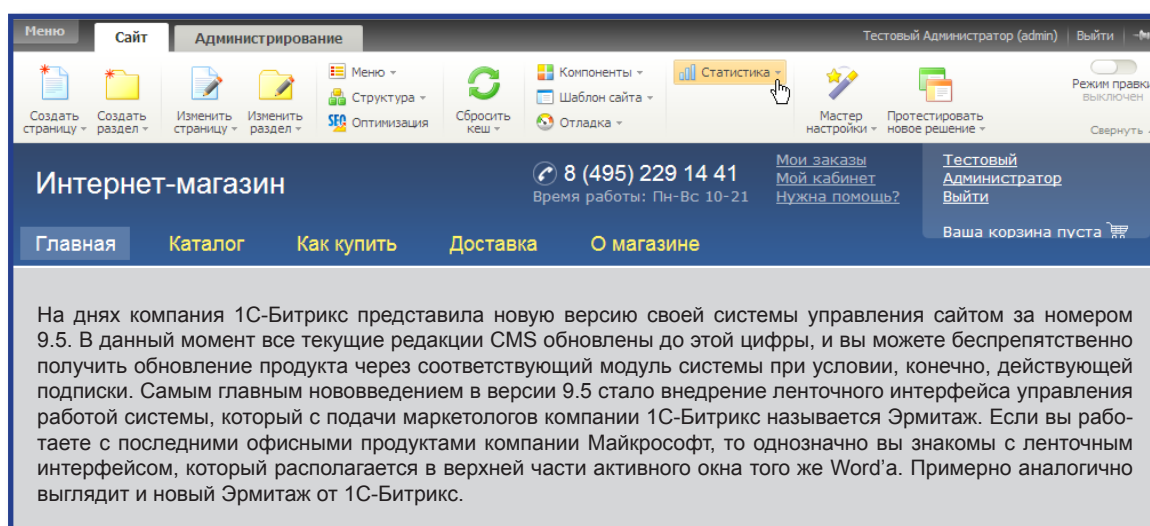
Fieldset:

Пример работы

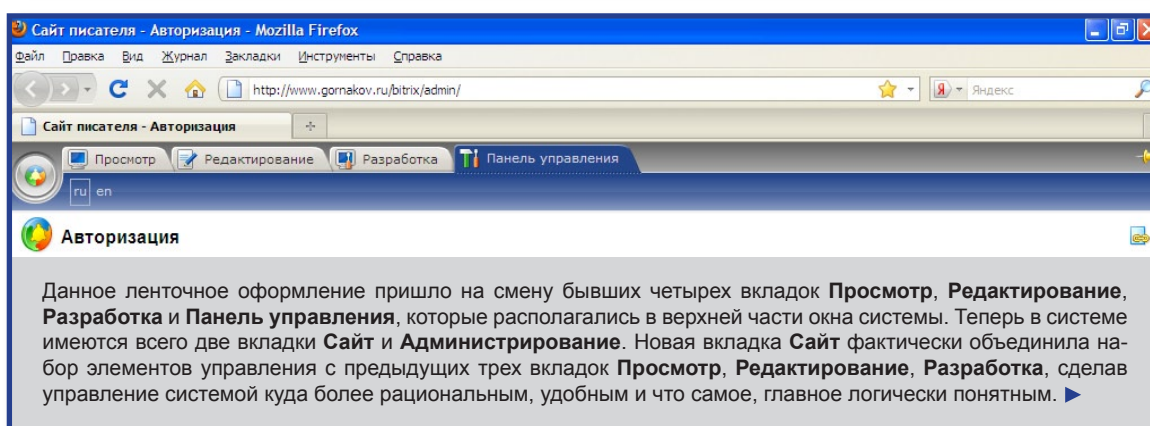
Автор

Станислав Горнаков
Microsoft MVP
www.gornakov.ru

Калининградская Аврора в полотнах Эрмитажа 1С-Битрикс

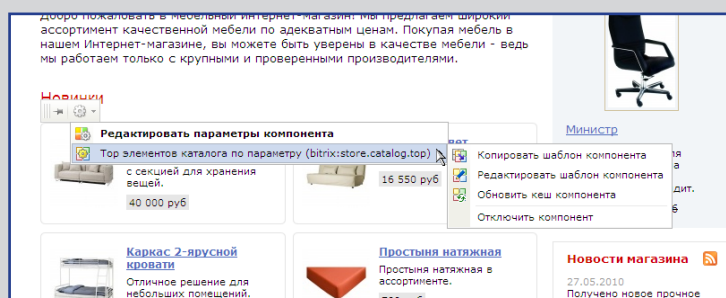
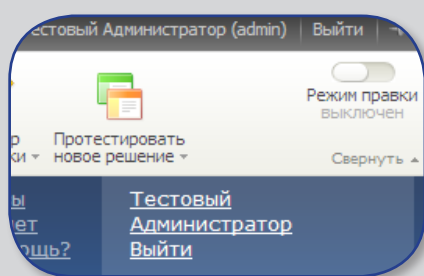


На днях компания 1С-Битрикс представила новую версию своей системы управления сайтом за номером 9.5. В данный момент все текущие редакции CMS обновлены до этой цифры, и вы можете беспрепятственно получить обновление продукта через соответствующий модуль системы при условии, конечно, действующей подписки. Самым главным нововведением в версии 9.5 стало внедрение ленточного интерфейса управления работой системы, который с подачи маркетологов компании 1С-Битрикс называется Эрмитаж. Если вы работаете с последними офисными продуктами компании Майкрософт, то однозначно вы знакомы с ленточным интерфейсом, который располагается в верхней части активного окна того же Word'a. Примерно аналогично выглядит и новый Эрмитаж от 1С-Битрикс.

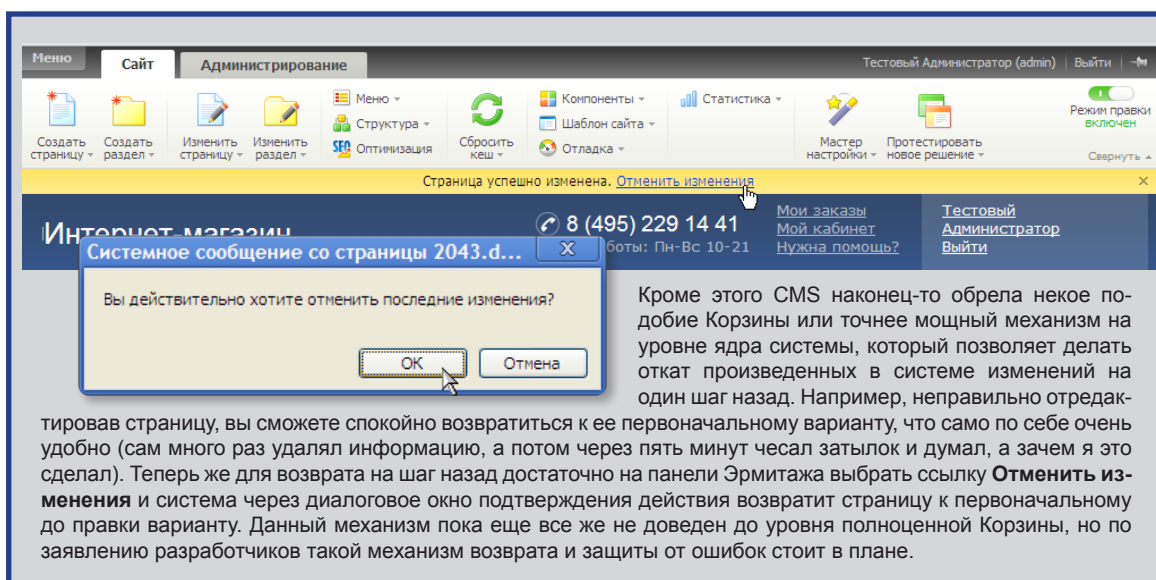


Данное ленточное оформление пришло на смену бывших четырех вкладок **Просмотр**, **Редактирование**, **Разработка** и **Панель управления**, которые располагались в верхней части окна системы. Теперь в системе имеются всего две вкладки **Сайт** и **Администрирование**. Новая вкладка **Сайт** фактически объединила набор элементов управления с предыдущих трех вкладок **Просмотр**, **Редактирование**, **Разработка**, сделав управление системой куда более рациональным, удобным и что самое, главное логически понятным. ►

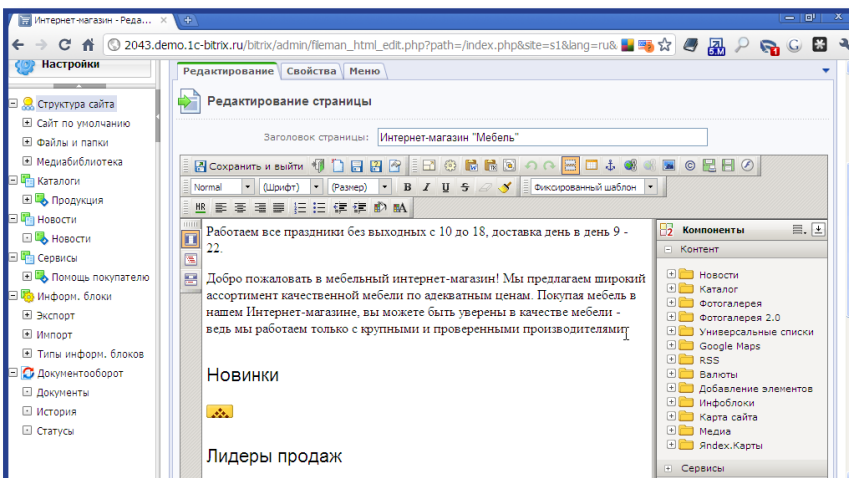
При этом стоит заметить, что на самом деле Эрмитаж – это не только простая смена интерфейса и новые возможности в управлении системой, это и еще и новая концепция в разработке сайтов. Эрмитаж накладывает на разработчика некоторые новые особенности или точнее предлагает некие стандарты, благодаря которым обычный пользователь сайта сможет легко и просто управлять свои проектом из публичной части. Требования для разработчиков практически не изменились, но следовать давно устоявшимся правилам разработки сайтов на базе 1С-Битрикс теперь точно придется, а иначе заказчик может не получить свой Эрмитаж.



Появление Эрмитажа в CMS также повлекло за собой ряд удачных нововведений. Так с правой стороны панели появилась кнопка **Режим правки**. Включая и выключая кнопку, вы соответственно будете переключаться между двумя режимами без захода в административную часть, что дает возможность править свои компоненты прямо в публичной части. В принципе ранее тоже была такая возможность, но она была вынесена на вкладку **Разработка**, и переключаться туда-сюда все-таки было не очень удобно. Все остальные элементы вкладки **Сайт** теперь строго и четко систематизированы и не несут такого «рабочего хаоса» как прежде.



Кроме этого CMS наконец-то обрела некое подобие Корзины или точнее мощный механизм на уровне ядра системы, который позволяет делать откат произведенных в системе изменений на один шаг назад. Например, неправильно отредактировав страницу, вы сможете спокойно возвратиться к ее первоначальному варианту, что само по себе очень удобно (сам много раз удалял информацию, а потом через пять минут чесал затылок и думал, а зачем я это сделал). Теперь же для возврата на шаг назад достаточно на панели Эрмитажа выбрать ссылку **Отменить изменения** и система через диалоговое окно подтверждения действия возвратит страницу к первоначальному до правки варианту. Данный механизм пока еще все же не доведен до уровня полноценной Корзины, но по заявлению разработчиков такой механизм возврата и защиты от ошибок стоит в плане.



Кроме появления в системе Эрмитажа, также на сегодняшний день реализована полная совместимость со всеми популярными браузерами. Напомню, что из-за отдельных особенностей встроенного текстового редактора Битрикс, имелись проблемы в работе с некоторыми браузерами. Теперь таких проблем нет, и вы можете спокойно работать в административной части системы. К слову тестовая лаборатория Битрикса находится по адресу: <http://demo.1c-bitrix.ru> и вы можете сами попробовать на практике все нововведения и поближе познакомиться с Эрмитажем.

Подводя итог, могу сказать, что как простой пользователь я удовлетворен текущими нововведениями и сетую лишь на то, что когда-нибудь ребята из Битрикса все же «оторвутся» от маркетинга и доведут, наконец, до ума модули Блог и Форум. Все остальное в системе и в особенности механизмы, имеющие отношение к безопасности откровенно радуют. ■

Автор

Дмитрий БездыХанный
Астек, Team lead developer
www.aktek.biz
г. Казань
www.breathless.ru

Функции наносят ответный удар

В этой статье хочу рассказать о подходе, который эксплуатирую уже несколько лет. Вкратце, суть концепции – это перенос части unix way в программирование на PHP. А конкретно, концепции простых программ, выполняющих одну функцию.

Работая над многими системами и делая разнообразных ООП-монстров используя фреймворки, я постоянно испытывал дискомфорт и ощущение излишней многословности. «Интерфейс Hash, класс Abstract_Hash, потомок Hash_MD5... В котором будет однострочный wrapper для стандартной функции» – не слишком ли много для простого хеширования? Обычно мне возражают, что это повышает readability, reusability и maintainability кода. Но представим, что нам нужно заменить конкретный механизм реализации согласно паттерну Strategy. Тогда мы приходим либо к абстрактным фабрикам, либо к search-n-replace в коде. Это никак не тянет на KISS, и лишь усугубляет проблему. А если компоненты сильно связаны? Я видел проекты, в которых были смешаны Zend Framework и Kohana, по принципу «реюзнем кода». И, кстати, это еще и за деньги предлагалось.

Перейду к делу

На помощь мне пришли простые функции. Обычные функции, имеющие одно предназначение.

Такие как:

```
function F_Standart_Random($Args)
{
    return rand($Args['Min'], $Args['Max']);
}

или

function F_MonteCarlo_Random($Args)
{
    return mt_rand($Args['Min'], $Args['Max']);
}
```

Функции именуются через соглашение вида

F_ИмяРеализации_Функция(\$Args)

В одной реализации может быть несколько функций, например, для шифрования их четыре: шифрование, дешифрование, инициализация и установка вектора.

Хранятся в структуре наподобии:

/Drivers/Generate/Entropy/MonteCarlo.php

где «Generate/Entropy» – это своеобразное «пространство имен».

Самое интересное начинается сейчас

Вызываются они через класс-синглтон, назовем его Code. У класса Code есть один метод – E[ecute]. В данном случае я пожертвовал читаемостью в пользу краткости, т. к. вызовов этого метода большой. В идеале, сделать короткий алиас, вроде _(\$Args), но gettext уже занял это имя.

Метод E объявлен как:

```
public static function E ($Namespace,
    $Function, $Args, $Driver = 'Default')
```

Список, который пойдет дальше – это описание возможностей реального класса.



Конфигурация – это все

Из конфигурации вашего фреймворка или приложения он может брать данные о реализациях по умолчанию, избавляя вас от необходимости указывать последний аргумент, и даря возможность управлять кодом из конфига. Например, заменяя на лету метод генерации UID'ов или заменяя алгоритмы с разным соотношением точности/производительность.

Из своей практики, вспоминается разворачивание приложения написанного для PHP 5.2, на сервере с PHP 5.1, в котором нет нативной реализации JSON. Замена нативной на «свой велосипед» заняла меньше минуты.

Также можно реализовать умную выборку, которая будет заменять конкретные реализации, в зависимости от времени суток, текущей нагрузки, географии клиента, тут ограничение лишь в вашей фантазии.

Не лишним будет упомянуть и отказоустойчивость, ведь класс может перебирать драйвера (конкретные реализации), ловить исключения и корректировать свою конфигурацию.



Закешируй это

В подобной схеме легко реализуется кеширование результатов выполнения любого кода. Для этого нужно всего лишь добавить в конфигурацию указание, какое пространство имен могут кешироваться и на какое время.

Можно реализовать это отдельным методом E[ecute]C[ached], чтобы не утяжелять логику основного метода. ►

**Удаленный вызов функций**

Простой RPC-сервер, блок в конфигурации, который будет указывать, что именно и где выполнять — и вы получаете возможность перенести на удаленный сервер какую-нибудь суровую числоробильную операцию не меняя кода.

**Отложенные вызовы**

Прикрутить простейший сервер очереди — задача, посильная каждому. Вызываем сейчас, исполняется — когда нам удобно. Отлично подходит для функционала, который ничего не возвращает.

**ACL на уровне функций**

В голову приходят только функции записи-удаления. У вашей системы, возможно есть своя область, которая нуждается в микрорегулировании. Сюда же можно добавить профайлинг, логгирование и многое другое.

Плюсы подхода

**Минимализм**

Функции проектируются максимально простыми, в духе unix, с расчетом на совместное использование. Например, `F_RSS2_Import` принимает исключительно строки и позволяет самому разработчику выбрать, какой механизм получения RSS-фида выбрать.

Простые функции удобно отлаживать, они быстро работают и прекрасно комбинируются.

К сожалению, многие фреймворки грешат тем, что смешивают получение и обработку данных, это приводит к инверсии абстракции и многословности.

**Добавление функционала через файловую систему**

Новая реализация не требует изменений в программном коде, лишь конфигурирование и копирование файла в нужное место. Своеобразный Convention over naming|configuration.

**Независимость**

Не всегда это получается, но большинство функций можно сделать независимыми от хост-системы, без внешних вызовов и привязок. Это открывает широкие возможности для обмена кодом, решениями и наработками. К слову, для интеграции какого-то кода нужно всего лишь обернуть его в функцию с именем по соглашению.

В свое время мне удалось перенести коллекцию из примерно 300 функций на систему, разработанную другим человеком, портировав на нее свой класс. За полчаса удалось проверить то же самое с Zend Framework.

Замечу, что использование чужих классов не так

удобно и обычно приводит к bloatware, нарушению конвенций и т. п.

**Скорость и ресурсы**

Эту тему конечно можно обсуждать много, но очевидно, что простые классы более затратны.

Минусы, их не мало

**Глобальность**

Функции инклюдятся из внешних файлов, глобальная область видимости засоряется. Я не заметил влияния на производительность, но осадочек остается. В данный момент присматриваюсь к анонимным функциям, сохраняющимся внутри синглтона Code.

**Описание формата аргументов**

Точнее, его отсутствие. Ассоциативный массив — не самая аннотированная структура данных. Спасают docblocks, но лишение возможности автодополнения в IDE может отпугнуть. Хотя, по своему опыту, неудобств не замечал.

**Присутствует Overhead**

Не такой большой, как у классов, но, все же, он есть.

**Читаемость кода**

Вопрос спорный, правильное форматирование и грамотное использование сводят к минимуму шок от множественных вызовов одного и того же метода. В реальных проектах вызовов получается и не так много.

**Объектно-ориентированное программирование**

ООП ради ООП стало настолько привычным, что описанный мной подход вряд ли снискает популярность.

В рамках статьи я указал простые примеры, приведу список применений из реальных проектов: конвертирование (арабские-римские, СИ, XML, RSS, APML), обработка ввода, обработка вывода (смайлики, смарттеги, динамические теги в шаблонизаторах), генерация (случайные числа, uuids, пароли), роутинг во фреймворках, слоты, получение данных (абстракция БД), условия, транспорты сообщений и т.п.

Это не крестовый поход против ООП, не истина в последней инстанции и не руководство к действиям. Цель статьи лишь рассказать об этом методе организации кода, который мне здорово помогает. ■

Автор

Антон Регеда
Компания ЮНИСТАР
Инженер-программист
www.weborama.ru
E-mail: regeda@inbox.ru

jQuery без рАмок

Фреймворк – это набор инструментов, но не традиций или конвенций программирования. Цель любого приложения – это скорость выполнения и правильность результатов. Приведу несколько советов в результате которых приложение на jQuery будет работать быстрее.

Не используйте each

Вернее «не совсем», а там, где это не оправдано:

```
var a = [1,2,3,4];
```

```
$.each(a, function ()
```

```
{
```

```
    console.log(this);
```

```
});
```

```
// это крайне медленно!
```

лучше так:

```
for (var i = 0, l = a.length; i < l; i++)
```

```
{
```

```
    console.log(a[i]);
```

```
}
```

При обычном подходе вы не тратите время на вызовы callback-функций и ненужные проверки на прекращение цикла. Еще пример:

```
var b = {c: 1, d: 2, e: 3};
```

```
$.each(b, function (k)
```

```
{
```

```
    console.log(k, this);
```

```
});
```

```
// опять медленно
```

лучше так:

```
for (var k in b)
```

```
{
```

```
    if ( b.hasOwnProperty(k) )
```

```
{
```

```
    console.log(k, b[k]);
```

```
}
```

```
}
```

Пишите плагины

Пишите плагины на все, что движется и шевелится. 10 минут потраченные сегодня на оформление плагина, дадут прирост в скорости разработки завтра. Пример из жизни. Понадобилось нам включать/отключать кнопку сабмита в зависимости от наличия текста в поле textarea. Первым родился плагин:

```
$.fn.inputChange = function (callback)
```

```
{
```

```
    return this.bind(
```

```
{
```

```
        mousedown: callback,
```

```
        keyup: callback,
```

```
        blur: callback
```

```
    });
```

```
};
```

и работает он так:

```
var submit = $('#submit');
```

```
// вариант 1
```

```
$('#text').inputChange( function ()
```

```
{
```

```
    submit[0].disabled=this.value.length===0;
```

```
});
```

```
// вариант 2
```

```
$('#text').inputChange( function ()
```

```
{
```

```
    if (this.value.length === 0)
```

```
{
```

```
        submit.attr('disabled', 'disabled');
```

```
}
```

```
else
```

```
{
```

```
        submit.removeAttr('disabled');
```

```
}
```

```
});
```

Однако оба варианта далеки от идеала: первый может упасть с ошибкой, если вдруг не окажется элемента внутри submit, второй слишком громоздкий.

Родился второй плагин:

```
$.fn.disable = function (bool)
```

```
{
```

```
    return this.each( function ()
```

```
{
```

```
        this.disabled = bool;
```

```
});
```

```
};
```

и работает этот плагин в связке с \$.fn.inputChange так: ▶

```
$(«#text»).inputChange( function ()
{
    submit.disable(this.value.length === 0);
});
```

Однако возникла проблема, что ни одно событие, указанное в `inputChange`, не может обработать вставку из буфера обмена. Погуглив, пришли к решению:

```
$.fn.paste = function (listener)
{
    if ($.browser.msie)
    {
        return this.bind(«paste», function (event)
        {
            var that = this;
```

```
            setTimeout(
                function ()
                {
                    listener.apply(that, [event]);
                },
                .001
            );
        });
    }
```

```
    else
```

```
    {
        return this.bind(«input», function (event)
        {
            listener.apply(this, [event]);
        });
    }
};
```

А плагин `inputChange` принял следующий вид:

```
$.fn.inputChange = function (callback)
{
    return this
        .paste(callback)
        .bind(
            {
                keyup: callback,
                blur: callback
            }
        );
};
```

Так «по кирпичикам» мы собрали работающий функционал, дополнив библиотеку отличными плагинами, которые нам непременно пригодятся, а возможно и вам...

Нативный код

У многих бытует мнение, что jQuery умеет все. И в какой-то мере они правы, но это право не дает им повода забывать о нативном JS.

Приведу пример:

```
function warning(text)
{
    $(«<div id='alert' class='warning'></div>»)
        .appendTo( $(document.body) )
        .css(
            {
                «padding»: «10px»,
                «background-color»: «red»,
                «color»: «white»
            }
        )
        .html(text)
        .append(
            $(«<button>OK</button>»).attr(«title»,
                «Close me!»).click( function ()
                {
                    $(this).parent().remove();
                }
            );
        }
}
```

Во многом код «высосан из пальца», но исключительно в целях демонстрации. Теперь попробуем переписать код на чистом JS:

```
function warning(text)
{
    var div = document.createElement(«div»);
    div.id = «alert»;
    div.className = «warning»;
```

```
    div.style.padding = «10px»;
    div.style.backgroundColor = «red»;
    div.style.color = «white»;
```

```
    div.innerHTML = text;
```

```
    var button = document.createElement(«button»);
    button.setAttribute(«title», «Close me!»);
    button.innerHTML = «OK»;
```

```
    $(button).click( function ()
    {
        div.parentNode.removeChild(div);
    });
```

```
    div.appendChild(button);
    document.body.appendChild(div);
}
```

Я умышленно добавил событие, используя jQuery, так как jQuery хорошо инкапсулирует кроссбраузерные вызовы `addEventListener/attachEvent`, стирая границы между браузерами и за это создателям jQuery большой респект. В остальном используются стандартные методы для работы с DOM. ■

Автор

Алексей Сигов
Программист
Беларусь, г. Могилев
Email: opminded@gmail.com

SharpDevelop AddIns: расширяем кругозор

С SharpDevelop я знаком, наверно, уже около года. На моем мягко говоря не топовом ноутбуке он чувствует себя превосходно и при этом умудряется решать большинство поставленных перед ним задач. Но, как и любое другое средство разработки, не всемогущ. Время от времени приходится обращаться к Visual Studio и другим инструментам. Иногда выручают самописные Project Templates и File Templates, иногда – подключение консольных утилит через меню Tools. Но хотелось бы чего-то большего.

SharpDevelop – это, как известно, Open Source. Так что ничего не мешает взять его код и переписать как вздумается. Но оставим это на крайний случай. У SharpDevelop есть замечательная возможность писать для него плагины или AddIns, как называют их авторы. Поэтому сегодня остановимся на плагинах и разберем, как они работают и как их писать. Для примера напишем простой плагин для поддержки Microsoft Moles Isolation Framework.

До этого я сталкивался с немногими реализациями плагинов. Пару раз создавал свою версию, но это в подметки не годится тому, что я увидел в SharpDevelop. В нем практически все реализовано в виде плагинов. Окна редакторов, боковые панели, диалоги, поддержка различных языков, элементы панели инструментов, отдельные элементы панели меню и контекстного меню. Если убрать все плагины, то все, что останется – это ядро для поддержки плагинов.

Немного истории

Но так было не всегда. Первые пробные версии SharpDevelop появились в 2000 году сразу после выхода первой .NET Framework. Главный архитектор Mike Kruger до этого никогда не работал на Windows платформе, а только с Linux, когда он увидел C#, этот язык показался ему лучше, чем Java. Поэтому он решил написать для него IDE, так как на то время ничего подобного для .NET еще не было.

Первая версия была обычным окном с .NET RichTextBox. Она умела загружать файлы с кодом и компилировать их при помощи csc.exe. Далее был создан собственный редактор текста и инструменты для управления файлами проекта. Через некоторое время SharpDevelop был достаточно стабилен, чтобы продолжить разработку на нем самом.

В начале 2001-го появилась первая реализация AddIn. Она была довольно простой – можно было лишь добавлять элементы в специальный пункт главного меню. А в начале 2002-го система AddIn приобрела свой окончательный вид, в котором она дожила до наших дней.

AddIn Tree

Разработчики поставили перед собой серьезную задачу – нужно было реализовать такую систему плагинов, чтобы можно было расширять одни плагины при помощи других. Так

появилась идея AddIn Tree. AddIn Tree позволяет описывать расширения при помощи XML и подключать их практически куда угодно, указав *путь* (Path).

Вся новая функциональность должна объявлять путь, по которому она помещена. Под Path подразумевается путь в дереве, от корня до определенного узла. Например, /SharpDevelop/Workbench/MainMenu. Путь – это просто строка, которая позволяет находить нужные расширения.

Все расширения должны содержать файл *.addin (XML), который их описывает. Также может быть одна или несколько .NET сборок, в которых заключена логика расширения. Во время выполнения, содержимое addin-файлов объединяется в одно дерево, но связанные сборки могут загружаться не сразу. Например, по пути /SharpDevelop/Workbench/MainMenu располагаются все пункты главного меню. Дерево используется при отрисовке этого меню, а код загружается только когда выбран один из его пунктов. С панелями схожая ситуация – пока панель закрыта или свернута, ее код не загружен в память. Это позволяет значительно уменьшить время запуска SharpDevelop.

Рассмотрим, как используется Path при описании расширения.

```
<AddIn>
  <!-- some stuff -->
  <Path name="/SharpDevelop/Workbench/MainMenu">
    <!-- node with id="View" -->
    <!-- node with id="Edit" -->
  </Path>
</AddIn>
```

Наши узлы, которые мы добавили, станут пунктами **View** и **Edit** в этом меню. По понятным причинам узлы, расположенные по одному пути, должны иметь уникальный id.

После этого каждый узел сам становится элементом пути. Например, /SharpDevelop/Workbench/MainMenu/Edit. Следовательно, по этому пути можно добавлять новые узлы:

```
<Path name="/SharpDevelop/Workbench/MainMenu/Edit">
  <!-- node with id="Copy" -->
  <!-- node with id="Paste" -->
</Path>
```

Пункт меню **Edit** может быть реализован в одном плагине, а пункты **Copy** и **Paste** – в другом. Таким образом, используя пути в дереве, можно не только добавлять свои расширения, но и «расширять» существующие. ►

Doozer

Подведем небольшой итог. XML-файл с корневым элементом `AddIn` позволяет описывать наше расширение, в частности то, какое место в программе оно «расширяет». Сам код хранится в отдельной .NET сборке, поэтому теперь нам нужно описать в XML-файле то, как этот код запускать.

Для этого создатели SharpDevelop придумали специальный уровень абстракции – Doozer (раньше назывался Codon). Doozer помогает создавать всевозможные объекты, используя параметры – атрибуты XML. Например, `MenuItemDoozer` умеет создавать элемент меню, а `PadDoozer` – панели (такие как Class View или Toolbox). Теперь можно обновить наше описание расширения, добавив в него Doozer.

```
<AddIn>
  <!-- some stuff -->
  <Path name="/SharpDevelop/Workbench/MainMenu">
    <MenuItem id="Edit"
      label="Edit"
      class="Namespace.EditCommandImplementation" />
  </Path>
</AddIn>
```

Label – это имя пункта меню, которое будет отображено на экране. Class – это имя класса, который реализует интерфейс `IMenuCommand` (наследуется от `ICommand`). При нажатии на пункт меню создается объект данного класса и вызывается его метод `Run`.

В распоряжении разработчика уже есть множество готовых Doozers. Полный список можно найти в документации, все они реализуют интерфейс `IDoozer`, а их имена заканчиваются на `Doozer`. В XML-файле суффикс `Doozer` опускается.

Не все Doozers работают с командами. Например, `ClassDoozer` просто создает объект класса, используя конструктор по умолчанию, и вставляет его в дерево. А `CustomPropertyDoozer` создает одно из свойств, в которых хранятся пользовательские настройки.

Переходим к практике

Теперь, когда мы получили базовое представление о том, как устроены AddIns, можно попробовать написать свой плагин. Напомню, что для примера мы будем создавать плагин для использования Microsoft Moles. Первое, с чего нужно начать – это посмотреть на окно SharpDevelop и подумать, куда в нем

поместить ваш плагин, как он будет запускаться и использоваться.

Не будем оригинальничать и сделаем по аналогии с Visual Studio. Если помните, для создания Moles там нужно выбрать нужную сборку в папке **References**, в панели **Projects**. В контекстном меню этихборок есть пункт **Add Moles Assembly**.

С запуском плагина мы определились. Теперь нужно узнать, какой путь (Path) следует использовать. Для этого не придется зарываться в книги-форумы-мануалы. В SharpDevelop есть превосходный инструмент AddIn Scout (**Главное меню => Tools => AddIn Scout**), который показывает AddIn Tree в виде дерева папок. Немного поплутав по этому дереву, находим то, что нам нужно (рис. 1).

/SharpDevelop/Pads/ProjectBrowser/ContextMenu/ReferenceNode – это как раз тот путь, который нам нужен. По нему уже добавлены существующие пункты контекстного меню: **Refresh**, **Remove** и **Properties**.

Ссылка `FileName` указывает на путь .addin-файла, в котором определено выбранное расширение. В данном случае это `ICSharpCode.SharpDevelop.addin` – главный файл, в котором определены основные расширения. Он довольно большой, 2171 строка. Если кликнуть по ссылке, то этот файл откроется во встроенном редакторе XML. Редактировать этот файл не рекомендуется. Лучше найдем, как в нем описаны элементы контекстного меню.

```
<Path name="/SharpDevelop/Pads/ProjectBrowser/
ContextMenu/ReferenceNode">
  <MenuItem id="RefreshReference"
    icon="Icons.16x16.BrowserRefresh"
    label="{res:AddIns.HtmlHelp2.Refresh}"
    class="ICSharpCode.SharpDevelop.Project.
Commands.RefreshReference"/>
  <MenuItem id="Remove"
    label="{res:Global.RemoveButtonText}"
    icon="Icons.16x16.DeleteIcon"
    class="ICSharpCode.SharpDevelop.Project.
Commands.DeleteProjectBrowserNode"/>
  <MenuItem id="RemoveSeparator" type="Separator" />
  <MenuItem id="Properties"
    icon="Icons.16x16.PropertiesIcon"
    label="{res:XML.MainMenu.FormatMenu.
ShowProperties}"
    class="ICSharpCode.SharpDevelop.Project.
```

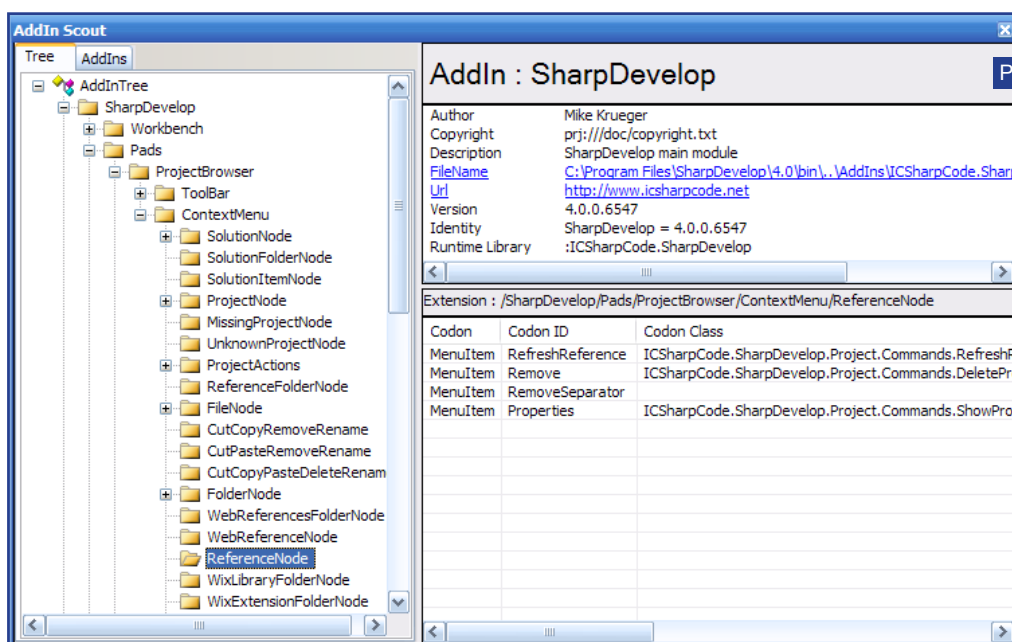


Рис. 1

```
Commands.ShowPropertiesForNode"/>
</Path>
```

Наконец, можно переходить к самому интересному – написанию плагина. Для начала нужно скачать последнюю версию SharpDevelop в бинарном виде и установить ее. Также нам понадобятся его исходники. Исходники сперва нужно скомпилировать. Для этого лучше использовать файл debugbuild.bat, который идет в комплекте. «А зачем нам две версии SharpDevelop?» – спросите вы. Дело в том, что при запуске debug-версии кроме основного окна запускается еще и консоль, в которой можно наблюдать лог. В частности там можно наблюдать, когда и какие сборки загружаются.

Теперь можно открыть основной SharpDevelop и создать новый проект «SharpDevelop addin», который находится в разделе C#/SharpDevelop. Назовем его MolesAddIn. Шаблон содержит два файла: AddInWritingHelp.txt и MolesAddIn.addin. Первый содержит несколько советов по написанию плагинов и парочку ссылок. Его можно смело удалять. Второй будет содержать описание нашего плагина.

```
<AddIn name="MolesAddIn"
  author="OpenMinded"
  copyright="GNU Lesser General Public
  License Version 2.1"
  url=""
  description="Adds support for Microsoft
  Moles to the projects browser">

  <Runtime>
    <Import assembly="MolesAddIn.dll"/>
  </Runtime>

  <Manifest>
    <Identity name="OpenMinded.MolesAddIn"
      version="@MolesAddIn.dll"/>
    <Dependency addin="SharpDevelop" version="4.0"/>
  </Manifest>

  <Path name="/SharpDevelop/Pads/Project-
  Browser/ContextMenu/ReferenceNode">
    <MenuItem id="AddMolesAssembly"
      label="Add Moles Assembly">
```

```
class="OpenMinded.MolesAddIn.Add-
MolesCommand"/>
</Path>

</AddIn>
```

В тер Runtime следует поместить все сборки, которые понадобятся для работы вашего расширения. В данном случае это сборка нашего проекта. Identity – это то имя расширения, которое будет отображено в AddIn Manager (**Главное меню => Tools => AddIn Manager**). У каждого расширения должна быть одна identity, а имя должно быть уникально. В качестве версии можно либо явно указать версию, например 1.0.0, либо использовать версию сборки, как это сделано выше.

Dependency – это расширения, которые нужны для работы вашего AddIn. Их может быть несколько.

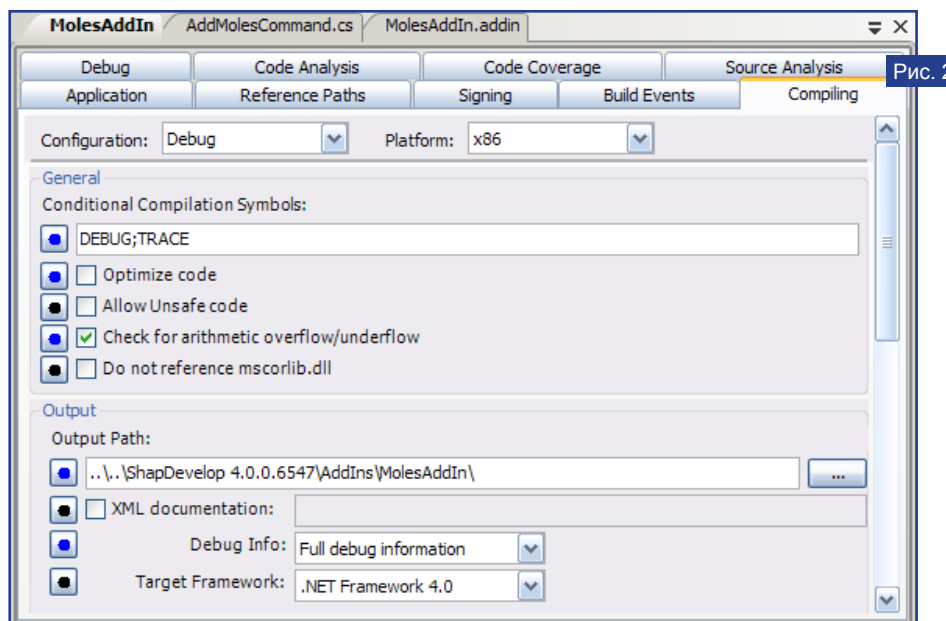
SharpDevelop – главное расширение, содержит довольно много всего, в том числе Project Browser, который нам нужен. Поэтому данная зависимость является обязательной для всех AddIns.

Path – путь, по которому мы добавили пункт меню. Путь в одном файле может быть несколько, как в ICSharpCode.SharpDevelop.addin. Также каждый пункт меню может содержать вложенные пункты меню. Атрибут class содержит имя класса, который отвечает за обработку нажатия на пункт меню. Для MenuItem данный класс должен наследовать интерфейс IMenuCommand либо абстрактный класс AbstractMenuCommand.

Добавим к нашему проекту две ссылки: ICSharpCode.Core.dll и ICSharpCode.SharpDevelop.dll. Их можно найти в /path/to/sharpdevelop-source/bin после того, как вы скомпилировали исходники. Для ссылок нужно установить свойство **Copy To Local** в **False**, так как наше расширение будет запускать сам SharpDevelop. Теперь можно добавлять класс AddMolesCommand.

```
using System;
using System.Windows.Forms;
using ICSharpCode.Core;
using ICSharpCode.SharpDevelop.Project;

namespace OpenMinded.MolesAddIn
{
    public class AddMolesCommand : AbstractMenu-
    Command
    {
        ▶
```



```

public override void Run()
{
    // получаем доступ к выделенному элементу папки References
    ReferenceNode node = Owner as ReferenceNode;

    if (node != null)
    {
        // получаем доступ к объекту, который представляет саму ссылку
        ReferenceProjectItem item = node.ReferenceProjectItem;

        if (item != null) {
            string fileName = item.FileName;

            MessageBox.Show(fileName);
        }
    }
}
}
}
}
}

```

Пока что команда не делает ничего полезного. Мы лишь получили доступ к сборке, по которой кликнули правой кнопкой мыши и узнали ее имя файла. Для запуска расширения нужно немного подредактировать свойства проекта MolesAddIn. Первое, что нужно сделать – это изменить путь, по которому будет скопировано готовое расширение, чтобы SharpDevelop его автоматически запускал. Для этого есть специальная папка AddIns, создадим в ней вложенную папку с именем нашего расширения.

Таким образом Output Path нужно установить в /path/to/sharpdevelop-source/AddIns/MolesAddIn (рис. 2).

Чтобы запускать debug-версию SharpDevelop при нажатии **F5**, нужно в свойствах проекта изменить способ запуска (рис. 3).

Теперь все готово. **Ctrl+F5** – и у нас уже запускается второй SharpDevelop. Откроем в нем проект MolesAddIn и кликнем правой кнопкой по любой из его ссылок (рис. 4).

Пункт меню **Add Moles Assembly** можно наблюдать сразу после **Properties**. К этому моменту сборка MolesAddIn.dll еще не загружена, в этом можно убедиться, просмотрев лог в консоли. После клика по выбранному пункту меню, откроется MessageBox, а в логе появится строка **Loading AddIn MolesAddIn.dll**. В AddIn Manager (**Tools => AddIn Manager**) можно найти запись о нашем расширении: имя, версия и описание. Также его можно наблюдать в AddIn Scout.

А что же дальше?

Вот так и выглядит разработка расширений для SharpDevelop. После того, как AddIn становится достаточно стабильным, его можно подготавливать для установки в вашу рабочую версию IDE. Делается это просто – все сборки, которые нужны для его работы (в нашем случае одна MolesAddIn.dll) вместе с файлом *.addin пакуются в архив zip. Далее этот архив переименовывается в *.sdaddin. Все, установочный пакет готов. С помощью AddIn Manager его можно установить SharpDevelop.

Разумеется, данное описание системы AddIns далеко не полное. Оно лишь дает общее представление и помогает взглянуть на написание плагинов со стороны разработчика.

Более полное описание можно найти в книге Dissecting a C# Application: Inside SharpDevelop. В цифровом виде ее можно скачать бесплатно. С тех пор, как она появилась на свет, не произошло серьезных изменений в архитектуре, в основном только переименования. Например, Path в книге называется Extension, а Doozer – Codon. Более актуальную информацию можно найти в дистрибутиве SharpDevelop, в папке doc/tech-notes. ■

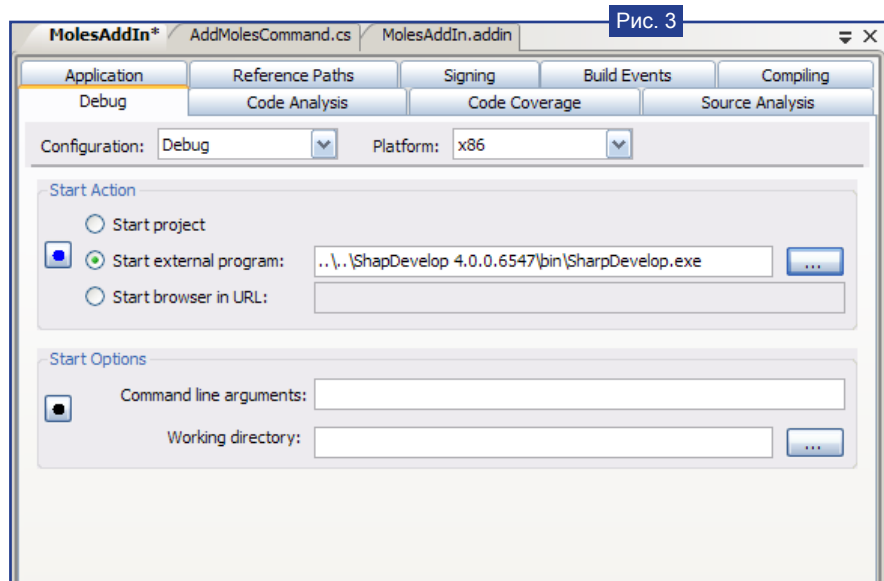


Рис. 3

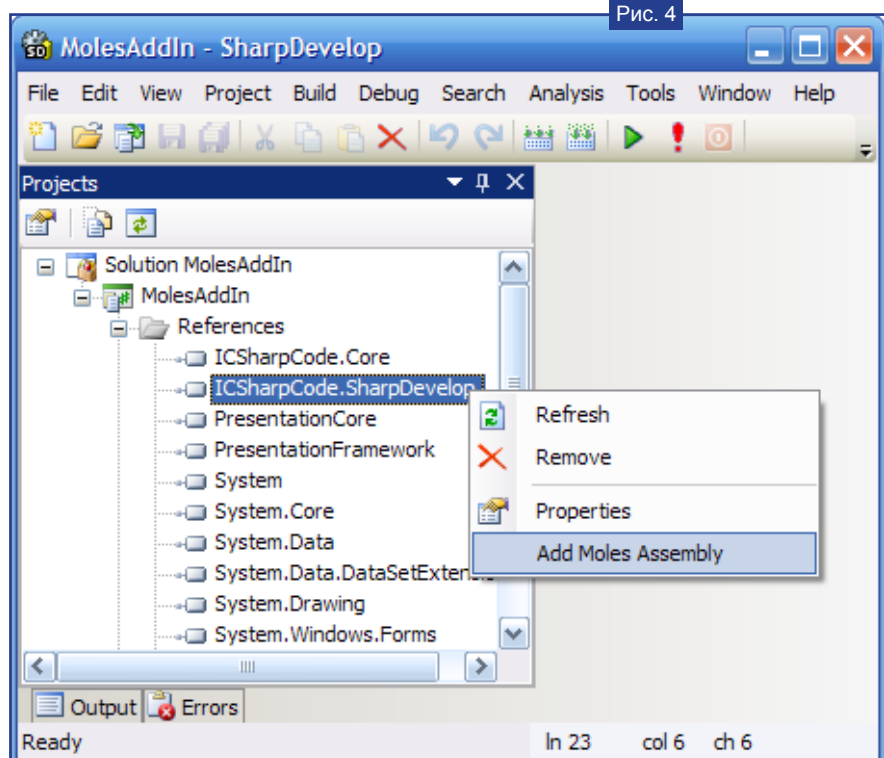


Рис. 4

Автор

Вадим Марковцев
Студент ФУПМ МФТИ
г. Долгопрудный
Email: gmarkhor@gmail.com

7-Zip из .NET

или как я делал open source проект на CodePlex

Ниже пойдет рассказ о моем полуторагодовом опыте разработки библиотеки SevenZipSharp с открытыми исходниками, выложенной на CodePlex в феврале 2009. Эта библиотека – обертка над 7-Zip, позволяющая с легкостью применять его в .NET.

Использование SevenZipSharp

В библиотеке два главных класса – SevenZipExtractor и SevenZipCompressor. Шаблон использования первого:

```
// Синхронная распаковка

using (var extr=new SevenZipExtractor(@"путь\к\
архиву"))
{
    extr.Extracting += DoExtractingEvent();
    extr.ExtractArchive(@"куда\распаковы-
вать");
    DoFinishEvent();
}
```

```
// Асинхронная распаковка

var extr = new SevenZipExtractor(@"путь\к\ар-
хиву");

extr.Extracting += DoExtractingEvent();

extr.ExtractionFinished += (s, e) => { DoFi-
nishEvent(); extr.Dispose(); extr = null; };

extr.BeginExtractArchive(@"куда\распаковы-
вать");
```

Шаблон использования второго:

```
// Синхронная упаковка

var cmpr = new SevenZipCompressor();

cmpr.CompressDirectory(@"путь\к\пакуемой\
папке", @"имя\архива");

DoFinishEvent();

cmpr = null;

// Асинхронная упаковка

var cmpr = new SevenZipCompressor();

cmpr.CompressionFinished += (s, e) => { DoFi-
nishEvent(); cmpr = null; };

cmpr.BeginCompressDirectory(@"путь\к\пакуе-
мой\папке", @"имя\архива");
```

Не хочу превращать статью в документацию по SevenZipSharp, так что просто перечислю ее некоторые возможности:

- Полная поддержка многопоточности.
- Оборачиваются все форматы архивов, поддерживаемые 7-Zip-ом.
- Неприхотливость к оборачиваемой библиотеке – никаких

жестких привязок.

- Поддержка многотомных архивов при распаковке и упаковке.
- Распаковка большинства SFX архивов.
- Весь код тщательно снабжен комментариями

Как все началось

В феврале 2009 года мне понадобилось работать с 7-Zip архивами в одном из платных проектов, над которым я трудился. Несколько дней я безуспешно искал готовое удобоваримое решение, но только прочитал много сообщений, где люди жаловались на его отсутствие. И тогда, собрав волю в кулак, я мужественно принялся писать свою реализацию обертки над 7-Zip. Я решил выложить код на недавно открывшемся CodePlex под лицензией LGPLv3. Поначалу работа кипела, и я выпускал релиз за релизом раз в несколько дней, потом мой пыл несколько поугас и я стал стабилизировать код. В сентябре 2009 релизы перестали выходить часто, и с тех пор поддерживаю проект как могу.

Рассматривался вариант с компиляцией 7-Zip в смешанную сборку (mixed assembly) с флагом /clr. Этот вариант был отвергнут, т. к. во-первых, интерфейс получился бы низкоуровневый и не пригодный для быстрого использования и все равно пришлось писать «добавку», а во-вторых, чтобы собрать код с флагом /clr:pure, требовалось бы переписывать много кода, и unmanaged части все равно остались.

Когда SevenZipSharp только появился, мне хотелось рассказать о нем потенциально заинтересованным пользователям-разработчикам. Я оставлял краткое описание библиотеки везде, где было можно: в ответах на вопросы StackOverflow, на программистских форумах (в т. ч. MSDN, Channel 9), и даже на английской Wikipedia. Это все принесло результаты, и вскоре поисковые выдачи Google вышли по трафику на первое место. Думаю, рекламировать свои проекты должны все, иначе большинство попросту не узнает об их существовании. Эффективность рекламы оценивается по статистике загрузок и посещений, которая публично доступна.

SevenZipSharp и 7-Zip

Как многие знают, 7-Zip написан на C++ с небольшим количеством C и пресловутой ассемблерной функцией, вычисляющей CRC32, которую Игорь Павлов реализовал для x86, x86-64 и ARM. Какая либо документация по коду отсутствует, что в стиле русских программистов, участвующих в open source движении. Кода много, он совсем не прост и требуется некоторое время, чтобы разобраться в изобилии define-ов, интерфейсов и классов. Реализации алгоритмов сжатия называются кодеками (Codecs). Кодеки стандартным способом встраиваются в библиотеку, как подключаемые модули протоколов в мессенджерах ala Miranda/Pidgin. Архитектура 7-Zip неотделима от COM; именно это препятствует развитию p7zip - 7-Zip для POSIX систем, которым также занимается Игорь Павлов. В p7zip, COM заменен костылем, который симулирует его работу, попутно объявляя половину типов windows.h. Сами алгоритмы написаны безупречно и очень стабильны, однако верхние уровни, как вы уже догадываетесь, оставляют желать лучшего. Если бы автор начал писать 7-Zip сейчас, думаю, он

придумал архитектуру ядра более понятную, универсальную и портируемую. В идеале – на языке вроде C# или Java, хоть даже на Python (ну не годятся для этих целей плюсы, чего уж там).

Кстати, 7-Zip для конечных пользователей (инсталляция, которая качается с 7-zip.org) собирается Visual Studio 6 образца начала века. Файлы решений успешно преобразовываются в форматы VS2008/2010, и после замены компилятора C/C++ на более новый, активации всех флагов оптимизации, а также использования профиля достигается ускорение около 15% (LZMA/LZMA2).

Вот как SevenZipSharp оборачивает 7-Zip. Через COM-овский CreateObject создается объект, поддерживающий указанный интерфейс (IInArchive, IOutArchive). Из этого объекта дергаются нужные функции, и достигается желаемый результат (например, IInArchive.Extract(...)). Во время длительных операций из unmanaged кода вызываются managed callback-и, и это приводит к проблеме, которую я не сразу осознал – обработка ошибок. Например, из-за ошибки в callback-е или исключении в вызываемом callback-ом пользовательском событии исполнение операции падает без предупреждений и какой-либо вразумительной информации, кроме странного 32-битного кода ошибки. Я решил обернуть все callback-и try/catch-ами и заносить все возникшие исключения в стек ошибок, который в случае неудачи показывается пользователю.

Попытки переписать весь код 7-Zip на C# энтузиастами предпринимаются регулярно, но ни одна не ушла дальше обсуждений. Переделывать алгоритмы с C++ на C# не выгодно: затраченные усилия и падение в скорости не окупаются кроссплатформенностью и религией, а переписать ядро с учетом всех тонкостей может только сам Игорь Павлов. Не буду голословен: LZMA на C#.NET из LZMA SDK по замерам работает в 4 раза медленнее unmanaged алгоритма. Поэтому, пожалуй, лучшим в такой ситуации было сделать обертку с понятным и простым интерфейсом.

В один прекрасный момент мне захотелось заставить работать SevenZipSharp под Mono (GNU/Linux). И тогда проблема привязанности 7-Zip к COM проявила себя во всем великолепии. Необходимо было заново написать низкоуровневую часть библиотеки почти с нуля. Т. к. код 7-Zip, как я уже писал, специфичный, инструменты для автоматической обертки вроде SWIG оказались бесполезными, а чтобы они вообще заработали, мне пришлось сначала пройти по всему коду препроцессором и убрать 10-ти этажные define-ы. В настоящее время я потихоньку пишу независимую от COM обертку.

Разработка

Наверное, многие начинающие C# разработчики повторяют одни и те же ошибки, и я не являюсь исключением. Когда мне стало известно о FxCop и StyleCop, я сразу попробовал их использовать. Казалось бы, логично поддерживать код библиотеки в хорошем состоянии. Однако StyleCop по умолчанию выдает слишком много предупреждений, а настраивать его мне не хотелось, и он был сразу отброшен. FxCop использовался некоторое время, но в один прекрасный момент я поймал себя на мысли, что на элементарные изменения кода я трачу слишком много времени, чтобы соответствовать правилам, которые, по сути ни на что не влияют. Вывод, который я для себя сделал – эти инструменты важны для выработки индивидуального стиля программирования, но разработчикам-одиночкам увлекаться ими не стоит.

Изначально SevenZipSharp писался на Visual Studio 2008 и работал под 2-ым фреймворком. Даже тогда я понимал, что чем меньше версия .NET, тем меньше проблем будет использовать библиотеку. Жаль, что многие разработчики на CodePlex этого не понимают, начинают писать код, используя ультрасовременные возможности .NET 4, и потом удивляются, почему у них так мало загрузок. Потом я узнал, что в Windows Mobile < 7 есть полноценный COM, и за несколько дней портировал SevenZipSharp на эти мобильные системы. Если кто-то не знает, фреймворки обычные и контраст имеют ряд различий, и код без небольших изменений не будет соби-

раться, а тем более работать. Поддерживать две почти идентичные ветки я считал неразумным, и решил эту проблему множественными #if/#else/#endif (стандартный подход в исходниках на C++).

Когда вышел Visual Studio 2010/C# 4, я обнаружил, что фичи новой версии языка можно эффективно применить к коду (например, появившиеся опциональные параметры устраняют 10+ перегрузок единственного логического метода). Для сохранения обратной совместимости я снова применил #if/#else/#endif. Код понемногу стал превращаться из изящных классов в ветвистого монстра. А когда пришла идея портировать SevenZipSharp на Mono, некоторые файлы с кодом я все-таки раздвоил, т. к. иначе сам бы через пару недель не смог в нем разобраться. В итоге, передо мной во всей красе встала проблема поддержки разных платформ и фреймворков в одном единственном файле. Пример:

```
#if !DOTNET20

/// <summary>

/// Unpacks the whole archive asynchronously to
the specified directory name at the specified priority.

/// </summary>

/// <param name="directory">The directory where
the files are to be unpacked.</param>

/// <param name="eventPriority">The priority of
events, relative to the other pending operations in the
System.Windows.Threading.Dispatcher event queue, the
specified method is invoked.</param>

#else

/// <summary>

/// Unpacks the whole archive asynchronously to
the specified directory name at the specified priority.

/// </summary>

/// <param name="directory">The directory where
the files are to be unpacked.</param>

#endif

public void BeginExtractArchive(string directory)

#if !DOTNET20

, DispatcherPriority eventPriority

#if CS4

= DispatcherPriority.Normal

#endif

#endif

)

{

SaveContext (

#if !DOTNET20

eventPriority

#endif

);

(new ExtractArchiveDelegate(ExtractArchive)
.BeginInvoke(directory, AsyncCallbackImplementation, this);

}
```

Отмечу, что разрабатывался SevenZipSharp стихийно. Если что-то хотелось добавить к функциональности, я брал и добавлял, и не советовался с начальниками, не согласовывал решения с руководством и т. д. В этом есть свои минусы, но зато баги исправлялись мгновенно и просьбы о новых функциях удовлетворялись в течение нескольких дней. Полная свобода действий и настоящая учеба на своих ошибках. ■

Каталог Веб-студий



Веб-студия ИТКЕМ

Страна: Россия
Город: Кемерово
Год основания: 2010
Сотрудников: 5
Сайт: <http://itkem.ru>



Imagz.ru

Страна: Россия
Город: Санкт-Петербург
Год основания: 2009
Сотрудников: 4
Сайт: imagz.ru



KamIT

Страна: Россия
Город: Каменск-Уральский
Год основания: 2008
Сотрудников: 2
Сайт: <http://kamit.net.ru>



Dart's

Страна: Россия
Город: Краснодар
Год основания: Краснодар
Сотрудников: 5
Сайт: www.studio-dart.ru



Shocos Design Team

Страна: Украина
Город: Николаев
Год основания: 2009
Сотрудников: 2
Сайт: www.shocos.com



Сайт Плюс

Страна: Россия
Город: Хабаровск
Год основания: 2007
Сотрудников: 5
Сайт: <http://site-plus.ru>



Delmar

Страна: Россия
Город: Москва
Год основания: 2006
Сотрудников: 15
Сайт: www.delmar.ru



Sandal

Страна: Россия
Город: Владивосток
Год основания: 2004
Сотрудников: 5
Сайт: www.sandal.ru



ArtMachine

Страна: Украина
Город: Николаев
Год основания: 2005
Сотрудников: 7
Сайт: www.artmachine.com.ua



Угол зрения

Страна: Россия
Город: Новосибирск
Год основания: 2008
Сотрудников: 8-10
Сайт: www.ugolzreniya.ru



CrazyOne

Страна: Украина
Город: Кривой Рог
Год основания: 2010
Сотрудников: 5
Сайт: <http://crazyone.net/>



SMSdesign

Страна: Украина
Город: Киев
Год основания: 2008
Сотрудников: 7
Сайт: <http://SMSdesign.com.ua>

Добавить веб-студию в каталог журнала бесплатно



ДОБАВИТЬ

Публикуемые в журнале компании взяты из каталога сайта издания. Данные о компаниях публикуются по мере их добавления в каталог. Добавить в каталог данные о компании может любая организация. Добавление данных в каталог сайта осуществляется бесплатно на добровольной основе самой организацией. Редакция журнала Веб-Аналитик.ИНФО не занимается добавлением данных в каталог сайта и не несет ответственности за предоставляемую информацию от организаций.



ИНТЕРНЕТ

**Интервью с
с Юрием Азовцевым
[Координатор Нижегородской
Группы Пользователей Linux
(NNLUG)]**

**Как заработать на поддержке
интернет-проектов?**

Где найти покупателей в Рунете?

**Ложь, большая ложь,
и антивирусы**



Интервью с Юрием Азовцевым

Координатор Нижегородской Группы Пользователей Linux (NNLUG)

Беседовал: Алексей Шаферов

N NNLUG это некоммерческая организация, которая занимается различными проектами так или иначе связанными с СПО. В настоящее время группа активно занимается внедрением СПО по школам Нижнего Новгорода и области.

Алексей Шаферов: Здравствуйте, Юрий! Рад нашей встрече на страницах журнала Веб-Аналитик.ИНФО. Юрий, расскажите, пожалуйста, немного о своем знакомстве с Linux и о том, как вы узнали об NNLUG.

Юрий Азовцев: День добрый, Алексей. Мне 26 лет, Linux я занимаюсь около 8 лет. Первое знакомство с Linux практически совпало с присоединением к NNLUG.

Все началось с создания домашней компьютерной сети. Году в 2001 у нас с соседом и еще несколькими ребятами была брошена небольшая локальная сеть между соседними домами. Один из домов находился на некотором удалении и для его подключения требовался промежуточный компьютер с несколькими сетевыми картами, компьютер соединял между собой сети разных типов. Для этой цели у нас стоял старенький системный блок с Windows 98, у которого имелась проблема с IP адресами. Дело в том, что каждый раз после перезагрузки компьютера мне приходилось дергать и перевтыкать провода чтобы определить какой IP на какое соединение «сел» на этот раз. Однажды мне это порядком поднадоело и я решил попробовать настроить такую же систему но с использованием Linux. На радиорынке удалось найти только RedHat на трех CD. При этом у ребят с рынка были только два вторых и третий диск. Первый установочный CD отсут-

ствовал.

Поспрашивав своих сокурсников в институте, я узнал что у нас в ННГАСУ есть интернет клуб и там работает парень знающий Linux. Так я познакомился с одним из участников NNLUG – Сергеем Смирновым (aka Sergous). Он мне рассказал когда и где происходят встречи, и я решил отправиться на одну из них.

Дело было поздней осенью, встреча проходила на площади Горького. На встрече было трое основателей NNLUG от которых я получил дистрибутив SlackWare. Около двух недель я проводился с дистрибутивом, ожидая увидеть после установки привычный графический интерфейс. Но, как позже узнал, на том старом компьютере были какие-то проблемы с видеокартой. За эти две недели я успел прикупить и прочесть книжку по SlackWare. Собственно, с этого и началось мое знакомство с Linux, так сказать, сразу же «на практике». Linux обеспечивал бесперебойную работу местной домашней сети, соединяя разные ее части между собой, и впоследствии, предоставляя доступ к файлам, фильмам и музыке размещенным на ней.

А. Ш.: Интересная история, вспомнилось как сам начинал с Red Hat 7. Юрий, расскажите нашим читателям о создании группы, ее деятельности и развитии.

Ю. А.: Группа появилась примерно за пол года до моего появления в ней. Основателями NNLUG были Звонилов Михаил, Сидоров Александр и Губанов Дмитрий. В первые несколько лет существования группы мы занимались изучением Linux. Практически у каждого участника NNLUG в то время была не-

большая локальная сеть. И на встречах мы не теряли возможности проконсультироваться по настройкам тех или иных вещей у Михаила Звонилина и других основателей Нижегородской LUG.

Года через полтора мы узнали, что в Нижегородском Политехническом Институте проходят свободные семинары по Linux. Как представители NNLUG мы не могли пропустить такое событие, и посетили второй семинар. Там познакомился с лектором – Самсоновым Евгением (aka SabutR), который во взаимодействии со СтудСоветом Нижегородского Политехнического Университета организовал и проводил курсы по Linux. Мы предложили свою помощь, и дальнейшие курсы уже велись с участием NNLUG. Так же Евгений стал одним из активных участников нашей группы.

За восемь лет существования группы наша деятельность многократно менялась. Мы осуществили и продолжаем большое количество проектов, среди которых хотелось бы отметить:

- Перевод на русский язык фильма RevolutionOS (2006 г.). ▶



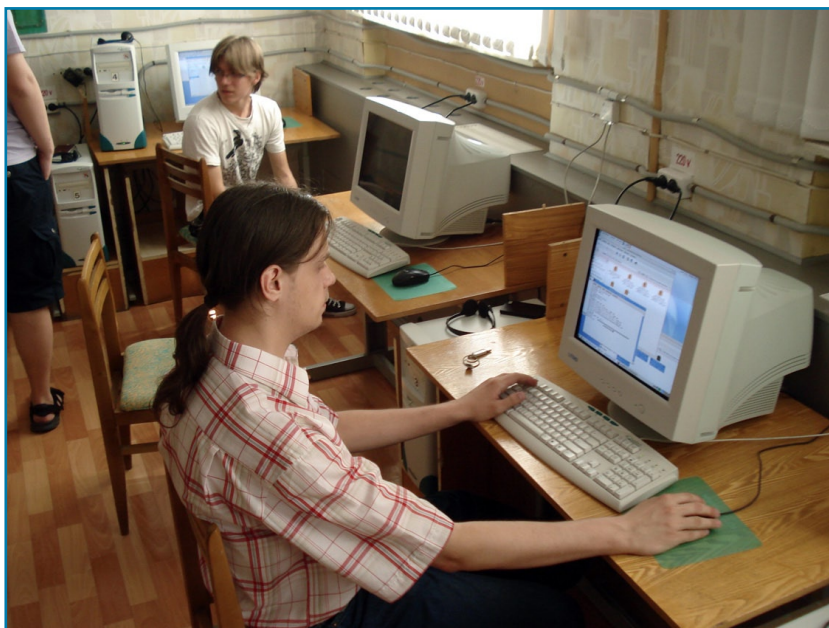
- Создание репозитория Ubuntu для пользователей нижегородского интернет кольца (2008 г.).
- Открытые семинары по Linux в Политехническом Университете, ставшие уже ежегодной традицией (2004-2010 года).
- Linux InstallFest проводимый на базе Нижегородского Радиотехнического Колледжа, на котором все желающие могут получить помощь в установке и настройке Linux (с 2007 г.).
- Перевод Нижегородского Радиотехнического Колледжа (НРТК) на СПО (с 2007 г. по настоящее время).
- Установка СПО в школах Нижнего Новгорода и области на безвозмездной основе (2010 г.).

А. Ш.: *Насколько я знаю, с Нижегородским Радиотехническим Колледжем у вас связано много проектов, самым крупным из которых является перевод колледжа на СПО. Пожалуйста, расскажите подробнее об этом процессе. С чего он начался?*

Ю. А.: Одним из самых крупных и долгих проектов, который длится и по сей день является перевод на Linux и СПО, и поддержка информационной системы НРТК. Проект начался в 2007 году, когда после «дела Поновоса» возник вопрос о лицензионной чистоте используемого ПО. Посчитав стоимость покупки лицензий на все ПО используемое на тот момент в колледже, директором колледжа было принято решение о начале перехода на СПО. Участники NNLUG в сотрудничестве с работниками ВЦ НРТК начали процесс по переводу инфраструктуры колледжа на Linux. Основные технические работы были выполнены за 4 месяца. Дальше началось обучение сотрудников и преподавателей, а также, поиск и изучение свободных аналогов программного обеспечения, использовавшегося ранее.

А. Ш.: *Юрий, как вы сказали, основным на данный момент является проект по установке СПО по школам города и области. Почему вы решили заняться этим? Как начинался проект?*

Ю. А.: Одной из задач, сформулированных в манифесте о группах пользователей Linux значится пункт «продвижение Linux и информирование пользователей». Если уж мы назвали NNLUG, то принимаем на себя некоторую ответственность и обязательства. Сообщество может существовать лишь при наличии реальной работы. С другой стороны у школ назрела проблема с лицензированием ПО. Мы поднимали этот вопрос и раньше, общались с



Установка в одной из школ

разными школами, но ответ был примерно одинаковым: «нет руководящих документов из мин. образования для использования Linux в учебном процессе». Два с половиной года назад по заказу мин. образования для школ был создан Пакет Свободного Программного Обеспечения, который с технической точки зрения является одним из дистрибутивов Linux. Инициатива и поддержка со стороны министерства открыло дорогу Linux в школы. Но тут как всегда в дело вмешалась наша действительность. Дистрибутив сделали, методический материал тоже, а вот кто все это будет внедрять, не подумали. Здесь NNLUG самая незаменимая структура. С одной стороны объединяющая заинтересованных и компетентных людей, с другой – имеющая связи и имя в академической среде.

Официально проект начался в конце мая месяца с предложения студентам, посещавшим Открытые Семинары «HowTo Linux 2010» принять участие в краткосрочном обучении по администрированию Школьного Linux.

Семинары по администрированию Школьного Linux начались с июня месяца в стенах НРТК. И первоначально на них приходило от 20 до 30 человек. Примерно через месяц осталось около 12 заинтересованных участников, большая часть из которых и вошла в основную команду проекта.

А. Ш.: *А какое состояние у данного проекта на текущий момент?*

Ю. А.: За три летних месяца мы выполнили установку комплексного решения в 20 общеобразовательных школах Нижнего Новгорода. В Павлово, на Бору и еще в нескольких городах области, школы так же получили воз-

можность плавно перевести учебный процесс на Школьный Linux в течение ближайшего полугодия. В августе месяце проект перешел на второй виток. Директор компании ООО «ЭЛСИС» и одновременно участник NNLUG Сергей Бессонов провел переговоры в Управлении Образованиием Города Дзержинск и рассказал им о нашем успехе в Нижнем Новгороде. Сотрудники администрации очень заинтересовались нашим опытом, и впоследствии было принято решение сделать внедрение Школьного Linux по всем школам города Дзержинска с дальнейшим обучением преподавателей. В течение двух недель при активной поддержке управления образованием города Дзержинска, участниками NNLUG был установлен Школьный Linux во всех 48 школах города Дзержинска.

Официально Школьный Проект NNLUG закончился 1 сентября. Но, не смотря на это, мы до сих пор получаем заявки от различных школ города и области.

А. Ш.: *Каким образом вы находите школы для внедрения?*

Ю. А.: Поначалу это были школы, в которых у кого-то из участников NNLUG работали родственники или знакомые, или же они сами там учились и у них остался контакт с их преподавателем информатики. Затем о нас рассказали сотрудники одной из школ, где мы уже были на заседании в НИРО и нам сами стали звонить.

Были попытки делать обходы и обзвоны школ с предложением об установке Школьного Linux. Но этот способ оказался не эффективным. Постоянно возникали вопросы «Кто вы? Откуда вы?» и мы наткнулись на стену непонима-

ния. Мы решили отказаться от «убеждения не желающих» и работать только с теми кто «хочет сам», а таких школ оказалось очень много.

А. Ш.: *Какое отношение вы встретили со стороны сотрудников и преподавателей школ?*

Ю. А.: Весь проект проходил летом и в школе мы встречались только с преподавателем информатики или директором. Другие сотрудники находились в отпусках, но те, кто присутствовал, принимали нас очень радушно. Все-таки IT-специалисты, готовые помочь решить технические вопросы с компьютерами в классе информатики появляются не часто. А участники>NNLUG по ходу установки умудрялись починить уже по полгода неработающие системные блоки, мышки, клавиатуры и прочую периферию.

А. Ш.: *Скажите, а почему вы выбрали именно Alt Linux, а не какой либо другой дистрибутив?*

Ю. А.: Мы рассмотрели различные дистрибутивы. Нужно сказать, что Нижегородский РадиоТехнический Колледж на протяжении уже двух лет работает на дистрибутиве Ubuntu, и Ubuntu нам был несколько ближе. Но посмотрев Школьный Linux мы нашли в дистрибутиве много интересного и полезного именно для использования в школьном классе информатики.

Во-первых, это была система управления классом iTalc. Она уже включена в дистрибутив и устанавливается вместе со всей системой.

Во-вторых, после выяснения техни-

ческой возможности установки Контент-Фильтра выяснилось, что «рекомендованный в обязательном порядке» для всех школ контент-фильтр NetPolice, существует только в варианте для Школьного Linux. В теории его можно установить и на Ubuntu, но это потребует дополнительных трудозатрат.

И в-третьих, все методические материалы и разработки сделанные в ходе конкурса «код свободы» и выложенные на сайте www.armd.ru были сделаны по Школьному Linux. Все снимки и иллюстрации демонстрируют именно его интерфейс.

По большому счету дистрибутив не так важен, важно чтобы разработчик поддерживал его. На первых порах преподавателям придется осваивать много нового, и желательно облегчить им этот процесс. Пусть сначала у них будет унифицированное решение на базе которого они смогут обмениваться опытом и использовать знания, полученные на наших семинарах. После того, как они освоятся, дистрибутив роли играть не будет, и они сами смогут выбрать, что им больше нравится.

А. Ш.: *Вы говорили, что компания «ЭЛСИС» помогла вам договориться с администрацией города Дзержинска. А кроме этого она вам помогает чем-то еще? Может быть вас поддерживают другие компании?*

Ю. А.: Школьный проект поддерживали несколько нижегородских IT-компаний. Основную поддержку мы получили от ООО «ЭЛСИС» (помощь в организации, информационная поддержка, хостинг для сайта проекта, контакты, связи) и «ГНУ/Линуксцентр-нн» (оборудование, контакты, связи).

Особую благодарность хотелось бы



выразить Баевскому Юрию Евгениевичу. При начале внедрений в Дзержинске нам понадобилось оборудование, которое он предоставил на безвозмездной основе в кратчайшие сроки.

А. Ш.: *Вы провели достаточно масштабное внедрение СПО, сейчас занимаетесь поддержкой. Расскажите о технических и организационных трудностях, которые возникли при осуществлении такого сложного проекта.*

Ю. А.: Основной технической трудностью была необходимость сохранить уже имеющуюся ОС на компьютерах. Windows очень капризная операционная система. Часто не удавалось изменить размер имеющегося раздела в виду некорректного завершения работы системы. Иногда нам приходилось упаковывать систему целиком в архив и, изменив всю геометрию диска разворачивать обратно. В основном эти процессы и съедали до 70% рабочего времени.

Еще к техническим трудностям я бы отнес наличие не очень хорошей периферийной техники. Например, принтеров от Canon. Производитель хоть и выпускает драйвера, но принтер работает из рук вон плохо. Однако в Дзержинске нам в этом плане повезло, и проблемной техники было не более пятнадцати процентов, а Управление Образовани-ем г. Дзержинск приняло решение просто заменить проблемную периферию.

К организационным трудностям я бы отнес два момента. Во-первых, это накладные расходы. Хоть проект и был волонтерским, мы все-таки несли накладные расходы на проезд (питание нам обеспечивали на месте). Это в некоторой степени обращение к тем, кто в дальнейшем будет работать с LUG'ами. Не забывайте, что ребята едут к вам и везут как минимум набор оборудования для работы. Это стоит денег и желательно это компенсировать, даже если они об этом не просят.

Во-вторых, иногда возникали вопросы «Зачем Linux?». Такие вопросы решались благодаря поддержке Управления Образовани-ем. Нам было гораздо проще отвечать на такие вопросы, ▶

Небольшие посиделки после завершения работ по Дзержинску





когда инициатива исходила от управляющей структуры.

А. Ш.: Вы говорите, что иногда возникали вопросы «Зачем Linux?». А как в целом вы могли бы охарактеризовать отношение преподавателей к СПО?

Ю. А.: По большей части «ровное». Преподаватели просто не знают, что это и не знают, как к этому относиться. Часть из них чувствует, что СПО – это для них дополнительная нагрузка за те же деньги и выражают некоторый негатив, но не из-за самих программ. Как нам сказал один из директоров школ «Мы готовы работать и на Linux, но дайте нам планы и методические материалы по нему». Если бы методические материалы были разработаны системно для всех лет обучения, то все бы просто полюбили Linux.

Не часто, но нам встречались по-настоящему заинтересованные преподаватели. Было очень приятно. Они обычно говорят «Linux уже появился, и школьники скоро будут его знать. Не могу же я знать меньше ученика!». Мы надеемся улучшить такую ситуацию с помощью проводимых нами курсов.

А. Ш.: Расскажите, пожалуйста, подробнее что это за курсы и как они проходят.

Ю. А.: В начале сентября совместно с НИРО мы проводили курсы повышения квалификации для 8 школ из Совета по Пилотному Внедрению ПСПО. Это был 36 часовый курс, включивший в себя все основные моменты использования Linux в школе. В течение недели мы прошли по офисному пакету OpenOffice.org, освоили основные приемы обработки цифровых фотографий в GIMP, поработали с векторной графи-

кой в InkScape, получили навыки обработки и монтажа видео в KDenlive и аудио в Audacity. И конечно, провели обзор программного обеспечения для преподавания астрономии (Celestia, Stellarium), физики (Step), химии (Kalzium), алгебры и геометрии (KAlgebra, Kig). В этом курсе мы не касались языков программирования.

В рамках этого курса 7 сентября проходила конференция для большинства школ Нижнего Новгорода. На этой конференции мы получили возможность поделиться с преподавателями накопленным опытом по использованию Linux в учебном процессе, и посетили с ними «Стажерскую Площадку» в Нижегородском РадиоТехническом Колледже, где они смогли вживую посмотреть на учебное заведение, которое работает на Linux уже третий год.

Так же одной из интересных тем на этой конференции было выступление Сергея Бессонова «Лицензирование программного обеспечения в Российской Федерации», где он дал обзор законодательства, касающегося лицензирования ПО, и рассмотрел условия свободных лицензий и различных лицензий от Microsoft. Видео выступления Сергея вы можете найти на сайте НИРО. Сейчас мы ведем 72 часовый курс для преподавателей школ города Дзержинск. Курс более расширенный, чем был в НИРО и рассчитан на 9 недель по одному дню в неделю. По опыту ведения курсов для преподавателей могу сказать, что многие проблемы с преподаванием информатики в школах, о которых говорят на форумах, связаны

больше с недостатком информации. Преподаватели в действительности хотят научиться и работе в различных системах, и редактированию фотографий, и обработке видео, но им некому об этом рассказать и научить.

А. Ш.: Какие дальнейшие планы по внедрению и поддержке? Будут ли платные услуги?

Ю. А.: Пока у участников>NNLUG будет желание и время, мы продолжим делать внедрения. К новому году большинство школ в Нижнем Новгороде окажутся в ситуации как в русской поговорке «пока петух не клюнет...», поэтому хотелось бы до этого не доводить. Платная поддержка? Да, пожалуйста.>NNLUG некоммерческая организация, поэтому техподдержку мы предлагаем от имени ООО «ЭЛСИС». Участники>NNLUG получают финансирование своей деятельности. Предложения по техподдержке для школ можно найти на сайте el-sys.org. Они очень демократичны и мы постарались сделать так, чтобы школы получали максимум сервиса за приемлемую сумму денег.

А. Ш.: Юрий, большое спасибо за интересную беседу. Желаю вашему проекту удачи и дальнейшего развития.

Ю. А.: Вам спасибо, Алексей. ■

Ссылки:

<http://www.lug.nnov.ru/> –>NNLUG.

<http://blog.freeschool.nnov.ru/> – блог проекта по установке СПО в школах.

<http://freeschool.nnov.ru/> – сайт проекта по установке СПО в школах.

<http://nnntc.nnov.ru/> – сайт НРТК.

Как заработать на поддержке интернет-проектов?

Автор

Василий Чуранов
Директор по развитию
ООО «Твинс»
www.twinscom.ru
г. Смоленск
E-mail: vs@twinscom.ru

Мы научились зарабатывать на услуге поддержки интернет-проектов. Оказывается это возможно. Несколько лет жизнь убеждала меня в обратном, и я почти с ней согласился, но... Выход есть.

Что не работает?



Абонентская плата

Казалось бы, самый благоприятный для студий вариант поддержки — это абонентская плата со стороны клиента и фиксированный набор услуг со стороны студий. Все кажется простым и прибыльным, пока не попробуешь. Мы начинаем оказывать обещанный набор услуг, но через какое то время понимаем, что забыли заложить в стоимость поддержки — затраты на консультации, документооборот, споры с бухгалтерией клиента, обучение новых сотрудников клиента, ночные звонки, свою ответственность в конце концов... В итоге поддержка даже одного клиента может вывести из равновесия всю студию. Высокую абонентскую плату закладывать бесполезно. Большинство клиентов не хотят платить просто так (по их мнению) и предлагают вариант оплаты за каждую доработку отдельно.



Решение отдельных задач

Не менее сложный вариант, когда клиент оплачивает каждую доработку отдельно. Как правило, решение отдельных задач становится нерентабельным уже на этапе обсуждения стоимости с клиентом. Опытный менеджер студии знает, что при таком подходе даже мелочь, которая делается 5 минут, требует участия как минимум 3-х человек в компании (менеджер, программист, бухгалтер). В результате споры с клиентом о сумме счета и недовольные две стороны.

Мой опыт показывает, что перечисленные способы поддержки клиентов накладывают на студию высокие риски уйти в минус, если держать цены, на которые будет соглашаться клиент.

Как же быть? Отказываться от поддержки клиентов тоже было бы глупо. Ведь самая дорогостоящая операция — заполнение клиента уже пройдена, осталось только работать и зарабатывать.

Решение

Решение, в котором мы нашли выход из ситуации — это **почасовая работа**. Эта схема подразумевает оплату работы по факту ее выполнения, в соответствии с отработанным разработчиками временем. В стоимость часа заложены накладные расходы студии и ЗП разработчиков. Мы применяем такую схему уже более 1,5 лет и, что самое интересное, мы перестали ругаться с клиентами, а они с нами. Мы начали зарабатывать на поддержке!

В чем основные плюсы для студии:



Минимальные риски

Все отработанные часы будут оплачены клиентом. Конечно, клиент не будет платить за очевидные баги. Оплату багов берет на себя сама студия, как гарант качества.



Понятное планирование

Мы знаем объем часов в месяце, соответственно легко подсчитаем возможную прибыль. Она будет конечно меньше (на бумаге), чем по «предоплатной» системе, но зато гораздо стабильней и, в итоге, больше.



Минимум бумажной работы

Клиенту отправляется отчет в конце месяца и счет на оплату отработанного времени. Я помню ужасные времена, когда на каждую задачу готовили счет, акты, отправляли курьером и пр. и так в течение месяца.

В чем минус:



Недоверие клиента

Самое трудное убедить клиента в том, что эта схема наиболее правильная для двух сторон. Для студии, при такой схеме, работать можно спокойно, а значит качественно. Собственно это нужно и клиенту. Клиенту нужно подешевле? Это именно тот вариант. При почасовой схеме у студии нет рисков, а значит, они и не закладываются в стоимость часа. У нас, например, при почасовой оплате стоимость часа почти в 2 раза ниже, чем по предоплате.

При всем этом, клиента пугает два момента,

1) студия будет накручивать часы, полагаясь на некомпетентность клиента,

2) неопределенность бюджета на доработки.

По первому пункту клиент должен просто вам поверить, а вы, в свою очередь, регулярно (в конце дня) отмечать отработанное время. По второму моменту — это не такие уж большие деньги, которые нужно планировать заранее. В любом случае, менеджер может называть клиенту предварительную оценку. Убеждайте клиента. Исключите все остальные варианты оплаты. Самое важное, что бы клиент попробовал эту схему работы один раз, дальше это как наркотик...

Что нужно чтобы почасовая схема заработала? Использовать таск-менеджер и дать клиенту в него доступ. Пусть клиент видит весь процесс разработки, комментарии разработчиков, получает оповещение о смене статуса задач на email и пр. Позвольте клиенту принимать участие в комментировании, тестировании результатов и он больше никогда не захочет возвращаться к предоплатной схеме. Клиент станет частью вашей команды, увидит, что проекты делаются не за 5 минут. Он поймет, насколько это сложный процесс и перестанет беспочвенно торопить, обвинять вас. Вы увидите, как исчезнет момент недоверия уже после первого месяца работы.

Конечно, эта схема эффективна не для всех проектов. Ее нужно применять по крупным проектам со стабильным пулом доработок, либо с клиентами у которых много небольших сайтов, отданных студии на аутсорсинг. В своей работе для постановки задач и контроля отработанного времени мы используем багзилу. Она позволяет делать сводку времени по отработанным часам за месяц с указанием проектов и разработчиков. Частично процесс работы с багзиллой описан на нашем [caйme](#). Для клиентов мы сделали отдельный документ, по которому они знакомятся с системой.

Сколько можно заработать? Сколько — каждый решает сам, регулируя стоимость часа. Тут ключевое слово «заработать». Если это направление не будет убыточным, то это уже большое достижение.

Надеюсь, что мои заметки помогут студиям зарабатывать на поддержке, а клиентам обрести стабильных партнеров и никогда с ними не ругаться. ■

Автор

Ксения Бобкова
Агентство интернет-
рекламы E-PROMO
www.e-promo.ru
e@e-promo.ru

Где найти покупателей в Рунете?

Вывав до капли весь одноцентовый трафик из Яндекс.Директа и Google Adwords, мы время от времени возвращаемся к вопросу – где взять еще покупателей. Особенно эта проблема мучает нас перед праздниками, потому что именно в это время шансы на превращение посетителя нашего сервиса (голосовые поздравления и розыгрыши www.voicescards.ru) в клиента существенно возрастают.

В этой статье мы расскажем, какие рекламные площадки мы нашли для усиления своих рекламных кампаний в праздники, что делали и что из этого вышло. Сообщу наши вводные по всем рекламным кампаниям: получить максимальный объем трафика по минимальной цене, желательная CPC до 1 руб. Добиться максимальной конверсии в продаже.

В качестве критериев для сравнения мы используем:

- объем трафика при минимальном CPC:
 - + – мало = менее 30 000 пользователей в месяц;
 - ++ – норма = от 30 000 до 100 000 пользователей в месяц;
 - +++ – отлично = более 100 000 пользователей в месяц;
- CPC – мы всегда стремились к минимально возможной;
- конверсия в продаже – укажу максимальную, которой мы добились. Понятно, что это были праздничные эксперименты, поэтому в обычные дни она может быть ниже.

Все рекламные площадки мы разделили на группы на основе удобства интерфейса для рекламодателя, возможностей системы и итоговой эффективности рекламы для нас.

Баннерные сети

До этого эксперимента мне казалось, что баннерных сетей с недорогим развлекательным трафиком огромное количество. Но на деле вышло, что половина из них не работает (или менеджеры просто не захотели отвечать на мои письма с мольбами о размещении рекламы), а другая половина не соответствует критерию «с недорогим трафиком»...

В итоге, попробовали следующие сети.

Рекламная площадка	Трафик	CPC	Конверсия
TBN	+	6,23 руб.	0,12%
Metronom	+	5,52 руб.	0,44%
Link.ru	++	0,61 руб.	0,25%

Проблема, с которой мы столкнулись везде: чем выше CTR баннера (можно добиться высокого, если делать их мигающими, крутящимися, с элементами интерфейса и завлекательными фразами), тем ниже конверсия в покупки.

Когда мы писали «Поздравь с Новым годом! Оригинальные открытки» – было мало переходов. Когда писали «Закажи звонок Снегурочки для друзей» – мало покупок.

Трафик

Самый бесполезный простой вариант, платишь деньги – получаешь трафик. И никаких тебе рекламных материалов и CTR... Добиться внятного и понятного ответа.

откуда конкретно этот трафик к нам придет, ни от одного менеджера так и не удалось. И поэтому понять, почему статистика системы так упорно не хочет совпадать со статистикой LiveInternet и почему пришедшие к нам пользователи не хотят ни покупать, ни даже просматривать страницы, не удалось тоже.

Рекламная площадка	Трафик	CPC	Конверсия
Readme	+	3,62 руб.	0%
PopupTraf.ru	+++	13,16 руб.	0%

Текстовые объявления

С ними мы не подружались. И главная причина – отсутствие трафика. Это несерьезно тратить свое время на ведение рекламной кампании, результатом которой являются 500 кликов в месяц. При этом работать приходится в не самом «рекламодатель friendly» интерфейсе, платить электронными деньгами (то есть сразу понятно, 100 000 рублей бюджета там не потратить) и с грустью наблюдать за низкой конверсией в продаже.

Рекламная площадка	Трафик	CPC	Конверсия
MediaTarget.ru	+	4,25 руб.	1,5%
Liveclicx.net	+	0,55 руб.	0,11%
Smotri.com – текстовые объявления в видео	+	3 руб.	0%
Gde.ru	+	1,5 руб.	0%

Тизеры (картинка + текст)

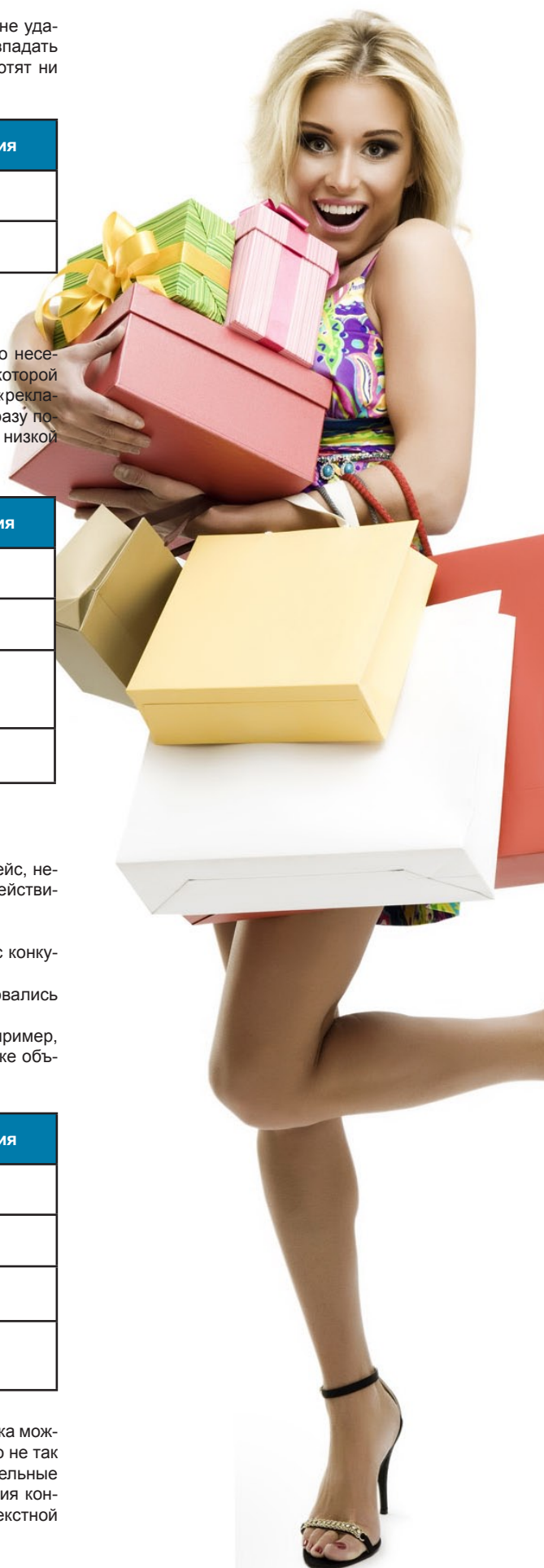
Для нас такие системы оказались наилучшим вариантом: удобный интерфейс, неплохая статистика и на некоторых площадках даже менеджеры, которые действительно помогают.

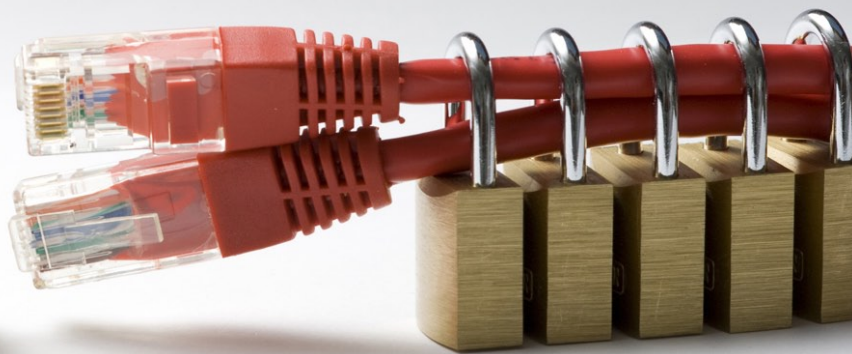
Вот наши советы по работе с тизерами:

- Пишите много объявлений постоянно, только это помогает справиться с конкуренцией и держать трафик на уровне.
- Найдите интересные темы, у нас, например, огромным успехом пользовались открытки от Равшана и смешные животные в картинках для объявлений.
- Приготовьтесь к некоторым дурацким немного странным правилам. Например, в Маркетгиде нельзя чтобы слова повторялись в заголовке, тексте и рубрике объявления (расширяйте тезаурус!)

Рекламная площадка	Трафик	CPC	Конверсия
Adlabs.ru	+++	0,52 руб.	1%
Marketgid.ru	+++	0,51 руб.	0,62%
Tx2.ru	++	0,4 руб.	0,27%
Fishki.ru – блок с тизерами	+	1,91 руб.	0,66%

ИТОГО: методом проб и ошибок найти дополнительные источники трафика можно. При этом трафик будет действительно продающим. Пусть его не так много из каждого отдельного источника, но в сумме дополнительные рекламные площадки дают необходимую страховку от изменения конкурентной ситуации, в традиционных любимых системах контекстной рекламы. ■





Автор

Илья Рабинович
SoftSphere Technologies

Ложь, БОЛЬШАЯ ЛОЖЬ, и антивирусы

Вот уже в который раз нас пугают страшилками в стиле «очередной вирус разошелся по миру миллионными тиражами. Мы все погибнем!». Вот только читая очередной бодрый пресс-релиз очередного производителя очередного антивирусного средства, недоумеваешь. Как же так? Нас так надежно защищают, столь всесторонне: тут и сигнатуры, и эвристики, и даже (писк сезона) поведенческий блокиратор. Тогда какого... люди продолжают заражаться? Откуда эпидемии? Действительно ли современные антивирусные средства эффективны?

Первые антивирусы возникли в районе 1985 года как ответ на первые файловые вирусы, заражающие исполнимые и интерпретируемые файлы и работающие в среде MS DOS. Кто еще помнит, это такая однозадачная операционная система, где ее ядро и прикладные программы работали на одном уровне привилегий. И именно антивирусы оказались на этой платформе наиболее рациональным инструментом борьбы с вирусами, причем как излечения уже зараженных машин, так и предотвращения заражения. Вирусы распространялись медленно, на дискетах, от пользователя к пользователю, а сигнатуры для поимки и лечения значительно быстрее, через сети (BBS, NNTP,...). И так продолжалось достаточно долго, до примерно начала 2000-х (то есть, пятнадцать лет минимум), когда произошло три кардинальных изменения

Кардинальное изменение №1: к нам пришел Интернет. А значит, среда распространения вирусов стала той же, что и распространения сигнатур. Опережение антивирусов над вирусами свелось к нулю.

Кардинальное изменение №2: вместо операционных систем на основе MS DOS (а это также вся линейка Win1.

xx–Win3.xx, Win95/96/98/ME) на десктопы пришло ядро Windows NT в реализации Windows 2000/XP. Теперь ядро операционной системы, ее код и данные, надежно отделены от адресного пространства обычных программ.

Кардинальное изменение №3: вирусы теперь пишут ради выгоды. Более того, именно «вирусы», фактически исчезли с компьютеров обывателей. Их заменили всевозможные «черви», «троянские кони», «блокираторы» и прочая нечисть, ориентированная на получение денег.

Именно с кардинальным изменением номер три связан первый сильный провал антивирусной индустрии – антивирусы не могли лечить уже зараженные троянами машины. Вот файловое заражение – сколько угодно, а когда исполняемые модули внедряются в операционную систему – нет. На этом выросла целая индустрия «anti-malware». Все тем, кто еще помнит такие названия как Spybot Search&Destroy, Ad-Aware, SpySweeper, думаю, ничего объяснять не надо. Антивирусная индустрия достаточно быстро сообразила, что деньги утекают из их рук и достаточно быстро намерстала упущенное.

Вот только из-за кардинального изменения номер один, уровень предотвращения заражения упал ниже всякой критики. Антивирусы катастрофически опаздывают. И ничего не спасает – ни эвристики, ни поведенческий блокиратор. Зловредописатели обходят все.

При этом на всех форумах периодически всплывают темы в стиле «Антивирус А пропустил заражение, он плохой. Посоветуйте хороший». Человеку советуют «хороший», который остается

в данном статусе вплоть до следующего пропуска. После чего цикл поиска «хорошего антивируса» повторяется.

Возникает парадокс – новые технологии предотвращения заражения, созданные благодаря «кардинальному изменению номер два» и показывающие абсолютные результаты в тестах на предотвращение заражения (так называемые «динамические тесты»), не могут пробить себе места под солнцем, ибо пришли на рынок довольно-таки поздно. Инерция сознания подавляющего количества пользователей просто не позволяет им искать ничего «защитающего», кроме антивирусов. Защита тождественно равна антивирусу. Точка. «Нужна защита. Посоветуй хороший антивирус». Знакомо?

А ведь антивирусы не лучшие в задаче предотвращения заражения! Просто они первыми начали!

Инновации и пресса

В первой части мы затронули аспект инерции сознания в выборе типа средств защиты от заражения зловерным программным обеспечением, зачастую ложно идентифицируемыми обычными пользователями как «вирусы».

Ну хорошо, скажете вы, мы тут такие все инертные, ничего нового не ищем. Но ведь есть компьютерные издания, журналисты которые должны следить за интересными новинками и писать о них. А они не пишут. Значит, новые технологии защиты плохие? Да нет, все не так однозначно. Когда кто-то создает новый, новационный продукт для рынка, где нет устоявшихся «хороших практик», технологий и лидеров, о нем, безусловно, напишут. Но если это не ▶

так, то никто ничего писать не будет. И причин тут несколько:

1. Журналисты такие же люди. Зачем что-то искать и о чем-то новом писать, если и так неплохо кормят? Да еще и в пресс-туры возят бесплатно...

2. Из-за инерции сознания, пользователи не просят журналистов писать статьи про новые средства защиты. Так зачем про них писать, если читателям не интересно и они не купят журнал?

3. Все издания существуют на доходы от рекламы. И антивирусная индустрия генерирует значительную ее часть. Если издание начнет печатать статьи, где говорится о ненадежности антивирусов, то на следующий же день к нему придут представители PR-отдела дистрибьютора или производителя антивируса и скажут, что сотрудничать с таким изданием им невыгодно. А за окном кризис...

Вот и получается замкнутый круг — пользователи не ищут ничего нового, более эффективного в предотвращении заражения, поскольку зачастую банально не знают о их существовании, а компьютерная пресса ничего не пишет, поскольку нет заказа со стороны читателя.

И даже если журналист и возьмется написать что-нибудь про вашу новацию, результат может быть, мягко говоря, неадекватным. Причина проста — в компьютерную журналистику идут или профессиональные журналисты, которые практически ничего не понимают в той теме, на которую пишут, ибо являются гуманитариями чистой воды. Либо бывшие IT-шники, вышедшие в тираж, оставшие от передовых технологических разработок и живущие прошлыми заслугами. Да и есть ли у пишущей братии время разбираться в вашем революционном продукте? Больше слов — больше денег. Значительно выгоднее написать десять тысяч знаков ерунды, не разобравшись в программе и получить причитающийся гонорар, нежели потратить это время на тщательное тестирование. Конечно, не все журналисты такие, но фактор налицо. Плавали — знаем.

Если же смотреть со стороны ре-

кламы в СМИ, тут тоже новационные продукты серьезно проигрывают антивирусам. Во-первых, маркетинговый бюджет любого традиционного производителя средств защиты значительно превосходит все то, что может вложить в канал любой стартап. Во-вторых, традиционные средства защиты можно пиарить в стиле «наши специалисты нашли страшный вирус, который никто не ловит, только наш антивирус». Инновационные продукты так пиарить не получится — они, зачастую, блокируют этот «страшный вирус» еще до того, как была написана первая строчка его кода...

ИННОВАЦИИ И ИНДУСТРИЯ

Ну ладно, посчитаем, что все инертные — и пользователи, и журналисты. Но почему крупные производители антивирусов ничего, фактически, нового не предлагают потребителям? Им что, все равно?

Может быть это странно звучит, но для крупных производителей средств защиты оные лишь средство ведения бизнеса. Кто-то производит и продает Кока-Колу, кто-то — антивирусы. И разницы между этими бизнес-процессами нет.

Это мелкие производители средств защиты, неистощимые энтузиасты, оперируют такими понятиями как «надежность средства защиты», «адекватность модели угроз» и, может банально, но «профессиональная гордость». Крупные производители оперируют лишь одним термином — ROI. ROI — это «уровень возврата инвестиций». Если ты вложил в производство сигнатурного движка сканирования, то сначала деньги нужно вернуть с прибылью, поделиться ею с акционерами, высшим менеджментом, а уже потом, может быть, сделать или закупить что-нибудь новенькое.

И мнение отдельно взятого потребителя никого тут не волнует, а податливое большинство настолько запугано, что его можно заставить поверить в надежность чего угодно. Например, фальшивые антивирусы. Они выглядят как

антивирусы, они сканируют как антивирусы, они вымогают деньги как антивирусы. Вот только ни от чего не защищают, скорее, наоборот. И знаете что? Даже без поддержки рекламы в СМИ, они неплохо продаются. Просто немного поиграть на страхах уже насмерть запуганного человека — и гребь деньги лопатой.

Хотите один интересный пример? В 2005 Лаборатория Касперского интегрировала в свою продуктовую линейку поведенческий блокиратор. На тот момент его не было ни в одном антивирусе. И была всего пара стартапов (CyberHawk, после покупки PC Tools'ом — ThreatFire, является на данный момент частью корпорации Symantec, и Sane Security Primary Response Safe Connect, что куплен AVG), безуспешно пытающихся пробиться в прессу и в массы. Через пять лет, к 2010 году, все те стартапы скуплены более крупными игроками, их программные продукты проинтегрированы в текущие продуктовые линейки. Не иметь у себя в антивирусе поведенческий блокиратор стало моветоном. Пробиться в массы тем стартапам до того, как их купили, так и не получилось, кстати. Люди просто не понимали, зачем им «еще один антивирус». Даже если и бесплатно.

Фактически, вся антивирусная индустрия оказалась в заложниках у себя самой. Маркетинговые отделы крупных производителей столь долго доказывали пользователям, что для защиты от вирусов и прочего зловредного кода им нужны антивирусы, что не только убедили в этом пользователей, но и сами в это поверили. И когда старая модель перестала работать, антивирусы становятся неэффективным средством предотвращения заражений, то возникает проблема. Пользователи хотят антивирусы (и ничего другого!) в качестве защиты, компании производят и продают именно то, что хотят пользователи и за что платят деньги, заражения становятся все более массовыми и никто не понимает, что делать. Вроде все хорошо, распаковщики, эвристики и интерфейсы продуктов становятся все лучше, но общая удовлетворенность от антивиру-

СЛУШАЕМ ПОДКАСТЫ НА ВЕБ-АНАЛИТИК.ИНФО

www.web-analitik.info/podcast/

- РУНЕТОЛОГИЯ с Максимом Спиридоновым
- ВРЕМЯ НОВОСТЕЙ с Владом Филатовым и Сергеем Болисовым
- PODCAST9 с Петром Диденко и Михаилом Черномордиковым
- APPLE МАНИЯ с Владом Филатовым и Сергеем Болисовым
- UBUNTU с Валентиной Мухамеджановой (Ubuntu.ru)
- Новости Hi-tech
- Новости IT



ДОБАВИТЬ СВОЙ
ПОДКАСТ



сов падает. Пользователи ведь ничего не понимают в безопасности, не знают, какими именно средствами стоит пользоваться, чтобы обеспечить безопасность своей цифровой жизни. Поэтому они возлагают эту задачу на экспертов, которым доверяют, а те остались своим созданием еще в начале-середине 90-х годов прошлого века.

Вот и получается, что если вам нужно что-то новационное прямо сейчас, то придется покупать ее у стартапов. А если вы доверяете только «большим именам», крупным производителям средств защиты, то вам придется подождать еще лет пять, пока текущие вложения отобьются им с прибылью.

Инновации и тесты

Средства безопасности — это черные ящики для их пользователей. Простой обыватель не способен понять, какова эффективность той защиты, которую он купил или собирается. Нужны тесты. Тесты, организуемые профессионалами.

До последнего времени, было всего два вида тестов средств защиты от заражений — прямой и ретроспективный тест сканирующего антивирусного движка. Больше ничего. Вообще. Прямой тест — когда берется коллекция из примерно миллиона образцов, непонятно как и где собранных, с неизвестным количеством мусора, который вообще не запускается, и на все это безобразие напускают сканирующий движок. Дешево и сердито. Ретроспективный тест — это тест с «замороженными» сигнатурными базами. Они перестают обновляться несколько недель, а потом на свежеступившее мясо натравливают тот же антивирусный сканер.

Как видите, в таких тестах нет места для новационных подходов предотвращения заражения. Тестируется только сканирующий движок, как будто ничего другого вообще не существует. Однако у большинства современных антивирусов появился поведенческий блокиратор и фильтр URL, тестировать которые старыми методами невозможно.

И тогда появилась организация AMTSO, которая занимается выработкой подходов для кардинально нового, так называемого динамического, тестирования. То есть, собираются ссылки на зловредные приложения и запускаются на тестовом стенде. Тут уже могут сработать и ссылочные фильтры, и поведенческий блокиратор, а сам тест становится более приближенным к реальности.

Динамические тесты технически значительно сложнее, чем простой прогон движка-сканера по коллекции зловредных модулей. Фактически, ни одна из тестирующих организаций не делает данный тест на регулярной основе, некоторые пока только находятся на этапе подготовки. Динамические тесты на предотвращение заражения, видимо, плотно войдут в нашу жизнь лишь в 2011-2012 годах, становясь основным мерилом эффективности предотвра-

щения заражения.

Полагаю, что многие производители постараются замолчать либо исказить результаты подобных тестов, плотно сосредоточившись на устаревших «сигнатурных», поскольку результаты будут отнюдь не в их пользу. В ноябре 2009 вышел первый динамический тест, произведенный российским порталом в области безопасности Anti-Malware. По [ссылке](#) его результаты. Как видно, старые подходы к обеспечению защищенности компьютеров с треском проиграли новым, пробивающим себе дорогу в жизнь. А теперь посмотрим на реакцию производителей средств защиты. Откликнулась только Лаборатория Касперского, маркетологи которой легким движением руки превратили второе место Kaspersky Internet Security в ... первое! Не верите — вот вам еще одна [ссылка](#). Все остальные просто проигнорировали тест, как будто его вообще не существует.

Инновации и решение проблемы вирусного заражения

Что управляет этим миром? Пресса? Телевидение? Монстроидальные корпорации? Нет — экономика! Именно она управляет людьми, которые хотят денег. А люди, которые хотят денег, составляют подавляющее большинство всех тех людей, которые вообще хоть чего-то еще хотят и готовы действовать ради этого.

Откуда берется экономика на зловредном программном обеспечении? А берется она из положительного сальдо между украденными у простого пользователя ресурсами минус стоимость обхода средств защиты. Все очень просто и логично, не так ли?

Какова же стоимость обхода современных антивирусных средств? Учитывая, что эпидемии случаются регулярно, а большинство пользователей все же имеют антивирус с регулярно обновляемыми сигнатурными базами, то результат неутешителен — стоимость обхода современного антивируса достаточно низка, чтобы экономика на зловредном программном обеспечении не имела возможности существовать.

Происходит это потому, что все антивирусы реализуют «черносписочный» подход к обеспечению безопасности. Это значит «я знаю, что это плохой модуль (плохое поведение программы), я его блокирую». Достаточно замаскировать модуль, сбив сигнатуру и притвориться «хорошо себя ведущей» — все, оборона прорвана. Пока информация доходит до производителя, пока там среагируют... Можно действовать, никого и ничего не опасаясь. А чтобы антивирус больше не мешал, взять, да и выключить его. Ничего сложного в этом нет. Немного расходов — и дальше только выгода, выгода, и ничего кроме выгоды.

И только принципиально иные подходы способны настолько увеличить стоимость обхода средства защиты, что продолжать бизнес на зловредном программном обеспечении станет вообще невыгодным. Почему? Давайте смотреть экономику пробоя новых средств защиты.

Если брать «не черносписочные» подходы, то их всего два основных:

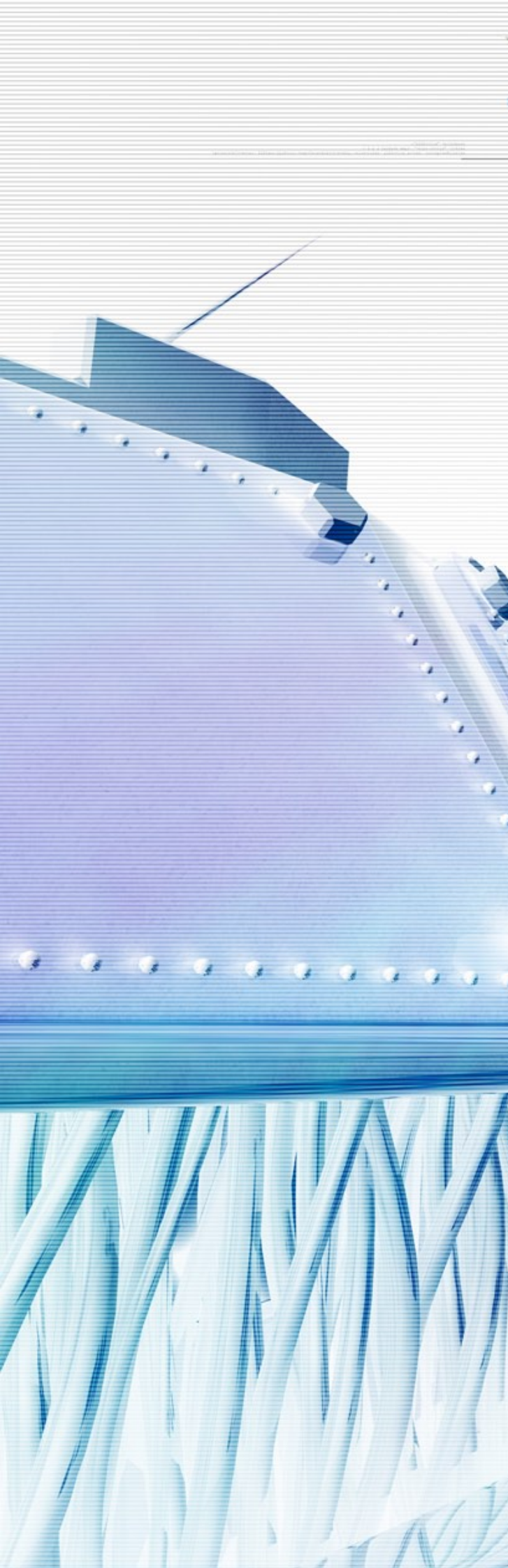
1. На основе «белых списков». То есть, запрещаем запуск всего того, про что мы не знаем, что оно заведомо хуже, незловредное.

2. На основе модели «песочницы», изолируя потенциально опасные процессы от всех остальных и от операционной системы.

Чистые белосписочные решения неприменимы в массовом продукте, поскольку пользователю очень тяжело ими пользоваться. Такого типа программы будут «ругаться» при обновлении используемого программного обеспечения, например, пока они не попадут в центральную базу, хранящую контрольные суммы «хороших» модулей. Кроме того, практически все подобные решения имеют врожденные недостатки в виде проблем в работе с файлами, содержащими скрипты. Поскольку скрипт — это лишь набор текстовых строк, а интерпретируют эти строчки команды к действию вполне себе легитимные (зачастую системные) исполняемые файлы. Так что обойти их либо достаточно тривиально, либо практически невозможно (но и работать с ними, при этом, также будет очень сложно).

Если же рассматривать песочницы, то нахождение дыр в них, которые уже достаточно давно на рынке, достаточно нетривиальная задача. Задача, посильная разве что профессиональному хакеру, время которого стоит достаточно дорого. Стоимость обхода хорошей песочницы может вылиться заказчику в тысячи долларов и довольно продолжительный период ожидания. Стоимость закрытия подобной дырки на стороне разработчика песочницы — несколько десятков долларов, полчас времени (с тестированием — ну пара часов максимум). Причем, поскольку старая дырка уже закрыта, то для обхода защиты нужно искать новую, что снова требует больших денег и времени. А поскольку все очевидные уязвимости обычно выбираются первыми, то дальше стоимость и время нахождения новых только увеличиваются.

Получается, что при использовании либо решений на белых списках, либо песочниц, бизнес на зловредном программном обеспечении становится все менее и менее рентабельным, с каждой следующей итерацией защита становится все крепче, а обход ее все дороже. И однажды будет пройдена «точка невозврата», когда такой способ заработка денег станет, банально, невыгодным. А что это, если не 100% защита от вирусов и прочих зловредных приложений? ■



ЖЕЛЕЗО

**Мультимедийный планшет
Toshiba FOLIO 100**

Медиаплеер с начинкой

Полтора месяца с iPad'ом



Автор

Игорь Снытко
Генеральный Менеджер
региона СНГ и Балтии
Toshiba Europe GmbH
www.toshiba.ru
г. Москва

Мультимедийный планшет Toshiba FOLIO 100

На выставке IFA2010 в Берлине Toshiba представила свой первый планшетный компьютер на базе Android – мультимедийный планшет FOLIO 100. Устройство с диагональю экрана 25,7 см (10,1 дюйма) предназначено для людей, которые желают иметь все любимые файлы и интернет под рукой. FOLIO 100 поступит в продажу в регионе EMEA в 4-м квартале 2010 г.

Планшет FOLIO 100 весом 760 грамм с толщиной корпуса всего 14 мм укомплектован мощным процессором Nvidia Tegra™ 2, поддерживающим идеальный баланс между производительностью и энергопотреблением.

FOLIO 100 поставляется с большим количеством приложений и сервисов, созданных для облегчения доступа к мультимедийным файлам и их воспроизведения. Таким образом, можно с комфортом заниматься веб-серфингом, и общаться в социальных сетях. Благодаря разработанному Toshiba интуитивно понятному графическому интерфейсу планшет FOLIO 100, работающий под управлением ОС Android 2.2 (Froyo), невероятно прост в эксплуатации.

В устройстве есть: SD-кардридер, интерфейсы HDMI, USB 2.0, Wi-Fi и Bluetooth. Кроме того, в ближайшее время компания планирует анонсировать версию планшета, в которую будет добавлена поддержка широкополосной мобильной связи (3G). ►



FOLIO 100 предлагается с большим набором предустановленных приложений, среди которых веб-браузер Opera Mobile, FBReader для чтения электронных книг, пакет для работы с офисными документами Document To Go, приложение Evernote для создания и синхронизации заметок. А так же имеется приложение Fring, позволяющее с помощью встроенной веб-камеры звонить друзьям или членам семьи и видеть их лица на экране. Кроме того, устройство поддерживает технологию Adobe Flash 10.1.

Время автономной работы этого устройства составляет до 7 часов, что позволяет использовать его на протяжении почти всего рабочего дня без подзарядки. Время включения устройства составляет всего 30 секунд, после чего оно полностью готово к работе. А встроенной флэш-памяти на 16 Гб будет достаточно для хранения фотографий, электронной почты, музыкальных и видеофайлов. В случае необходимости дополнительную память всегда можно получить благодаря поддержке SD-карт.

А вот еще один интересный шаг, подобного которому Toshiba, кажется еще не делала. Собственно, компания приглашает разработчиков присоединиться к экосистеме FOLIO 100 для создания и распространения приложений, игр и других материалов, предназначенных для использования на этом устройстве. Получить поддержку и оптимизировать свои приложения для FOLIO 100 разработчики могут, обратившись за помощью к специально созданному portalу Toshiba по адресу www.toshibatouch.eu. ■



Технические параметры

Процессор Nvidia Tegra 2

Емкостной мультисенсорный дисплей 25,7 см (10,1") с разрешением 1024 x 600 пикселей

OS Android 2.2

Встроенная флэш-память емкостью 16 Гб

Bluetooth 2.1+EDR, WLAN (802.11 b/g/n), поддержка широкополосной мобильной связи (будет добавлена позже в некоторых моделях и странах)

Соответствие DLNA

1 x mini HDMI, 1 x USB 2.0 (клиент/хаб), кардридер SD/MMC

Поддержка Adobe Flash 10.1

Автоматический поворот изображения благодаря акселерометру

Веб-камера 1,3 МП

Время автономной работы: 7 часов (65% веб-серфинг, 10% воспроизведение видео, 25% ожидание)

Вес: 760 г

Размер: 281 x 181 x 14 мм

ПО: Opera Mobile, Toshiba Media Player, FBReader, Fring, Document to Go, Evernote и др.





Автор

Сергей Кузнецов
г. Балашиха
www.kuuuzya.ru

Медиаплеер с начинкой

Пожалуй, сейчас уже никого не удивишь медиаплеером, их многообразие с каждым днем все увеличивается. Бывают и с HDD и без, и с поддержкой Lan и без, в общем, на любой вкус. Но давайте поговорим о стандартном представителе этого клана видеопроигрывающих устройств. Рассмотрим его в буквальном смысле со всех сторон и даже препариреуем.

Перед нами Egreat EG-R2A. Начну, пожалуй, с общего – комплектация. Помимо плеера в комплект поставки входит:

Адаптер и кабель питания

HDMI-кабель

Пульт

Винтики

Диск и инструкция

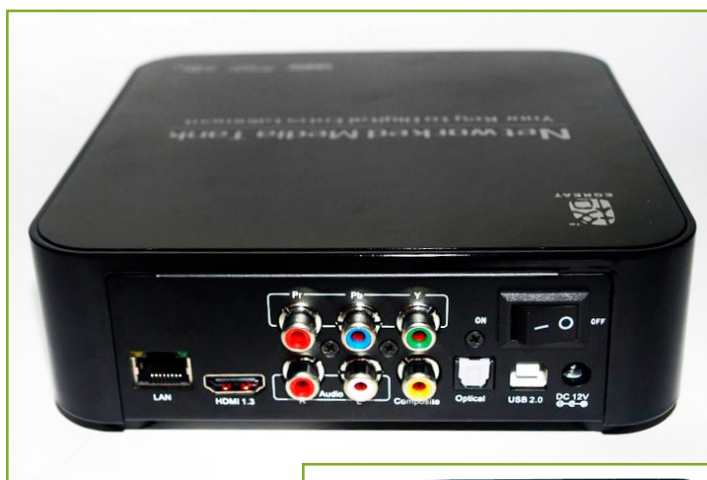
Немного странно, что в комплекте не было стандартного провода с RCA разъемами, ведь пока еще не все телевизоры умеют соединяться по HDMI.

Дизайн

Этот медиаплеер в отличие от многих своих собратьев представляет собой не прямоугольник, а квадрат со скругленными краями. В высоту около 5см, на резиновых ножках, глянцевый (увы), он прочно будет стоять, позволяя насладиться видео.

Спереди находится USB-hub и разъем eSata, так что плеер можно спокойно придвигать к стенке и не мучиться с тем, как вставить флешку или подключить жесткий диск. Также на передней части находятся ИК-порт и индикатор работы плеера. Боковые панели девственно чистые, а вот задняя содержит в себе различные интерфейсы.

Тут находятся Lan, HDMI, композитные, Optical, USB 2.0, ну и конечно, разъем для питания этой «коробочки».



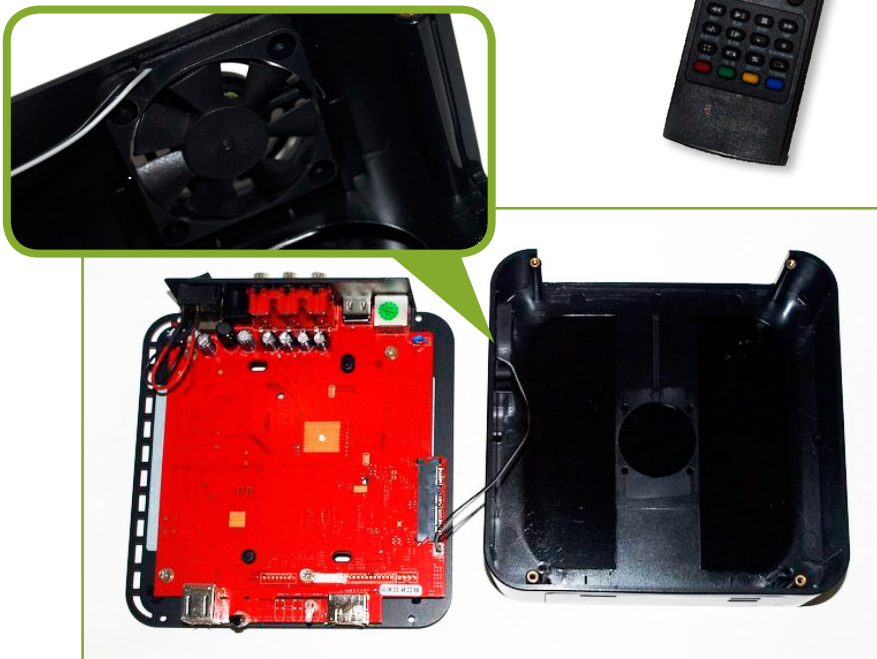


Как видите, никаких кнопок помимо On/Off на корпусе нет, так что все действия выполняются только с пульта. Он ничем не отличается от пультов остальных таких же устройств. Все те же кнопки, разве что расположение отличается.

Интересности

Первое, что удивило после открытия упаковки, так это то, что плеер разобран, а винтики лежат отдельно.

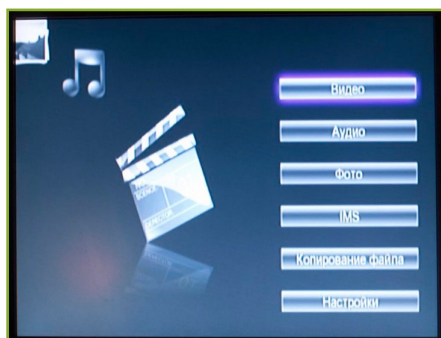
Сделано это для того, чтобы можно было сразу вставить внутрь 3,5" SATA HDD, который, к сожалению, в комплекте не поставляется. За охлаждение жесткого диска можно не переживать: сверху на корпусе находится теплоотводящая пластина, а сбоку есть маленький с виду, но достаточно мощный по оборотам кулер.



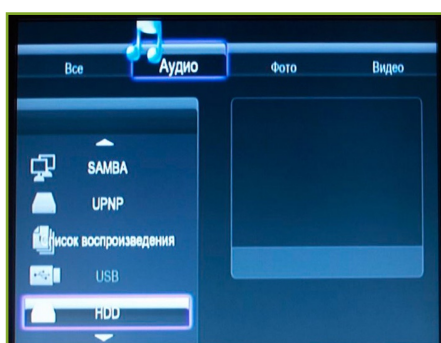
Меню и воспроизведение

Как и в каждом нормальном плеере здесь присутствует некое меню с разнообразными пунктами. Разберем несколько из них. Дизайн, конечно, не на 5+, но не в интерфейсе счастье.

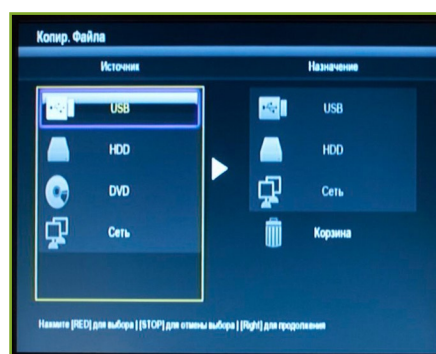
Итак, главное меню:



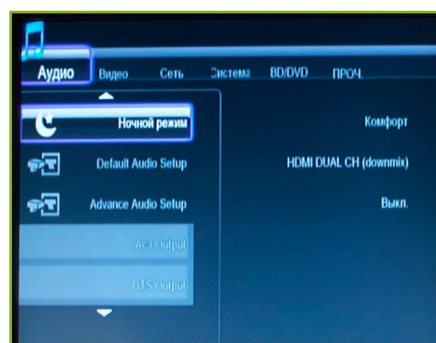
Аудио выглядит именно так. Как видите, воспроизводить можно из многих источников.



Понравилась функция копирования файла. Можно вставить флешку, скопировать на жесткий диск и смотреть, не травмируя свой USB-носитель.



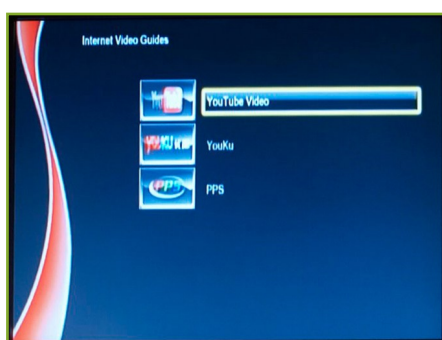
В настройках есть много дополнительных «фишек» и тонких настроек, например, яркость экрана, качество звука и т. п. ►



Ну и куда же мы в сетевом, фактически, плеере без интернет-ресурсов? Никуда! Тут вам и RSS, и сервис погоды, и Picasa, и поиск видео.



Ну а в интернет-каналах находится Youtube и еще два непонятных онлайн-проигрывателя, в которых я не нашел ничего интересного.



Попробовал несколько самодельно собранных MKV файлов – не пошел только один, хотя на PC запустился с помощью Media Player Classic, но как-то с зависаниями. Также, естественно, протестировал музыку и интернет-видео. Могу

сказать, звук действительно Dolby Digital, но это, конечно, если подключать колонки к плееру, а не слушать через ТВ. Воспроизведение же Youtube и иже с ним будет зависеть скорее от скорости интернета, чем от свойств проигрывателя.

Качество, как и положено плееру такого уровня – на высоте. Конечно, достаточно странное меню, но что касается воспроизведения, то никаких глюков и тормозов. Даже на удивление подхватились почти все субтитры, даже кастомные.

Скорость интернета у меня около 5-6 мегабит, но штук пять фильмов 1.3Gb DVD-Rip за ночь скачивает. Web-морда позволяет загружать torrent-файлы.

При покупке IDE-USB переходника за 10\$, можно подключить старый DVD-ROM и проигрывать фильмы с дисков. Все ставится само, ибо нуксы и дров в комплекте много.

Подведение итогов

На мой взгляд, это достаточно неплохая модель. Не хуже и не лучше своих собратьев по цеху. Радует в ней то, что можно установить HDD внутрь, обилие разъемов и возможностей подключения, а также то, что проигрываются почти все форматы. Я думаю, что за цену в районе 4000 р. это достойный экземпляр HD плеера, особенно, если его подключить к широкоформатному ТВ и получить мощный медиацентр.

Плеер поставляется в двух цветах: черный и белый. Это позволит каждому выбрать тот цвет, который больше вписывается в интерьер. ■



Технические параметры

Комплектация	без HDD
Поддержка HD	1080p (Full HD)
Менеджер зачак	BitTorrent
Выходы	аудио стерео, аудио оптический, видео композитный, видео компонентный, HDMI 1.3
Интерфейсы	USB Type A x2, USB Type B, Ethernet, eSATA Host
Поддержка HDD	3.5", SATA
Поддержка файловых систем	FAT32, NTFS
Форматы файлов	MKV, AVI, WMV, ISO, IFO, VOB, MOV, ASF, TP, TS, TRP, M2T, M2TS, MP4, MPG
Кодеки	MPEG1, MPEG2, MPEG4, XviD, DivX, H.264, WMV9, AVCMD, VC1
Аудиофайлы	MP3, WMA, AAC, Ogg, PCM, M4A, WAV, FLAC, AC3
Графические файлы	JPEG, GIF, BMP, PNG
Поддержка субтитров	sub, srt, ssa, smi, idx+sub, pgs
Процессор/чипсет	Realtek 1073
Оперативная память	128 Мб
Flash-память	256 Мб



Дизайн

Звук

Воспроизведение и мультимедийность



Небольшая комплектация

Меню

ложений бесплатных или lite-версий. Весь основной функционал был из коробки (браузер, видео, плеер, фотографии, карты, календарь, заметки, почта). Из действительно важных приложений скачал: iBooks, Яндекс Карты, DOCUMENTS 2 FREE, Dropbox, Калькулятор, Диктофон – все бесплатные.

За два месяца накопилось три экрана игрушек, из них только одна платная. Понравилась интересная модель – сама игрушка идет бесплатно, но в ней ограниченный набор уровней или режимов сложности, остальные можно приобрести отдельно. Но во многих игрушках это совсем не нужно, ограниченного количества уровней вполне хватает.

Еще в iTunes есть такая практика, многие приложения становятся бесплатными на несколько дней (чтобы накрутить количество скачиваний, рейтинг, подняться в рейтинге и снова стать платными). В этот момент самые классные игрушки можно получить совершенно бесплатно, есть даже специальное приложение (называется: free app tracker), которое показывает, что сегодня можно урвать за ноль долларов.

Для просмотра видео я использую стандартный плеер он меня вполне устраивает. Единственное, он поддерживает только .mp4, другие форматы не подойдут. Есть и другие плееры, но я их не ставил. Мне проще сразу закачать кино в нужном формате, благо это не составляет проблемы, да и самому это сделать не сложно. Я бы посоветовал утилиту Leawo Free MP4 converter,



которая умеет конвертировать все что угодно в MP4, не искажает пропорций видео (многие утилиты пытаются подогнать входящий файл под соотношение экрана 4х3), может конвертировать несколько видео-файлов параллельно на разных ядрах (может перегонять сразу целый список файлов), что особенно полезно при перегонке сериала.

Я нахожу большие плюсы в предварительной легкой подготовке видео, 1 минута предварительной подготовки дает 100% надежность просмотра. Этого мне больше всего не хватало на нетбуке, когда всеядный плеер мог начать показывать фильм с глюками и притормаживаниями, расхождением видео со звуком, из-за какого-то нестандартного кодека или кривых рук кодировщика. В итоге бывали моменты, когда оставался в дороге без кино, хотя фильм на нетбуке был в наличии.

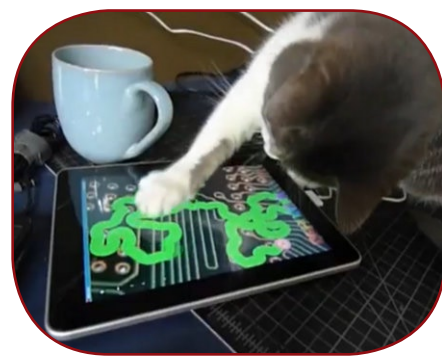
Железные впечатления

Начнем с батарейки – она фактически бесконечна, заявленные 10 часов в режиме просмотра видео отработывает честно, что на самом деле означает «ваш iPad никогда не сядет». Мне иногда приходится ездить в командировки, которые занимают один день и ни разу еще заряд батареи не опускался ниже 30%. Единственный раз, когда мне его удалось разрядить до 10% заряда – это когда на нем целый день и всю ночь игрались друзья, разводя в шесть рук кораблики по разным портам и мучая аэрохоккей, а потом я еще все утро сидел в интернете. Но это был стресс тест, в ходе которого из него каждую секунду вжимали максимум.

3G и Wi-Fi заметного влияния на энергопотребление не оказывают, видимо есть какие-то алгоритмы энергосбе-

режения. Возможно что-то «допилили» в софте, так как первые версии iPhone довольно сильно сажали батарейки при использовании Wi-Fi, увеличивая потребление энергии раза в два.

Экран классный, сказать о нем нечего, он действительно нереально хорош. Да он глянцевый, но в помещении проблем не возникает, т. к. он довольно яркий, а на улице я iPad практически не использую. Легко чистится от отпечатков обычной микрофиброй для оптики, тряпочка продается в любом магазине, в котором продают очки. Единственный минус – экран таки царапается, хоть и довольно тяжело. У меня уже появилась небольшая царапина, которую правда не видно при включенном экране. По моим подозрениям это дело когтей кошки, хотя она и не признается.



Wi-Fi и 3G работают без нареканий к устройству, но с нареканиями к нашим 3G провайдерам, уровень сигнала которых зачастую оставляет желать лучшего.

Micro-SIM делается из самой обычной симки за 10 минут, для этого потребуются канцелярский ножик и пилка для ногтей. Я уже сделал две (от разных ►





демонстрация портфолио клиентам.

Плюсы:

- Все очень качественно и добротно сделано, алюминиевый корпус делает устройство цельным. Устройство маленькое, легкое и крепкое. Экран в девять дюймов идеален, меньше было бы неудобно читать (особенно сайты, где часто верстка не резиновая и нужно охватить по ширине весь блок), а больше — не помещалось бы в обычную сумку.
- Долгоживущая батарейка, которая отрабатывает свои 10 часов активного использования, этого более чем достаточно (можно успеть посмотреть 6 фильмов).
- Удобство в использовании — никаких настроек и переустановок для обычной работы не нужно.
- Книги в iBooks закачиваются в формате ePub, в который их можно легко сконвертировать из любого текстового формата.
- Идеальный фотоальбом для демонстрации «как я съездил туда-то» родственникам. Можно закатать фотоальбомы и листать их на хорошем экране,

операторов), ни каких проблем не возникло.

У устройства всего одна колонка, т. е. внешний звук МОНО, это можно было бы записать недостатком, но эта колонка настолько хороша, что рука не поднимается — громкая, звук очень чистый. Acer Aspire One, со встроенными стерео-динамиками, нервно и негромко похрипывает в сторонке.

Заключение

Устройство, как это ни странно, является идеальным нетбуком. Если вам нужно что-то для серфинга в интернете, работы с почтой, чтения книг, просмотра фильмов, игр в дороге или зале ожидания — iPad создан для вас.

Для меня устройство оказалось крайне полезным, но я не строил никаких иллюзий и знал что мне от него нужно. Работать на нем у вас не получится, максимум — работа с почтой и

увеличивая и показывая детали, при чем это смогла делать даже моя 80-ти летняя бабушка, которая ко всей технике относится крайне холодно.

- Серфить в интернете ОЧЕНЬ удобно, даже удобнее чем на компьютере.
- Огромное количество приложений, утилит и игр, значительная часть из них вам не будет стоить ни копейки. При этом все они прошли проверку перед принятием в AppStore и случаи «глюков» можно пересчитать по пальцам одной руки, это я говорю по опыту работы с более чем сотней приложений. Ни одного сколько-нибудь серьезного глюка я не встретил вообще.
- Вау-эффект в наличии, все знакомые непременно захотят его потрогать и посмотреть, т. к. таких железяк пока не много в стране.

Минусы:

- Экран все-таки царапается, хоть и устойчив к воздействию, песком его лучше не полировать.
- Без Джейлбрейка не получится закатать готовый кэш Яндекс-Карт. ■



Обязательно возьмите кейс, стандартный эппловский вполне подойдет, он очень качественный и сравнительно недорогой. Выглядит значительно лучше, чем на фотографиях, где он кажется пластиковым, на самом деле это не так. Он сделан из микрофибры, как тряпочки для протирки оптики, на ощупь нечто среднее между мягким шершавым пластиком и тканью, плотный, крепкий, но хорошо гнется на сгибах. Кейс позволяет ставить iPad под двумя углами, с ним легче держать (не выскальзывает), и можно спокойно кидать в сумку рядом с ключами и всякой другой царапающейся ерундой

ХОСТИНГ



Супер Старт —

полноценный хостинг начального уровня,
идеально подходящий для размещения
вашего первого сайта.

1 сайт, 1 база данных MySQL, 1 Гб дискового
пространства, 10 почтовых ящиков
с антивирусом и антиспамом, PHP, SSI, CGI,
круглосуточная тех. поддержка.

99 Р/мес.*

+ домен
в подарок *



* При подписке на 1 год

(495) 508-99-59
www.rusonyx.ru/superstart

DockBar

или почему я стал строить свой велосипед

Операционные системы семейства GNU/Linux всегда славилась гибкостью и приспособляемостью под предпочтения пользователя, и как правило, для решения какой-либо задачи существует множество альтернатив. Графический интерфейс пользователя не является исключением, существует несколько графических сред, множество оконных менеджеров и просто куча утилит, апплетов, плазмоидов и прочих приспособлений, чтобы сделать работу комфортной и продуктивной. Об одном из таких апплетов и пойдет речь в этой статье.

Мне всегда нравился док, считаю это средство управления окнами и приложениями более удобным, чем панель задач. Удобное, потому что док ориентирован на работу с приложениями, а не с отдельными окнами, следовательно:

- Удобная работа с приложением, у которого открыто множество окон (зависит от конкретного дока).
- Значок в доке это не только группа окон, но и средство быстрого запуска, а в некоторых случаях и средство вывода полезной информации.
- Быстрый доступ к функциям приложения (конечно, если это поддерживается доком и приложением).

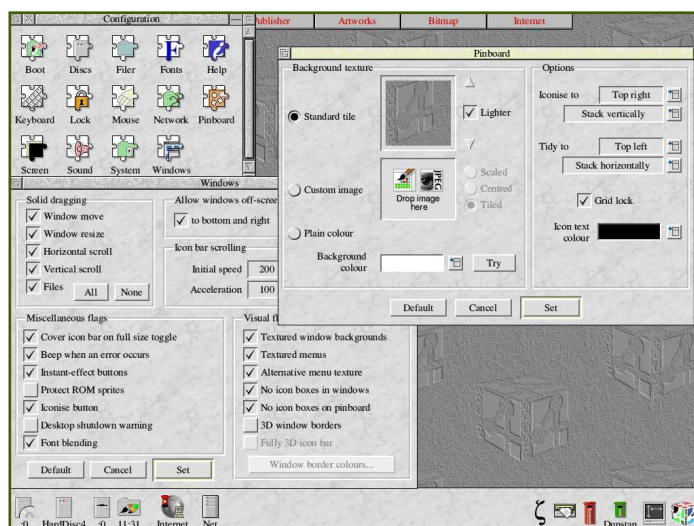
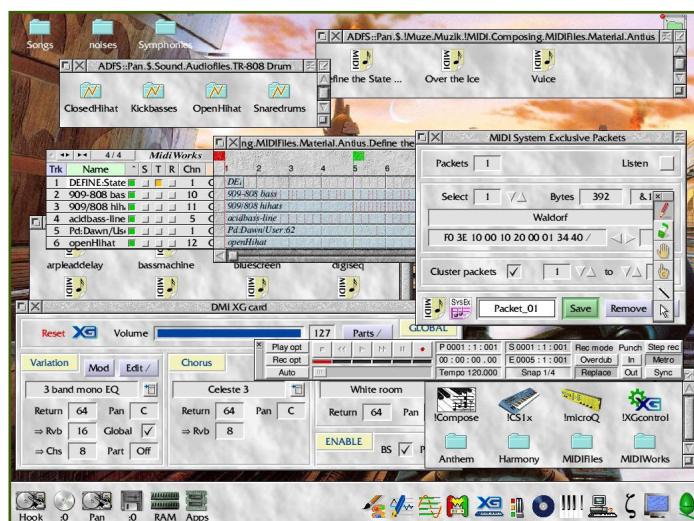
Первая система, которая приходит на ум при слове «док» это конечно Mac OS X, возможно кто-то вспомнит о NeXTSTEP или GNUstep (свободное окружение для unix-подобных ОС, реализация стандартов OPENSTEP), но мало кто знает, что прообраз дока появился достаточно давно. А было это в 1987 году, когда британская фирма Acorn Computers выпустила операционную систему Arthur, позд-

нее переименованную в RISC OS. Прообраз дока в ней называется IconBar и представляет собой панель со значками, которая располагается внизу и занимает все пространство по ширине. Значки, расположенные на панели выравниваются по левому или правому краю, в зависимости от типа программы владеющей значком. Так значки, относящиеся к дискам и файловым системам должны помещаться в левой части панели, а значки остальных приложений – в правой. Если на панели становится слишком мало места, то она расширяется за пределы экрана, размер значков при этом не меняется, а скрытая часть становится доступной

при подведении мыши к одному из краев экрана.

Интересным является то, что через значок можно получить быстрый доступ к функциям приложения. Так по щелчку левой кнопкой открывается чистый документ или окно файлового менеджера (в зависимости от типа приложения), а по щелчку средней – появляется меню приложения.

Другая операционная система прошлого, в которой использовалась концепция дока – NeXTSTEP. Первая версия этой операционной системы была выпущена компанией NeXT в 1989 году. Док расположен в правой части экрана и представляет собой набор кнопок со



значками часто используемых приложений. Он всегда располагается поверх обычных окон, но есть возможность временно переместить док за край экрана. Пользователь может добавлять приложения в док перетаскивая их из файлового менеджера и удалять приложения из дока, перетаскивая значок за пределы дока. Значок **Workspace Manager** (самый верхний) содержится в доке всегда. Если приложение еще не запущено, то в нижней левой части значка отображаются три точки. Двойной щелчок на значке в доке запускает приложение, а если приложение уже запущено, то его окна выводятся на передний план. Щелчок правой кнопкой мыши показывает меню приложения (меню приложений в NeXTStep не привязаны к окнам и являются плавающими). Приложение может отображать не просто картинку, а выводить какую-либо полезную информацию, например приложение для настройки параметров операционной системы показывает часы вместо простого значка. Значки запущенных, но не добавленных в док приложений располагаются у нижнего края экрана, там же располагаются значки свернутых окон.

И, пожалуй, наиболее известная система, в которой док является неотъемлемой частью интерфейса – Mac OS X. А про нее вы и так все знаете...

Идея дока достаточно популярна, и как для Windows так и для GNU/Linux существует большое количество сторонних приложений, так или иначе реализующих подобный функционал. Должен заметить, что я пользуюсь ОС Ubuntu и работаю в окружении Gnome. Собственно то, что у меня не получилось ужиться со стандартной нижней панелью Gnome и послужило толчком для поисков чего-то более удобного, а впоследствии, и для начала работы над собственным апплетом.

Не найдя желаемого удобства при работе с панелью Gnome я стал искать подходящие реализации дока для этого DE. Не буду во всех подробностях описывать, что и в какой программе мне не понравилось, просто обобщу:

- Многие программы того времени представляли собой панель задач с «закосом» под док, т. е. просто панель задач но с большими значками вместо обычных кнопок.

- Если все же окна группировались, то это происходило неудобно. Например, были значки окон без

подписей, т. е. не понятно где какое окно, или для доступа к отдельным окнам надо было лезть в далекое выпадающее меню.

(Некоторые из тех программ за время существования моего проекта преобразились и обросли новыми функциями.)

Перепробовав несколько программ и не найдя ничего подходящего, я решил что надо браться за работу над собственным велосипедом. Были поставлены следующие цели:

- Это должен быть апплет для стандартной панели.

- Все окна одного приложения должны группироваться, как и положено для дока.

- У каждой группы должен быть всплывающий список окон.

- Все действия для окна или группы должны быть доступны максимально быстро (т. е. без дополнительных меню, долгого удержания нажатой клавиши мыши, клавиш модификаторов и т. п.) и легко настраиваться под предпочтения пользователя.

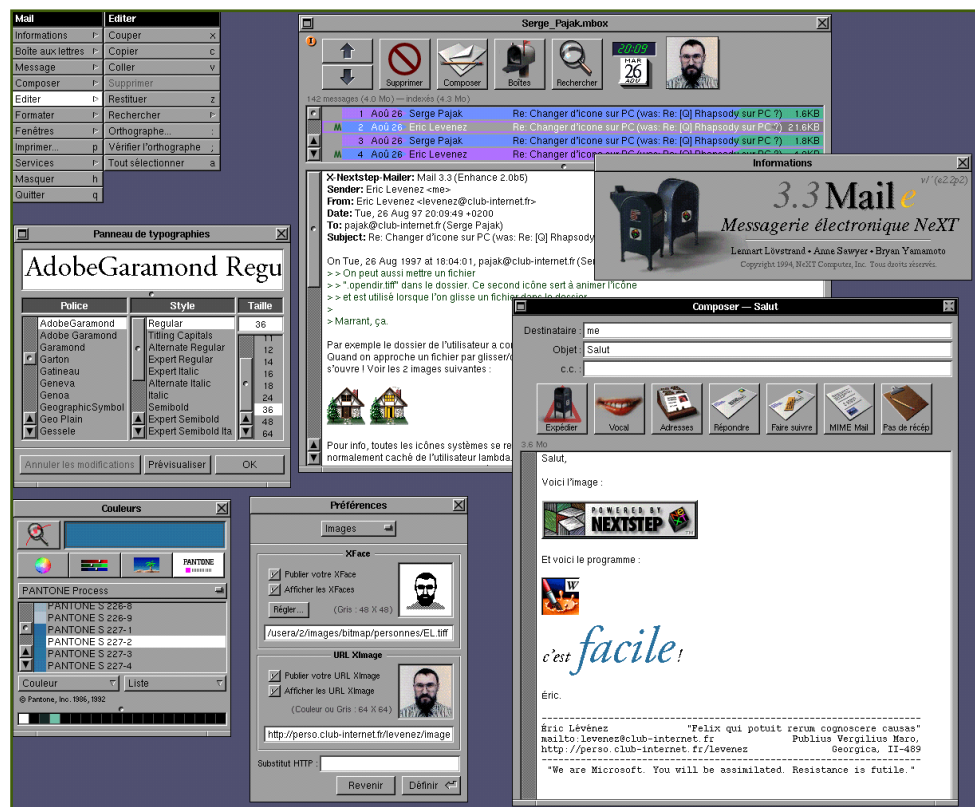
Опыт программирования с использованием GTK (библиотека виджетов в Gnome) на тот момент у меня отсутствовал полностью, также я не особо хорошо представлял, как происходит взаимодействие различных доков, панелей и переключателей столов с менеджером окон. Поэтому я начал с поиска документации по созданию апплетов для Gnome и описания работы оконных менеджеров. С поисками документации везло не особо, но зато мне

попался апплет панели задач (сейчас, к сожалению, не скажу его название) для XFCE написанный на языке Python. Из чтения его исходных кодов я узнал о существовании библиотеки libwnck (Window Navigator Construction Kit), которая сильно облегчает разработку подобных приложений, кстати, она же и используется в стандартных панелях Gnome и XFCE. После этого принялся за набросок своего приложения на Python'e.

Почему Python? Потому, что это простой и удобный ОО язык. Исходный код программы на Python с использованием GTK гораздо компактнее и легче читается по сравнению с кодом на C (сама библиотека GTK написана на C, несмотря на это, является объектно-ориентированной). Кроме того, Python и биндинги для различных библиотек присутствуют в большинстве дистрибутивах из коробки.

В начале января 2009 появился первый рабочий апплет, а 18 января 2009 я выложил на www.gnome-look.org первую доступную версию, проект получил название DockBar. Первая версия не обладала большим функционалом, действия на нажатия кнопок мыши были жестко прописаны в коде, возможность закрепления приложения на панели отсутствовала вообще. Однако апплет заинтересовал пользователей окружения Gnome и мне стали приходить пожелания и патчи.

Среди прочих, в марте 2009 года мне написал программист из Финляндии Matias Särs, он предложил дополнить апплет возможностью прикрепления приложений к панели, но я отказался



т. к. решение было неудобным, и требовало от пользователя ввода класса окон (`WM_CLASS`) приложения. Таким образом, появилась отдельная ветка — DockBarX. Позже в основной ветке была добавлена возможность прикрепления приложений простым перетаскиванием, без необходимости ввода класса окна приложения, теперь такая система используется в обеих ветках.

Далее я хотел бы немного рассказать о том, как работает DockBar и о том, почему с доками для GNU/Linux не все так гладко. Для того чтобы DockBar работал именно как док, а не как панель задач должно быть выполнены две задачи: во-первых, нужно сгруппировать окна одного приложения; во-вторых необходимо определить что же это за приложение. С первой задачей как правило не возникает проблем, окна группируются по классу (свойства окна `WM_CLASS`). Согласно стандарту ICCCM каждое окно имеет свойство `WM_CLASS` типа `STRING`. Свойство содержит две строки, завершающиеся нулевым символом:

- **Имя экземпляра** (англ.: Instance name, также обозначается `res_name`) — имя конкретного экземпляра приложения. На POSIX-совместимых системах значение должно быть одним из следующих:

- Аргумент параметра `-name`, переданного при запуске приложения.

- Если параметр `-name` не задан, то значение берется из переменной окружения `RESOURCE_NAME`.

- Если и переменная окружения не задана, то значение должно совпадать с именем исполняемого файла, из которого было запущено приложение.

- **Имя класса** (англ.: Class name, также обозначается `res_class`) — имя класса приложения.

Например, для приложения «Терминал» значения будут «`gnome-terminal`», «`Gnome-terminal`»; для браузера Chromium — «`chromium-browser`», «`Chromium-browser`»; для приложения «Блокнот» (запущенного с помощью Wine) — «`notepad.exe`», «`Wine`».

К сожалению не все приложения следуют стандарту и корректно выставляют значения для `WM_CLASS`, окна таких приложений могут быть сгруппированы некорректно, например для интерактивной среды разработки на языке Python SPE, значения будут «`python`», «`Python`», что не позволяет корректно группировать окна, и так же решение следующей задачи становится невозможным.

Со второй задачей все несколько сложнее. Дело в том, что не существует стандартного механизма для идентификации приложения по известным свойствам его окон, также нет стандартных средств для работы с приложениями (запуск нового экземпляра,

получение нормального значка, локализованного имени для приложения, и т. п.). Поэтому приходится искать обходные пути. В окружениях рабочего стола Unix-подобных операционных систем для запуска приложений и формирования основного меню используются специальные конфигурационные файлы, описанные стандартом «Desktop Entry Specification». Такие файлы имеют расширение `.desktop` и содержат следующие сведения о приложении:

- **Name** — имя приложения, например «Firefox».

- **GenericName** — общее имя приложения, например «Веб браузер».

- **Comment** — комментарий, всплывающая подсказка, появляющаяся при наведении указателя мыши на пункт соответствующий приложению в главном меню.

- **Icon** — значок (иконка). Указывает название значка для отображения в меню или файловом менеджере.

- **Exec** — командная строка, используемая для запуска приложения.

- **Terminal** — признак того, что приложение должно быть запущено в терминале.

- **StartupWMClass** — свойство `WM_CLASS` которое должно присутствовать хотя бы у одного окна этого приложения.

- Другая информация, которая не имеет отношения к работе DockBar, и я не буду останавливаться на ее рассмотрении.

Решение второй задачи сводится к поиску `.desktop` файла для приложения по известным свойствам его открытых окон. К сожалению, присутствующий в спецификации `.desktop` файлов параметр `StartupWMClass` не является обязательным и не используется, а он бы значительно облегчил задачу. В виду того, что не существует способа для однозначного сопоставления окна (или группы окон) приложения и `.desktop` файла, из которого это приложение было запущено, применяется несколько методов для нахождения `.desktop` файла. В лучшем случае `res_class` в поле свойства `WM_CLASS` окна соответствует названию `.desktop` файла (на этом поиск `.desktop` файла заканчивается), но не для всех приложений это так. Если `.desktop` файл с именем соответствующим `res_class` не был найден, то DockBar пытается найти `.desktop` файл, сравнивая `res_class` и различные ключи известных `.desktop` файлов. К сравниваемым ключам от-

носятся имя приложения и командная строка приложения. Результат поиска может быть некорректным, но такие случаи встречаются не часто.

После того как обе задачи решены корректно и для группы окон однозначно определено приложение, к которой группа принадлежит (найден `.desktop` файл), то становится возможным:

- работать с группой окон,

- запускать новые экземпляры приложения,

- в заголовке группы выводить корректное и локализованное имя,

- отображать значок (иконку) для приложения из текущей темы оформления и нормального разрешения, а не тот, что присутствует в свойствах окна,

- прикреплять приложение к DockBar для быстрого запуска,

- получить доступ к последним открытым файлам (используется в ветке DockBarX).

Поскольку не существует стандартного метода для идентификации приложения, то это получается не всегда. Например, все приложения из состава OpenOffice.org имеют одинаковый `WM_CLASS` и, соответственно группируются вместе. К счастью таких приложений не так уж и много, и в основном DockBar работает корректно.

Как мне кажется, универсальным решением всех проблем DockBar'a и ему подобных приложений (в других доках GNU/Linux тоже встречаются проблемы с идентификацией приложения) стало бы изменение спецификации «Desktop Entry Specification». Например, перевод ключа `StartupWMClass` в список обязательных, или добавление нового ключа, так или иначе позволяющего однозначно идентифицировать приложение по свойствам его окон. Об этом я писал в freedesktop.org, но, к сожалению, они не считают такие изменения нужными.

На данный момент обе ветки (Dockbar и DockBarX) существуют и разрабатываются параллельно, обе доступны для скачивания как на gnome-look.org, так и на Launchpad'e в виде пакетов для Ubuntu. За время существования проекта только с сайта gnome-look.org было произведено более 49000 скачиваний. X ветка обладает некоторыми дополнительными возможностями (поддержка тем, доступ к недавним открытым файлам) и разрабатывается более активно, по всей видимости, она и станет основной в будущем. ■

Ссылки:

DockBar <http://gnome-look.org/content/show.php?content=97822>

DockBarX <http://gnome-look.org/content/show.php/DockbarX?content=101604>

На Launchpad: <https://launchpad.net/dockbar>



OpenSource Java Math Library

Автор

Игорь Сычев
ЦНИТ МГУПИ
Разработчик,
Microsoft Student Partner
г. Москва
sychevigor.habrahabr.ru
E-mail: sychev-igor.90@mail.ru

Изучая математические библиотеки для .Net я заметил, что многие из них — портированные библиотеки с Fortran/C++/Java. Я не являюсь фанатиком какой-либо технологии или платформы. Если есть решение поставленной мне задачи на языке отличном, от того на каком я работаю и нет внешних ограничений, то я буду писать его свою задачу именно на этом языке и с этой библиотекой. В этой статье я хочу рассказать о некоторых библиотеках на JAVA.

Apfloat

(<http://www.apfloat.org>)

Это библиотека для работы с числами с произвольной точностью. Есть две версии библиотеки: C++ и Java. Все бы не плохо — и исходные коды есть, и тестами они покрыты полностью. Документация к библиотеке стандартная для Java, качество документации отличное, только бы еще в коде было больше комментариев. И примеры есть. Основной пример использования: вычисление числа π с произвольной точностью. Мне вот в жизни никак нельзя жить без знания 100 знаков числа π . В Netbeans в режиме отладки результат был такой:

```
3.14159265358979323846264338
327950288419716939937510582097
494459230781640628620899862803
4825342117067
```

```
Final value took 0.024
seconds
```

Более 100 знаков я не считал, хотя в gui версии по default указано 100 тысяч знаков, но gui версия честно падает, а заниматься отладкой интерфейса мне было не интересно. Версия для C++ правда меня шокирует. Во-первых, последнее обновление: April 24th 2006, то есть 4 года библиотека не развивается вообще. Естественно, что библиотека устоявшаяся, но это значит, что четыре года разработчики ее уже не поддерживают.

О самом коде указано, что в нем есть оптимизирующие ассемблерные вставки, но вот проблема — есть версии сборки только под (стыдно сказать) reptum4. И даже есть сборки для не Windows а именно для MSDOS. Есть сбор-

ки под Linux, но не известно под какой. Я сам C++ версию не использовал, но на сайте указано, что возможность считать с точностью «Maximum number of digits: 226 million».

ArciMath BigDecimal

(<http://users.belgacombusiness.net/arci>)

Название говорит само за себя. Библиотека для операции с большими числами. Указано, что она может работать с числами в 999999999 знаков. Но и слово АРХИ тоже очень актуально. Последнее обновление для jdk 1.3 это 2002 год. Действительно архи-старая библиотека, даже есть мнение, что разработчики ее давно бросили. Исходных кодов нет. Ах да. Еще могу сказать, что разработчики этого проекта юмористы, у них почта на домене skynet.be.

COLT

(<http://acs.lbl.gov/~hoschek/colt/>)

Как и все проекты, связанные с математикой он очень давно не обновлялся. Последнее обновление указано на 2005 год. Но пусть ни кого это не пугает, проект использовался в CERN и писался для физиков (Большой Адронный Коллайдер — их детище. Если появится черная дыра, то это скорее всего Java выбросила Exception).

Что она умеет:

- ✓ Различные матричные операции (в том числе матрицы можно использовать разреженные).
- ✓ Различные статистические распределения.
- ✓ Генераторы случайных чисел.
- ✓ Некоторые спецколлекции (не очень понимаю, зачем реализовывать свои List и Map? Хотя может они в итоге в jdk вошли, сколько лет-то прошло).
- ✓ А главной фишкой является пакет с возможностью делать ГИСТОГРАММЫ.

Документация у проекта отличная, комментариев в коде более чем достаточно. Тестов нет совершенно, зато есть некоторые бенчмарки, находящиеся непосредственно в своих пакетах. Для изучения можно использовать их

как точки входа.

Commons-Math

(<http://commons.apache.org/math>)

Вот эта библиотека мне понравилась сразу двумя вещами. Ее поддерживает Apache Foundation, следовательно, мало вероятно, что проект умрет, и в исходных кодах последние обновления были в 2009 году (из тех, что видел я), а в репозиторий последний комит был уж совсем недавно.

Что эта библиотека умеет:

- ✓ Статистика (Регрессия, распределения).
- ✓ Генераторы случайных чисел.
- ✓ Линейная алгебра.
- ✓ Интеграция, интерполяция.
- ✓ Комплексные числа.
- ✓ Преобразование Фурье и другие преобразования.
- ✓ Геометрия и преобразования примитивов. ▶

Лирическое отступление по поводу вычислений и Европейского Ядерного Центра

На сколько мне известно, от людей из С-Пб, которые имеют отношения к работам в европейском ядерном центре, для обработки данных вместо традиционного вычислительного кластера (с высокоскоростной сетью, с низкой латентностью), можно использовать распределенные вычисления (GRID). Для распределенных вычислений в комментариях была указана библиотека *gridgain cloud computing* (<http://www.gridgain.com/index.html>). В качестве непосредственно счетной части, библиотека использует нативный код на C или Fortran, вызываемый через JNI. Не могу точно сказать, используют ли при обработке данных с БАК именно эту библиотеку, но она каким-то образом связана с их вычислительным центром.

✓ Есть свой фреймворк для организации Генетических алгоритмов.

JMSL Numerical Library

(<http://www.vni.com/products/jmsl/jmsl>)

Обожаю такие библиотеки. В июле месяца я не смог получить даже как указано на сайте: «Новую версию C# библиотеки». А версия для Java это вообще шедевр! Кто поймет, как отправить им эту формочку, тому можно памятник ставить. Можно конечно обратиться в тех поддержку, но раз они за месяц ничего не сделали то, наверное не сильно люди этим интересуются. И библиотека платная.

Drej

(<http://www.gregdennis.com/drej>)

Совсем небольшая библиотека для линейной и не линейной регрессии. Последний релиз 2005 года. Документация есть, но ни тестов, ни примеров использования. Можно конечно поискать, но результатов не так много, раздвигать и обчелся.

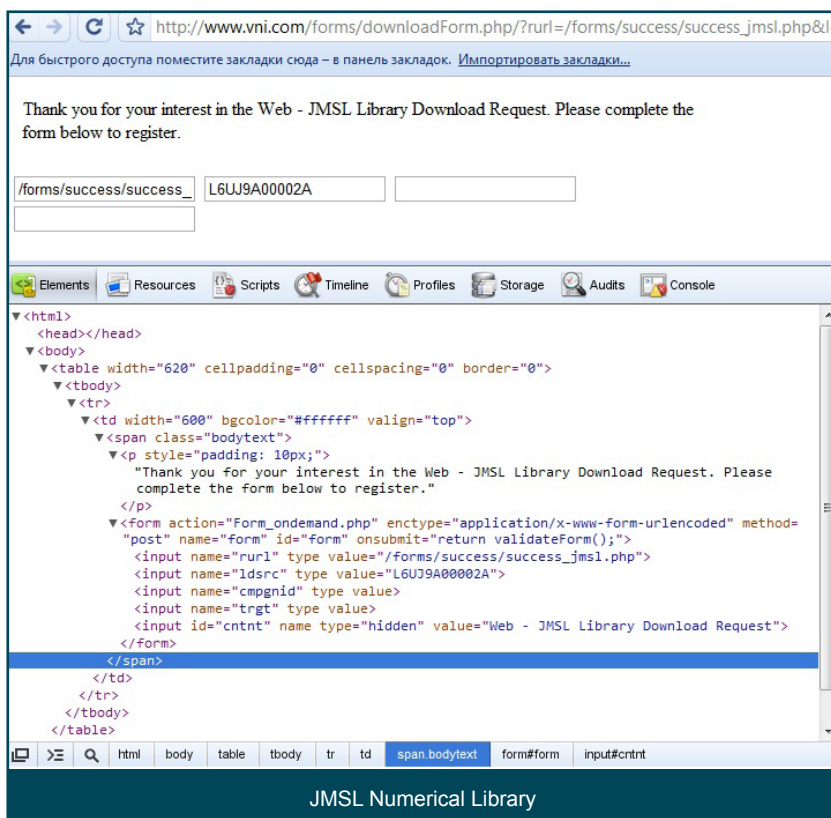
К слову этот проект тесно завязан на другую библиотеку – Java3d (<https://java3d.dev.java.net>). Из названия понятно, что библиотека для 3d графики, но Drej использует из нее только небольшую часть, а именно классы базовых примитивов типа векторов, матриц, точек.

JTransforms

(<http://sites.google.com/site/piotrwendykier/software/jtransforms>)

Библиотека для произведения различных преобразований.

- ✓ Быстрое Фурье преобразование (FFT).
- ✓ Дискретное Fourier преобразование (DFT).
- ✓ Дискретное Cosine преобразование (DCT).



✓ Дискретное Sine преобразование (DST).

✓ Дискретное Hartley преобразование (DHT).

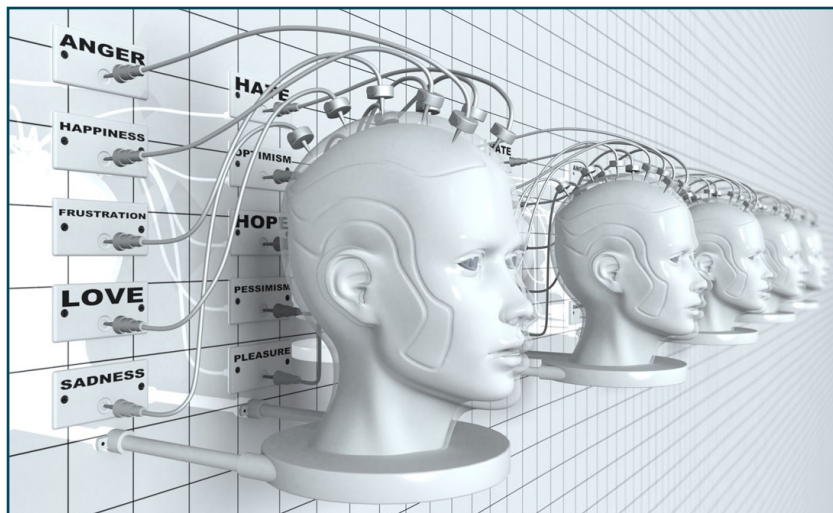
Естественно, что это очень краткий обзор и очень не большой части. Библиотек очень много, наилучший способ изучать – взять и попробовать. Я видел некоторые библиотеки под .Net и их состояние, мне есть с чем сравнивать и я не являюсь врагом Java. Нельзя быть врагом Java и писать на ней. Мне очень нравится, что существует огромная масса библиотек, многие с открытым исходным кодом. Их на порядок больше чем на .Net, разве что C++ библиотек больше чем Java, но нельзя

не заметить, что:

● Большая часть библиотек не поддерживается совершенно. Последние комиты были в 2003-2005 годах. Очень не многие библиотеки поддерживаются нормально, одна из этих немногих Common-Math. Можно конечно говорить, что библиотеки стали настолько хорошими, что им уже просто не нужна поддержка. Но, как известно, что в любой программе есть хотя бы один баг. Не думаю, что в корпоративном сегменте согласятся использовать библиотеку, которая не поддерживается своими создателями.

● По опыту участия в школах и конференциях по параллельным вычислениям и т. п., могу сказать так – на .Net и на C++ сейчас много интересных работ. Для .Net: Провайдеры для вычисления на GPU, различные системы легкого распараллеливания такие, как TPL, но ни одной работы связанной с Java я не увидел. Я думал, что буду белой вороной, когда рассказывал о своей работе, завязанной на .Net (в последствии была сделана Java копия). Только от сотрудника ИСП РАН я слышал про ParJava, но этот проект года так 2002. Неужели все молодые Java разработчики вымерли или перешли в разряд Энтепрайз девелоперов?

● После нескольких неуклюжих шагов Oracle и очередной отсрочкой выхода Java 1.7, что делать тем, кто любит Java? Переходить на .Net или надеяться, что Oracle и Google оживит Java? ■





Автор

Максим Олейник
maxim-oleinik.moikrug.ru
maxim.oleinik@gmail.com

Почему Git

Было время, когда я ничего не знал про VCS, ни что это такое, ни тем более, зачем это мне. И верхом своих достижений считал папку с архивами версий. К моменту осознания необходимости системы контроля версий, я уже набил шишек и прочувствовал нужность такого инструмента. Но борландовский аналог CVS меня не впечатлил. У каждого файла свой номер версии, как мне получить срез определенного релиза я так и не разобрался. А в это время SVN победоносно шла сквозь умы разработчиков. Это было то, чего мне так не хватало, прочитав доку и начав работать, я просто влюбился в нее. Да, были трудности и определенные неудобства, но куда без них.

Так я и работал бы в SVN, но ничего не стоит на месте. В интернете уже потекли тонкие ручейки новостей про Git. Я не кидаюсь за каждой новой технологией, и прошло уже достаточно много времени, пока мне не прожужжали этим Git'ом все мозги. Мне стало любопытно, я вначале присматривался, примерялся, а потом плюнул и начал новый проект на Git. Мучился с ребятами две недели, накачал литературы, написал шпаргалку... ничего, привыкли, а потом меня прорвало. Теперь регулярно просят рассказать про Git и что в нем такого. Поскольку мне уже надоело рассказывать, то эта статья для тех, кто еще сомневается.

Все локально

Отсюда вытекает два важных следствия: все очень быстро, и второе – вы получаете абсолютный контроль над репозиторием.

- Создать репозиторий в текущей директории – «git init». Все.
- Всего одна папочка .git и никаких заморочек в виде .svn в каждой директории.
- Интернет не нужен. Работать одному вообще шикарно – локальный репо-

зиторий разворачивается за доли секунды. Работа в распределенной команде – у каждого своя копия репозитория, никто не зависит от доступности центрального сервера.

- Не надо задумываться о создании системы резервного копирования. У каждого участника есть копия, из которой можно поднять оригинал (поднимите руки те, кто поднимал потерянный svn-репо из своей копии git-репозитория).
- В своем репозитории вы делаете все, что хотите. Любые ветки, и никто их не увидит.

Контроль

В Git можно делать с коммитами и историей все что угодно:

- Удалить коммит, как будто его и не было.
- Изменить комментарий коммита.
- Поменять коммиты местами.
- Объединить несколько коммитов в один, особенно когда нужно «докоммитить» что-то забытое.
- Разбить коммит на несколько.

Ветки:

- Перенести/скопировать коммит из одной ветки в другую.
- Переместить ветку в любую точку.

Очень хорошо это описано [здесь](#).

От таких возможностей буквально «сносит голову» и кому-то это может показаться слишком опасным. Да, пару раз я терял коммиты из-за своих неопытных манипуляций и потом рылся в «корзине», чтобы их восстановить. Но по мере того, как привыкаешь и понимаешь что и как работает, вырастают крылья, и работа происходит намного эффективнее, доставляя попутно кучу удовольствия.

Ветки

Git – это ветки. Это настолько просто, гибко и удобно, что все делается в ветках. От релиза, до маленьких задач.

- Создать ветку, переключиться в нее, объединить ветки, вернуться назад –

это рутинные операции.

- Мерджить одно удовольствие. Есть конфликты или нет, все проходит очень легко и никогда ничего не теряется.
- За исключением релизных, большинство веток очень короткие и живут 1-3 дня.

Коммит

Чтобы сделать коммит, надо указать какие именно изменения нужно туда положить. Изменения, а не файлы. «git add» надо вызывать как для измененных, так и для новых файлов. Таким образом, подготавливается временное состояние, которое я называю «временный коммит» (в оригинале «staging area» или «index»).

В чем преимущество:

- Я всегда могу посмотреть, что буду коммитить, а что не попадет в коммит.
- Поскольку git принимает «изменения», я могу положить в коммит только часть изменений в одном файле, а другую часть в этом же файле откатить или положить в другой коммит.

Последнее время я использую только «git add -p» – Git показывает чанк за чанком (набор изменений в рамках файла) и спрашивает добавить в коммит или нет. Так я вижу, что я изменил и буду коммитить. Поэтому в коммит у меня никогда не попадут отладочный код, комментарии, лишние правки или пробелы, и коммиты получаются четкими и атомарными.

Диф и лог

- Я могу посмотреть диф всей ветки или группы коммитов, чтобы не листать по коммиту, особенно, если они неграмотно оформлены.
- Могу посмотреть разницу между двумя ветками, посмотреть какими коммитами они отличаются.
- Посмотреть диф без учета пробелов (git diff -b -w), если кто-то увлекся форматированием и превратил диф в одну сплошную кашу. ►

● Могу в конфиге настроить кучу алиасов, которые будут выводить логи коммитов в разных форматах, например, сформировать change-log.

Лог — это вообще бездна возможностей.

Stash

Я работаю над задачей, разобрал проект на части и тут мне надо все бросить, переключиться на другую ветку, чтобы пофиксить баг или провести ревью. Я сохраняю свои правки во временном состоянии (stash), переключаюсь, делаю все, что мне надо, возвращаюсь обратно, восстанавливаю свои правки и продолжаю работать.

Bisect

Команда для поиска коммита, в котором была допущена ошибка. Я указываю коммит, в котором точно знаю, что этой ошибки нет, и коммит, где эта ошибка уже есть. Далее Git бинарным поиском выбирает коммит для проверки, переключается на него и предлагает ответить, есть здесь ошибка или нет. И так до тех пор, пока не будет найден ошибочный коммит. Если есть возможность, то можно написать скрипт, который будет автоматически проверять наличие ошибки и возвращать соответствующий код. Отдать этот скрипт в bisect и получить битую правку.

git-svn - для тех, кто в подполье

Вы можете локально работать в Git и коммитить при этом в SVN. Никто даже не догадается, если вы сами не проколется по своей счастливой физиономии. Вы можете легко объединиться со своими коллегами подпольщиками и делиться ветками и правками из своих репозиториях.

Субмодули

Git позволяет включать в проект другие проекты. Подключаются они как субмодули:

```
$ git submodule add git://github.com/maxim-oleinik/sfDoctrine2Plugin.git plugins/sfDoctrine2Plugin
```

Git создаст директорию «plugins/sfDoctrine2Plugin» и развернет там еще один репозиторий, а под контроль поставит хеш ревизии. Если зайти в эту директорию, то мы попадем уже в другой проект. Там можно работать и

коммитить как обычно, только в родительском проекте Git покажет, что подпроект обновился и предложит сохранить ссылку на новый коммит.

● Удобно работать в основном проекте и одновременно коммитить в разные субмодули.

● В Git нельзя подключить указанную директорию из другого проекта, как это делает svn:externals. Только весь проект целиком.

В том и суть, чтобы выделять самостоятельные модули и оформлять их в отдельные репозитории. Правда, становится немного неудобно работать, когда у субмодулей есть свои субмодули и т. д.

GUI и консоль

Я, честно говоря, не знаю, какой есть GUI у Git. Я никогда им не пользовался и рекомендую не лишать себя удовольствия работы с Git'ом в консоли. Не превращайтесь в мартышек, которые ищут нужную кнопку или галочку в интерфейсе, вместо того, чтобы просто набрать команду в консоли.

Ладно, без обид. Я просто хотел сказать, что работа в консоли намного богаче и удобнее.

У кого нет нормальной консоли... ну, тогда выбирайте GUI, IDE или... OS, у которой эта консоль есть.

Единственным окошком, которым я регулярно пользуюсь — это «gitk -all &». Он показывает дерево коммитов с ветками и тегами. Очень удобно смотреть, где ты сейчас находишься, где какие ветки и как они переплетаются. Там есть простой поиск и просмотр дифа.

Работа в команде

● В Git нельзя закрыть доступ к определенной ветке или модулю. Или все, или ничего. Но это еще ни разу не создало мне проблем. Если кто-то ошибся и закоммитил не туда, куда ему следовало (как мы заранее договорились), тогда я просто откачу эти правки так, как будто их и никогда и не было, и проведу разъяснительную беседу.

Как вариант, можно выделить отдельный репозиторий с ограниченным доступом, и рабочий — для всех.

● Все задачи делаются атомарно в ветках и в таком же виде передаются на ревью и мердж. Т. е. при желании, ни один коммит не попадет в основной ствол без ревью.

● Очень удобно проводить ревью задачи, которая выполнена в отдельной

ветке. Посмотреть полный диф всех коммитов, что-то исправить, прокомментировать, отправить обратно на доработку, а потом объединить отдельные коммиты и слить в основную ветку.

GitHub

Это классная платформа, где можно бесплатно разместить и опубликовать свой репозиторий. Где живет куча проектов, которые форкаются, развиваются и обсуждаются. Сейчас там лежат все мои проекты.

Бесплатно для открытых проектов и мин. 200 руб/мес, чтобы закрыть доступ.

Это конечно, далеко не все, что может Git. Есть еще куча всего, чего я не рассказал и чего я до сих пор еще не знаю. Я постарался осветить принципиальные моменты. У Git'a, конечно, есть и определенные минусы и трудности, но без них не бывает. Да они и не так существенны — чаще всего это вопрос git-way. Правда у Git'a есть один серьезный минус — он не для ленивых. Поэтому подумайте заранее, прежде чем предлагать Git таким ребятам.

Что дальше

Дальше начинать работать. Можно читать кучу обзоров и обсуждать сколько угодно, но до тех пор, пока не начнешь работать, ничего не изменится. За один день я перерыл доку и начал работать, две недели привыкал, через месяц учил остальных, через три расправил крылья.

Как начать.

1. Скачать, установить и настроить (я думаю, ни у кого не возникнет проблем).

2. Прочитайте любую книгу из списка указанного ниже. Правда, прочитайте. Хотя по диагонали, зато будете знать, что вообще есть. Это легко сделать за пару часов.

3. И начинайте работать над новым проектом или переносите существующий (импортировать не составит труда).

Есть чит-шит, который поможет с командами на первое время. Есть «git help команда». Рекомендую не лениться и регулярно читать хелп для каждой новой команды, а потом еще читать и перечитывать. Так я узнал большинство возможностей, тонкостей и нюансов. ■

Ссылки

Книга «Pro Git» <http://progit.org/book/>

Книга «Git Community Book» <http://book.git-scm.com/>

Книга «Pragmatic Version Control Using Git» <http://pragprog.com/titles/tsgit/pragmatic-version-control-using-git>

Книга «Git Magic», нашел здесь: <http://habrahabr.ru/blogs/Git/80909/>

Сравнение команд Git-SVN <http://git.or.cz/course/svn.html>

Редактирование истории в git <http://gg.net.ru/2009/12/16/git-history-rewrite/>

Мои правила оформления коммитов и веток: <http://github.com/maxim-oleinik/symfony-dev-rules/blob/master/git.txt>

Мой cheat-sheet: <http://github.com/maxim-oleinik/symfony-dev-rules/blob/master/git-cheat-sheet.txt>

Тест популярных браузеров на совместимость с HTML5

Автор

Станислав Горнаков
Microsoft MVP
www.gornakov.ru

Прочитав очередную статью о его величестве HTML5, решил протестировать текущие версии известных и наиболее широко распространенных браузеров. В обиход тестируемых браузеров попали следующие монстры:

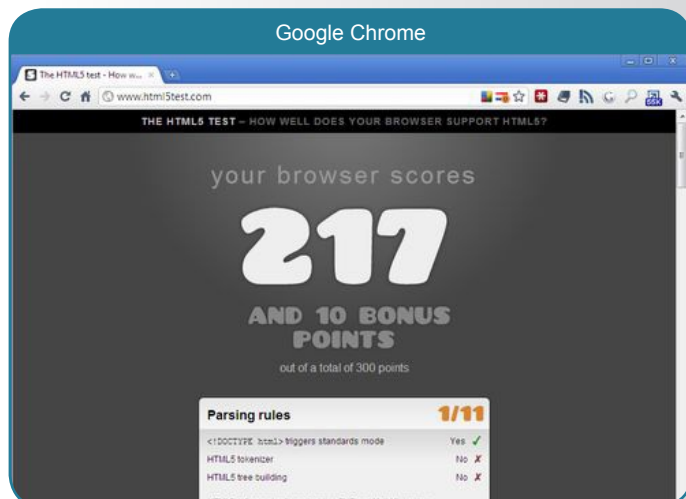
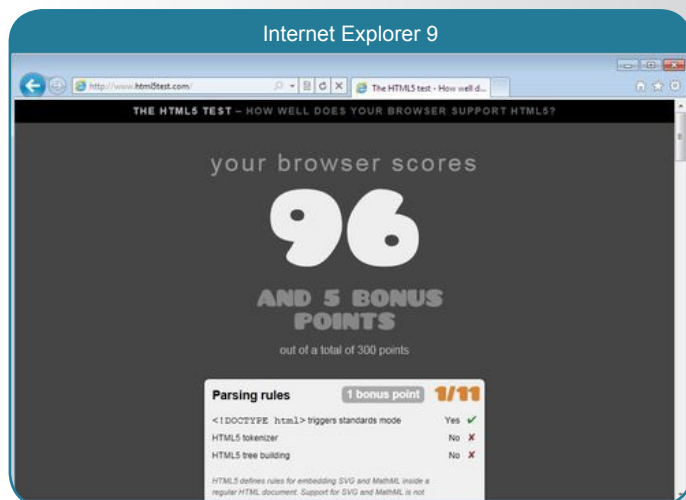
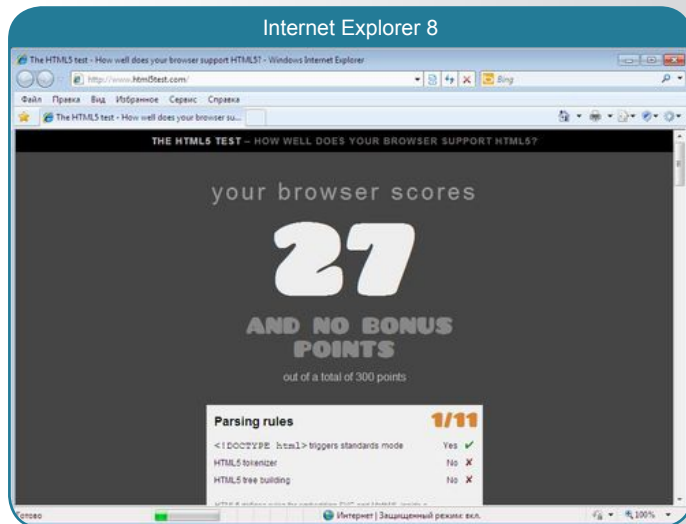
- IE8 в сборке 8.0.7600.16385
- IE9 (beta) 32-х разрядная версия для Windows 7 в сборке 9.0.7930.16406
- Google Chrome версии 6.0.472.63
- Firefox в сборке 3.6.10
- Opera 10.62
- Safari в версии 5.0.2

Замечу, что все вышеперечисленные браузеры на текущий момент (6 октября 2010 г.) были доступны именно в этих версиях, то есть все версии — это самые последние сборки на данный момент. В качестве операционной системы использовалась Windows 7, которая установлена на моем рабочем ноутбуке. Такой себе старенький Acer с 2 Гб памяти, мобильным Intel Celeron M 370 (1.5 ГГц) и встроенным видео на чипсете Intel 915 GM.

В качестве тестовой площадки был выбран сайт www.html5test.com. Как мне показалось это наиболее независимый на данный момент сайт с тестом HTML5. Другие аналогичные проекты как-то уж больно однобоко принимают ту или иную сторону разработчиков. Сама же суть теста заключается в простом для пользователя действии. Необходимо зайти на сайт в одном из браузеров и дождаться итогов теста, который автоматически отобразится на экране через несколько секунд.

Итоговый результат измеряется в очках, с максимальным значением равным трем сотням каких-то там своих попугаев. Эти самые попугаи — то есть, очки складываются из целого набора более мелких тестов, сумма которых и выдает конечный результат. Все промежуточные результаты также будут доступны на экране в браузере, и вы сможете проанализировать полученные результаты тут же.

Первыми в бой пошли продукты от Microsoft IE8 и IE9 (beta). Кстати, если вы собираетесь установить IE9 (beta) то помните, что он устанавливается прямо поверх IE8, то есть после установки IE9 предыдущая версия браузера Internet Explorer вам будет не до- ➤



ступна. Полученные результаты, мягко говоря, сильно удивили. Браузер IE8 выдал 27 очков, ну тут ладно, а вот новый IE9 дал всего 96 очков. Напомню, максимальный результат в тесте равен 300 очкам. По результатам эти два браузера заняли два последних места. Ожидал, честно говоря, большего.

После этих двух тестов сразу же возникло желание опробовать браузер Google Chrome, которым я пользуюсь по умолчанию и отдаю предпочтение в серфинге по Интернету. Как не крути, а скорость и адекватность работы этого браузера на высоте. Итоговые результаты подтвердили мои впечатления, тест выдал 217 очков и это самый лучший результат на текущий момент.

Далее в дело пошли три оставшихся браузера, Safari от Apple, Opera, с которого я слез пару лет назад и Firefox, от которого я отказался этой зимой. К слову Firefox использую и до сих пор. Этот браузер с полностью отключенными надстройками, дополнениями и другими «опасными» фишками служит мне для входа в административную часть моих сайтов, в купе с аппаратным USB ключом Aladdin eToken PASS, благо СМС 1С-Битрикс (а взрослые дяди выбирают тяжелые, суровые и платные решения) дает такую возможность. Ни с каких других браузеров я туда не хожу, равно, как и не пользуюсь этим браузером в Интернете. Безопасность и осторожность еще никому не помешала. Так вот Opera показала 159 очков, Firefox 139 очков, а Safari дал удивительные 207 попугаев. Примечательно, что на сайтах этих браузеров говорится, что это самые быстрые браузеры в мире. В конечном счете после теста мы имеем следующие результаты.

Итоговая таблица

IE9 (beta) – 27 очков

IE8 – 96 очков

Firefox – 139 очков

Opera – 159 очков

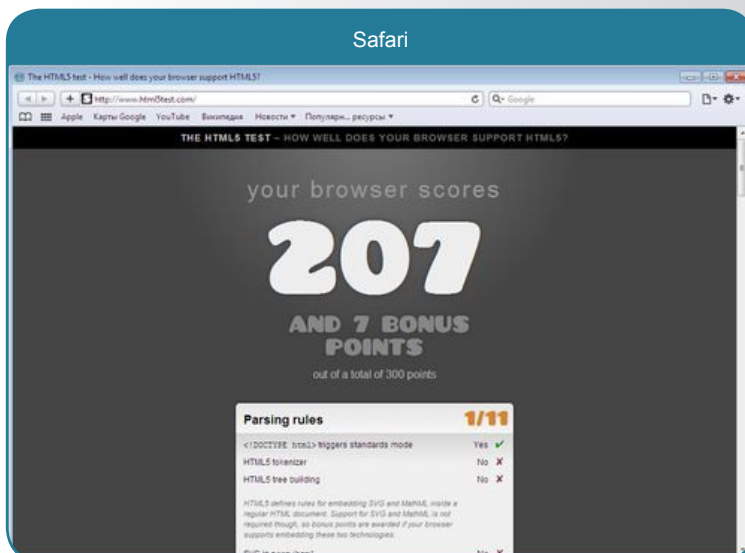
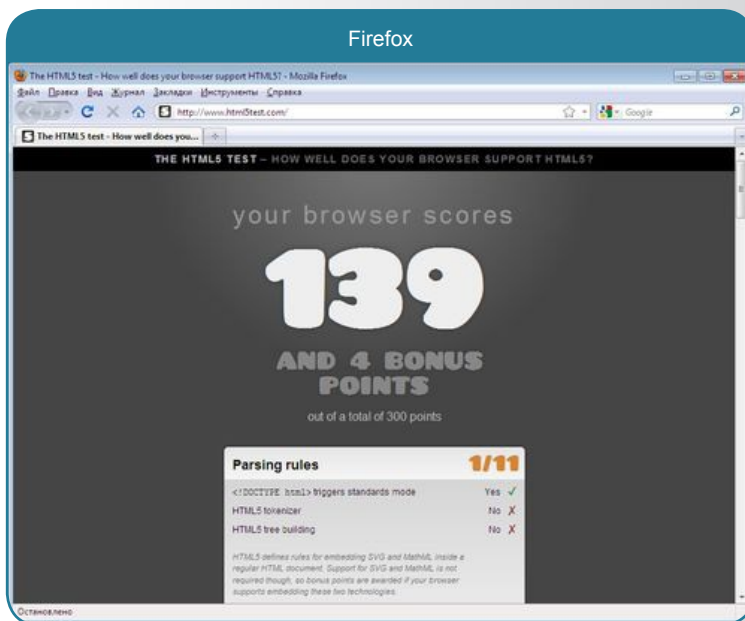
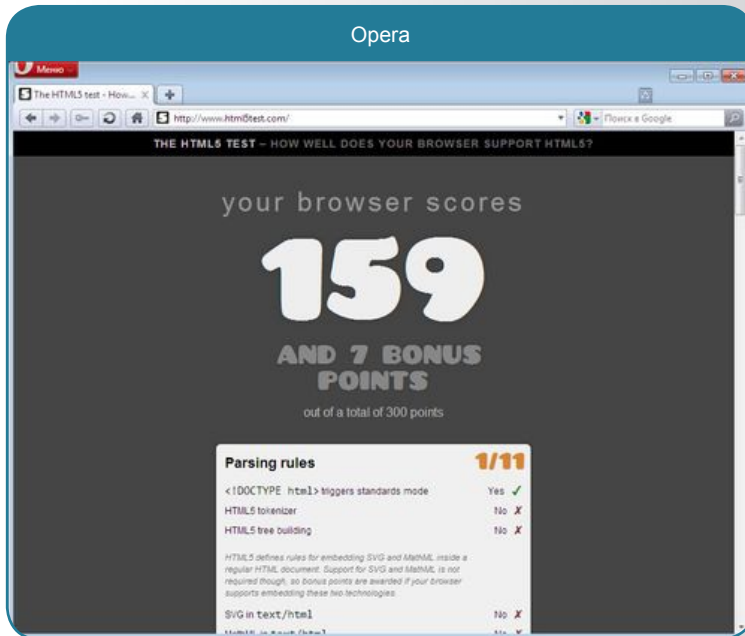
Safari – 207 очков

Chrome – 217 очков

И напоследок еще немного цифр. На днях в мировых СМИ проскочили независимые результаты о распределении долей рынка среди браузеров во всем мире, которые выглядят следующим образом.

1. IE - 49,87%
2. Firefox - 31,5%
3. Google Chrome - 11,54%
4. Safari - 4,42%
5. Opera - 2,03%

P.S. если у вас есть желание высказаться по теме, то жду вас у себя в блоге на www.gornakov.ru. ■



FastVPS.ru

VPS от 129 рублей

- 100 MB RAM
- 300 MHz CPU
- 6 GB HDD

**Dedicated от
2199 рублей**

- Intel® Core™ i7-920
Quad-Core incl
- 8 GB DDR3 RAM
- 2 x 750 GB HDD



- *один из лучших
дата-центров Германии*
- *служба поддержки
на русском языке*
- *уникальная система
восстановления*
- *удаленный reboot/reinstall*
- *установка сервера
в течение 24 часов*