Плоха Windows 8? Есть Mint 14 KDE

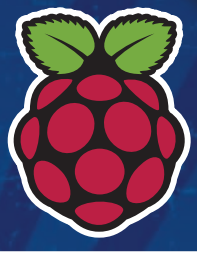
НА ВАШЕМ БЕСПЛАТНОМ DVD

Fuduntu: имя смешное, система серьезная

Плюс: GHostBSD, Netrunner и более того!

# LINUX FORMAT

Главное в мире Linux



**Raspberry Pi:**

» Scratch: язык программирования для детей с. 62

Март 2013 № 3 (168)

# Linux vs Windows 8

## LINUX ПОБИВАЕТ ПОСЛЕДНИЙ ПЛОД MICROSOFT

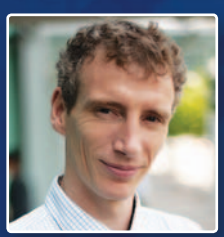
» Толковый интерфейс! » Нету нудных недочетов!
» Файлы найти легко! » Незачем использовать Skype!

**X System** Этот загадочный Икс
**OpenSUSE** Релиз покатится, покатится...
**Django** Web-приложения для ~~ленивых~~ умных
**Erlang** Отсылаем задачи процессам

**ПЛЮС!** Параллельные вычисления с. 88

### Умники на пленке

» Как закрасить бывших супругов на отпускных фото с. 20



### От GIMP к MeeGo

“В мире СПО мы привыкли считать: все то, что мы делаем — ново”

Дейв Нири — о Gnome, переменах и Библии с. 34

### Также в номере...

**UEFI как оно есть**
Грузите свое железо с оглядкой на Microsoft и без нее с. 42

**Diaspora**
Исследуйте мир новых возможностей и приключений с. 14

**KDE 4: недостающее звено**
Курочим монстра, для забавы и работы с. 38

4 607130 825167 13003

Безопасность
**Shellinabox**
» Безопасное удаленное подключение

Сети
**Понять пакеты**
» Разберитесь, из чего строятся сети

Виртуализация
**VirtualBox**
» Прививка старого ПК к молодому стволу

ПОДПИСНЫЕ ИНДЕКСЫ В КАТАЛОГАХ
Агентство «Роспечать» – 36343,
«Почта России» – 11932, «Пресса России» – 90959

Российский Интернет Форум

# РИФ+КИБ 2013

# #RIF2013 ПРИБЫВАЕТ ПО РАСПИСАНИЮ!

17-19 АПРЕЛЯ | ПАНСИОНАТ «ПОЛЯНЫ»

**WWW.RIF.RU**

LINUX FORMAT
Главное в мире Linux

# Что мы делаем


» Мы поддерживаем открытое сообщество, предоставляя источник информации и площадку для обмена мнениями.

» Мы помогаем всем читателям получить от Linux максимум пользы, публикуя статьи в разделе «Учебники»: здесь каждый найдет что-то по своему вкусу.

» Мы выпускаем весь код, появляющийся на страницах раздела «Учебники», по лицензии GNU GPLv3.

» Мы стремимся предоставлять точные, актуальные и непредвзятые сведения обо всем, что касается Linux и свободного ПО.

# Кто мы

Linux всерьез становится привлекательной платформой для игр. А какую новую игру изобрели бы вы?

**Гэри Уокер**
*Симулятор журнала 2013*. «Повысить цену! Уменьшить полосность! Подлить пивка!»

**Эндрю Грегори**
*Mega Landlord*. Вы сдаете бедолагам обшарпанную однокомнатную конуру и пытаетесь содрать с них по максимуму.

**Эфраин Эрнандес-Мендоса**
Мексиканскую рукопашную. А главного героя назвал бы «Чили Бандидо».

**Бен Зверард**
Нечто вроде *BSD Robots*, но без всей этой затейливой графики.

**Маянк Шарма**
*Frogger*, но только с Туксом, который гуляет по закоулкам Мира ПК, побивая логотипы Apple и Windows.

**Джонатан Робертс**
Стратегия реального времени: заполонить Девоншир ядовитыми змеями и грядками с луком.

**Майк Сондерс**
Я бы воссоздал *Mario Kart* для Nindendo DS. Но – никаких красных ракушек!

**Валентин Синицын**
Игры? А разве их не было? Я уверен, что если хорошенько покопаться в недрах *Emacs*...

**Ник Вейч**
Стрелялка от первого лица, где я – в смысле, главный герой – уничтожает пользователей Facebook.

**Сюзан Линтон**
*Дистро-Войны*: эпическая битва за последний уцелевший ПК в пост-апокалиптическом мире.

**Шашанк Шарма**
Обезьяний теннис.

**Нейл Ботвик**
Полную трилогию Майка Сингл-тона «Властители полуночи».

# Проблема интерфейса

» Многие статьи этого номера так или иначе касаются графического интерфейса Linux – истории его создания и наиболее современных реализаций. Во многих случаях именно интерфейс, то есть удобство работы, является для пользователя решающим (а то и единственным) аргументом при выборе системы. Так, при сравнении Ubuntu 12.10 с Windows 8 главные претензии к последней связаны именно с «революционным» интерфейсом Metro, пришедшим с мобильных устройств.

Многолетние наблюдения показывают, что понятия об удобстве у двух произвольно взятых пользователей не совпадают никогда, а у каждого по отдельности существенно меняются с годами. Скажем, алфавитно-цифровой терминал EC 7920 (отечественный аналог IBM 3270) с интерфейсом 25 строк по 80 символов (для удобства – как на перфокарте) кажется ныне столь же архаичным, как телефон с диском, а ведь не прошло и четверти века. Сейчас наблюдается очередная революция: «классический» десктоп с клавиатурой и мышью уступает место планшетам, смартфонам и другим устройствам, на которых традиционный «оконный» интерфейс уже не удобен. Unity и Metro – попытки найти решение-компромисс для «железа» обоих поколений. И не факт, что удачные. А на подходе уже новое поколение устройств – еще более миниатюрных и мобильных. Пример – очки от Google. Там с интерфейсом управления вообще беда.

По данным новостных лент, работы ведутся во всех направлениях: управление голосом, мимикой, жестами; анализ энцефалограмм и томограмм мозга. Только пока выходит дорого или неудобно. А чаще – и то, и другое. Что ж, подождем...

**Кирилл Степанов**
Главный редактор
» info@linuxformat.ru

## Как с нами связаться

Письма для публикации: letters@linuxformat.ru
Подписка и предыдущие номера: subscribe@linuxformat.ru
Техническая поддержка: answers@linuxformat.ru
Проблемы с дисками: disks@linuxformat.ru
Общие вопросы: info@linuxformat.ru
Вопросы распространения: sales@linuxformat.ru
Web-сайт: www.linuxformat.ru
› Адрес редакции: Россия, Санкт-Петербург, Лиговский пр., 50, корп. 15
› Телефон редакции: (812) 309-06-86. Дополнительная информация на с. 112

LINUX FORMAT

# Содержание

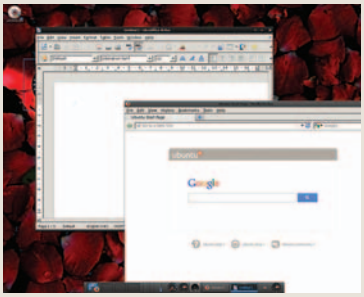
Интернет обезлюдил улицы! Давайте прогоним Интернет!

## Обзоры



› ***Звездный крейсер «Галактика»*: явно не *Star Wars* для бедных.**





› **12 лет и цистерны кофе спустя, вот он — стабильный релиз.**

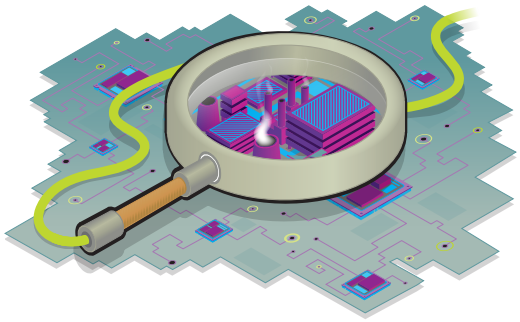


› **Сердце вашей армии роботов с нетерпением ждет паяльника.**

# Linux vs Windows 8

Платить за просмотр DVD? Мы так не думаем с. 26

Сравнение: Фоторедакторы с. 20



Что за штука — System on a Chip? с. 54

## Люди говорят

> Говоря по-простому, с MeeGo случилось то, что Nokia потеряла веру в проект

**Дейв Нири** поясняет таинства мобильников с. 34

# На вашем бесплатном DVD


## Linux Mint 14 KDE

» Просто работает — с кодеками и прочим

## Netrunner 12.12

» Комбинация KDE и свободы


**ПЛЮС:** Главное, документы и прочее с. 106

## Ищите в этом номере




## Пропустили номер?

Узнайте на с. 108, как получить его прямо сейчас!


## Учебники



## Постоянные рубрики

ЗАСЛУЖЕННОЕ ПРИЗНАНИЕ

# Лавры российской компании

## Свободный проект ScratchDuino получил премию Google RISE Award 2013.

» Рубрику готовил АРТЕМ ЗОРИН

Команда ЗАО «Тырнет» (группа компаний «Мезон.Ру») стала одной из 30 в мире, кто получит грант Google RISE Award 2013 на поддержку образовательных инициатив в 2013 г. RISE (Roots in Science and Engineering – «Основы науки и техники») финансирует и поддерживает организации по всему миру, занимающиеся образованием в области науки и техники. Грант употребят для разработки англоязычной версии продукта и для его продвижения в мировое образовательное сообщество вне России.

Проект ScratchDuino был задуман как средство взаимодействия физической среды с визуальной средой программирования Scratch, развивающей идеи Лого, разработанной в Массачусетском технологическом институте (MIT).

Конструктор «СкретчДуино.Робоплатформа», часть проекта ScratchDuino, нацелен на обучение школьников основам программирования, используя данные, получаемые непосредственно из окружающей среды. Набор законченных модулей позволяет создать роботизированный механизм, передающий информацию о внешних воздействиях в компьютер, и посредством созданных школьниками программ решать поставленные задачи. С его помощью можно освоить работу с основными электронными компонентами, разобраться, как работает тот или иной датчик, произвести его калибровку и настройку, затем применить эти данные в программе. Робоплатформа может управляться из *Scratch*, из *Lazarus* (на языке Pascal), из *Кумир* (через транслятор) или просто с пульта управления (к примеру, с вашего Android-смартфона).

Робоплатформа полностью свободна и открыта. Схемы, чертежи, список деталей, инструкции можно скачать с сайта проекта и собрать себе своего робота, или купить детали для его сборки и получить готового робота, который сможет ездить уже через час.

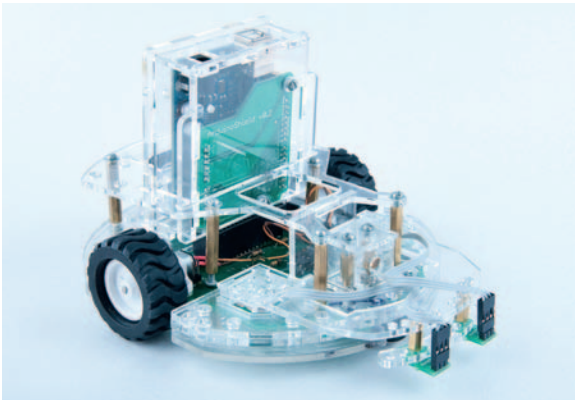
За управление и взаимодействие с реальным миром отвечает Arduino – плата с микроконтроллером AVR, очень популярная у «самоделкиных». «Изначально идея состояла в том, чтобы дать детям в школах поработать с Arduino, но потом мы поняли, что микроконтроллерный модуль там не выживет – его могут уронить, облить водой или положить в груду скрепок, – говорит генеральный директор "Мезон.Ру" Павел Фролов. – Плату нужно упаковать в некую защитную среду, чтобы ребенок ее нечаянно не сломал». Поэтому Arduino и специальную плату расширения (т. н. shield) для подключения внешних устройств убрали в прочный картридж из прозрачного оргстекла.

Ранее проект ScratchDuino получал поддержку Фонда содействия развитию малых предприятий в научно-технической сфере (Фонд Бортника).

Компания ЗАО «Тырнет», начавшая с детского портала http://tirnet.ru, созданного для детей младшего возраста и подростков, а также их родителей, активно занимается проблемой качества компьютерных игр и сайтов для детей. Портал является призером конкурса «Продукт года-2007» в номинациях «ПО для образования» и «Лучший детский сайт»; на нем есть возможность не только играть, но и создавать компьютерные игры, предназначенные для самых маленьких пользователей.

> «Идея состояла в том, чтобы дать детям поработать с Arduino.»

Платформа ScratchDuino была разработана с целью познакомить молодое поколение с возможностями программирования и получить опыт в создании программ, решая конкретные задачи школьного курса с использованием внешних данных. Устройства ScratchDuino протестированы в российских школах и получили хорошие отзывы. Сейчас компания готовится начать их регулярные поставки в образовательные учреждения и собирает предварительные заказы через сайт Linuxcenter.ru. Фролов также отметил высокий интерес, проявленный к ScratchDuino родителями учеников.



» **«СкретчДуино. Робоплатформа» в собранном виде. Прочный корпус печатается на 3D-принтере.**

Главным идеологом и руководителем разработки ScratchDuino является старший научный сотрудник ЗАО «Тырнет» Александр Казанцев – человек, хорошо известный в отечественном сообществе СПО. С его именем связывают инициативу Edumandriva и ряд других свободных проектов. Сейчас он работает над версией робоплатформы для вузов и предприятий.

Стоимость одного комплекта «СкретчДуино.Робоплатформа» – 10000 рублей. Заказать новинку можно уже с 1 марта этого года в магазине Линуксцентра.

Ежегодная программа Google RISE предоставляет гранты в размере $5000–25000. Размер гранта, доставшегося ЗАО «Тырнет», пока неизвестен.

КЛИЕНТСКАЯ И СЕРВЕРНАЯ ОПЕРАЦИОННЫЕ СИСТЕМЫ СО ВСТРОЕННЫМИ СРЕДСТВАМИ ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА К ИНФОРМАЦИИ, НЕ СОДЕРЖАЩЕЙ СВЕДЕНИЯ, СОСТАВЛЯЮЩИЕ ГОСУДАРСТВЕННУЮ ТАЙНУ.

Могут использоваться при создании автоматизированных систем до класса защищённости 1Г включительно и при создании информационных систем персональных данных до 1 класса включительно.


НЦПР

ООО «Национальный центр поддержки и разработки»
125375, Россия, Москва, Тверская ул., д. 7, под. 7, этаж 2
Тел.: +7 495 988 27 09; факс: +7 495 745 40 81
e-mail: office@ncpr.su
www.ncpr.su

ПЕРВОЕ МЕРОПРИЯТИЕ ГОДА

# Подружились в Москве

## Red Hat и «Аксофт» провели совместный пресс-брифинг по вопросам виртуализации.

30 января сего года в Москве в отеле Mariott Courtyard две компании – американская Red Hat и российская «Аксофт» – провели совместный пресс-брифинг, на котором были озвучены планы совместного сотрудничества и рассказано о преимуществах платформы виртуализации Red Hat Enterprise Virtualization 3.1 для предприятий. По словам представителей обоих компаний, версия 3.1 стала важным шагом вперед. Внесенные улучшения делают эту платформу виртуализации еще более масштабируемой, интерфейс управления пользователями и всей системой – более удобным, расширяют ее возможности в области работы с сетями, хранилищами и виртуальными рабочими столами. Новый выпуск корпоративной среды виртуализации Red Hat Enterprise Virtualization позволяет создавать еще более гибкие системы в сочетании с другими продуктами Red Hat, включая хранилище Red Hat Storage и платформы Red Hat Enterprise Linux для предприятий.

Основной темой пресс-брифинга были результаты сотрудничества и дальнейшие планы развития компаний Red Hat и «Аксофт» в России.

Главными докладчиками пресс-брифинга были Михаил Прибочий, генеральный директор «Аксофт», Фил Эндрюс [Phil Andrews], вице-президент Red Hat по Северной и Восточной Европе, Карл Стивенс [Karl Stevens], старший технический специалист EMEA Red Hat.

> «Платформа виртуализации доступна обладателям подписки.»

Среда Red Hat Enterprise Virtualization – единственное комплексное инфраструктурное решение для корпоративных пользователей, доступное сегодня на рынке продуктов на базе открытого исходного кода. Версия Red Hat Enterprise Virtualization 3.0, которая появилась в январе 2012 г., ознаменовала важный этап в развитии отрасли благодаря уникальному набору возможностей и преимуществ для корпоративных пользователей, а также своей высокой производительности и масштабируемости. С появлением версии 3.1 этого продукта обновленный гипервизор *KVM* (*Kernel-based Virtual Machine*) лидирует по 19 показателям производительности из числа 27 опубликованных эталонных показателей SPECvirt_sc2010 (www.spec.org), продемонстрировав лучшие результаты.

В решении Red Hat Enterprise Virtualization 3.1 реализовано несколько важных улучшений. В версии 3.1 этой платформы увеличена масштабируемость гостевых виртуальных машин (поддерживается до 160 логических процессоров и до 2 ТБ памяти на машину), а гипервизор *KVM* способен работать с новейшими моделями процессоров x86. В решении Red Hat Enterprise Virtualization используется та же базовая технология *KVM*, что и в платформе Red Hat Enterprise Linux.

Red Hat Enterprise Virtualization 3.1 обладает более удобным пользовательским интерфейсом, в нем реализован усовершенствованный кросс-платформенный web-портал для администрирования, улучшенная информационная панель с отчетами, новые возможности для работы с сетями и более совершенная подсистема дискового хранилища.

Одним из важнейших улучшений в версии Red Hat Enterprise Virtualization 3.1 является интеграция с масштабируемым открытым программным решением Red Hat Storage для организации хранилища, благодаря чему можно эффективно управлять как структурированными, так и неструктурированными данными на уровне блоков, файлов и объектов.

Платформа виртуализации для предприятий Red Hat Enterprise Virtualization 3.1 доступна обладателям подписки Red Hat во всем мире. Кроме того, на сайте www.redhat.com/promo/rhev3 представлена 60-дневная пробная версия этого выпуска с полной поддержкой.

По словам Поля Кормье [Paul Cormier], президента по вопросам продукции и технологий компании Red Hat, платформа Red Hat Enterprise Virtualization – ключевой элемент семейства продуктов Red Hat, который позволяет предприятиям использовать все преимущества модели разработки на базе открытого исходного кода для внедрения инновационных решений в области виртуализации. Выпуск версии 3.1 является серьезным этапом в развитии этого продукта, поскольку она поддерживает важнейшие функции и возможности интеграции, которые необходимы разработчикам. К примеру, интеграция с платформой Red Hat Storage позволяет предприятиям создавать гибкие виртуальные хранилища, которые пока недоступны в традиционных решениях для виртуализации», а Джерри Хэкетт [Gerry Hackett], вице-президент Dell по вопросам проектирования серверных платформ, сказал: «Мы рады продолжить наше успешное сотрудничество с компанией Red Hat в сфере виртуализации и облачных вычислений. Для пользователей систем Dell, ориентирующихся на решения с открытым исходным кодом, платформа виртуализации Red Hat Enterprise Virtualization является естественным выбором и позволяет нам добиваться исключительной производительности, масштабируемости, гибкости и экономии».


› **Сотрудничество двух компаний станет успешным, уверены руководители компаний.**

Дуг Фишер [Doug Fischer], вице-президент и генеральный менеджер по вопросам системного ПО, Intel отметил, что решение Red Hat Enterprise Virtualization на серверных платформах с процессорами Intel® Xeon® помогает организациям добиться исключительной производительности, которая необходима им для обслуживания важнейших систем и задач».

softline®

Services Software Cloud

# ИТ-архитектура вашего бизнеса

CLOUD

SOLUTIONS

SaaS

SAM

юридическая поддержка

remote workplace

PROJECTS

VIRTUALIZATION

LINUX

программное обеспечение

hardware

legal support

education

ERP&CRM

защита персональных данных

Mirapolis

SERVICES

CAD
CAM
CAE

SOFTWARE

САПР и ГИС

информационная безопасность

SECURITY

учебный центр

техническая поддержка

search

flexible

CONSULTING

решения Microsoft

data leak prevention

корпоративный портал DeskWork

open source

PLATFORM

protection

hosting

technical support

+7 (495) 232-00-23 www.softline.ru info@softline.ru

НОВОВВЕДЕНИЯ КАЖДЫЙ ДЕНЬ

# Дистрибутив и телефон от космонавта

## В Canonical хотят отказаться от LTS-модели и выпустить свои телефоны.

Лиэнн Огасавара [Leann Ogasawara], менеджер команды поддержки ядра в компании Canonical, рассказала в рамках семинара Google Hangouts о рассмотрении возможности перехода к новой модели разработки, при которой классические обособленные выпуски будут формироваться только для LTS-релизов, а вместо промежуточных версий будет доступен непрерывно обновляемый Rolling-репозиторий. Используя данный репозиторий, пользователи смогут установить в LTS-выпуске последние версии программ, не дожидаясь очередного релиза дистрибутива. По словам Лиэнн, Canonical может перейти к новой модели не раньше, чем после выпуска весной следующего года очередного LTS-релиза 14.04.

Итак, предлагается выпускать отдельные релизы Ubuntu раз в два года, а в промежутках между LTS-выпусками прекратить формирование раз в 6 месяцев обособленных релизов. Такой подход позволит сохранить стабильность LTS-выпусков и доступность инноваций промежуточных версий, при этом пользователям не придется ждать отдельных релизов при желании использования новых версий программ, а компания Canonical сможет не тратить лишние ресурсы на поддержку каждого промежуточного выпуска в течение 18 месяцев. LTS-релизы будут формироваться как стабилизированный срез Rolling-репозитория, для которого, по мнению Canonical, можно обеспечить высокий уровень качества и стабильности.

Подобный репозиторий должен быть постоянно в целостном состоянии, все доступные пакеты должны всегда быть работоспособными и сочетаться друг с другом. По словам Лиэнн, это задача большая, но выполнимая. В настоящее время при разработке Ubuntu уже практикуются некоторые методы ежедневного контроля качества, делается автоматизированное тестирования работоспособности сборок.

Тем не менее, обсуждаемое нововведение выглядит нереалистично с учетом возможности негативного влияния на пользователей, которым недостаточно релиза раз в два года и возможности использования rolling-выпусков в остальное время. При rolling-выпусках теряется возможность контроля за появлением инноваций, а пользователь обычного релиза имеет возможность решить, переходить сразу на новый выпуск или подождать какое-то время. Rolling-выпуски непредсказуемы для пользователя, нововведения могут обрушиться в неожиданные и неподходящие моменты, и не всегда могут устраивать пользователя (например, в новой версии программы может быть изменено поведение или появятся регрессивные изменения). Кроме того, невзирая на все усилия по стабилизации и тестированию, rolling-выпуски по своей сути менее стабильны, чем обычные релизы (особенно с учетом того, что многие пользователи не устанавливают релиз сразу, а выжидают примерно месяц, за который успевают устранить вовремя не выявленные ошибки).

Заметим, что подобная инициатива уже высказывалась около полутора лет назад – в октябре 2011 года, но дальше обсуждения дело не пошло.

В интервью журналу CIO Марк Шаттлворт также заявил о намерении уже в октябре 2013 г. начать поставки потребителям первых устройств, оснащенных анонсированной в январе редакцией дистрибутива Ubuntu для смартфонов (Ubuntu Phone OS). В итоге дистрибутив Ubuntu будет доступен для широкого спектра устройств, включая настольные и планшетные ПК и смартфоны, что позволит корпорациям использовать единую платформу на всех типах используемых сотрудниками устройств и обеспечить полноценный доступ к корпоративной информации с одного портативного устройства.

> «В итоге Ubuntu будет доступен для широкого спектра устройств.»

Пока непонятно, будет ли представлен одновременно с телефонами на основе Ubuntu Phone OS и магазин приложений для этой ОС.

В рамках проекта Ubuntu for phones была предпринята предпринята попытка создания новой редакции Ubuntu Linux, специально адаптированной для использования на смартфонах.


**› Пока главный минус всей экосистемы Ubuntu for phones — отсутствие магазина приложений.**

Шаттлворт предполагает занять новой ОС нишу в относительно маломощных устройствах, предоставляя простой и гибкий интерфейс для пользователей.

Хотя Canonical видит свой главный рынок в поставке готовых телефонов с Ubuntu, систему можно будет свободно установить на любое из поддерживаемых устройств. В частности, планируется опубликовать сборку для установки на телефон Galaxy Nexus, используемый как эталонное устройство в процессе разработки. Список вендоров, готовых поставлять устройства с новой ОС, пока не оглашается. Однако у компании есть надежда привлечь крупные компании для производства и выпустить первые устройства на рынок в конце 2013 или начале 2014 г.

Как сообщает Wall Street Journal, основатель Canonical планирует запустить телефон в двух крупных регионах (“two large geographic markets”) в октябре. Ранее сообщалось, что первые устройства должны появиться в начале 2014 г. По мнению многих аналитиков, это слишком поздний срок для вывода продукта на рынок. В этом году Mozilla также представит свою Firefox OS в Южной Африке. Это систему планирует поддержать множество производителей телефонов, в том числе ZTE, а также свыше 7 основных национальных сотовых операторов, которые выпустят устройства на ее базе.

Несмотря на то, что Canonical пока не признается, на какие два рынка они будут нацелены в первую очередь, Шаттлворт бегло отметил, что считает Северную Америку ключевым рынком для Ubuntu.

НОВАЯ МОБИЛЬНАЯ ПЛАТФОРМА

# Первые телефоны на Firefox OS

## Основной рынок Mozilla в данный момент – страны Азии и Африки.

Проект Mozilla представил Keon и Peak, ознакомительные [Developer Preview] модели мобильных телефонов на платформе Firefox OS от проекта Geeksphone совместно с компаний Telefonica. Унифицированный Web API использует стандартные HTML5-технологии, CSS и JavaScript.

Недорогой Keon построен на процессоре Qualcomm Snapdragon S1 1 ГГц, поддерживает UMTS 2100/1900/900 (3G HSPA) и GSM 850/900/1800/1900 (2G EDGE), оснащен 3,5-дюймовым сенсорным экраном (HVGA) с поддержкой мультитач, 4-ГБ Flash, 512-МБ ОЗУ, 3-мегапиксельной камерой, GPS, MicroUSB, MicroSD, Wi-Fi N, датчиками освещенности и приближения, гироскопом, аккумулятором на 1580 мАч. Более продвинутая модель Peak характеристики использует двуядерный процессор Qualcomm Snapdragon S4 1,2 ГГц, и имеет 4,3-дюймовый экран (qHD IPS Multitouch), две камеры (8 и 2 мегапикселей), аккумулятор на 1800 мАч.

Телефоны не привязываются к оператору, поддерживают автоматическую установку обновлений по сети и стоят от $80.

Опробовать написанные для Firefox OS приложения можно на телефоне с платформой Android, установив на нее мобильную версию *Firefox* и приложение *Marketplace for Android*. Не боящиеся трудностей энтузиасты могут собрать Firefox OS для смартфонов, поставляемых с платформой Android, типа Samsung Galaxy S2.

Мобильная платформа Firefox OS базируется на идее использования браузера вместо рабочего стола.

Обновления Firefox OS распространяются по опробованной технологии проекта *Firefox*. Как бесплатные, так и платные приложения будут распространяться через каталог-магазин Mozilla Marketplace.

В состав приложений включены программы из набора web-приложений Gaia: web-браузер, калькулятор, календарь-планировщик, приложение для работы с web-камерой, адресная книга, интерфейс для осуществления телефонных звонков, клиент электронной почты, система поиска, музыкальный плейер, программа для просмотра видео, интерфейс для SMS/MMS, конфигуратор, менеджер фотографий, рабочий стол и менеджер приложений с поддержкой нескольких режимов отображения элементов (cards и grid).


**Peak — «флагманская» модель телефона на основе ОС от Mozilla — стоит очень дешево.**

Приложения для Firefox OS формируются с применением стека HTML5 и расширенного интерфейса Web API, позволяющего организовать доступ приложений к аппаратному обеспечению, телефонии, адресной книге и другим системным функциям. Доступа к реальной файловой системе нет: программы ограничены внутри виртуальной ФС, построенной на Indexed DB API и изолированной от основной системы. Планируется сформировать набор стандартов для создания универсальных мобильных web-приложений, способных обеспечить функциональность, свойственную обособленным мобильным стекам, обычно контролируемым отдельными производителями (Android, iOS, Apple iOS и Windows Phone).

Firefox OS станет «по-настоящему свободной» благодаря использованию открытых технологий CSS, HTML и JavaScript. Среди других достоинств – нетребовательность к техническим характеристикам и то, что программы, разработанные для данной ОС, смогут работать на любом устройстве с современным браузером. Магазин приложений Mozilla Marketplace позволит обогатить состав приложений. LXF

## Новости короткой строкой

Фонд СПО объявил о вводе в строй своего сайта микроблогов status.fsf.org на базе платформы StatusNet.
Источник: www.fsf.org

Компания Valve выдала первый релиз клиента Steam для Linux, отметив это событие большой распродажей Linux-игр, со скидкой 50–80 %.
Источник: store.steampowered.com/news

Вслед за *Team Fortress 2*, *Half-Life* и *Counter-Strike 1.6* началось распространение Linux-версии 3D-стрелялки *Counter-Strike Source* от компании Valve.
Источник: www.phoronix.com

Поступила в продажу модель A одночипового компьютера Raspberry Pi, по цене $25. Производство новой модели организовано на заводе компании Sony в Великобритании.
Источник: www.raspberrypi.org

Алан Кокс [Alan Cox], известный разработчик ядра Linux, лауреат престижной премии Free Software Awards за вклад в разработку ядра, сообщил о своем уходе из Intel и прекращении участия в разработке Linux, по семейным обстоятельствам.
Источник: plus.google.com

Проект OpenPhoenux, развивающий инициативы OpenMoko, покажет на конференции FOSDEM 2013 два новых устройства – планшет и КПК.
Источник: blog.slyon.de/2013/01/21/openphoenux-at-fosdem-2013/

На Kickstarter собирают средства для производства пробной партии портативной игровой консоли GCW-Zero для Linux-дистрибутива OpenDingux, ориентированной на ретро-игры.
Источник: www.kickstarter.com

Начались поставки миниатюрных плат Odroid U2, по размеру близких к Raspberry Pi, но пригодных для роли обычного компьютера. Цена платы $89.
Источник: www.hardkernel.com

Готова первая партия модели телефона OpenMoko GTA04, сменившего Neo FreeRunner. Пока для продажи доступно 10 эксклюзивных устройств, по цене €699.
Источник: permalink.gmane.org

Intel показала на выставке Mobile World Congress новую платформу двуядерных чипов Atom Clover Trail для смартфонов и планшетов на Android.
Источник: www.intel.ru

LINUX FORMAT

# Обзоры

**Новинки программного и аппаратного обеспечения в описании наших экспертов**

**АЛЕКСЕЙ ФЕДОРЧУК**
**Тэг <сарказм> по умолчанию, смайлики по вкусу.**

## Немножко о DragonFly...

Не секрет, что далеко не все линуксоиды в восторге от текущих изменений своей операционки. И потому начинают потихоньку посматривать по сторонам в поисках запасного аэродрома. Одно из привлекающих взгляд направлений – DragonFly BSD. Отпочковавшись от FreeBSD 4-й ветки (см. **LXF157**), она почти 10 лет развивалась самостоятельно и успешно. Однако – не вполне интересно для конечного пользователя.

В частности, из рук вон плохо было в ней с приложениями и управлением оными. В момент зарождения этой ОС ее автор Мэтт Диллон [Matt Dillon] декларировал и свою систему пакетного менеджмента, а в качестве паллиатива предложил порты FreeBSD. Увы, не сложилось: на свою систему просто не хватило сил, а сборка бинарников из Free'шных портов напоминала шитие на живую нитку.

Тогда за следующий паллиатив в DragonFly приняли систему pkgsrc из NetBSD – надежную и безотказную, но с двумя неискоренимыми недостатками: малое количество поддерживаемых приложений и сильное отставание их от апстрима. Что фактически закрывало DragonFly дорогу на десктопы пользователей.

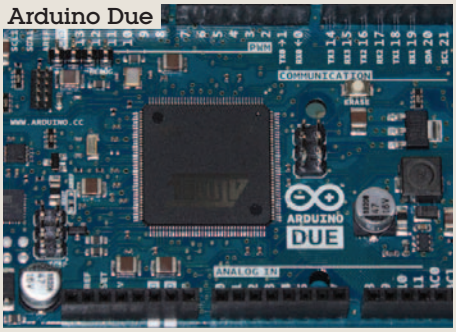
Но колесо фортуны свершило оборот: система управления пакетами в DragonFly – опять вариация на тему портов, dports. Плюс к ней – pkgng из той же FreeBSD, версии 9.1. Нет, не сказать, что все в одночасье похорошело. Но есть шанс, что хорошо таки будет. А у применителей есть повод глянуть на DragonFly в работе. Потому что это хорошая система...

alv@posix.ru

## Сегодня мы рассматриваем:




› **Мрак бездонного Космоса навевает на нас мысли о музыке Эдварда Грига.**




› **Arduino приглашает не дать застояться знаниям, добытым из серии статей в LXF!**

## Сравнение: Фоторедакторы с. 20

**AfterShot Pro**

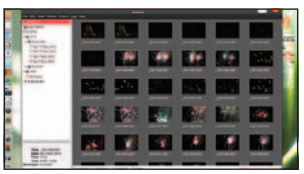
**Darktable**

**DigiKam**

**Fotoxx**

**GIMP**

**Shotwell**



# Diaspora: Shattered Armistice R1

**Алекс Кокс** садится за штурвал звездолета, тут же врезается в другой, подстрелен третьим, а дальше только чертыхается.

## Вкратце

» **Полная модификация движка *Open Freespace 2* по модели вселенной *Battlestar Galactica*. См. также: *The Babylon Project, Wings of Dawn*.**

*Battlestar Galactica* [в российском прокате – «Звездный крейсер “Галактика”»] – это оплот научной фантастики, с глубоким смыслом, пронзающим даже самого недалекого фаната. Это история о сражении людей с человекоподобными роботами. Дело происходит в космосе. Кругом звездолеты и лазеры. Настоящая классика жанра. И пусть у самого ТВ-шоу не всегда все шло гладко – его закрывали в 1979, 1980, 2009 и в середине сезона 2010 – его фанаты и наследие продолжают жить. Ну, а скоротать время до выхода давно обещанной экранизации поможет вот это: верная оригиналу и сделанная с любовью модификация движка *Open FreeSpace2*, резко вбрасывающая вас в разгар битвы между Двенадцатью колониями и подлыми Сайлонами.

› **Планеты в *Diaspora* — всего лишь фон, но они придают эпический масштаб вашим баталиям.**

> «Скоротать время до обещанной экранизации поможет вот это.»

Прежде чем перейти к игре *Diaspora: Shattered Armistice* [Рухнувшее перемирие], давайте сначала избавимся от подкатившей было к горлу паники. Да, здесь замешан *FreeSpace2*. Нет, он не ощущается как технология 10-летней давности, хотя по сути и является таковой.

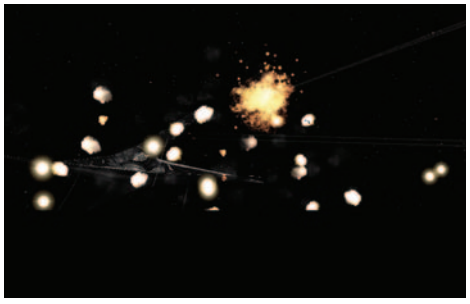
В целом, графика в *Diaspora* вполне приличная, по крайней мере в контексте игры, как-никак происходящей в необъятной черной бездне. Гигантские планеты, звезды, туманности и неуклюжие космические мега-шаттлы заполняют эту бездну, и по большей части выглядит все неплохо, особенно в пылу яростных сражений.

Однако при подлете ближе к большому кораблю, например, к базе Сайлонов, текстуры немного теряют четкость. Способность вилять и скрываться за большими объектами – важный фактор успеха в космическом бою, так что эта небольшая и в общем простительная размытость картинки довольно часто сказывается на ходе игры.

Теперь к делу: блужданию по космосу с оружием и звездолетами. Запомните крепко: настроек в *Diaspora* множество. Мы трижды проходили обучающие миссии, пока освоили самые азы пилотирования. Это не попытка походя пнуть руководство пользователя – оно вполне толковое и дает внятное представление об игре – а скорее попытка подчеркнуть нездоровую одержимость разработчиков подробностями и вариациями, итог которой – фолианты описаний органов управления.

## Свойства навскидку


**Простите, что?**
Целые тома настроек. Половина используется с модификаторами. Уму непостижимо.



**Забудьте об этом**
Действуйте наобум, полагаясь на то, что запомнили. И погибните в чудовищном взрыве.

## Ключевые маневры

Если вы мечтаете о безумном маневре – дерзайте, насколько у вас хватит памяти удерживать нужные комбинации ключей и модификаторов для каждой мелочи.

Настройки вам тоже захочется реформировать, поскольку многое из того, что мы сочли бы существенным, разбросано по разным сторонам клавиатуры. Возьмем, к примеру, противоракетную защиту: прицел наводится клавишей [H], а поворот корабля осуществляется клавишами [Q] и [E], но все контрмеры при этом выполняются через клавишу [.]. Возможно, это имитация пребывания в кабине Viper'а в разгаре битвы, когда пилот судорожно шарит по пульту управления в поисках спасительной комбинации. А возможно, просто досадное упущение.

Поведение корабля подчиняется не то чтобы законам ньютоновской физики, а скорее изменчивым законам невесомости: не без подвохов, чтоб была рисковость, но и с большой долей реализма. И, невзирая на излишнюю мелочность, выходит занимательно.

## Скользя по космосу

В игре есть несколько весьма симпатичных и нечасто встречающихся функций, таких как режим Скольжения, когда ваши двигатели отключены, но вы продолжаете двигаться тем же курсом и не снижая скорость. А значит, вы можете резко развернуть свой корабль и убить преследующего вас врага, или исполнять нелепые маневры, чтобы избежать столкновения.

Сами миссии очень напряженные. Вам даются цели, которые необходимо выполнить за заданное время – как правило, все из той же нескончаемой оперы об обманках и уловках вашей армии в борьбе со злодеями. Хорошо написанные легенды, вкупе с полной и атмосферной озвучкой, действительно затягивают.

Но при всем внимании к оформлению – ну, кроме пары мелких промахов – человеческий фактор не учтен. Вот вы вникли в цели, поняли, что от вас требуется... а потом с воплями будете напропалую палить из пушки, даже не замечая, что сама миссия прошла мимо вас. Шансы выполнить задание у вас ничтожно малы.

Скорее всего вы, как и мы, будете просто метаться, отстреливаясь от плохих парней, отвечая нелепыми рывками на призывы терпящих бедствие союзников; послушаться удастся только тогда, когда звездолет пригрозит уйти без вас, если вы немедля не приземлитесь.

Постараться выйти из неофициальной роли сорвавшегося с цепи межзвездного орудия станет вашей главной задачей при втором и более прохождении игры.

## Вид из кабины

**Средства связи**
Союзники и враги будут с вами постоянно болтать, сквозь треск космического эфира.

**Приказы, сэр**
Здесь то, что вы должны делать. Но не сделаете: вам будет не до этого.

**Будьте начеку**
Приборная панель держит вас в курсе всего: скорость, уровень топлива, прицелы и цели.

**Подсистема наведения**
Если хотите, можете выбирать для удара конкретные зоны противника.

**Время целиться**
Можете палить по друзьям, врагам и вообще чему угодно.

**DRADIS**
Система 3D-наведения иногда подводит в нужную минуту.

**Во всеоружии**
В многопользовательском режиме можно управлять своей амуницией.

› **Аутентичное вооружение *BSG* делает каждую миссию и стильной, и неистовой.**

## Коротко, но мило

И весьма печально, что наступит этот второй раз так скоро: *D: SA* – игра очень короткая. Это только первый релиз – пока про нее не пронюхали специалисты по авторскому праву и не пресекли продолжения – но не успеете вы вникнуть, что и как, а все уже кончено.

К счастью, к игре приложен редактор миссий, и зарождающееся сообщество любителей *FreeSpace* уже принялось вносить реформы в контент. Редактор довольно замысловатый – мы провозились с ним битый час, так и не создав ничего ценного, но это прекрасная возможность вытянуть немного больше из лучшей игры на основе *FreeSpace*, когда либо существовавшей.

*Diaspora* – быстрая, бурная и невероятно веселая игра. Находящийся пока что в бета-версии многопользовательский режим, хотя и кривоватый, делает ее еще веселее – пока вам не встретится настоящий ас пилотирования Viper с 12-ю пальцами, который покончит с вашими мечтами стать звездой.

Но так и задумано: хоть мы и критиковали игру за сложность, зато в ней всегда есть чему поучиться, и каждый раунд будет приближать вас к совершенству как пилота.

За прекрасную цену в £0 вы просто обязаны встать на крыло настолько хорошо, насколько позволят возможности вашего ПК: игра хорошо масштабируется, но для достижения наилучшего результата не обойтись без хорошей видеокарты и избытка ОЗУ. LXF

## LINUX FORMAT Вердикт

### Diaspora: Shattered Armistice Release 1 (v1.04)

**Разработчик:** Diaspora Development
**Сайт:** http://diaspora.hard-light.net
**Лицензия:** Free Creative Commons (BY-NC-SA 3.0 unported)

| | |
|---|---|
| Сюжет | 10/10 |
| Графика | 9/10 |
| Длительность | 6/10 |
| Документация | 9/10 |

» Яростная битва во вселенских масштабах. Короткая, но многообещающая.

**Рейтинг 9/10**

# Enlightenment E17

Оригинальный и радующий глаз менеджер окон обретает, наконец, стабильный релиз. Расследование ведет **Бен Эверард**.

## Вкратце

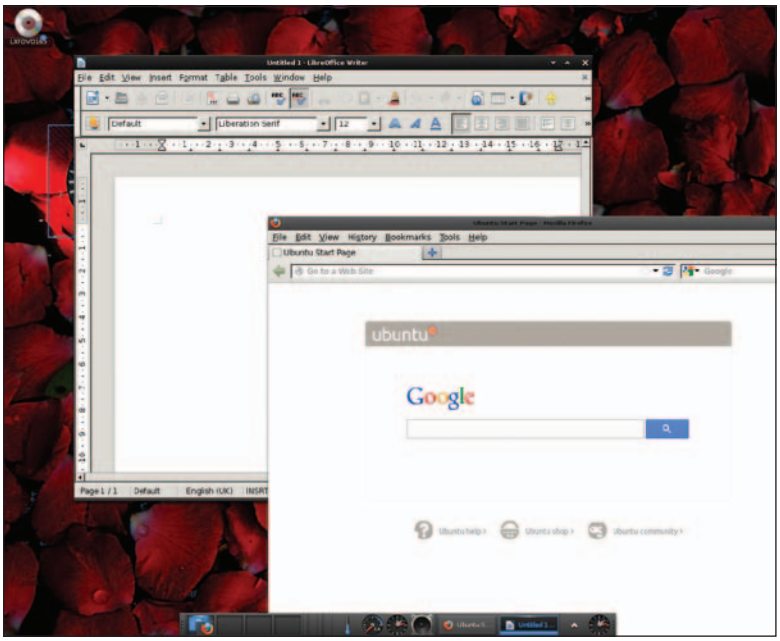
» **Оконный менеджер X11. См. также: *Kwin* (KDE), *Mutter* (Gnome), *Compiz*, *IceWM* или *OpenBox*.**

Если вы предпочитаете именитые дистрибутивы, не знать о существовании *Enlightenment* вам простительно, не говоря уж о том, что недавно вышла версия *E17*. *Enlightenment* не используется ни в одном дистрибутиве из первой десятки Distrowatch (хотя почти везде он есть в репозиториях, если вы пожелаете установить его самостоятельно). На момент написания статьи, за дистрибутивом с *Enlightenment* вам придется спуститься до 15-й строки: это Bodhi.

Что нельзя считать признаком неготовности или негодности Enlightenment к выходу в большой свет – хотя, возможно, своим пристрастием к альфа- и бета-версиям команда *Enlightenment* кого-то и отпугнула. Данная конкретная версии разрабатывалась 12 лет, прежде чем увенчалась стабильным релизом. Конечно, рабочие версии все равно активно использовались в эти годы, и без особых проблем.

Желая прочувствовать эту систему, мы опробовали ее и на Bodhi, и на Ubuntu 12.04. Первый представляет этот оконный менеджер более выигрышно, поскольку дает любопытствующим возможность поиграть с несколькими вариантами макетов.

Первое, что вы отметите уже при запуске *E17* – то, что он великолепно выглядит. Это не новшество релиза, а фирменный стиль *Enlightenment*. Похоже, графическим эффектам, вроде растворения и появления меню, здесь уделяют больше внимания, чем в других оконных менеджерах. Более того, эффекты гладко работают и на более старом оборудовании, где другие настольные окружения задыхаются из-за своих виджетов. Кстати, виджетов, которые можно добавить на рабочий стол (называемых здесь гаджетами), в *Enlightenment* множество. Что для него и характерно, гаджеты хорошо прорисованы и гармонируют в систему в целом.




› **Из-за стилевых расхождений с *GTK* при запуске программ Enlightenment теряет весь свой лоск.**

## Трата времени

Стандартная конфигурация меню немного запутана. Автор этого обзора чаще всего пользуется меню, чтобы открыть приложения, и думает, что прятать их в подменю нерационально. Если каждый раз набавлять на запуск программы несколько секунд, в сумме набежит порядочно. Эти две лишних секунды, при том, что приложения вы открываете сотни раз на дню, складываются в 20 часов в год, потраченных на возню с меню, когда вы могли бы их потратить... ну, на другое.

Смысл существования *Enlightenment* в том, что он красив. По-настоящему красив. Хотя, в отличие от Gnome, KDE и прочих, это просто оконный менеджер, а не рабочее окружение. То есть собственных приложений у него нет. Можно использовать арсенал других настольных систем – работают они нормально... но выглядят довольно-таки неказисто. Здесь-то и кроется фатальная ошибка *Enlightenment*. Он великолепен, пока на рабочем столе ничего нет, но стоит вам наоткрывать окон и приняться за работу, как возникает ералаш. Обидно, что вся красота разрушается чуть ли не от первого прикосновения. Так что рекомендуем его тем, кому хочется, чтобы рабочий стол радовал глаз, но лень добиваться такого эффекта другими путями. LXF

## Свойства навскидку

**Анимация**
И при открытии окон, и при работе с меню или иконками эффекты анимации гладко отображаются даже на слабых системах.

**Гаджеты**
Добавьте их на свой рабочий стол, чтобы фон этого самого рабочего стола сделался более содержательным.

## LINUX FORMAT Вердикт

### Enlightenment E17

**Разработчик:** Enlightenment Dev Team
**Сайт:** www.enlightenment.org
**Цена:** Бесплатно по BSD-подобным лицензиям

| | |
|---|---|
| Гладкость | 9/10 |
| Производительность | 9/10 |
| Удобство использования | 8/10 |
| Гармония с приложениями | 3/10 |

» Симпатичный оконный менеджер с фатальным недостатком.

**Рейтинг 7/10**

# Arduino Due

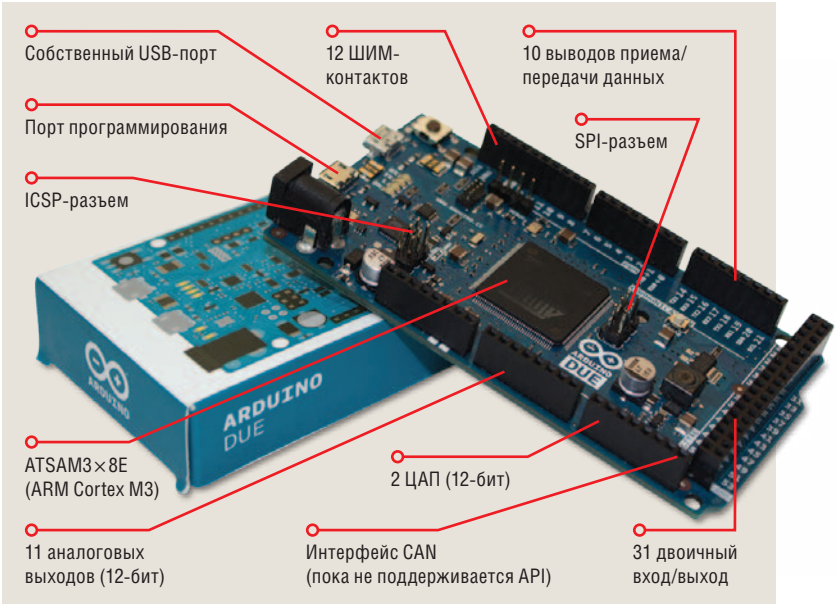
Способен ли новый микроконтроллер Arduino нести мощь ARM в массы? **Бен Эверард** разбирается.

## Вкратце

» **Микроконтроллер на базе ARM. См. также: Maple или Mbed.**

Arduino сделал себе имя на производстве плат микроконтроллера, простых и удобных в использовании. До сих пор они все шли на базе процессоров AVR. Однако за последние восемь лет в мире микроэлектроники многое изменилось, и пользователи начали отказываться от AVR в пользу более мощных чипов на базе ARM. К нам в руки попала первая плата Arduino с этими микроконтроллерами.

Начнем с плохих новостей. В отличие от предыдущих продуктов Arduino, Due работает от 3,3 В и напряжения в 5 В не выдерживает. Как следствие, он несовместим с некоторыми платами расширений Arduino, а также схемами, разработанными для других платформ. Более того: если с ним обращаться как с обычным Arduino, он может просто поджариться. Вывод: Due не для новичков, а для тех, кто полюбил Arduino и нуждается в мощи ARM.




> «Мощность ARM дает вам широкий спектр новых функций.»

Но не все чипы ARM одинаковы, и это не универсальный процессор SoC, как в Raspberry Pi. Это микроконтроллер ARM, и вы программируете его в среде разработки (IDE) Arduino, а затем в ней и запускаете. Дополнительная мощность процессоров ARM дает вам широкий спектр новых функций, каждая из которых может стать «бомбой», в зависимости от специфики вашего проекта:

» Повышение вычислительной мощности, благодаря процессору Cortex-M3 с тактовой частотой 84 МГц.

» Два USB-порта (один – для программирования, другой – для обычных целей), это значительно облегчает разработку для данного типа соединений.

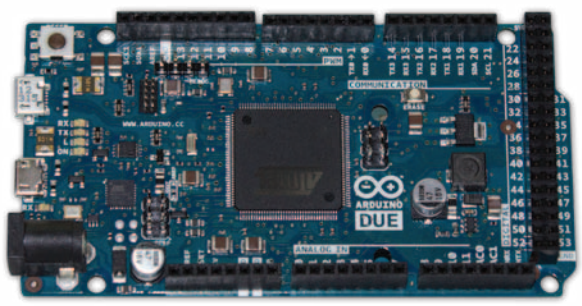
» Диспетчер, чтобы управлять множеством задач.

» Два 12-битных цифро-аналоговых преобразователя (ЦАП) позволяют подключать аналоговые устройства. Например, можно слушать музыку, подключив колонку напрямую к платформе.

» Совместимость с Android ADK позволяет создавать периферийные устройства для телефонов и планшетов Android.

Это, безусловно, большой шаг вперед по сравнению с устройствами AVR Arduino, но микроконтроллеры ценятся не за свою вычислительную мощь и широкий функционал. Как правило, лучший микроконтроллер – тот, где есть все для решения конкретной задачи, и ничего лишнего.

› **Due немного больше Uno, но гораздо лучше укомплектован.**



## Новый уровень

Чтобы использовать Due, вам необходимо установить IDE Arduino версии 1,5 или выше. Ее нужно ставить вручную, так как в репозиториях большинства дистрибутивов пока есть только версия 1. Но это всего лишь распаковка архива и запуск исполняемого файла (никаких там **./configure && make && sudo make install**).

Для новичков самыми подходящими по-прежнему остаются Uno и Mega, и вряд ли это скоро изменится. Новые функции Due большинству их проектов просто ни к чему, а совместимость и поддержка более ранних платформ гораздо ценнее невостребованных возможностей. Тем не менее, Due действительно сделал выполнение нескольких классов проектов намного легче и дешевле (сократив количество добавочных компонентов).

Нас в Башнях **LXF** более всего порадовали ЦАПы и два USB-порта, но это наша специфика. Несомненно, блогосфера вскоре забурлит хитроумными штуками, большими и малыми, созданными на Due.

Если вы затеваете грандиозный проект, Due непременно обязан быть в вашем списке покупок. LXF

## LINUX FORMAT Вердикт

### Arduino Due

**Разработчик:** Arduino
**Сайт:** www.arduino.cc
**Цена:** €39

| | |
|---|---|
| Функциональность | 9/10 |
| Производительность | 9/10 |
| Удобство использования | 9/10 |
| Оправданность цены | 9/10 |

» Due — наш фаворит среди микроконтроллеров на базе ARM.

**Рейтинг 9/10**

GOOGLE NOW В ДЕЛЕ

# Подсказки не только голосом

Не так давно представленная функция уже стала лучше, чем у конкурента №1.

Google Now – это персонализированный сервис поиска. Анонс услуги состоялся на конференции Google I/O в 2012 году. Впервые реализован в ОС Android 4.1 "Jelly Bean". Google Now отображает информацию согласно предпочтениям пользователя, изучив его привычки и режим дня, с учетом его текущих GPS-координат, личной информации из календаря, истории поисковых запросов, перемещений и посещенных страниц, и т.д. Интерфейс оформлен как «информационные карточки» – их число на момент реализации достигало 10 и планировалось к увеличению. Пользователь может настроить карточки под свои нужды и удалить лишние. По словам разработчиков, такой интерфейс наиболее удобен для постоянного обновления информации.

Один из пользователей сайта Reddit заявил, что новый сервис от Google всего за три дня установил его место жительства. Причем он не вводил поисковых запросов в Google и нигде не указывал свой адрес. Недавно в поиске Google появилась функция Instant – уже по первым буквам запроса поисковик выдает релевантные результаты. Но в компании сочли, что пользователю вообще незачем задавать вопросы: нужную информацию ему выдаст сам поисковик, зная, что пригодится пользователю сейчас.


**› Интерфейс Google Now реализован в виде так называемых «карточек», что очень удобно для пользователя.**

Google Now уже получил множество положительных отзывов. Он позволяет узнавать погоду, расписания поездов, может быть гидом в незнакомом месте или переводчиком. Хотя Google Now объективно лучше Siri от Apple, стабильнее и надежнее, Siri существенно известнее благодаря агрессивной рекламной кампании.

Сейчас Google активно развивает Google Now – похоже, готовится новый сюрприз в этом отношении, в виде особого виджета серии Nexus. Виджет вкратце отображает содержание текущих карт Google Now и может быть установлен как на домашнем экране, так и на экране блокировки.

Ложка дегтя: система Google Now пока недоступна русскоязычным пользователям, а также пользователям Android версии ниже 4.1, каковых пока большинство.

ТАБЛЕТКА ОТ ГИГАНТА

# Вторая попытка выйти на рынок

HP выпустит Android-планшет на базе NVIDIA Tegra 4.

Компания Hewlett-Packard готовится выпустить Android-планшет на основе NVIDIA Tegra 4, который станет одним из самых быстрых устройств на рынке. Первый планшет HP может быть анонсирован уже очень скоро, говорят источники портала ReadWrite. При успехе продаж компания может выпустить и Android-смартфон, пишут *Вести*. Но генеральный директор компании Мэг Уитмен [Margaret «Meg» Whitman] уверила, что в планах на текущий год такого нет.

**› HP TouchPad был великолепен с точки зрения программной платформы, но не снискал успеха.**

HP пережила провал 9,7-дюймового TouchPad – первого планшета компании на ОС webOS. Тот поступил в магазины США в июле 2011 года, однако уже через несколько месяцев HP заявила, что прекращает выпуск продуктов на базе своей платформы. Когда не нашел отклика у покупателей и webOS-смартфон Veer 4G, гендиректор HP Лео Апотекер [Leo Apotheker] ушел в отставку. Сменившая его Мэг Уитмен передала проект webOS сообществу, открыв исходный код ОС.

Сейчас HP выпускает 2 планшета на основе Windows 8: ElitePad 900 и гибридный ноутбук Envy x2. В 2012 году компания планировала выпустить «таблетку» на Windows RT, но в итоге отказалась от проекта. Новые продукты HP будут применять платформу Google, и будущее webOS остается неясным. Кстати, webOS была успешно портирована на планшет Google Nexus 7, но проблема со сторонним ПО, вернее, с его отсутствием, пока не решена, и не факт, что данная ОС всерьез заинтересует вендоров, несмотря на свою инновационность и открытость.

Сейчас работа с Android – стратегическое направление работы HP, где компания надеется преуспеть. В выборе Android ничего удивительного нет – сейчас это самая популярная ОС для планшетов и смартфонов, обладающая обширной библиотекой стороннего ПО и активно поддерживаемая разработчиками. По сути, HP необходимо лишь предложить разумную цену за качественное устройство, чтобы застолбить для себя часть планшетного рынка.

ПЛАНШЕТ ДЛЯ ИГРОКОВ

# Производство уже начато

## Игровой планшет Wikipad вышел на Android и NVIDIA Tegra 3 с 7” экраном.

В последнее время в сегменте смартфонов и планшетов просматривается тенденция к стремительному увеличению производительности комплектующих этих устройств. Кроме того, что большинству пользователей это совершенно не требуется, в мобильной сфере практически отсутствуют приложения, способные реально задействовать новейшие четырехъядерные мобильные процессоры... но есть ведь игры!

Именно для геймеров, предпочитающих ОС Android, и был представлен, а сейчас уже доступен для предзаказа, уникальный планшет Wikipad. Его оригинальность – в наличии специального подключаемого блока управления, который по сути представляет собой огромный геймпад.

Производитель, компания Wikipad, планировал начать продажи планшета еще до конца прошлого года, но в последний момент выпуск был отменен, а сам планшет ушел на доработку.

**› Самое интересное в Wikipad — настоящие геймерские джойстики — по достоинству оценят адепты мобильных игр.**

Бывший 10-дюймовый планшет, Wikipad стал более удобным, компактным 7-дюймовым устройством; габариты корпуса с геймпадом – 286×145×63,3 мм и 195,6×125,7×10,6 мм без него, и вес, соответственно, 760 и 320 грамм. Начинка осталась практически без изменений: платформа NVIDIA Tegra 3, с четырехъядерным процессором с частотой 1,4 ГГц и графическим ускорителем GeForce, объем ОЗУ – 1 ГБ, внутренней памяти – 16 ГБ, две камеры с матрицами 8 и 2 Мп, адаптеры Wi-Fi, Bluetooth и 3G, приемник GPS/ГЛОНАСС, динамики, разъем для наушников, слот для карт microSD, порты microHDMI и microUSB, а также аккумулятор емкостью 4100 мАч и док-станция, представленная в виде геймпада.

Подковообразный геймпад охватывает три из четырех сторон планшета и располагает двумя аналоговыми джойстиками, крестовиной и блоком кнопок. Также на нем расположились динамики. Планшет способен загружать игры с сервисов Tegra Zone, Google Play и PlayStation Mobile.

Новинка получила название Wikipad 7. Примечательно, что она стала в два раза дешевле: теперь ее цена $250.

В комплект поставки изначально будут включены такие игры, как *Dead Trigger*, *Hockey Nations Tournament* и *ShadowGun: Dead Zone*.

---

WINDOWS НА ANDROID

# Android набирает силу

## Samsung реализовал поддержку многооконного режима для платформы Android.

Компания Samsung выпустила обновление прошивки на базе Android 4.1.2 для планшета Galaxy Note 10.1 с поддержкой многооконного режима запуска приложений – одновременно можно разместить на экране до 16 окон.

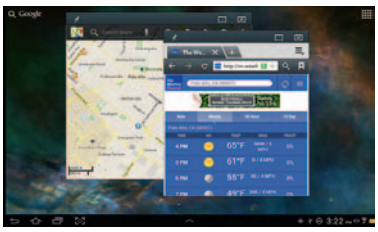
Предусмотрено два режима группировки окон: разделение экрана на две части и каскадное размещение, плюс открытие окна на весь экран. Поддержка фреймворка, дополняющего TouchWiz, экранную оболочку в устройствах Samsung, добавлена в 18 базовых приложений: браузер, почтовикл, офисный пакет, календарь, будильник, видеоплейер, и т. д. В заголовке окна появились три кнопки: закрепление позиции, раскрытие на весь экран и закрытие программы. Для запуска многооконных программ в интерфейс добавлен специальный лоток, позволяющий выбирать приложения и переключать режимы.

Обновление доступно для загрузки через Wi-Fi или программу Samsung KIES.

**› Многооконности действительно не хватало многим пользователям Android.**

Реализованный фреймворк не нарушил требования консорциума Open Handset Alliance, запретившие неконтролируемое расширение API Android отдельными производителями, чтобы предотвратить фрагментацию платформы: он ограничивается базовыми внутренними приложениями и не предлагается сторонним разработчикам. Программный интерфейс сохранен, и разработчики по-прежнему должны использовать стандартный Android SDK.

Теперь пользователи смогут работать в Интернете и общаться в Skype, не переключаясь при этом между приложениями.



Александр Джульяр [Alexandre Julliard], создатель и руководитель проекта *Wine* и технический директор компании CodeWeavers, рассказал на конференции FOSDEM 2013 о работах по адаптации *Wine* к системам с архитектурой ARM и создании варианта *Wine* для Android, продемонстрировав готовый прототип для архитектуры x86 и Android-планшетов с процессорами Intel Atom. CodeWeavers заинтересована в создании коммерческого продукта для платформы Android и считает данное направление перспективным. Появление *Wine* для смартфонов и планшетов на базе ARM не связано с желанием обеспечить запуск обычных Windows-приложений для комьютеров на базе архитектуры x86: расчет делается на приложеня для ARM-устройств на базе Windows 8.

Итак, Android становится все больше похож на Windows... Если вспомнить о растущем количестве вирусов для Android, становится ясно, что это пока не плюс. LXF

# Сравнение

» Каждый месяц мы сравниваем тонны программ — а вы можете отдыхать!

# Фоторедакторы

От профессионального «проявления» RAW до обычного управления библиотеками, Linux прошел долгий путь, детка. **Адам Оксфорд** исследует...


## Про наш тест...

Напрямую сравнивать наши программы тяжело, поскольку они разрабатывались с разными целями. Однако базовые тесты были максимально стандартизованы.

Мы ставили каждому приложению задачу каталогизировать библиотеку из нескольких тысяч изображений, и смотрели, как оно справится с обработкой файлов RAW, редактированием фоток в формате JPG, изменением размера пакетом и фильтрами. Конечно, не все приложения из нашего списка на это способны, но, хотя функциональность и важна, куда важнее скорость, простота в использовании и конечный результат. Если быстрее будет взять одно приложение, чтобы позаботиться о том, где хранятся фотографии, и для их поиска, и другое – для редактирования, нас это устроит.

## Наша подборка

- AfterShot Pro
- Darktable
- digiKam
- GIMP
- Fotoxx
- Shotwell

Среди профессионалов и любителей фотографии до сих пор существует сильное предубеждение против Linux, однако время, когда никто не принимал всерьез умение нашей любимой ОС обрабатывать изображения, давно прошло. Возможно, пока и нет программ от Adobe или Apple, разработанных для Linux – а надо ли это нам? Ведь имеются более чем серьезные, зрелые пакеты абсолютно для всего, начиная с обычного управления библиотеками и до обработки RAW-форматов. В наше время вполне возможно обзавестись профессиональной фотостудией, не обращаясь к Windows или Mac.

По сути, выбор настолько богат, что ограничиться всего шестью программами для нашего Сравнения было задачей не из легких.

Среди них вы неизбежно встретите старых знакомцев. Например, хотя упоминания о *GIMP* уже на зубах навязли, нельзя же его пропустить, если нужен всеобъемлющий пакет для пост-обработки снимков. Подобным же образом – вы, наверно, знаете даже больше, чем хотелось бы, о *Shotwell* и *digiKam* – столпах Gnome и KDE, но и они по ряду причин являются выбором де-факто. Оставить их за бортом Сравнения означало бы отказаться от лучших.

Самым спорным было решение о включении или не-включении в наше сравнение *AfterShot Pro* от Corel – урожденного *Bible*. Это золотой стандарт редактирования изображений RAW в Linux, но у него закрытый код, и он отнюдь не дешев: полная версия стоит $59,99. Однако с того момента, когда мы в последний раз его рассматривали (**LXF156**), он обновился, так что мы вновь его рассмотрим – ценой отказа от истинной альтернативы FOSS. Может, это и покажется неправильным, но истина все же за нами.

Если вы не согласны с нашим выбором, пишите нам на форумы: www.linuxformat.com/forums.

# Любителям или профи?

## Кому предназначена программа и как это влияет на простоту использования?

Наши шесть кандидатов намеренно отобраны с охватом разных целевых групп, но ни один из них не страдает недостатком функций. В большей мере это относится к тому, как они представлены и как взаимодействуют друг с другом. И если вам нужно всего лишь по-простому организовать фотографии и устранить эффект красных глаз, то оплоты дистрибутивов – *digiKam* и *Shotwell* – вполне удовлетворят ваши потребности. Оба концентрируются на основных функциях программы управления фотографиями, и хотя в них есть встроенные редакторы изображений, вас никто не заставляет их применять.

*Shotwell*, в частности, умышленно ориентирован на простоту использования – с тем же голым минимализмом, который разработчик Yorba внес в свой превосходный клиент электронной почты, *Geary*. Откройте его, и справа будут фотографии, слева – опции сортировки, а посередине – почти ничего. Однако редактор изображений крайне близок к приложению по своей простоте – здесь есть кнопка Enhance, которая по одному щелчку приводит в порядок изображение перед его публикацией на выбранном вами сайте.

Однако *digiKam* намного полнее, и в нем масса опций и функций. Хотя трудно сказать, для кого он предназначен. Исключительно удобный инструмент Library позволяет выполнять сортировку по дате, тэгу, распознаванию лица и даже по «нечеткому» – он же «умный» – поиску [fuzzy search], который пытается соотнести общие фигуры с существующим изображением: вам предлагается «набросать» в пустом окне фигуру, которую нужно найти, и *digiKam* постарается это сделать.

Помимо не совсем стабильных результатов нечеткого поиска и совпадения лица, *digiKam* может показаться случайному пользователю довольно пугающим, не предлагая при этом инструментов редактирования, достаточно мощных, чтобы удовлетворить профессионалов.

*AfterShot Pro*, как и подразумевает его название, достойный инструмент для достойных фотографов. В нем имеется мудрый менеджер библиотек, хороший редактор изображений и бесспорно лучшие инструменты для обработки RAW среди всех программ всех платформ. Кроме того, он выглядит недорогим на фоне аналогов *Lightroom* или *Aperture*, хотя и дешевым его не назовешь.

> «AfterShot Pro — достойный инструмент для достойных фотографов.»

*Darktable* создавался чисто для работы с файлами RAW, но он, как нам кажется, больше подойдет любителю-энтузиасту, чем фотографу-профессионалу. В нем невероятно много функций, и он способен на все подвиги *AfterShotPro*, но... бесконечная тонкость его индивидуальной настройки, в духе открытого кода, слишком необъятна для применения в промышленном рабочем процессе. Добиваться совершенства каждого снимка можно при обработке двух-трех фото, но не сотен их в день.

› ***У AfterShot Pro*** **множество простых в использовании инструментов профессионального уровня.**

А вот *GIMP* по-прежнему остается для новичков ошарашивающим, несмотря на недавний пересмотр. Однако благодаря отсутствию конкуренции в плане функций редактирования изображений – таких, как клон кисти и управление слоями – он попадет в инструментарии большинства фотографов, независимо от способа его применения: чтобы осветлить телефонные фотографии с вечеринки или создать рекламные щиты для центральных улиц.

### Вердикт

**AfterShot Pro** ★★★★★
**digiKam** ★★★★☆
**GIMP** ★★★★☆
**Shotwell** ★★★★☆
**Darktable** ★★★☆☆
**Fotoxx** ★★☆☆☆

» Инструменты профи вовсе не обязаны быть сложными, и AfterShot получил пятерку.

# Управление фотографиями

## Скопилась куча изображений... И кто о них лучше позаботится?

Большинство из нас начинают управлять библиотеками изображений посредством стандартного менеджера файлов и умного именования папок. А некоторые придерживаются этого способа на протяжении всей своей карьеры. Но зачем усложнять жизнь, если можно просто дважды щелкнуть по «Рождеству 2013», и *Nautilus* или *Dolphin* и иже с ними отобразят страницу с миниатюрами изображений?

Кроме одного заметного исключения, все наши кандидаты имеют некую форму управления библиотеками. *GIMP* – это редактор в чистом виде, и он разумно избегает попыток гнаться за чересчур многим.

Управление библиотеками заложено в *Shotwell* и *digiKam*, и оба прекрасно справляются с этой работой. *DigiKam* рассортирует ваши снимки по любой комбинации папки, даты и EXIF или тэга, хотя если отклониться от дерева файлов, интерфейс может показаться несколько замусоренным. *Shotwell* чуть более ограничен, поскольку сортирует все в первую очередь по дате – что может оказаться проблематичным, если вы не снабжаете свои снимки информацией о теме.

У *AfterShot Pro* имеется весьма впечатляющая функция работы с библиотеками, позволяющая сортировать фотографии практически по любому параметру. К сожалению, ее несколько портит функция импорта – очень сложная и без автоматического просмотра папки на предмет обновлений; поэтому встроенный браузер файлов лучше. Возможно, именно поэтому управление библиотеками *Darktable* отражает структуру файлов на жестком диске – так проще, и это позволяет приложению легко справляться с задачей быстрого создания миниатюр для просмотра.

*Fotoxx* здесь вновь демонстрирует свою непохожесть. Он должен быть менеджером библиотек, но действует по большей части как браузер файлов, где нет явной возможности перейти на уровень выше. Довольно странно.

### Вердикт

**digiKam** ★★★★☆
**AfterShot Pro** ★★★☆☆
**Shotwell** ★★★☆☆
**Darktable** ★☆☆☆☆
**Fotoxx** ★☆☆☆☆
**GIMP** ☆☆☆☆☆

» Менеджеры библиотек — лучший способ быстрой сортировки множества снимков.

# Интерфейс

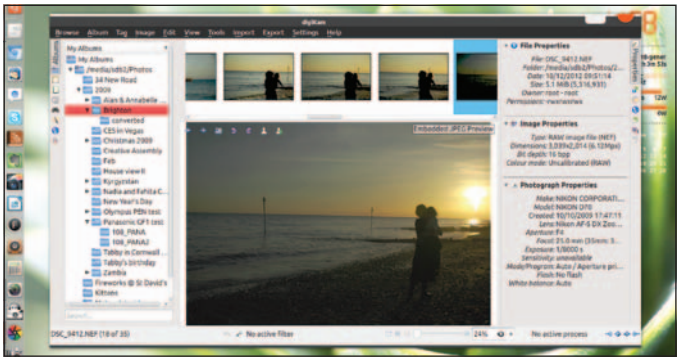
Итак, пакет вы добыли. Легко ли в нем разобраться?

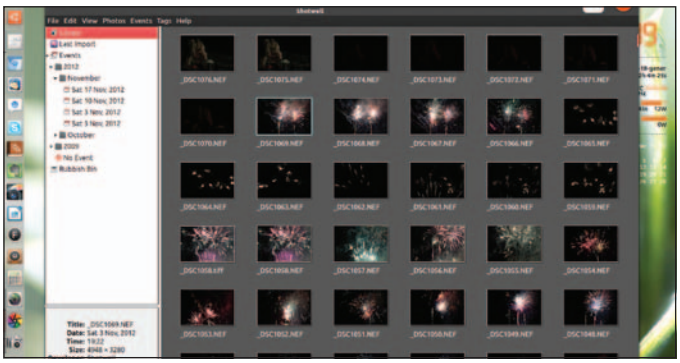
## AfterShot Pro ★★★★★

Если за последние лет десять вы использовали хоть какой-то редактор RAW, структура *AfterShot Pro* покажется вам знакомой с ходу. Слева – программы управления просмотром: менеджер файлов, программа просмотра библиотек и ряд настроек для фильтрации и поиска снимков. В середине – цифровой световой стол, отображающий миниатюры или увеличенные фото для обработки. А справа – инструменты редактирования. Они явно не озаботились экономией площади экрана, зато наименованы и расположены в интуитивно удобном порядке: большинству снимков хватит редактирования из вкладки по умолчанию, а более продвинутые опции лежат уровнем дальше. Эта программа не самая красивая или современная, но не выглядит архаично и не заставит пользователя искать основные инструменты с собакой. Высший балл.

## digiKam ★★★½☆

Более одного щелчка требуют немногие функции. Открывается *digiKam* в менеджер библиотек, и можно создать новый вид по названию, дате, метатэгам, рейтингу и т.д., или перейти в полноценную программу импорта RAW и фоторедактор. Это в чем-то проблема: по умолчанию значки идут вниз по обеим сторонам основной рамки, и массу кнопок управления надо запоминать. Тем, кто хочет делать всю обработку кучи изображений, не покидая *digiKam*, это удобно, но вы не будете применять добавочные функции управления, они слишком загромождены. Вдобавок некоторые особо крутые функции – например, Вертящийся глобус для присвоения снимкам тэгов GPS – не лишены ошибок. Панель сортировки по дате мила, но тупое дерево папок *Shotwell* удобнее. Зато, как приложение KDE, *digiKam* отлично настраивается.





## Shotwell ★★★☆☆

Так как *Picasa* от Google больше не поддерживается для Linux, *Shotwell* становится основным среди менеджеров библиотек, которые в первую очередь позволяют быстро найти фото. По умолчанию он сортирует изображения по дате, используя временную шкалу справа – выглядит она рудиментарной, но зато ею проще пользоваться, чем таковой в *digiKam*. Можно настроить просмотр по тэгам или данным EXIF – хотя эти опции не столь явны: чтобы их найти, не обойтись без меню View. Здесь есть кнопка исправления фото одним щелчком, и снимки и альбомы загружаются напрямую на немалое число сайтов. Простота *Shotwell* лишена элегантности. Скругленные значки, применяемые для отображения в виде папки и альбома, выглядят несколько неуместно, а колонке сортировки слева не помешала бы лишняя забота.

# Управление цветом

Важнейший аспект редактирования фото стал обычным делом.

Все приложения, кроме *Fotoxx*, справляются с полным управлением цветом. Пусть это не кажется столь важным – но для фоторедакторов Linux это гигантский прыжок вперед.

Они используют пользовательскую гамму по стандартам ICC, меняя способ отображения цветов согласно уникальным характеристикам монитора и тестированным пространствам цветов камеры и изображения – обычно получается один из стандартных профилей RGB, например, sRGB или Adobe RGB. Без этого профессионалам не жить, ибо тогда цвета, видимые на экране, точно воспроизводят то, что напечатается. Особого спасибо заслуживают разработчики Gnome за включение управления настройкой цвета на базе *Argylla* в панели управления по умолчанию. Ныне полностью откалибровать монитор с помощью устройства вроде ColorVision Spyder 2 проще в Gnome, чем в Windows. Это касается и среды Canonical Unity, применяющей тот же инструмент.

KDE старается не отставать – *Oryanos* и *KCM* почти сравнимы с инструментами по умолчанию Gnome, но требуют ручной компиляции и установки; а единственная реальная опция *Xfce* – инструмент командной строки **xcalib**. Если ваш монитор неверно откалиброван, наличие редактора с управлением цветом бессмысленно. Но у фотографов, недолюбливающих Gnome или Unity, есть вариант – Kubuntu по умолчанию включает калибровку цвета, основанную на тех же пакетах, что и Unity.

### Вердикт

**AfterShot Pro** ★★★★★

**GIMP** ★★★★★

**Darktable** ★★★★☆

**digiKam** ★★★★☆

**Fotoxx** ★★★☆☆

**Shotwell** ★★★☆☆

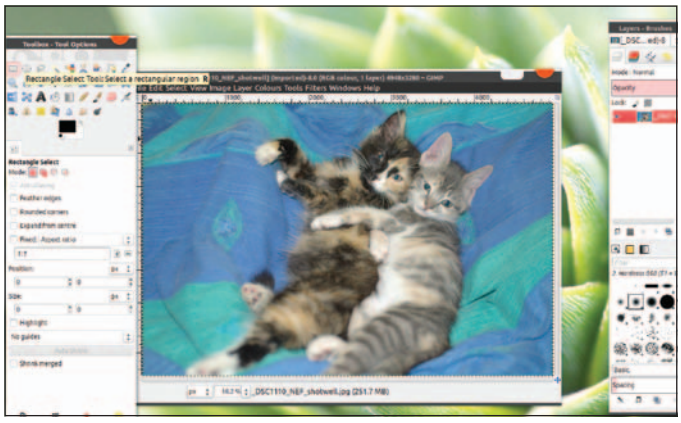
» DigiKamImpress явно впечатляет, оправдывая свое название.


## Darktable ★★★☆☆

Похоже, основной принцип *Darktable* – брать сложившийся дизайн традиционного редактора RAW и упрощать его. И все получается отлично. Отображение библиотеки – сама простота, да и инструмент редактирования тоже сперва кажется простым. Плюс небольшие выкрутасы, некие завихрения, напоминающие панель немого кино и украшающие черный фон. Но при внешней простоте, использовать *Darktable* не так уж просто. Меню редактирования отмечены значками, а не словами, и названия им даны по интуитивному принципу. Функции управления делятся на Basic Group [Основную Группу], Tone Group [Тоновую Группу] и Explicitly Specified by User [Явно определяемые пользователем]. Часть наиболее популярных функций управления недоступна – их нужно добавить из скрытого меню Plugins. При должной настройке *Darktable* станет эффективным инструментом, удобным именно для вашего образа действий. Но для большинства уровень его сложности высоковат.

## Fotoxx ★★☆☆☆

Интерфейс *Fotoxx* слегка отдает стилем Windows 98, что отбивает у большинства пользователей охоту им пользоваться. Однако не дать ему шанса означает пропустить изумительное безумие, заложенное в его дизайне. Это инструмент управления библиотеками, который скрывает свою основную функцию. Вы получаете серый экран, который совершенно ничего не делает. И только при исключительной наблюдательности вы углядите и нажмете на крошечную букву G в углу – она запустит менеджер файлов со структурой папок и эскизов изображений, которая будет работать, пока вам не понадобится перейти на уровень выше. Есть ряд кнопок для функций и навигации по файлам. И только когда вы начнете исследовать опции файлового меню, перед вами откроются все функции во всей своей полноте. В любой другой программе столь сознательное запутывание пользователя считалось бы грехом. Но есть в *Fotoxx* некое веселое чудачество, и перед ним трудно устоять.

## GIMP ★★★★☆

Кхе... что бы такого написать о *GIMP*, чего еще не писали? Большинство улучшений в версии 2.8 касаются движка редактирования и среде GEGL для модулей расширения, приводя к аппаратному ускорению и управлению цветом с плавающей точкой. Основной интерфейс по-прежнему суров – кажется, что он создавался ради запугивания новичков до полной подавленности. Снизошли и до уступок жалобщикам насчет простоты в использовании. Неплохое начало – функция единственного окна, которая фиксирует положение перемещаемых панелей инструментов, и можно отключить лишние для вас диалоговые окна. Но зато *GIMP* умеет делать все, что угодно, от простой корректировки экспозиции до маскировки и вырезания целых частей изображения для вставки в другое место. Нарекание по поводу интерфейса одно: он, похоже, по-прежнему перемещает D3 при каждой загрузке, прежде чем решить, какие модули он откроет. Возможно, слои и гистограммы? Кто знает...

# Модули расширения

## Редактировать изображения вы можете, а еще что?

Идеальных фоторедакторов нет, и среди прочего профессиональные инструменты от любительских отличает поддержка расширений, которые добавляют функций или работают как макрос для достижения определенных эффектов. Модули-плагины, увеличивающие контраст и делающие цвета блеклыми, имелись для *PhotoShop* задолго до появления *Instagram*.

Так, в *GIMP* есть большая и солидная библиотека плагинов, включая те, что придают ему сходство с *PhotoShop*. А *Fotoxx* рассматривает плагины как команду настраиваемого меню для запуска внешнего редактора. В *Shotwell* и *digiKam* большинство имеющихся плагинов предустановлены – и многие обеспечивают доступ к сайтам фотохостинга, не покидая свою среду. В номенклатуре *Darktable* и функции, и набор регуляторов изображения – это плагины. Он отличается высочайшей расширяемостью, позволяя добавлять функции к набору по умолчанию – это если таковые есть, поскольку скачивать особо нечего, как мы поняли. Если *Darktable* сумеет привлечь достаточно большую аудиторию, плагины уж точно появятся.

Когда *AfterShot Pro* выступал под именем *Bibble*, было полное энтузиазма сообщество разработчиков плагинов, как свободных, так и коммерческих. Хорошая новость: оно вместе с проектом перешло к Corel, и на форумах полно самодельных пакетов для геотэгов, обрамлений и общей работы с изображениями.

### Вердикт

**AfterShot Pro** ★★★★★
**GIMP** ★★★★★
**digiKam** ★★★★☆
**Darktable** ★★★☆☆
**Shotwell** ★★★☆☆
**Fotoxx** ★☆☆☆☆

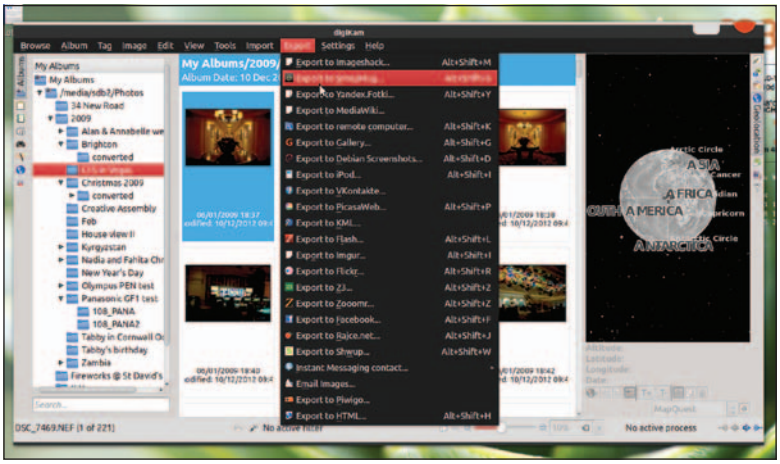
» Расширяемая среда характерна для Linux, и все, кроме Fotoxx, ее поддерживают.

# Конечный результат

## Довольны результатами редактирования? А как насчет формата файлов?

Здесь насущны два вопроса. Во-первых, каково качество итогового фото? И что с ним делать дальше?

*GIMP* пока не обладает способностью 16-битной точности цветопередачи, и для профессионалов, работающих с печатью, это проблема. А еще он с раздражающим упорством держится за свой формат файлов, **.xcf**. Почти все в *GIMP* 2.8 улучшено, кроме решения о переносе других форматов файлов из Save в диалоговые окна меню Export. Тут есть логические основания, но это мешает привычному рабочему процессу и вводит два лишних уровня диалогов, чтобы всего лишь вывести JPG в формат, из которого он открыт. А вот *DigiKam* по части формата файлов являет собой полную противоположность, и с радостью загружает фото на сайты и скачивает с них. Некоторые из его онлайн-плагинов не надежны, и шансы неповреждения тэгов и метаданных переменны. У *Shotwell* меньше онлайн-плагинов, но главные социальные сайты охвачены прекрасно.



› ***DigiKam*** **предоставляет экспорт уймы форматов файлов, и умеет загружать и скачивать изображения с разных сайтов фотохостинга.**

Для конвертора RAW у *AfterShot Pro* удивительно богатый набор опций конечного результата. Здесь нет плагинов для сайтов, но можно создать все, что угодно, вплоть до 16-битных TIFF и готовых web-галерей или обзорных листов. Для многих снимков не требуется внешнего редактора, чтобы предоставить отличный снимок с камеры клиенту. Но не обошлось без причуд: диалоговое окно вывода результата чересчур сложное и предлагает добавить эффекты, типа увеличения резкости, без предпросмотра. В свежем обновлении *AfterShot Pro* разработчики учли самый крупный недочет: цветовой баланс изображений по умолчанию не стал натуральнее, но и не столь сногсшибательно «контрастен», как раньше. И стало проще сразу получать качественные снимки.

Что подводит нас к главной критике насчет *Darktable*: при видимой простоте интерфейса он сложен в применении, что увеличивает шанс ошибок. Так, изображение сразу после редактирования не сохранить – вы возвращаетесь к отображению в виде эскизов, затем находите Export Selected Images. Это подходит для ряда рабочих процессов, но снижает гибкость. В плане качества *Darktable* способен на результаты не хуже, чем у *AfterShot Pro* – если вы разберетесь с управлением.

### Вердикт

**AfterShot Pro** ★★★★☆
**GIMP** ★★★★☆
**digiKam** ★★★★☆
**Shotwell** ★★★★☆
**Darktable** ★★★☆☆
**Fotoxx** ★★★☆☆

» Большинство наших редакторов дают вам желаемое, но сложнее, чем вы рассчитывали.

# Многопоточность и производительность

## Интересные функции – это мило, но быстро ли они работают?

Сенсоры камер увеличиваются, большинству из нас нравится работа в форматах RAW, и ограничения полосы пропускания больше не причина для урезания качества и размера изображения перед загрузкой его на сайт. И мы делаем гораздо больше фотографий, чем раньше. Файлы изображений меньше не становятся, и наши библиотеки быстро растут. Поэтому фоторедакторы ведут себя как участники гонки Красной Королевы из «Алисы в Зазеркалье»: им приходится работать эффективнее, чтобы не потерять в качестве.

Кроме *Fotoxx*, все наши программы отличаются многопоточностью и могут использовать более одного ядра процессора. *DigiKam* и *Shotwell* очень резво работают с большими библиотеками фотографий, помогая найти нужный снимок. Однако ни один из них не идеален: в *Shotwell* имеются ошибки, и он периодически тормозит, а интерфейс *digiKam* часто становится камнем преткновения. Например, для открытия файла RAW придется тащиться через нудное и старомодное окно импорта, вместо того, чтобы сразу перейти к основным инструментам редактирования.

Говоря о редакторах RAW – *AfterShot Pro* невероятно быстро справляется с каталогизацией и редактированием файлов, и берет на себя максимум этапов вашего рабочего процесса. Однако в работе со слоями изображения он изрядно тормозит, и вам, вероятно, захочется вернуться к *GIMP*, чтобы добиться высокоточного редактирования. А вот *Darktable* весьма легок на ногу при просмотре эскизов, но стоит начать редактирование слоев изображения – его прыть скисает, и он становится несносно вялым. Даже простое увеличение масштаба длится слишком уж долго (и нет бегунка, чтобы вы могли видеть, насколько увеличился масштаб).

Триумфом последнего релиза *GIMP* стала поддержка многопотоковых процессоров и – если вы готовы заняться настройкой – OpenCL для ускорения GPU. В результате вы не найдете в *GIMP* 2.8 ничего похожего на рутинную работу, если только вы знаете, что делать.

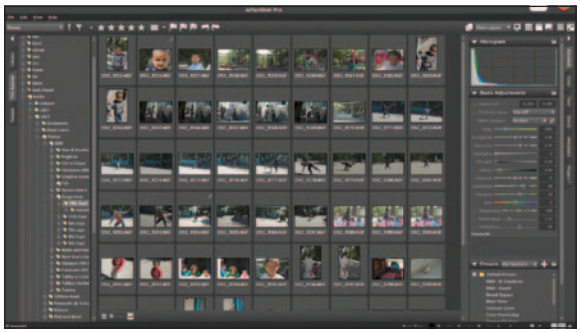
### Вердикт

**AfterShot Pro** ★★★★★
**GIMP** ★★★★★
**digiKam** ★★★★☆
**DarkTable** ★★★☆☆
**Fotoxx** ★★★☆☆
**Shotwell** ★★★☆☆

» Многопоточность очень важна, но она – не единственный фактор.

Фоторедакторы

# Вердикт



› *AfterShot Pro*: ах, если бы он был свободной программой...

Мы здесь не коснулись качества изображений. *GIMP* остается единственным серьезным инструментом для всесторонней работы; *Darktable* – реальный конкурент *AfterShot*, но ему не хватает мощного CPU за спиной.

Ни один инструмент не в силах справиться со всем: попытки провести снимок по пути от жесткого диска до конечного результата единым духом неизбежно где-то терпят неудачу. Управление библиотеками *AfterShot Pro* чрезмерно для повседневных нужд, если просто просматривать старые снимки, а обработка RAW в *digiKam* может довести вас до белого каления.

Лучший выбор для работы со снимками RAW – *Shotwell*, благодаря управлению изображениями, и *AfterShot Pro* или *Darktable*, благодаря обработке изображений.

Для случайных пользователей – *digiKam* прекрасно работает и в KDE, и в Gnome, хотя вам, вероятно, захочется залепить пленкой две трети его значков. Инструменты редактирования *Shotwell* чересчур примитивны для большинства, а *digiKam* может показаться перебором.

Что приводит нас к *GIMP*. То, что он столько лет остается единственным реальным выбором и для беглого, и для полномасштабного редактирования, может показаться упущенной возможностью, но единственная программа, которая может выжить без его функций ретуширования, это *AfterShot Pro* – и даже здесь проще вернуться к *GIMP* для более тонкой работы. Нечасто бывает в Linux всего один способ что-то сделать, но нам кажется, что *GIMP* должен быть установлен у всех, кто занимается редактированием фотографий. Отсутствие 16-битной точности цветопередачи – это ограничение для профессионалов, но для обычных задач в нем есть все, что нужно, и даже более того.

А что же *AfterShot Pro*? Нам больно это признавать, но явно сказывается тот факт, что это продукт крупной компании с большими ресурсами. Только *GIMP* приближается к его уровню разумного выбора дизайна и безошибочной реализации, но даже тогда он норовит свернуть в свой странноватый мир. Мы благодарны Corel за поддержку для Linux столь превосходной программу.

*Fotoxx* мил своими причудами и имеет совершенно безумный набор функций для приложения, столь простецкого на вид при первой загрузке, но пока не созрел для конкуренции с известными программами.

> «Нечаст в Linux всего один способ что-то сделать, но GIMP должен быть у всех.»

## I AfterShot Pro ★★★★★

**Сайт:** www.aftershotpro.com

» Как ни больно это признавать, но проприетарный AfterShot доказывает, чего он стоит.

## II GIMP ★★★★☆

**Сайт:** www.gimp.org

» Обновление GIMP исключительно тонкое, но во многом это большой шаг вперед.

## III digiKam ★★★★☆

**Сайт:** www.digikam.org

» Мощный и профессиональный, digiKam блестяще управляет библиотеками.

## IV Shotwell ★★★☆☆

**Сайт:** www.yorba.org

» Менеджер Gnome по умолчанию со всеми основными функциями; навести бы лоск.

## V Darktable ★★★☆☆

**Сайт:** www.darktable.org

» Все, что нужно для обработки изображений, но с большими пакетами тормозит.

## VI Fotoxx ★★☆☆☆

**Сайт:** www.kornelix.com/fotoxx.html

» Умелая и многообещающая альтернатива Shotwell, хотя требует доработки.

# Рассмотрите также...

Альтернатив для обработки фото хватает, но применимы они с переменным успехом. Для простого редактирования есть незатейливые программы рисования, типа *Pinta*, *GNU Paint*, *KolorPaint* и пр., но стоит ли ограничиваться их функциями управления, если нужно только изменить размер? Нечто помощнее – *RawTherapee*; при работе с изображениями RAW его настраиваемость не уступит *Darktable*. Среди старых фаворитов, все еще достойных внимания – *F-Spot* от Gnome; хотя его популярность идет на убыль, хоронить его рано.

Программа рисования KDE, *Krita*, явно упущена из вида. Ее цель – скорее создание произведений искусства, чем ликвидация эффекта красных глаз, но здесь есть поддержка 16-битных цветовых пространств и масса столь же продвинутых функций, чтобы вы, осмелев, занялись ею. Реальной конкуренции надо ждать от онлайн-приложений – не меньше, чем для приложений рабочего стола. Социальные сайты дают основные инструменты редактирования фото, среди них есть похожие на *Pixlr*, которые используют скорость и гибкость HTML5, и зачем вообще редактировать фотографии на жестком диске? Пока их время не пришло, но это не значит, что оно не придет.

Пожалуйста, сообщите нам свое мнение о фоторедакторах: lxf.letters@futurenet.co.uk. LXF

# Linux vs Windows 8

**Мы знали, что она плоха, но чтобы настолько?! Маянк Шарма разбирается со свежей попыткой Microsoft.**

**Релиз Ubuntu 12.10 появился как раз тогда, когда пользователи Windows в недоумении таращились на совершенно несходный с предыдущими релизами Windows 8 и взвешивали свои возможности.**

Согласно бродящим в Интернет отчетам, прогнозы для Windows 8 были весьма неутешительными: Гартнер [Gartner, исследовательская компания в сфере информационных технологий, – *прим. пер.*] даже предсказал, что порядка 80 % пользователей вообще его не примут.

Похоже, абсолютно все согласились на том, что самым большим изменением в Windows 8 стал новый экран Start, изначально названный Metro Interface, но в окончательном релизе переименованный в Modern UI.

Мы в Башнях **LXF** провели свои собственные сравнительные тесты, еще в **LXF159**, и обнаружили, что Ubuntu побивает своего соперника от Microsoft по всем статьям. Излучая самодовольство, мы наблюдали, как наши подопытные, в большинстве своем – давние пользователи Windows, еле справлялись с тем, что было пустяком в Windows 7. Спустя четыре месяца после выхода и Ubuntu 12.10, и Windows 8 мы решили оценить ситуацию снова.

> «Подопытные еле справлялись с тем, что было пустяком в Windows 7.»

Нельзя сказать, что только недостатки Windows 8 делают Ubuntu 12.10 лучшим выбором. Этот дистрибутив рабочего стола прошел долгий путь с момента представления своего радикально нового интерфейса, и он гораздо эффективнее и дружелюбнее к пользователю. Даже пользователи, в свое время шокированные Unity, имели возможность воссоздать привычные для себя интерфейсы, сохранив все радости основной ОС. Увы, структура ОС Windows OS не предполагает подобной гибкости.

## Унифицированные релизы

Среди изменений интерфейсов и Windows 8 и Ubuntu ясно прослеживается одно: обеим операционным системам нужен унифицированный пользовательский интерфейс для всех устройств. Но Windows 8, пытаясь ублажить пользователей рабочего стола, в своем виде Desktop внедрил в отличную схему странную аномалию, которая теперь грозит ему взрывом.

Разработчики Ubuntu, напротив, всегда сплоченно считали Unity единственным общим интерфейсом для всех устройств, работающих на Ubuntu. Пару лет назад это могло казаться неудобством, но сейчас, когда Ubuntu обретает популярность на портативных устройствах, таких как планшетные компьютеры и телефоны, это однозначно обеспечивает большую продуктивность вашей работы, поскольку у вас один и тот же интерфейс на всех ваших устройствах и вам не надо перестраиваться.

Еще одна проблема с Windows – искусственное разделение его релизов. Отдадим ему должное – в отличие от предыдущих релизов, у Windows 8 все же меньше версий. Большинству пользователей придется выбирать между двумя опциями: простой версией Home, где нет шифрования файловой системы и удаленного сервера рабочего стола, и версией Pro, в которой это все есть.

Ubuntu, как и большинство других дистрибутивов Linux, поставляется в единственном варианте, который можно установить на нетбук, ноутбук или настольный ПК, плюс серверная версия. Также, в отличие от Windows, стандартный дистрибутив Linux идет со всеми компонентами, необходимыми для продуктивной работы с ним на всех этих устройствах.

Дожидаетесь ли вы обновления Windows 8 или подумываете о возврате к более старому релизу, Ubuntu 12.10 и его собратья являют собой отличную альтернативу. Вам мало доказательств? Читайте следующие страницы, и вы увидите, что Linux превосходит Windows 8, нисколько не уступая в производительности.

# Рабочий стол

## Как успехи у интерфейсов?

Бесспорно, самая обсуждаемая функция нового релиза Windows 8 – его интерфейс. Исторически он был единственным, что постоянно сохраняли релизы Windows с прошлого века. Даже пробудившийся от криогенного анабиоза пользователь Windows 95 мог воспользоваться Windows 7 без всяких затруднений.

В Windows 8 все изменилось. Если вы не впали в ступор от нового экрана Start, то уж точно впадете при появлении с виду традиционного Desktop, на котором отсутствует кнопка Пуск. Переключитесь пару раз с экрана Start на Desktop и обратно – и вам захочется обратно в криогенную камеру.

Радикальные изменения интерфейса – вещь для пользователей Linux привычная. В некотором пункте мы все пали жертвами превосходства стиля над содержанием, будь то в KDE 4, Gnome 3 или Unity. Но интерфейс Windows 8 – это куда более радикальный уход от своих предшественников.

Для начинающих наши рабочие пространства по-прежнему придерживаются структуры интерфейса в стиле WIMP (Windows, Icons, Menus, Pointers). Пускай кнопки и переехали на другую сторону окна, но нам, по крайней мере, не приходится осваивать новые трюки, чтобы их закрыть. И наши панели инструментов, обозначенных только значками, много лучше, чем идея Windows 8 заменить значки мозаикой.

Но главным промахом Windows 8, похоже, стали его двойные интерфейсы. Намерения Microsoft были похвальны. Они соорудили прибежище для пользователей на время привыкания к новому, мозаичному интерфейсу. Однако уже через пару минут мы поняли, что стратегия двойного интерфейса – прямой путь к катастрофе для удобства пользователя. Две среды Windows 8 ведут себя совершенно по-разному, что не способствует формированию навыка работы у пользователя. Вдобавок игнорировать одну среду, обходясь исключительно другой, у вас не получится. Придется лавировать между двумя, так как есть вещи, которые можно сделать только в одном интерфейсе, и есть вещи, достижимые только в другом.


› Windows 8 смотрится как планшетная ОС, насаженная поверх Windows 7 с парой доработок.

Кроме того, приложения, запускаемые из нового интерфейса Windows 8, работают в полноэкранном режиме, что весьма неплохо для планшетных компьютеров с их ограниченной площадью. Однако пространства широкоэкранного монитора full-HD вполне достаточно, чтобы разместить многочисленные приложения и окна, и навязанное полноэкранное поведение выглядит оскорбительным. А поскольку в режиме полного экрана не отображаются кнопки управления, позволяющие свернуть или закрыть приложения, придется освоить новые навыки и жесты для управления окнами.

Сравните это с первой версией Unity в Ubuntu 11.04. Разработанный изначально для нетбука, этот интерфейс старательно экономил «жилплощадь», скрывая Launcher при запуске приложения. Что вызывало ярость, хотя вы и могли изменять размер окон и уместить несколько окон на экране; и это поведение по умолчанию в следующем релизе было изменено.

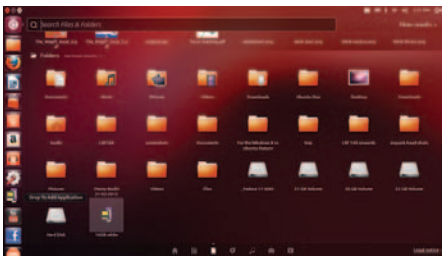
## Разбитая мозаика

Еще одна функция Windows 8, которую не назовешь успешной – мозаика. На первый взгляд выглядит она как значки приложений, пока вы не осознаете, что некоторые приложения могут также иметь свои мозаичные плитки, которые обновляются, получая новейшую информацию для приложения, например, список недавних сообщений электронной почты.

Мозаичная плитка может также быть папкой или ссылкой на библиотеку или на сетевой ресурс. Но если они вам кажутся превосходными значками быстрого запуска, ждите еще одного сюрприза: нельзя создать мозаики для файлов, например, для PDF, которые имеют для вас большую важность и к которым вы периодически обращаетесь, и хотели бы разместить их на экране Start.

И хотя плитки легко упорядочить, раздражает их ограниченная настраиваемость. Плитке можно разве что задать один из двух предустановленных размеров. Кроме того, легко упустить какие-то


› До покупки музыки в музыкальном магазине Ubuntu можно сперва ознакомиться с треками.



› В линзах Ubuntu появились значки для добавления файловых систем, например, устройств USB.

### Выберите свой Linux

Ubuntu 12.10 – один из самых популярных дистрибутивов Linux, который чисто случайно вышел почти одновременно со знаковым релизом Windows . Но для тех, кто решил покинуть Windows 8, Ubuntu – не единственный шанс.

Еще одна популярная возможность для пользователей рабочего стола – Fedora с рабочим столом Gnome 3. Пользователи, которых не прельщают Gnome 3 и Ubuntu Unity, выбирают Linux Mint с его рабочими столами Mate и Cinnamon. Дистрибутивы, использующие рабочий стол KDE, например, OpenSUSE и Chakra, тоже достойны вашего внимания.

Системные требования Windows 8 не чрезмерны, но все же требуется довольно современный компьютер. Однако если вы пока не готовы расстаться со своей старой рабочей лошадкой, есть несколько дистрибутивов Linux, которые прекрасно будут работать на старом компьютере. Наши фавориты для подобных машин – Bodhi Linux и Puppy Linux.

функции – например, ту, что присваивает имена колонкам мозаики. И эта обманчиво-простая задача потребует от вас весьма сложной гимнастики с мышью, что справедливо для большинства задач, если вы используете в Windows 8 мышь. Windows 8 автоматически создаст плитку для любого нового приложения, которое вы установите, тогда как в Ubuntu вы можете выбрать, прикреплять приложение к Launcher после установки или нет.

## Unity спешит на помощь

Unity в Ubuntu тоже не похож на традиционную ОС. Но все же не столь чужероден, как экран Windows 8 Start. Два самых выдающихся компонента на Ubuntu Desktop – это Launcher и Dash. Считайте Launcher панелью задач исключительно со значками. Для быстрого доступа можно прикрепить к Launcher значки приложений, которыми вы пользуетесь чаще всего. Значки некоторых приложений в Launcher снабжены настраиваемым контекстным меню, которое появляется после щелчка правой кнопкой и обеспечивает доступ к разным интересным функциям приложения. Так, значок браузера *Chromium* в Launcher может запускать новое окно в режиме Incognito или со временным профилем. Unity предлагает очень удобную среду как для сенсорных, так и для традиционных интерфейсов. Можно, например, изменить расположение значков в Launcher, подцепив их пальцем или мышью и передвинув на новое место.


› **Если у вас открыто несколько окон, переключение в режим Spread (Super+W) отобразит их всех.**

> «Unity совершенствуется, невзирая на мучительные пересмотры.»

А есть еще Dash, позволяющий искать приложения, файлы, музыку и видео. Некоторые приложения, например, *Gwibber*, устанавливают собственные линзы, увеличивая удобство Dash. В новом релизе Ubuntu имеются также Unity Shopping Lens, которые показывают, что вы можете приобрести онлайн на Ubuntu One и Amazon.

С ходу становится понятно, что Windows 8 предназначен для мобильных устройств с сенсорными экранами. Все меню расположены по краям экрана, в пределах досягаемости для пальцев. Удобство использования тоже в первую очередь ориентировалось на работу тачскрина, а потом переносилось на традиционные устройства, типа мыши и клавиатуры. Закрыть приложения и открыть меню можно простым тычком в экран, но те же самые задачи требуют многочисленных щелчков мышью и нажатия комбинаций клавиш.

А вот Unity, хотя и не обладает тем же удобством работы с сенсорным экраном, что Windows 8, намного лучше справляется с управлением как традиционными, так и новыми устройствами. Украшенный значками Launcher прекрасно подходит для сенсорного запуска приложений, но также можно осуществлять навигацию по всему рабочему столу Unity с клавиатуры. И еще имеется инновационный интерфейс HUD, используя который, вы избежите раскопок в меню, особенно в приложениях вроде офисных пакетов, где меню многоуровневые.

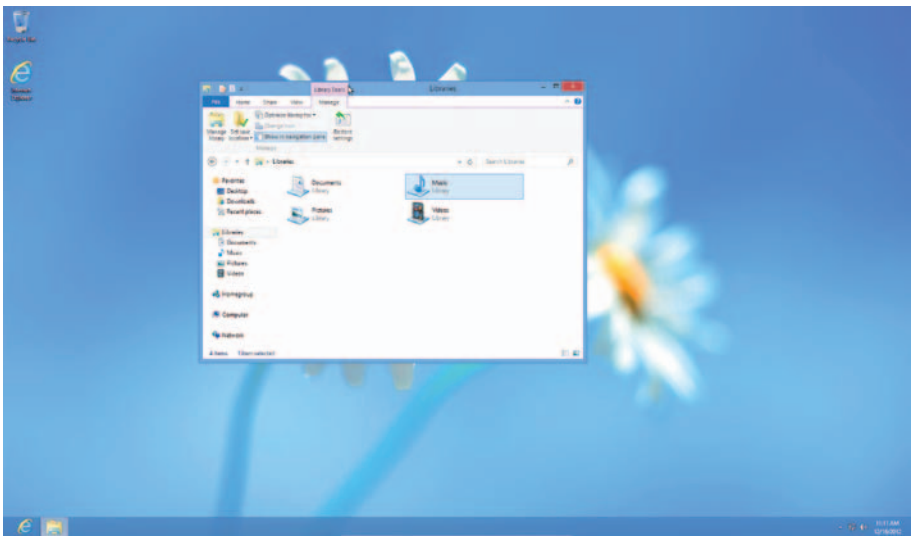
Сравнение нового интерфейса Windows 8 с Ubuntu Unity показывает: в последнем есть все, что надо. Это основной рабочий стол Ubuntu уже года два, и он постоянно совершенствуется, невзирая на рост мучительных проблем неприятия и пересмотров. Критики остаются, но их число явно уступает количеству активных пользователей.

Фактически, в наших экспериментах еще тогда, когда Microsoft делал предварительное ознакомление с Windows 8, наши испытатели Windows ощущали себя более комфортно в Unity, чем в новом интерфейсе Windows 8. С тех пор у Windows 8 были дополнительные демо-релизы без особых изменений в интерфейсе пользователя. С другой стороны, Unity в Ubuntu 12.10 искоренил массу проблем с удобством работы, и сейчас он лучше, чем когда бы то ни было.

## Хитроумный предпросмотр

Лучшее в Unity в Ubuntu 12.10 – различные способы предпросмотра в Dash. В каком-то смысле это новый шаг в эволюции контекстного меню правой кнопки. При поиске через линзы в Dash, Unity предоставит вам интерактивный предпросмотр результатов. Какая здесь отобразится информация и что можно с ней делать, зависит от выбора линзы. Например, линза Files позволит осуществлять предпросмотр изображений и прослушивать аудиофайлы, не открывая их. У вас также есть возможность пересылать по электронной почте определенные виды файлов, если вы настроите в *Thunderbird* свою учетную запись; также вы сможете открывать папку с файлом и сам файл.

Если вы ищете приложения, программа предпросмотра покажет вам краткое описание приложения, в том числе номер его версии, рейтинг и количество изменений. Программа предпросмотра разработана так, чтобы вы избежали лишнего обращения в Software Centre. И если приложение установлено, вы можете удалить его прямо отсюда, а если не установлено, есть опция его отсюда и установить.



› **Панель меню в Windows Explorer заменил интерфейс Ribbon.**

# Приложения

## Что лучше – комплекты программ или программные центры?

Если честно, никто не покупает Windows ради его приложений по умолчанию. Однако Windows 8 ломает эту традицию. Новая ОС включила-таки несколько программ, изначально разработанных для ее нового интерфейса. К сожалению, большая их часть явно не доделана, хоть они и прошли через несколько тестовых релизов.

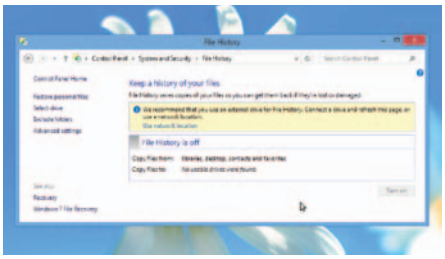
Вероятно, главная проблема в образе действий Microsoft та, что во время установки Windows они требуют предварительно создать бесплатную учетную запись Microsoft Live; но если вы зайдете в Windows через учетную запись оффлайн, вы не сможете использовать никаких новых приложений Metro или магазина приложений, пока не зарегистрируетесь на Microsoft Live.

Как ни странно, версия Windows Media Player, поставляемая с Windows 8, не поддерживает воспроизведение DVD, потому что ОС разработана для устройств без приводов оптических дисков. По тем же причинам в Windows 8 вы не найдете Windows Media Centre. Но вы все же получите программы, а также поддержку воспроизведения DVD... оплатив дополнение Media Centre Pack.

Естественно, подобных ограничений в Ubuntu 12.10 нет, как и в любом другом популярном дистрибутиве рабочего стола. Дистрибутивы уважают лицензионные требования различных кодеков, и для воспроизведения какого-нибудь экзотического формата вам, возможно, придется загрузить кодек из репозиториев, но для этого потребуется всего один-два щелчка – и не придется доставать кошелек.

Как мы уже упоминали ранее, Windows 8 включает приложения, позволяющие вам общаться


**› В приложения Windows 8 беззастенчиво встроена коммерческая реклама!**



**› Новая функция Windows делает резервные копии файлов только в библиотеках Windows.**

с друзьями и коллегами. Вместо *Outlook Express* вы получаете приложения *Mail*, *Messaging* и *Calendar*, и еще есть приложение *People*, которое собирает контакты из разных учетных записей онлайн.

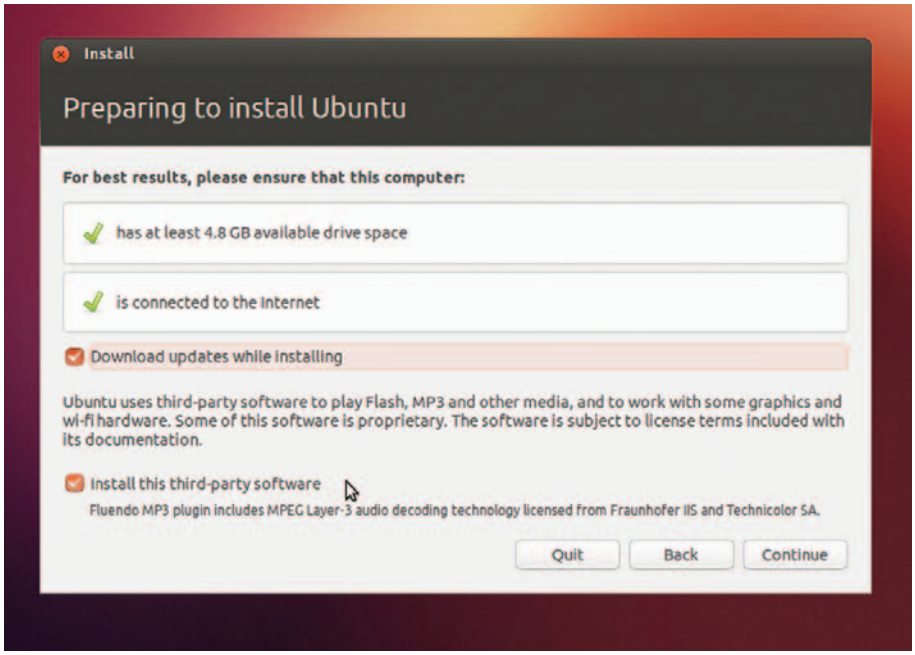
Однако некоторые из этих приложений разят своей ограниченностью. Не считайте их заменой подлинно эффективных приложений. По сути, они сравнимы с соответствующими приложениями на планшетном компьютере с Android. Но, в отличие от Android, Windows 8 предназначена в равной мере и для обычных компьютеров, и тут-то и выясняется, насколько эти приложения хромают.

Например, если вы затеете поделиться новостью из приложения *News*, она будет отправлена через приложение *Mail*, но получатель не сможет открыть ссылку из сообщения ни на какой платформе, кроме Windows 8.

Как приложение рабочего стола, *Mail* обладает, наверное, худшим пользовательским интерфейсом. Если вы запустите его без всяких учетных записей, все, что перед вами появится – пустой белый экран, без всяких значков, кнопок или подсказок, как его настроить. Более того, при составлении нового сообщения у вас будет занят весь экран, и вы не сможете просматривать другие сообщения. Даже сетевой сервис *Gmail* намного разумнее.

Точно так же и другие приложения редко предлагают что-либо за пределами базовых функций. Например, приложение *Messaging* работает только с собственным сервисом *Messenger* от Microsoft и сервисом обмена сообщениями Facebook, игнорируя столь популярные сервисы, как, например, Google Talk.

Сравните это с Ubuntu 12.10, который включает лучшие программы с открытым кодом, в том числе браузер *Firefox*, клиент электронной почты *Thunderbird*, эффективный пакет *LibreOffice*, *Empathy* для быстрого обмена сообщениями, *Gwibber* для микроблогов, фотоменеджер *Shotwell*, *Rhythmbox* и *Totem* для воспроизведения мультимедиа и многое другое!




**› Ubuntu позволяет установить мультимедиа-кодеки во время инсталляции.**

## Онлайн-магазины приложений

Помимо комплектов программ, и Ubuntu 12.10, и Windows 8 имеют магазины приложений, чтобы помочь пользователям найти нужные им программы.

Хотя пользователи Linux находят и устанавливают программы из онлайн-репозиториев с незапамятных времен, концепция Центра приложений, которая выставляет приложения и дает пользователям возможность просматривать их и создавать для них рейтинги, довольно нова. И все же Ubuntu и его Software Centre более чем на три года опередили Windows 8, и это заметно.

С помощью *Ubuntu Software Centre* пользователи могут устанавливать тысячи свободных приложений и приложений с открытым кодом, драйверов и системных библиотек, а также платные коммерческие приложения, в том числе, например, *Wine* или приложение *CrossOver* от CodeWeaver, которое позволяет запускать приложения Windows в Linux.

А вот в *Windows Store* представлены только приложения, но не драйверы. Здесь имеются как бесплатные, так и платные приложения, но некоторые разработчики, среди которых есть такие

## Снято ли с Linux игровое проклятие?

Глядя поверхностно, Windows 8 подходит для игр ничуть не меньше предыдущих версий. Зато разработчики игр смотрят на новую операционную систему Microsoft с серьезными опасениями.

Первый звоночек прозвенел в июле, когда Гейб Ньюэлл [Gabe Newell], представлявший Valve Software на игровой конференции Casual Connect в Сиэтле, заявил: «Windows 8 – это катастрофа в области ПК для всех». Valve – один из самых популярных разработчиков игр для ПК, среди которых – *Half Life* и *Team Fortress*, и им также принадлежит успешная платформа по продажам и распространению игр под названием Steam.

Разработчики инди-игр, например, Маркус Перссон [Markus Persson], создатель такого хита среди игр, как *Minecraft*, согласен с Ньюэллом. В сентябре Перссон написал в Twitter: «Получил сообщение от Microsoft с просьбой помочь "сертифицировать" *Minecraft* для Win 8. Я предложил им перестать разрушать ПК как открытую платформу».

Из себя разработчиков вывели новые правила Microsoft по разработке и распространению приложений для Windows 8 через *Windows Store*, которые называют огораживанием вольера. Поэтому сейчас Valve портирует свою инфраструктуру Steam в Linux, вместе со множеством классных игр.

Ход работ был продемонстрирован на Ubuntu Developer Summit в Дании разработчиком из Valve Дрю Блиссом [Drew Bliss], который добавил, что в Linux есть все технологии, необходимые для самых современных игр. Он также упомянул, что к Valve обратились несколько сторонних разработчиков, которые хотят, чтобы их игры работали в Steam для Linux.

крупные, как Google, сообщили, что они не будут разрабатывать приложения для Windows 8.

Другое ключевое различие между двумя магазинами приложений касается безопасности. Ubuntu 12.10 использует ключи GPG для проверки аутентичности пакетов. Приложения в *Windows Store* проверяются и получают «добро» только по усмотрению Microsoft.

Еще одна область, в которой Windows 8 крупно проигрывает Ubuntu 12.10 – это интеграция. Некоторые из выставленных в магазине новых приложений Windows 8 не особо хорошо интегрированы с Desktop. Например, щелчок по «живой плитке» не даст вам прямого доступа к тому контенту, который на данный момент ею отображается.

> «Windows 8 крупно проигрывает Ubuntu в области интеграции.»

Приложение *Mail* тоже не интегрировано с рабочим столом. И хотя в файловом менеджере Windows имеется возможность отправки выбранного файла по электронной почте, он не работает с приложением *Messaging*. Чтобы отправить по электронной почте файл из Windows 8 Desktop, вам нужно приложение электронной почты рабочего стола, а в Windows 8 такого нет.

В Ubuntu 12.10 вы можете не только напрямую отправлять файлы по электронной почте из файлового менеджера *Nautilus*, но еще и заархивировать их в нескольких форматах для облегчения пересылки.

Кроме того, вы можете отправлять их на свои контакты IM, или на подключенный диск USB, или сетевой диск, записывать на CD, DVD или присоединенные устройства Bluetooth.

Кстати о действиях над файлами: настроив встроенный инструмент резервного копирования *Deja Dup*, вы также можете вернуть отдельные файлы к их предыдущим скопированным версиям. *Deja Dup*, бесспорно, самый простой инструмент резервного копирования на планете. В Ubuntu он также будет копировать файлы на онлайн-сервис хранения Ubuntu One. A Dash умеет давать советы по приложениям, что пользовалось большой популярностью у нашей предыдущей группы испытателей, когда мы сравнивали Ubuntu с Windows 8 перед их выходом, в **LXF159**. В Ubuntu 12.10 эта функция расширена так, что теперь пользователи могут устанавливать приложения прямо из Dash, не обращаясь в *Ubuntu Software Centre*.

› **В Ubuntu Software Centre тысячи свободных приложений.**

Щелкнув правой кнопкой по предлагаемому приложению, вы получите информацию по приложению и кнопку для его установки. Или, если вы хотите выбрать опциональные компоненты, например, модули расширения, просто щелкните левой кнопкой по приложению, и вы попадете в Центр приложений.

### Web-сервисы как приложения

В Ubuntu 12.10 дистрибутив интегрирует web-приложения с остальной частью Desktop. И мы отнюдь не говорим о простом помещении значка в Launcher, способного ненамного больше, чем просто открывать окно соответствующего web-браузера с помощью web-приложения.

Суть идеи с web-приложениями заключается в том, что, несмотря на тот факт, что они работают в web-браузере, они будут интегрированы в различные компоненты рабочего стола Ubuntu Unity, например, в Launcher, систему уведомлений, Dash и HUD.

Например, web-приложение Google Docs полностью интегрировано с HUD. Поэтому вы можете осуществлять навигацию по меню Google Docs с помощью HUD, точно так же, как и в любом оффлайн-приложении, например, *LibreOffice Writer*. Еще есть web-приложение *Last.fm*, которое интегрировано в меню громкости в области уведомлений, точно так же, как *Rhythmbox*, позволяя вам менять треки и переключаться на другие станции.

В настоящее время насчитывается более 30 сайтов и сервисов, которые могут быть интегрированы как web-приложения. При посещении вами сайта с web-приложением браузер предложит вам добавить этот сайт в качестве приложения. Затем вы сможете запустить его из Dash, или присоединить к Launcher, подобно всем остальным приложениям.

Некоторые web-приложения, такие, как Facebook, Twitter и Gmail, интегрируются с меню сообщений, и будут отслеживать любую новую активность на сервисах.

Еще есть сайты новостей, такие, как CNN и BBC, которые отображают новостные заголовки в виде уведомлений рабочего стола.


› **Ubuntu обращается с web-приложениями как с обычными локальными.**

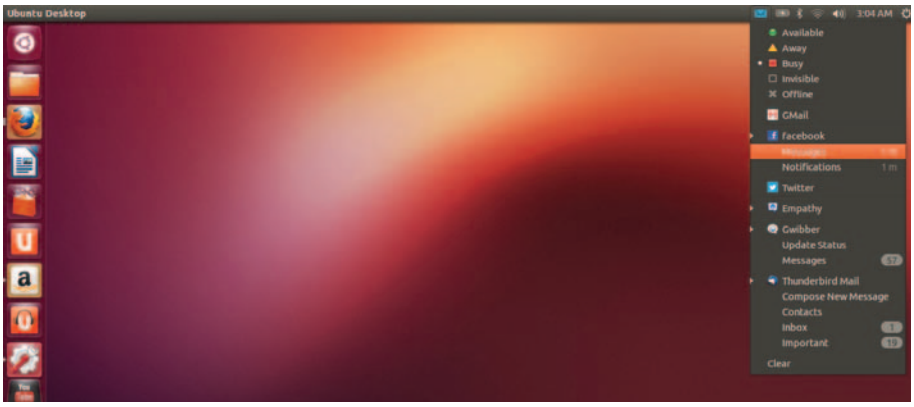
# Интеграция с облаком

## Достиг ли Windows в этой области уровня Ubuntu?

Windows 8 – первая попытка Microsoft интегрировать сервисы на основе облака с рабочим столом. Сюда относится все, начиная с потокового контента с таких онлайн-сервисов, как Twitter, Facebook и Flickr, и заканчивая синхронизацией файлов с локальных дисков с онлайн-дисками.

У Ubuntu есть преимущество перед Windows благодаря большему опыту в этой области. Ubuntu Messaging Menu существует с начала 2010 года. Меню разработано с целью облегчения общения и предоставляет доступ к почте, чату и другим приложениям для коммуникаций. Вы можете настроить свой статус IM на нескольких сервисах для обмена сообщениями, и получить доступ к непрочитанным сообщениям. Он связан с разными приложениями, установленными в Ubuntu по умолчанию, *Empathy*, *Gwibber* и *Thunderbird*, а также с новыми web-приложениями типа *Gmail*.

Приложение *People* в Windows 8 ближе всего к Ubuntu Messaging Menu. Основное различие заключается в том, что в дополнение к отслеживанию всех уведомлений на добавленных вами учетных записях, приложение выступает как универсальный репозиторий контактов. Поэтому оно будет перечислять всех, с кем вы общались в Twitter, а также друзей и членов семьи, с которыми вы постоянно общаетесь; и нет способа объединить их в логические группы. Хорошо в приложении *People* то, что его можно использовать для связи с кем угодно из вашего списка контактов, независимо от того, в какой сети те находятся. Оно также позволяет вам разместить первое или повторное сообщение в Twitter, оценить сообщение или оставить комментарий по поводу последний сообщений. Однако данное приложение помещает каждый отдельный твит в отдельную плитку мозаики, и это – основной недочет.




› **Меню Ubuntu Messaging сообщит вам об активности на всех настроенных вами учетных записях онлайн.**

### Учетные записи онлайн

Используя диалоговое окно Ubuntu Online Accounts, можно создать ссылку на свои учетные записи в разных популярных сервисах, в том числе на Facebook, Flickr, Gmail, Google Docs, Google+, Picasa, Twitter, AIM, Windows Live, Identi.ca и Yahoo!

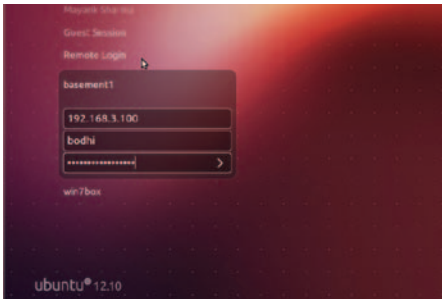
Когда вы добавляете учетную запись, диалоговое окно покажет вам список локальных приложений, с которыми будет интегрирован сервис. И если вы добавили учетную запись Google, информация о логине будет использоваться *Empathy* для ваших контактов IM. Подобным же образом, при добавлении учетной записи Twitter, клиент микроблогов *Gwibber* будет отображать вашу ленту новостей Twitter. Вы также сможете публиковать свои фотографии из *Shotwell* прямо в альбоме в Picasa или Facebook, или на Flickr, если добавите соответствующие учетные записи.

Лучшее всего то, что вы теперь можете искать контент на этих онлайн-сервисах из Dash. То есть линза Search будет заодно отображать документы в Google Docs; линза Photos покажет изображения из Facebook, Flickr и Picasa; а линза *Gwibber* позволит осуществлять поиск частных и публичных сообщений по всем настроенным учетным записям.

И Windows 8 с помощью Microsoft SkyDrive, и Ubuntu с помощью Ubuntu One позволяют пользователям хранить файлы и резервные копии онлайн. Но реализация Microsoft по сравнению с Ubuntu One выглядит ограниченной. Утилита резервного копирования Windows не настолько интегрирована в файловый менеджер, как утилита Ubuntu. Вы можете настроить Ubuntu One на обновление и синхронизацию любой папки на компьютере. Плюс к тому, она умеет возвращать отдельные файлы из онлайн-копий.

Ubuntu One делает намного больше простого копирования файлов, и работает на большем количество платформ, чем Windows SkyDrive, в том числе на Windows, OS X, iOS и Android. Ubuntu One имеет превосходный музыкальный магазин и позволяет распределять доступ к музыке среди нескольких устройств. Приобретая трек, вы получаете возможность воспроизводить его на устройствах на Android или iOS через сеть. Пользуясь этим сервисом на мобильном телефоне, вы можете напрямую поделиться своими фотографиями с такими сервисами, как Facebook и Twitter.


› **Ubuntu One выдает бесплатно 5 ГБ хранилища.**




› **Вход на удаленные компьютеры из окна логина.**

### Мгновенный удаленный вход

Одна из еще более интересных функций для опытных пользователей в Ubuntu 12.10 – возможность войти на удаленную машину прямо из окна приглашения (логина), что облегчает использование Ubuntu как простого клиента.

Еще важнее то, что вы отсюда же можете добавлять и редактировать настройки для многочисленных удаленных машин. Чтобы добавить информацию об удаленном сервере, нажмите на кнопку '?' в окне приглашения, что приведет вас в гостевую учетную запись, и запустите *Firefox* с помощью страницы Ubuntu Universal Client Configuration Server (UCCS).

Как только вы добавите информацию об удаленных компьютерах, они появятся как пункты в окне приглашения. Эта функция позволяет избежать использования рабочего стола, где вам, прежде чем соединиться с удаленной машиной, пришлось бы вручную запустить удаленный клиент рабочего стола и ввести настройки. В настоящее время, используя эту функцию, вы можете только соединиться с компьютерами, на которых работает сервер RDP (Remote Desktop Protocol).

# Индивидуальная настройка

## Легко ли создать себе идеальные условия для работы?

В этом аспекте Windows 8 и Ubuntu различаются весьма значительно. Индивидуальная настройка всегда была неприятностью в Windows, а в Windows 8 эта проблема еще усугубилась. Главный источник проблем для постоянных пользователей Windows – новый экран Windows 8 Start. Причем это не та функция, которую можно отключить. И загрузиться напрямую в Desktop тоже нельзя, что сразу делает ПК с Windows 8 совершенно неподходящими для ситуаций, когда нужно напрямую загрузить приложение, например, для рабочих станций торговых точек.

В готовом Windows 8 слишком много мозаики, и живая мозаика начинает раздражать и лезть в глаза, как только она привязывается к соответствующим учетным записям онлайн и начинает притягивать контент. Да, Windows 8 позволяет индивидуально настроить некоторые аспекты среды по умолчанию. Например, можно изменить размер живых мозаичных значков, переместить их или сгруппировать для упрощения доступа, и даже открепить ненужные, но это, собственно, и все!

Зато Ubuntu Unity, благодаря самой структуре дистрибутива Linux, отличается намного большей гибкостью. Если он вам не нравится, вы можете легко заменить его другим пользовательским интерфейсом – скажем, KDE, Gnome 3, Cinnamon, MATE *Xfce*, *LXDE*... список можно продолжить.

> «Главный источник проблем — новый экран Windows 8 Start.»

### Шарм ли это?

Опции настройки в Windows 8 разбросаны повсюду. Некоторые располагаются в панели Charms, приткнувшейся с правой стороны экрана, и размещает значки для доступа к таким функциям, как управление оборудованием, разделение доступа к ресурсам и системные настройки. Количество настроек, доступ к которым можно получить через панель Charms, отличается от интерфейса к интерфейсу. И если взять эти настройки на экране Windows 8 Start, то они ограничатся только возможностью изменения мозаики. Но если вызвать их из отображения в виде Desktop, то опций у вас будет намного больше, в том числе и ссылка на классическую Control Panel.

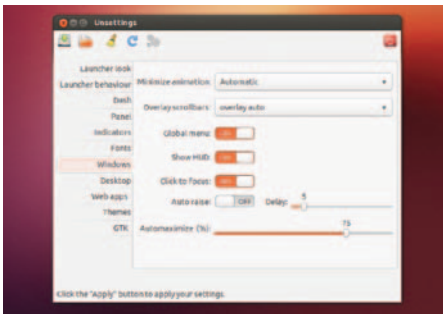
Если вам кажется, что это довольно путано, то есть многочисленные способы выполнения простых задач, таких, как обновление ОС и ее компонентов. Вы удивитесь, но Windows Update не может обновлять новые приложения Windows 8. Чтобы обновить их, нужно перейти в *Windows Store*, а затем щелкнуть по Updates при наличии таковых.

### Правильный способ

Какой контраст с настройками Ubuntu! Все они собраны в опции System Settings, доступ к которой открывает значок с гаечным ключом и шестерней в Launcher. Можно также вызвать индивидуальные настройки из Dash. В System Settings различные настройки объединены по категориям под тремя разделами. В разделе Personal размещены настройки, которыми может воспользоваться по своему усмотрению каждый пользователь и которые относятся только к учетной записи этого пользователя. Среди них такая мелочевка, как обои рабочего стола, размер значков в Launcher и возможность автоматического скрытия Launcher и управления расположением и чувствительностью точки, которая будет снова показывать его. Вы также можете настроить отсюда все ваши учетные записи онлайн, а также облачный сервис хранения данных Ubuntu, *Ubuntu One*. Раздел Hardware содержит настройки для различного оборудования в вашем компьютере. Здесь можете настроить любое устройство, присоединенное к компьютеру, например, Bluetooth, беспроводной сетевой адаптер или принтер, и настроить поведение Unity на нескольких дисплеях. И, наконец, в разделе System размещаются настройки для всей системы, и для доступа к ним могут потребоваться права администратора. Здесь есть опции добавления и удаления пользователей, изменения исходников программ, настройка опций доступа и настройка инструмента резервного копирования *Deja Dup*.

Этих опций уже должно хватить среднему пользователю рабочего стола, но есть еще и сторонние инструменты индивидуальной настройки для опытных пользователей, в том числе постоянно набирающий популярность *Ubuntu Tweak* и *Unsettings* (см. врезку).

› **Навигацию по рабочему столу Unity можно осуществлять с помощью одной клавиатуры.**



› **Для настройки Unity весьма популярны инструменты *Ubuntu-Tweak* и *Unsettings*.**

## Настройка Unity

Приложение *Unsettings* – это один из нескольких инструментов для дополнительной настройки Unity за пределами возможностей, предоставляемых встроенными опциями.

С его помощью можно изменить вид и работу Launcher и Dash большим количеством способов, чем с помощью встроенных инструментов. Этот инструмент также можно использовать для отключения глобального меню Unity и HUD, или изменения перекрытия полос прокрутки. С ним также можно добавлять виртуальные рабочие области и размещать значки Home, Trash и Network на рабочем столе. Также он позволяет легко включать и отключать определенные web-приложения. *Unsettings* нет в официальных репозиториях Ubuntu. Чтобы его установить, введите следующую команду в терминале:

```
sudo add-apt-repository ppa:diesch/testing
sudo apt-get update
sudo apt-get install unsettings
```

Другим популярным инструментом настройки является *Ubuntu-Tweak*, который воскресили из мертвых после отчаянной мольбы пользователей. Но более простым инструментом по-прежнему является *MyUnity*, и его можно установить из официальных репозиториев.

› **Все параметры для индивидуальной настройки Ubuntu аккуратно сложены в раздел System Settings.**

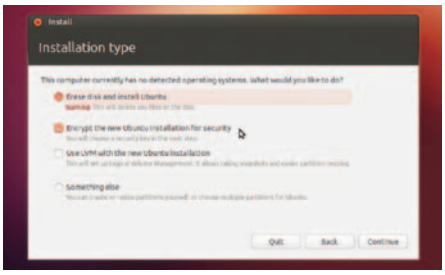
# Безопасность и приватность

## В какой ОС вы будете чувствовать себя спокойнее?

Побочным эффектом стремления операционных систем привнести на рабочий стол онлайн-сервисы стало повышение заботы о конфиденциальности и безопасности. И Windows 8, и Ubuntu 12.10 подвергались серьезной критике со стороны защитников приватности. Но, в отличие от Ubuntu 12.10, Windows 8 сделал весьма немного, чтобы развеять эти опасения.

Помимо спорного сервиса SmartScreen, который теперь выходит за рамки *Internet Explorer* и проверяет также скачанные файлы, ряд приложений Windows 8 также хранят информацию о ваших учетных записях онлайн. Например, при добавке учетной записи Twitter в приложение *People* Microsoft уведомит вас, что будет хранить информацию о вашей учетной записи Twitter на своих серверах, не особо вдаваясь в подробности.

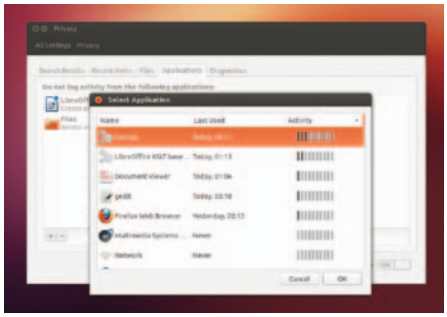
Внутри самой ОС вы вряд ли найдете способы управления конфиденциальностью. Все, что у вас есть – это возможность не позволить приложениям использовать ваше местоположение, имя и аватар учетной записи. Имеется также опция, не позволяющая Windows 8 отправлять URL web-контента, который вы просматриваете с помощью приложений, загруженных через *Windows Store*.



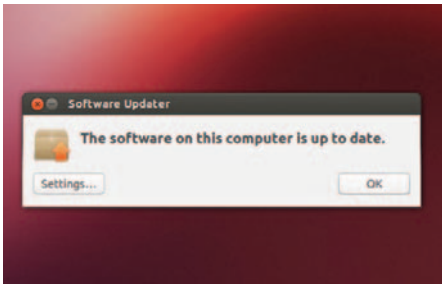

**Ubuntu 12.10 делает полное шифрование диска.**

Ubuntu 12.10 также критиковали за нарушение конфиденциальности, причиной которого стало включение линз Amazon в Dash. Однако большая разница в том, что эту функцию можно отключить, или даже удалить линзы. Все опции, касающиеся защиты конфиденциальности, в Ubuntu размещены в разделе Privacy в System Settings.

Перейдите во вкладку Search Result, чтобы отключить онлайн-предложения при поиске файлов или приложений в Dash. Тогда отпадут результаты для Amazon и для сервиса Ubuntu One Music; но при этом будут показываться приложения, которые вы можете загрузить из Software Centre.



**Онлайн-результаты отключаются в настройках конфиденциальности Ubuntu.**




**Регулярно проверяйте наличие обновлений, чтобы обеспечить надежность своего компьютера с Ubuntu.**

## Что делать, установив Ubuntu 12.10

Установив Ubuntu 12.10, вам, возможно, надо будет последовать следующим советам для обновления и настройки некоторых аспектов дистрибутива, чтобы добиться более эффективной его работы.

Первое, что надо сделать всем пользователям после установки дистрибутива – проверить наличие обновлений. В Ubuntu перейдите в Dash и запустите Software Updater, который просканирует репозитории и предложит вам установить имеющиеся обновления.

Хотя у вас есть возможность установить кодеки для воспроизведения мультимедиа в разных форматах, если вы этого не сделали, можете установить их из https://apps.ubuntu.com/cat/applications/ubuntu-restricted-extras. Если в вашем компьютере имеется оборудование, требующее проприетарных драйверов, например, видеокарта Nvidia, можете включить их из вкладки Additional Drivers в Software Source.

Перекрывающие полосы прокрутки стали одной из характеристик пользовательского интерфейса Unity, которая сильно раздражает некоторых пользователей на больших дисплеях. Чтобы заменить их традиционными полосами прокрутки, откройте терминал и введите команду **gsettings set com.canonical.desktop.interface scrollbar-mode normal**.

**gsettings** пригодится и для изменения других настроек. Если вы хотите видеть на панели имя текущего пользователя, как в предыдущих релизах Ubuntu, введите следующее: **gsettings set com.canonical.indicator.session show-real-name-on-panel true**.

## Нужна помощь?

Лучшее в огромном сообществе пользователей Linux – то, что при проблеме с любым приложением в любом дистрибутиве, вы получите всю необходимую помощь. Все ведущие дистрибутивы рабочего стола снабжены руководствами и справочниками для начинающих. Также у них имеются собственные форумы или списки рассылки (или и то, и другое) и каналы IRC. Дистрибутивы вроде Ubuntu и Fedora, также имеют сайты “ask”, являющие собой базу данных из вопросов и ответов с функцией поиска.

Помимо этих специализированных сайтов, есть также форумы, не ограничивающиеся каким-то одним дистрибутивом – там обсуждаются все вопросы насчет Linux. Самый популярный – вероятно, LinuxQuestions.org: там есть темы, посвященные практически всем дистрибутивам Linux. Там также работает Live Linux Help Desk, который предлагает голосовую поддержку для начинающих пользователей Linux. Загляните также на http://lug.org.uk, чтобы узнать, нет ли локальной (в географическом смысле) группы Linux Users Group неподалеку от вас.

Вкладка Recent Items основана на предположении о том, что при каждом обращении к файлу или приложению она может сохранять некую информацию. На этой вкладке вы можете или ограничить длительность хранения этой информации, или отключить ее записи. А если вы не хотите просто наложить запрет на все файлы и приложения, у вас есть возможность точной настройки критериев выбора. Еще можно перейти во вкладку Applications и выбрать приложения, которые не должны вести запись недавно просмотренных файлов. При просмотре списка приложений Ubuntu обозначит уровень их активности и дату последнего использования. И, наконец, есть еще безвредная вкладка Diagnostics, которая расскажет вам, что Ubuntu собирает информацию об установке и анонимно отправляет ее разработчикам.

Ubuntu 12.10 также позволяет зашифровать свой раздел Ubuntu с помощью пароля. Эта опция отображается во время установки дистрибутива. При шифровании вам предложат ввести пароль перед загрузкой Ubuntu. Но не забывайте пароль, иначе подмонтировать зашифрованный раздел или использовать данные на нем будет нельзя.

Помимо этого, Ubuntu 12.10 включает подписанный загрузчик *Grub 2*. Это позволяет установить дистрибутив на компьютеры, где используется функция UEFI Secure Boot, например, на те, которые идут с установленным Windows 8, без необходимости отключать безопасную загрузку.

Итак, чего же вы ждете? Хватайте Ubuntu 12.10 live CD и знакомьтесь с ним. LXF

# Дейв Нири

# Управлять сообществом

## LXF беседует с «садовником» сообщества Red Hat, Дейвом Нири, знающим секрет, как быть всегда впереди.


**Вот этот секрет: бегите быстрее. Так Дейв Нири [Dave Neary] советует выигрывать гонки. Что не удивляет, если учесть тему его прошлогоднего доклада на OSCON — «Приведите себя в форму: бег как средство повышения производительности», да и сам он — профессиональный тренер по атлетике. И как технарь Дейв тоже состоялся. За последние 10 лет он приложил руку к *GIMP*, Gnome, Meego, а теперь и Red Hat, где он налаживает сотрудничество между сообществами.**

**LXF: Вы описывали свою новую роль в Red Hat как идеальную работу для себя.**

ДН: [*Смеется*] Ну, я несколько лет организовывал сотрудничество компаний и сообществ. Что делаю и в Red Hat. Я работаю со всеми открытыми проектами, которые Red Hat финансирует, помогаю им развиваться на благо сообщества. Не знаю миссии почетнее!

**LXF: Но разве у Red Hat были с этим какие-то проблемы?**

ДН: Хотите верьте, хотите – нет, но специально этим в Red Hat никто не занимался. Пока в Red Hat не взяли людей, умеющих мыслить в масштабах сообщества. А оно, тем временем, продолжало расти. И на сегодня Red Hat объединяет почти 5000 человек.

### О ДОКУМЕНТАЦИИ

«Документация — это способ сообщить пользователям, что вы делаете.»

**LXF: Но ведь наверняка это отчасти пересекается с аудиторией Fedora?**

ДН: Точно. Ну, Fedora тоже не идеал. Думаю, большинство руководителей проектов первыми бы это признали. Динамику продвижения продукта в сообществе всегда есть куда улучшать. Даже при такой богатой истории и наследии, как у Fedora.

Кроме того, в Red Hat приходит много новых людей через приобретение проектов, выросших вне компании, и мы стараемся, чтобы особая культура, особая философия – философия Red Hat передалась и стала присуща им тоже.

Так что работы всегда хватает. Поддержка и развитие здорового сообщества требуют постоянного внимания. Бдительность терять нельзя.

**LXF: Говоря о делах, в чем именно состоят ваши обязанности?**

ДН: Первый проект, которым я плотно занимаюсь – oVirt. В частности, мы работаем над его адаптацией. Над тем, чтобы документация соответствовала тому, что пользователи хотят найти на сайте oVirt – что это за проект, и что он предлагает? И как его улучшить, чтобы сайтом интересовались те, для кого предназначен oVirt? Как расширить аудиторию разработчиков oVirt? Какие еще компании и частные лица заинтересованы, или потенциально заинтересованы в развитии oVirt или другого подобного проекта? И привлечение их к проекту.

Это все внутренняя работа. Не особо увлекательная. Но если смотреть глобально, с oVirt мы делаем большое дело – виртуальный центр обработки данных; идея состоит в том, чтобы параллельно поддерживать сотни виртуальных машин на десятках узлов в режиме ЦОД на базе одного контроллера, одной машины, управляющей всеми остальными.

Движок oVirt используется в качестве контроллера, информационной панели, а каждый следующий узел сети может быть операционной системой как на базе Fedora, так и oVirt Node, а также Red Hat Enterprise. Сейчас мы работаем над тем, чтобы в системе поддерживалась любая ОС – конечно, включая Fedora, которая уже поддерживается; но также Ubuntu и OpenSUSE, поскольку мы действительно хотим, чтобы проект был универсальным.

**LXF: Насколько потенциальным пользователям важна документация?**

ДН: Как хлеб. Для многих проектов, документация – это внутренняя работа, а я считаю, что она выполняет маркетинговые функции. Это способ сообщить пользователям о том, что вы делаете, какие задачи решаете, и им сразу ясно: «Так, этот проект мне подходит».

Именно в этом я вижу задачу сайта, равно как и документации и форумов – донести до людей: вот проект, который будет вам интересен, которым вы хотите пользоваться, который справится с вашей задачей.

**LXF: В том, что oVirt — это открытый продукт, тоже делается ставка на успех?**

ДН: Начнем с того, что бесплатных проектов среди виртуальных ЦОД не так уж много. Но помимо этого, oVirt еще и основан на *KVM*, который, в свою очередь, основан на *libvirt*, а значит, справляется с теми же задачами, что и *libvirt*. Для хранения данных можно использовать блочную запись, NFS, распределенные хранилища вроде Gluster FS; а функции сервера выполняет приложение *Jboss* – оно полностью на Java и полностью открытое. Я полагаю, что компании, заинтересованные в использовании продукта, могут быть заинтересованы и в том, чтобы вносить в него изменения, и вот здесь вступает в силу фактор его открытости: кому это будет интересно?

## О GNOME 3

## «По-моему, Gnome 3 становится лучше с каждым релизом.»

Навряд ли какой-нибудь студент колледжа сможет внести весомый вклад в oVirt. Это не тот проект – чтобы оценить его преимущества, требуется некий минимум оборудования. И мы вновь возвращаемся к роли сообщества в моем представлении – кому выгодно внести вклад в этот проект? И не пытайтесь продать проект, выдав его за то, чем он не является, тому, кто в нем не заинтересован и кто непременно пожалеет, если, связавшись с ним, не получит ожидаемого.

**LXF: А что было не так с MeeGo?**

ДН: Говоря по-простому, с MeeGo случилось то, что Nokia потеряла веру в проект. Вопрос только в том, как это вышло? Мне кажется, это произошло потому, что MeeGo был детищем двух независимых платформ: Moblin и Maemo. В этом отношении мне более импонирует подход Дэвида Ивза [David Eaves] (активист открытого правительства): лучше совместно найти новый подход, чем каждому тянуть одеяло на себя.

**LXF: Вы про Intel и Nokia?**

ДН: Да, про них; мне кажется, что при создании проекта они как раз этим и занимались, определяя, чьи компоненты должны войти в программу. И те, и другие много вложили в проект и не хотели терять свои инвестиции. Это более чем понятно. Результатом стала некая платформа, и если на момент запуска проекта MeeGo до выхода будущих N900 и N9 оставалось всего несколько месяцев, то после старта MeeGo эти модели вышли более года спустя.

Мне кажется, имели место проблемы разницы культур, как бывает всегда, когда речь заходит о сотрудничестве двух крупных организаций, но также и проблемы качественной и слаженной работы.

**LXF: Nokia и Intel даже разработали разные пользовательские интерфейсы, так?**

ДН: Да. И вновь, это решение Nokia было абсолютно обосновано, на мой взгляд. При том, что у них уже был разработан интерфейс для Maemo 6 – и оставалась буквально пара месяцев до выхода устройства – и всю эту работу предстояло выполнить заново. Они решили: «Давайте пойдем на компромисс для этой первой версии, а потом возьмем свое во второй».

Но даже с учетом этого компромисса, интеграция потребовала гораздо больше времени, чем все предполагали.

**LXF: Думаете, с выходом Gnome 3 ситуация стабилизировалась?**

ДН: Думаю, что да. По-моему, Gnome 3 становится все лучше и лучше с каждым релизом. Я сам им пользуюсь, и даже зная, что часть аудитории Gnome 2 перешла на что-то другое, мне кажется, все идет правильно. И Gnome 3, и Unity выбрали верное направление...

**LXF: Что нам показалось печальным...**

ДН: Да, любопытно видеть, насколько похожи стали Gnome 3 и Unity в плане визуального представления и пользовательского интерфейса.

**LXF: Досадно, что Canonical не удалось поучаствовать в разработке Gnome 3, да?**

ДН: Лично я изначально выступал только «за», когда проект стартовал: «Почему нельзя всем просто сотрудничать»? А на случай, когда у каждого свое *в*идение, и не просто о деталях реализации – у нас есть две группы, исследующие вопросы интерфейса – и они очень схожие, вернее, стали таковыми со временем. Вот и посмотрим. Допустим, однажды одна из них найдет лучшее решение; в итоге, оно и победит. Это не соревнование. И не гонка.

**LXF: Это разделение. А ненавистники, как вы замечали в своем блоге, всегда найдутся.**

ДН: Конечно, проще от них отмежеваться и не думать о них. Здесь уместно вспомнить о том, что говорил Дэвид Ивз: не только в Open Source, но и в целом мире все строится на двух вещах: на человеческих отношениях и на взаимодействии. А та статья, на которую вы ссылаетесь – «ненавистники всегда найдутся» – простейший способ разрушить взаимодействие. Вы как бы говорите: «С такими я общаться не намерен, такие мне не нужны». И обсуждение отменяется. А без этого вы никуда не сдвинетесь.

**LXF: Не опишете ли нам вкратце суть вашего бесподобного блиц-доклада о «Принципе Экклезиаста»?**

ДН: Вообще-то это придумал Саймон Фиппс [Simon Phipps], а теперь требует с меня авторские [*смеется*]. Из Книги Экклезиаста 1:9: «Что было, то и будет; и что делалось, то и будет делаться, и нет ничего нового под солнцем», – это вошло в поговорку. «Нет ничего нового под солнцем» стало духом времени. Мы привыкли к этой фразе. И для меня она означает, что у прошлого есть чему поучиться, а в мире свободного ПО мы привыкли считать: все, что мы делаем – ново. Мы пробуем

и видим, что работает, а что нет, и море усилий тратится на вещи, которые уже ранее доказали свою неэффективность в других сферах.

Непосредственно в том докладе я коснулся двух сфер: архитектура и градостроительство. Суть архитектуры в том, чтобы создавать здания, удобные для сообществ, семей. А градостроительства – та же, но в больших масштабах: планирование районов как экосистем, где люди должны работать, сосуществовать и взаимодействовать в едином пространстве.

Есть, в частности, две книги, из которых, по-моему, мы могли бы многое почерпнуть. Во-первых, это “Pattern Language [Язык шаблонов]” Кристофера Александра [Christopher Alexander], где описывается устройство зданий: удачные архитектурные решения и их взаимодействие. Например: «хорошо, когда в гостиной окна на две стороны здания» – это улучшает атмосферу вокруг здания; или – как расположить жилые зоны, чтобы люди знали, где им собираться. Он также объясняет, как эти шаблоны соотносятся друг с другом.

Отдельно я рассматривал сообщества с градиентом интимности. Такой существует в зданиях, где вы постепенно движетесь от публичного к более приватному пространству. Но не наоборот. Случалось ли вам прийти в офис, где нет приемной, и вы вынуждены подойти к кому-то на рабочем месте и спросить: где найти такого-то? Неловко, правда? Потому что вы вторгаетесь в личное пространство. Хотя оно и открытое, но условно-приватное, а вы пришли из публичного. И чувствуете себя неуютно.

Так же и в сообществах. Можно войти туда и спросить: «Как мне сообщить об ошибке?» А вам ответят: «Напишите такому-то, он даст вам доступ к средству отслеживания». И это разрыв градиента – ведь не так легко обратиться лично к незнакомому человеку. Для того, что принято считать условно-публичным ресурсом, вроде отчета об ошибке, личный контакт не нужен.

## О СООБЩЕСТВАХ

## «Чтобы сообщить об ошибке, личный контакт не нужен.»

Второй источник – Джейн Джекобс [Jane Jacobs]. У нее есть книга под названием The Death and Life of Great American Cities [Смерть и жизнь больших американских городов]. В 1950-х она сама занималась градостроительством, а в 60-х переехала в Торонто, чтобы ее сыновей не призвали в армию [во Вьетнам, – *прим. ред.*]. Очень и очень интересная дама. В книге она описывает устройство успешных районов и то, как они ухудшались или улучшались; объясняет, почему в отдаленных спальных районах выше преступность, и т. д. Одна из ключевых ее идей – смешанная функциональность в рамках одной территории. Например, там расположены дома, магазины и офисы – и функционально это будет «здесь я живу», «туда я хожу на работу» или «делаю покупки». Если присутствуют все три компонента, на территории всегда будут люди. Одни идут на работу из дома, другие приезжают, матери ведут детей в школу или из школы, люди выходят пообедать. А вечером, так как есть бары и рестораны, в них тоже кто-нибудь заходит. Всегда есть деятельность, а это создает спрос на добавочную функциональность, делающую район привлекательным для жизни. Бары, рестораны, театры – магазинчики на углу, а не только супермаркеты и торговые центры.

Такой спрос улучшает и безопасность, ведь кругом всегда кто-то есть.

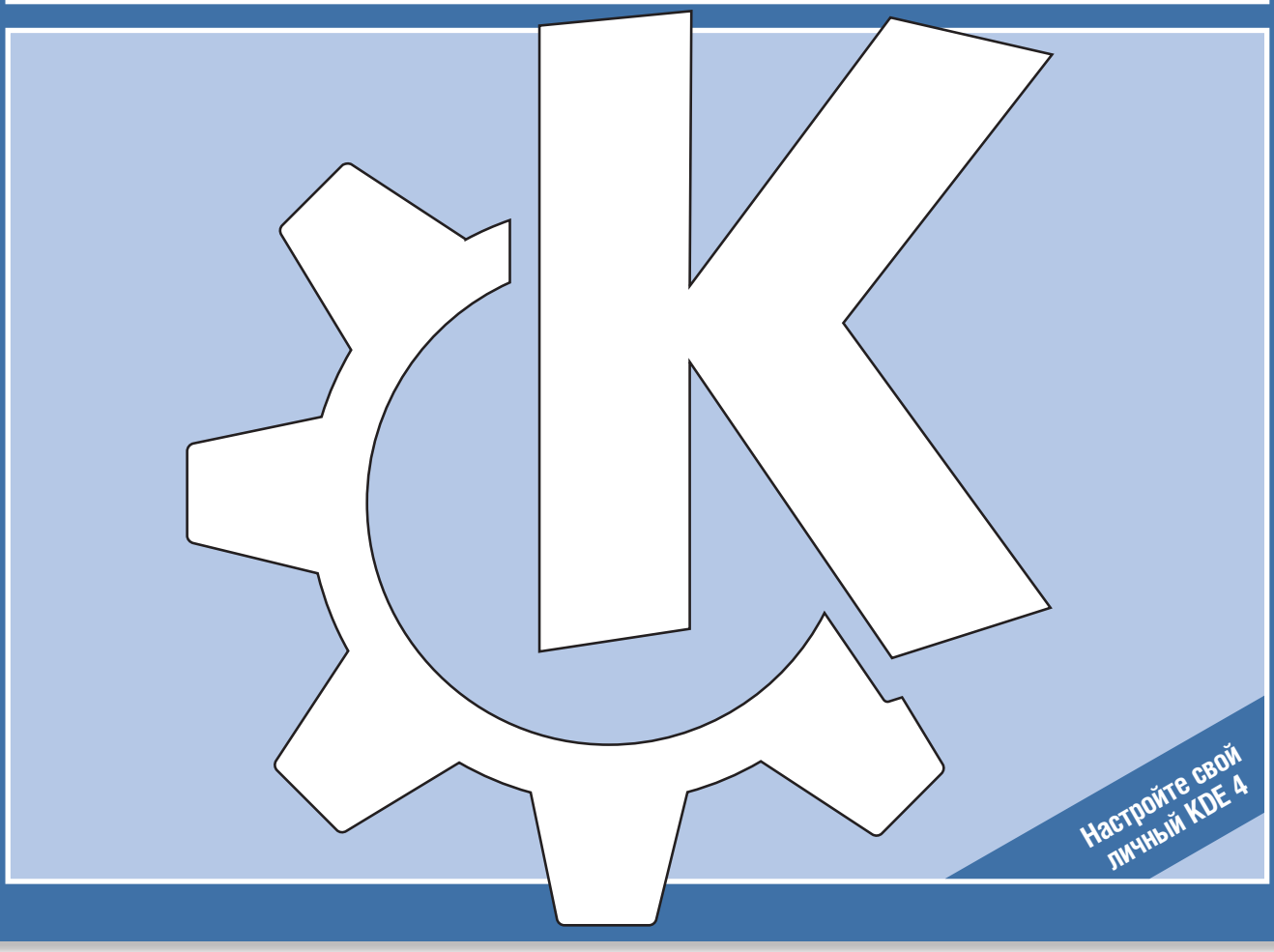
На том же канале IRC ведь не бывает простоя, чтобы кто-нибудь мог просто появиться там, потроллить и распугать публику. Смешанная функциональность проекта означает, что я занимаюсь им «постоянно» или «в свободное время». Потому и хорошо привлекать как профессиональных разработчиков, так и любителей – тогда ваш проект не будет умирать по выходным или по вечерам.

Хорошо, если у вас широкая география, ведь тогда у вас постоянно на уме, во-первых, что он работает во всех часовых поясах, а во-вторых, что мне нужно делать, чтобы задействовать всех, кто есть в проекте, будь они в Африке, Азии или Южной Америке? Наконец, проекты со смешанной функциональностью – которые можно по-разному применять – как правило, более модульные, более развитые и успешные. И для Open Source, я считаю, это очень важно. LXF

# KDE

## Недостающее звено

**Шашанк Шарма предлагает вам путеводитель по рабочему столу с высочайшей в мире настраиваемостью.**




**На первый взгляд, рабочий стол KDE 4 не похож на сородичей. В нем нет ни ярких программ запуска приложений, ни обзора горячих углов, ни панелей управления, ни линз. И хотя некоторые выбирают KDE именно за эти его отличия, из ряда прочих дистрибутивов KDE выделяет глубина индивидуальной настройки.**

KDE всегда стоял за индивидуальность. Сменить положение панели на рабочем столе или заполнить панели инструментов приложениями KDE можно одним-двумя щелчками. Потрудившись самую малость, вы перекрасите KDE 4 под Unity!

Вид рабочего стола KDE 4 по умолчанию привлекательным не назвать. Он не только скучен – он контр-интуитивен и не очень удобен для среднего пользователя. Но обширные возможности настройки делают его невероятно гибким: его легко подогнать под все ваши рабочие потребности. Настроим же себе автономный рабочий стол под названием Activities, со своей структурой и виджетами, через самые общие параметры KDE.

## Настроим «под себя»

Для затравки, сменим вид по умолчанию. Щелкните правой кнопкой по рабочему столу и в контекстном меню выберите Default Desktop Settings. В открывшемся окне выберите View и познакомьтесь с видами из выпадающего списка Layout.

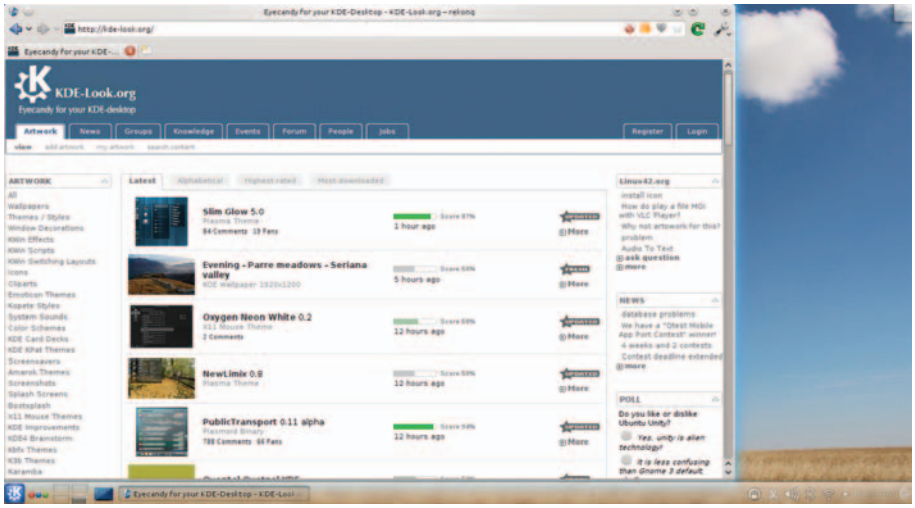
Вид Desktop – это раскладка по умолчанию; она позволяет разместить на рабочем столе виджеты. Если вам ближе рабочий стол ретро-стиля, с разбросанными по нему файлами и папками, берите вид Folder View. Вид Search and Launch идеально подходит для устройств с небольшим экраном.

В зависимости от выбора вида, у вас появятся добавочные настраиваемые элементы на левой панели. Вид Folder предоставляет опции указания местоположения требуемой папки и способ отображения значков. Меню Search and Launch позволит задать категории отображаемых приложений.

На всех компьютерах, кроме нетбуков, я предпочитаю вид по умолчанию и добавляю виджеты в зависимости от применения рабочего стола. А для пользователей-гостей задаю вид Folder View, поскольку тенденцию сохранять и создавать файлы на рабочем столе имеет большинство.

По умолчанию у рабочего стола KDE plasma одна панель, вдоль всей нижней части экрана. На ней несколько виджетов – например, программа запуска приложений *Kickoff*, менеджер задач, пейджер рабочего стола и системный лоток. Можно добавить панели или переместить эту панель в другую часть экрана, если вам так удобнее. Чтобы добавить новую панель, щелкните правой кнопкой по рабочему столу и перейдите к опции Add Panel – она предлагает панели разных типов. На панели по умолчанию располагаются

› **Простые инструменты индивидуальной настройки Linux Mint — одна из главных причин его успехов.**



› **kde-look.org — универсам, предлагающий все необходимое для KDE.**

те же самые виджеты, что и на панели внизу экрана, а на пустой панели ничего нет. Для настройки панели щелкните по значку-кэшью, расположенному с краю, и перед вами откроется диалоговое окно. Можете отрегулировать длину панели, потянув за стрелки. Нажмите и тащите кнопку Height, чтобы отрегулировать высоту, и перемещайте панель с помощью кнопки Screen Edge.

Опция More Settings поможет также изменить выравнивание панелей, которые не занимают всей ширины рабочего стола. По умолчанию панели всегда видны, но вы можете автоматически скрывать их, и тогда они появятся только при наведении на их местоположение мыши. Чтобы удалить имеющийся виджет, выберите этот виджет в диалоговом окне настройки панели и щелкните по X. Здесь вы также можете изменить расположение виджетов, перетаскивая их на другое место.

> «Настроив виджеты по своему усмотрению, выберите Lock Widgets.»

Чтобы добавить на панель виджет, выберите Add Widgets, и перед вами откроется диалоговое окно Add Widgets. Здесь десятки виджетов, а если вам этого мало, поиск в Интернет подбросит еще.

Опция Add Spacer весьма удобна, если нужно разместить виджеты с интервалами между ними. Эти разделители не видны при закрытом диалоговом окне настройки. KDE также позволяет добавлять виджеты на рабочий стол. Один у вас уже есть – он называется Folder View. Этот виджет отображает содержание папки в окне, которое вы можете поместить куда угодно.

Чтобы добавить виджеты, щелкните правой кнопкой по рабочему столу и выберите опцию Add Widgets, которая откроет перед вами знакомое диалоговое окно с десятками виджетов. Потом можете дважды щелкать по виджету или перетаскивать его на рабочий стол. Почти все виджеты предлагают несколько опций настройки. Чтобы перейти к ним, наведите мышь на виджет на рабочем столе. Откроется окно со значками. Щелкните по значку с гаечным ключом, чтобы отобразить опции настройки конкретного виджета.

Опции настройки для разных виджетов разные. Так, опции виджета Folder View позволяют выбрать папку, чье содержимое вы хотите видеть, фильтровать любые типы файлов, и пр. Можно настроить виджеты на панелях, щелкнув правой кнопкой по виджету и выбрав опцию Settings – рядом с ней также появится значок с гаечным ключом. Например, щелкните правой кнопкой по цифровым часам на панели по умолчанию и перейдите в Digital Clock Settings, чтобы настроить вид часов и другие параметры.

Настроив виджеты по своему вкусу, щелкните правой кнопкой по рабочему столу или панели и выберите Lock Widgets. Тогда окна настройки перестанут возникать при каждом наведении мыши на виджет на рабочем столе, и это также пресечет случайное перемещение виджетов. После этого, чтобы добавить виджеты или настроить имеющиеся, сперва придется щелкнуть правой кнопкой на рабочем столе/панели и выбрать Unlock Widgets.

## Виртуальные рабочие столы

Одна из старейших инноваций рабочего стола Linux, виртуальные рабочие столы по-прежнему занимают значительное место в KDE 4. Хотя KDE 4 вводит функцию Desktop Activities, которая переводит виртуальные рабочие столы на новый уровень, эта золотая функция полезна для распределения разных задач отдельным рабочим столам.

По умолчанию менеджер задач в KDE 4 будет отображать все окна открытыми на всех виртуальных рабочих столах. Меня это отчасти раздражает, ибо захламляет менеджер задач. Чтобы изменить это, щелкните правой кнопкой по менеджеру задач и выберите Task Manager Settings из меню Context. В диалоговом окне Settings выберите опции, которые будут показывать задачи с текущего рабочего стола и из текущей деятельности.

Если ваша работа предполагает использование нескольких экземпляров приложения, то опция Do Not Group в выпадающем меню Grouping обеспечит отображение каждого экземпляра отдельно в менеджере задач. Для дальнейшего разделения виртуальных рабочих столов щелкните правой кнопкой по пейджеру, а затем по Pager Settings. Во вкладке General находятся опции для изменения внешнего вида пейджера и его поведения, если вы щелкнете по текущему рабочему столу в пейджере. Я предпочитаю выбрать опцию Shows Desktop и удалить ненужный виджет Show Desktop. Перейдите во вкладку Virtual Desktops и выберите опцию Different Widgets for Each Desktop [Разные виджеты для каждого рабочего »

## Настройте внешний вид

Художественные достоинства и дизайн KDE вам не милы? Измените их инструментом System Settings.

Начните с листа Application Appearance – здесь настраивается вид и работа приложений. Во вкладке Styles выбираются темы для разных элементов, составляющих приложение: кнопок, полос прокрутки, вкладок и т.д. Выбранный стиль также может позволить более тонко настроить другие элементы, скажем, ширину полосы прокрутки. Изменить цветовую схему приложения позволит вкладка Colors. И снова, можно выбрать цветовую схему для всего рабочего стола или отдельных элементов, например, строки заглавия или выделенного текста, и т.д. Во вкладке Icons можно изменить размер значков и выбрать им темы, а также элементы анимации в зависимости от их применения.

На листе Workspace Appearance можно настроить темы элементов рабочего стола: фона панели, пейджера Kickoff и т.д., а также изменить тему курсора мыши, выбрать анимацию всплывающего экрана и сделать многое другое.

Поведением переключателя задач управляет лист Window Behaviour. Для настройки поведения краев экрана перейдите на лист Workspace Behaviour.

стола]. Она позволит вам украсить каждый виртуальный рабочий стол разными виджетами, обоями и типами отображения.

## Desktop Activities

Если вы всерьез решили создать разные рабочие столы для одного и того же пользователя на одной и той же машине, вам в KDE 4 Desktop Activities. Данная функция шлифовалась в разных релизах KDE 4.x и в KDE 4.9 стала весьма удобна.

Рассматривайте Activities [Деятельности] как разные экземплярыв вашего рабочего стола, каждый со своим вариантом структуры окон и своей настройкой для разных задач. Например, у меня есть Activity для письма, там работают *LibreOffice Writer* и виджет Dictionary; вторая, социальная Activity, автоматически направляет меня в Google Talk и на настраиваемый сервер Jabber, а виджеты на рабочем столе отображают обновление статуса и позволяют мне изменять свое присутствие в этих сетях; третья, изобразительная Activity показывает содержимое папки с моими картинками, и имеет виджет, который по всем по ним циркулирует. Название текущей деятельности отображается в правом верхнем углу рабочего стола.

Чтобы создать новую Activity или переключиться на другую, выведите Activity Manager, щелкнув по трем цветным точкам рядом с пейджером. Появится список действий, имеющихся в вашей системе. Большинство дистрибутивов KDE идут только с одной Activity по умолчанию – Desktop Activity. С помощью кнопки Create Activity можно создать другую Activity на базе шаблонов в вашей системе, или других шаблонов, которые можно найти в Сети и скачать с помощью опции Get New Templates.

Создавая Activities на базе шаблона, помните, что им могут потребоваться приложения. Так, Photos Activity использует просмотрщик изображений *Gwenview* и фотоменеджер *Digikam*, а также виджет Picture Frame. При первом запуске Activity отобразит список приложений, связанных с ней.

Каждая Activity, с которой связаны приложения, спросит вас также, хотите ли вы запускать эти приложения при каждом запуске Activity. Хотя можно использовать Activity даже в отсутствие связанных с ней приложений, все же лучше их установить, чтобы задействовать все возможности Activities с максимальной полнотой.


**› Интерфейс KDE для нетбука разработан с максимальной экономией площади экрана.**

Возникает закономерный вопрос: что происходит с приложениями, работающими в Activity, когда они не используются? Ради экономии ресурсов вы можете остановить любую – или все – из Activities, которые на данный момент не нужны. При этом сохранится состояние приложений и файлов, и вы легко получите к ним доступ позднее, щелкнув по этой Activity в Activity Manager.

## Согнать плазмоиды в кучку

KDE также имеет Activities типа Grouping Desktop и Grid Desktop. Они весьма удобны, если у вас много виджетов, разбросанных повсюду и захламляющих рабочий стол. Вместе взятые, эти Activities известны как изолирующие: их цель – держать наборы виджетов в контейнере, независимом от других контейнеров. Вам придется установить эти Activities через менеджер пакетов своего дистрибутива. В Kubuntu они входят в пакет *plasmacontainments-addons*. Activity Manager будет содержать два дополнительных шаблона: Grouping desktop и Grid desktop. Вы также можете использовать эти Activities на отдельном виртуальном рабочем столе, если у вас включена опция Different Widgets for Each Desktop.

Grouping desktop, независимо от того, используете вы его в качестве Activity или на виртуальном рабочем столе, сначала выглядит как обычная Activity без всяких виджетов рабочего стола. Но щелкните правой кнопкой по рабочему столу – и в меню Context вы увидите дополнительную опцию Add Groups. Она предоставляет пять типов групп, которые вы можете добавить к Grouping desktop. Группа – это обособленное единство, содержащее все виджеты. Количество типов групп в Grouping desktop можно увеличить.

Группа Tabbing group позволяет разместить виджеты в нескольких вкладках внутри группы, а Stacking group размещает виджеты один поверх другого. Можете использовать Grid group для размещения виджетов в виде сетки. Используя Grid Desktop Activity или структуру, вы сумеете разместить все виджеты на вашем рабочем столе в виде сетки. Flow group размещает виджеты по столбцам и строкам, а Floating group позволяет перемещать виджеты, размещенные там. Добавленная группа отображается в виде пустого квадратного контейнера на рабочем столе. Наведя на него мышь, вы получите доступ к управлению им, чтобы можно было изменить его размер и переместить его. Чтобы добавить виджеты в группу, просто перетащите их в контейнер. Теперь, если переместить контейнер, виджеты внутри него тоже переместятся.

Мой любимый тип групп – Tabbing group; я использую его для группировки общих типов виджетов. Так, вкладка Clock содержит всевозможные виды часов; вкладка System содержит виджеты, показывающие информацию о компьютере; и т.д. Меняя размеры виджетов внутри этой группы, будьте особо внимательны: вместо обычных кнопок виджеты в ней снабжены маленькими

## Пусть KDE выглядит, как Unity и Gnome 3

Разве не заманчиво разыграть друзей с помощью легендарного умения KDE менять свою форму, и сменить традиционный рабочий стол программой запуска, панелью управления и глобальными меню?

Сперва удалите панель по умолчанию внизу рабочего стола KDE. Затем добавьте новую панель Empty наверху. Теперь добавьте виджет Window Menubar, чтобы он действовал как меню Global в Unity. Также добавьте виджеты System Tray и Digital Clock.

Далее добавьте новую панель Empty слева, разверните список виджетов и добавьте виджет Icon-only Task Manager. Это будет программа запуска. Чтобы присоединить к ней значки любимых приложений, откройте приложение, щелкните правой кнопкой по его значку и выберите опцию Show A Launcher When Not Running. Чтобы значки выглядели, как их собратья в Unity, щелкните правой кнопкой по любому значку, пройдите в Icon Only Task Manager Settings и во вкладке Appearance выберите Use Indicators стиль Colored Background.

Чтобы получить программу запуска по типу Dash, установите Takeoff launcher. Если такого виджета нет в вашем дистрибутиве, возьмите с http://code.google.com/p/takeoff-launcher. Установив, добавьте эту программу на панель слева. Затем щелкните правой кнопкой по его значку, перейдите в Takeoff Settings, снимите отметку с опции Show Takeoff in Full Screen Mode и пометьте опцию Show background image.

Очень легко переместить кнопки управления окнами из верхней правой части окна в верхнюю левую. Перейдите в System Settings > Workspace Appearance, и во вкладке Window Decoration щелкните по кнопке Configure Buttons. Выберите место для кнопки Use Custom Titlebar и затем перетащите кнопки управления окнами в предпросмотре строки заглавия справа налево. Вот и все!

черными квадратиками по краям. При их использовании для изменения размера виджета они автоматически выровняют виджет по строкам и столбцам внутри группы. Чтобы добавить строки и столбцы, наведите мышь на края контейнера и щелкните по зеленому плюсу. Grid desktop работает по той же схеме.

Также по умолчанию Tabbing group предоставляет вам одну вкладку. Можно добавить другие вкладки с помощью значка New Tab в верхней левой части контейнера. Все вкладки по умолчанию именуются New Tab. Чтобы переименовать вкладку, щелкните правой кнопкой внутри контейнера и щелкните по опции Configure this Tabbing Group.

*Kickoff* является программой запуска приложений по умолчанию, которую ввел рабочий стол KDE 4, и предлагает полезные опции настройки. Одним из первых параметров, который я настроил, был ее размер. Для этого зацепите верхний правый угол *Kickoff* курсором мыши и растяните *Kickoff* по диагонали до желаемого размера. Далее идет вкладка Favourites, дающая быстрый доступ к ряду наиболее часто используемых приложений. Вы можете изменить последовательность приложений в списке по умолчанию, перетаскивая пункты списка выше или ниже других пунктов. Удаляется пункт щелчком по нему правой кнопкой и выбором Remove From Favourites.

А хотите – добавьте в это меню свои любимые приложения. Сначала найдите приложение во вкладке Applications. Теперь, вместо запуска приложения, щелкните по его значку правой кнопкой и выберите опцию Add to Favourites. Можно добавить приложение на рабочий стол и на панель с помощью подходящих опций в меню Context.

На традиционную программу запуска приложений KDE переключает щелчок правой кнопкой по значку Kickoff и выбор опции Switch to Classic Menu Style. Можете также установить и использовать другие программы запуска приложений, например, *Lancelot* и *Homerun*. Вероятно, вы найдете их в репозиториях вашего дистрибутива.

> «Kickoff по умолчанию предлагает удобные опции настройки.»

Еще одна хитрость, экономящая время – присвоение наиболее часто используемым приложениям клавиши быстрого запуска на клавиатуре. Для этого щелкните правой кнопкой по Kickoff и щелкните по опции Edit Application; запустится KDE Menu Editor. Найдите приложение, к которому вы хотите привязать клавишу быстрого запуска, в левой панели.

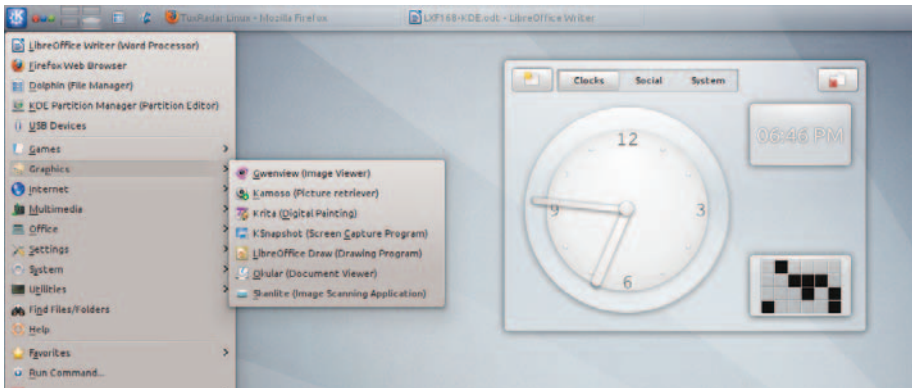
Щелкните по приложению, и в правой панели перейдите во вкладку Advanced. Текущая клавиша быстрого запуска, присвоенная этому приложению, будет обозначена внизу этой страницы. Щелкните по кнопке, отображающей клавишу быстрого запуска, и задайте собственную комбинацию клавиш, нажав на них. Перед выходом из редактора не забудьте нажать на кнопку Save.

Возможно, вы решите запускать приложение со специальным аргументом или сделать CLI приложения, не имеющего значка в меню. Или, возможно, вам проще использовать клавиатуру, чем пытаться найти что-то в меню с помощью мыши.

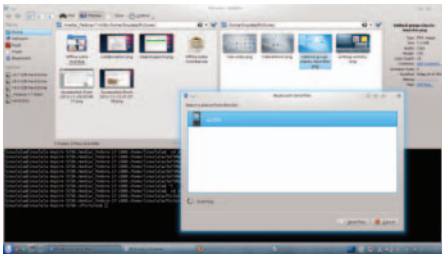
Здесь пригодится *Krunner*. Это удобный инструмент, позволяющий вводить команды, не переходя в терминал. При его запуске (комбинацией клавиш Alt+F2) *Krunner* появится посреди экрана в виде текстового окна, где вы можете ввести свою команду. Чтобы запустить приложение, введите первые три буквы, и вам покажут приложения, совпадающие с написанным. Он удобен также и для отмены приложений. Просто запустите *Krunner* и введите **xkill**, а затем щелкните по любому приложению, которое зависло.

Этот инструмент часто называют мини-командной строкой – и так оно и есть. В *Krunner* можно вводить любые команды. Более того, его можно расширить с помощью плагинов, чтобы он выполнял самые разнообразные задачи – например, поиск информации в Wikipedia или в словарях, или преобразование единиц, или вычисления.

В большинстве дистрибутивов множество плагинов *Krunner* уже установлено. В репозиториях своего дистрибутива вы найдете и другие. В Kubuntu пакет *plasmarunners-addons* устанавливается уже с добавочными плагинами. Чтобы включить или отключить плагины Krunner, щелкните по значку с гаечным ключом слева, и перед вами появится диалоговое окно его настройки.




› **Если вам нравится чистый рабочий стол, разложите плазмоиды по контейнерам со вкладками.**



› ***Dolphin* легко настроить под предпочтения как средних, так и опытных пользователей.**

*Krunner* способен также управлять вашим компьютером: переводить его в спящий режим, блокировать экран и даже менять его яркость. Чтобы использовать *Krunner* по максимуму, ознакомьтесь с синтаксисом командной строки. Просто щелкните по значку '?', и появится выпадающий список с пояснением синтаксиса всех наличных команд. Тщательно изучите все его пункты.

## Обязательный Dolphin

Обязанности по управлению файлами KDE 4 передал от любезного многим *Konqueror* более миловидному *Dolphin*. Как и многое другое в KDE 4, новый файловый менеджер улучшался от релиза к релизу. Для большинства пользователей рабочего стола, которые не хотят смешивать управление файлами и просмотр web-страниц, *Dolphin* работает превосходно.

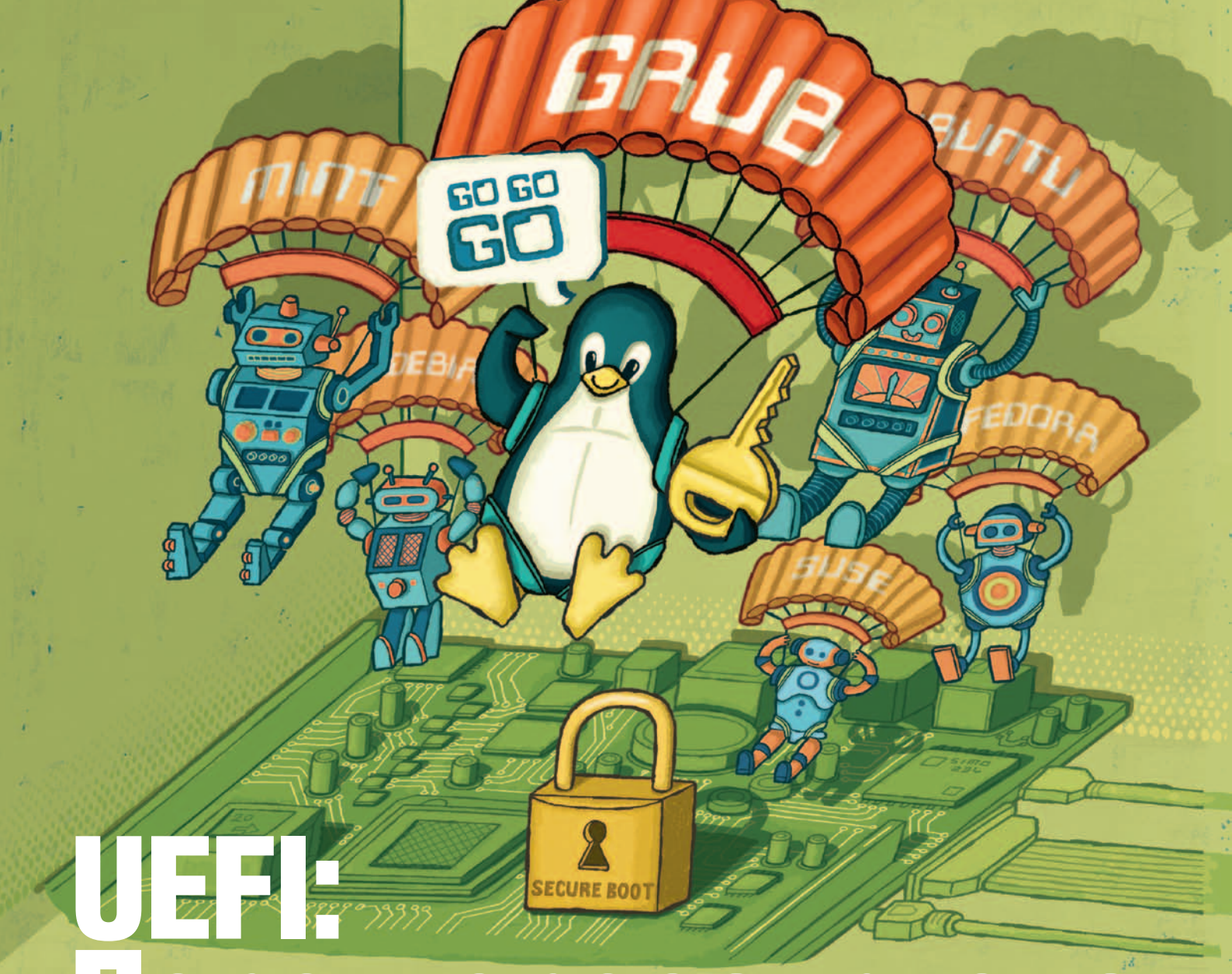
Лучшее в нем – его гибкость. Поскольку я постоянно прыгаю с папки на папку, я предпочитаю поделить вид по умолчанию, который отображает текущую папку, чтобы он отображал несколько папок, в стиле *Midnight Commander*.

Щелкните по кнопке Split, чтобы поделить окно напополам. Радует, что обе панели можно настроить независимо друг от друга. Например, пусть в одной панели режим View показывает подробную информацию, а в другой – скрытые файлы.

Поскольку я работаю с файлами удаленно, мне в файловом менеджере нужна адресная строка. По умолчанию *Dolphin* ее скрывает. Вы можете временно включить ее клавишами Ctrl+L, а чтобы закрепить этот результат, зайдите в Control > Configure Dolphin и выберите опцию Editable Location Bar.

Настраиваются и другие параметры Dolphin – скажем, возможность открывать архивы в виде папок из вкладки Navigation. Вкладка Services перечисляет различные сервисы, которые будут появляться в виде опций в меню Context по щелчку правой кнопкой в Dolphin. Можете загрузить и дополнительные, с помощью кнопки Download New Services.

Работая в командной строке, вы можете открывать окно терминала в своем файловом менеджере. Перейдите в Control > Panels > Terminals, после чего окно *Dolphin* разделится на две части горизонтально, и добавьте окно терминала в нижнюю часть экрана. Терминал будет автоматически переключаться на папку, которую вы откроете в файловом менеджере над ним. Если вы включили вид Split, то терминал будет переключаться на директорию активной панели. LXF




# UEFI: Пере-перезагрузка

BIOS загружал ПК 30 лет, но настала пора перемен. **Джон Лэйн** знакомится с преемником.

**С момента, когда IBM создала свой первый Персональный Компьютер в 1981 году, тот изменился едва ли не до неузнаваемости, но при его включении происходит, по большей части, все то же самое. В каждом ПК есть BIOS, ответственная за запуск компьютера и загрузку операционной системы. Суть в том, что BIOS (сокращение от Basic Input/Output System — Базовая система ввода/вывода), прежде чем загрузить программу загрузки из записи MBR (master boot record) или на устройстве хранения, осуществляет серию тестов Power On Self Tests (именуемых POST) и исполняет ее. А уж загрузчик (bootloader) затем принимается за вашу операционную систему.**

Две новых технологии покушаются изменить этот порядок: BIOS потихоньку вытесняется UEFI, Unified Extensible Firmware Interface, а MBR – GUID Partition Table [Таблица разделов GUID], или GPT.

Изначально BIOS разрабатывалась как интерфейс между оборудованием и Disk Operating System (DOS, более известной как MS-DOS). Она была, и остается, 16-битной программой реального режима. Поскольку за эти годы операционные системы эволюционировали до 32-битного, а сейчас и 64-битного кода, они больше не используют интерфейс BIOS, содержа вместо него собственные драйверы устройств. Роль BIOS сократилась до запуска процесса загрузки операционной системы, а после этого – по сути, бездействия.

> «BIOS потихоньку вытесняется UEFI, а MBR — GPT.»

MBR служит двум целям: она содержит bootloader, исполняемый BIOS для загрузки компьютера, и таблицу разделов, определяющую расположение файловых систем на диске. Вся эта информация хранится в самом первом секторе диска (именуемом сектор 0) и по этой причине ограничена до 512 байт: 446 байт под загрузчик, плюс таблица разделов, содержащая до четырех 16-байтных записей. Последние два байта – это подпись,

## Большой диск

Размеры жестких дисков стали такими большими, что схема деления на разделы MS-DOS уже не справляется с ними. Она не умеет обращаться к дискам размером более 2 TiB, потому что таблицы разделов хранят адреса секторов в виде 32-битных чисел, то есть более 2^32 секторов быть не может.

Так как в каждом секторе 512 байт, это дает максимальный объем 2 TiB. Новая схема разделов GPT обходит данное ограничение, используя 64-битные области, что позволяет хранить 2^64 секторов по 512 байт, или 8 ZiB (9,64 ZB). Это превосходит сумму всех ныне существующих хранилищ.

по которой BIOS распознает действующую MBR. Таблицы разделов используют адресные поля с 32-битным сектором, а значит, не могут обращаться к дискам размером более 2 TiB. Такой тип таблицы разделов часто именуется таблицей разделов MSDOS, с целью отличить его от нового GPT.

Недостатки такого рода настройки – то, что она ограничена одним загрузчиком, четырьмя разделами и максимальным размером диска 2 TiB. UEFI поддерживает множественные загрузчики, а GPT поддерживает до 128 разделов и работает с дисками, по объему превышающими 2 TiB. Кроме того, UEFI дает таблице возможность сделать процесс загрузки безопасным.

Поскольку BIOS подразумевает, что загрузчик расположен в первом секторе диска, тот может быть только один на каждый диск. Большинство BIOS разрешают выбирать способ загрузки диска, и поэтому могут поддерживать столько загрузчиков, сколько имеется физических дисков. UEFI, напротив, игнорирует загрузчик в секторе 0, но зато разрешает существование нескольких загрузчиков на одном диске, используя особый раздел вместо абсолютного сектора. Этот особый раздел известен как EFI System Partition, или ESP, и форматируется он файловой системой FAT (обычно FAT32) и имеет размер от 100 до 250 MiB. Согласно системным требованиям UEFI, это должен быть первый раздел, и его флажок загрузки должен быть установлен. В системах Linux принято монтировать ESP под **/boot/efi**. Традиционно загрузчик хранятся в ESP в поддиректориях в соответствии с их производителями, а на форуме UEFI имеется список этих поддиректорий в www.uefi.org/specs/esp_registry.

> «Прошивка UEFI умеет читать таблицы разделов и системы FAT.»

Сердцевиной UEFI является его прошивка, и она отвечает за загрузку и исполнение приложений UEFI. Это отдельные программы, зависящие только от сервисов самой прошивки – ОС им не нужна. Они могут быть инструментами предзагрузочной диагностики/техобслуживания, но большинство из них являются загрузчиками ОС. Прошивка UEFI содержит менеджер загрузки, позволяющий пользователю запустить приложения UEFI. Менеджер загрузки напоминает обычный загрузчик, например, *Grub*, но грузить он может только приложения EFI. Можно скомпилировать ядро Linux (с версии 3.3) как приложение EFI, тем самым устранив необходимость в загрузчике – том же *Grub*, потому что тогда UEFI сможет загружать ядро напрямую.

Прошивка UEFI умеет читать таблицы разделов и файловые системы FAT, в отличие от BIOS, которая полагается на загрузчик. Она настраивается через переменные EFI, хранящиеся в NVRAM. Главная переменная, управляющая менеджером загрузки EFI, именуется BootOrder; она определяет, какие пункты меню отображать. Если ее не настроить, прошивка будет следовать стандартной процедуре, обнаруживающей загрузчик по указанному пути к ESP. Путь, который зависит от архитектуры, существует в виде:

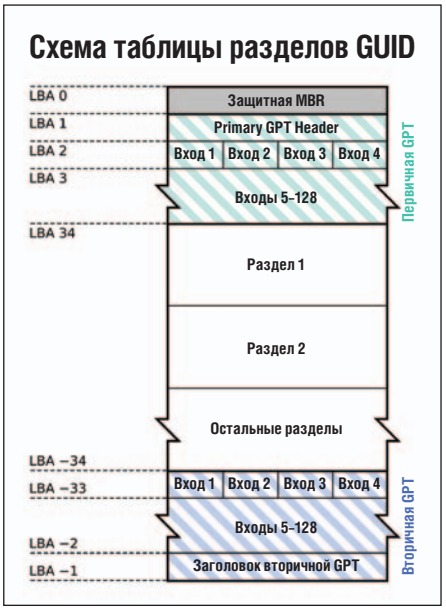
```
\EFI\BOOT\BOOT[architecture name].EFI
```

Буква G в GPT означает GUID, то есть Globally Unique Identifier [Глобально уникальный идентификатор], реализацию стандарта Universally Unique Identifier, определенного RFC4122. GPT использует GUID для идентификации дисков и разделов. Он начинает с сектора 1, с заголовка, который определяет количество и размер записей о разделах в идущей далее таблице. Заголовок содержит GUID диска, уникальным образом идентифицирующий диск, и каждый раздел, определенный в GPT, содержит два GUID: первый представляет тип раздела, а второй уникальным образом идентифицирует его. Помимо этого, заголовок содержит контрольную сумму CRC32, которая может использоваться прошивкой или операционной системой для подтверждения целостности данных. Не вздумайте редактировать таблицу разделов вручную!

GPT может иметь 128 разделов, и это устраняет надобность в логических и расширенных разделах. Расширенный раздел был способом решить проблему ограничения MBR четырьмя разделами: один из этих разделов использовался как расширенный, и его можно было разбить на множество логических разделов.

Каждая запись раздела имеет длину 128 байт и содержит уже упомянутые GUID, адреса начального и конечного секторов раздела, некоторые атрибуты и имя, понятное человеку. Вторая копия GPT хранится в конце диска в качестве резервной копии. Чтобы нормально разместить 128 таких разделов, требуется 16384 байта. Это 32 сектора, а следовательно, первый используемый сектор на диске GPT – сектор 34.

Схема GPT также включает master boot record, и ее цель – помочь предотвратить повреждение диска GPT инструментами, которые GPT не распознают. Эта Protective [Защитная] MBR, или PMBR, содержит таблицу разделов MSDOS с одним разделом на весь диск (или 2 TiB, если диск больше). Этот раздел отмечен типом 0xEE. Системы, не распознающие GPT, будут видеть неизвестный раздел, занимающий весь диск и не оставляющий свободного места.




**Диск GPT содержит две копии таблиц разделов плюс защитную таблицу разделов MS-DOS.**

Еще одно применение защитной MBR – разрешить BIOS загружать диск GPT. Если он содержит загрузчик, то BIOS слепо загрузит и исполнит его. Если этот загрузчик понимает GPT, то сможет его загрузить. Один из таких загрузчиков – *Grub2*, используемый многими дистрибутивами Linux. Это позволяет системам, работающим на BIOS, использовать диски размером более 2 TiB.

Диски с разделами MS-DOS обычно оставляют пропуск (именуемый областью совместимости DOS), начиная с сектора 1 и доходя до начала первого раздела. Традиционно загрузчики употребляли это свободное место для размещения на нем кода (*Grub 1* записал сюда свою программу загрузки в стадии 1.5). В GPT таблица разделов начинается с сектора 1, сразу после PMBR – и здесь нет незанятого места, чтобы его можно было использовать подобным образом. Вместо этого спецификация GPT предусматривает специальный раздел загрузки BIOS для использования теми загрузчиками, которые в ином случае должны были бы использовать область совместимости DOS. *Grub 2* пишет свой загрузчик прямо на этот раздел (он не содержит файловой системы). Принимая во внимание, что наличие области совместимости DOS было определено скорее соглашением, чем стандартом, наличие специально отведенного раздела более отказоустойчиво.

Все это была теорией. Теперь перейдем к практике – и настроим новую систему на основе UEFI и GPT. Наша новая система будет иметь двойную загрузку: она будет работать как с прошивкой UEFI, так и с прошивкой BIOS.

Мы воспользуемся *VirtualBox*, и вы сможете следовать нашим инструкциям, даже если у вас »

пока нет компьютера с прошивкой UEFI. А возьмем мы ISO Arch Linux от ноября 2012 – то есть вам нужно будет выполнить настройку вручную, это поможет разобраться в нужных действиях. Скачайте его с https://www.archlinux.org/download.

Сперва создайте новую 64-битную виртуальную машину и жесткий диск для тестов (10 ГБ должно хватить). Отметьте Enable EFI option на странице System настроек виртуальной машины, чтобы на компьютере появилась прошивка UEFI. Присоедините ISO-образ и запустите виртуальную машину. Она должна отобразить приглашение для root. Зайдите – пароль не нужен.

Вам нужно создать четыре раздела. Тут вам пригодится *gdisk*. Мы предлагаем размеры, достаточные для этого упражнения, но можно задать и другие. ESP создается из сектора 2048 (параметр по умолчанию, предлагаемый *gdisk*), а раздел загрузки BIOS располагается перед ESP. В таблице вы найдете предложения по созданию разделов.

Запустите *gdisk* командой **gdisk /dev/sda**, затем используйте **o** для создания новой пустой таблицы разделов. Используйте **n** для создания раздела, **t** – для настройки его типового кода, и (опционально) **c** для изменения его имени. Запишите новую таблицу разделов с помощью **w**. Или воспользуйтесь *parted*:

```
# parted /dev/sda
(parted) unit s
(parted) mktable gpt
(parted) mkpart primary 2048 411647
(parted) set 1 boot on
(parted) name 1 “EFI System Partition”
(parted) mkpart primary 34 2047
(parted) name 2 “BIOS Boot Partition”
(parted) set 2 bios_grub on
(parted) mkpart primary ext2 411648 821247
(parted) name 3 “Linux /boot filesystem”
(parted) mkpart primary ext4 821248 20971486
(parted) name 4 “Linux /root filesystem”
(parted) quit
```

## Таблица разделов

| Номер | Начало | Конец | Размер | Кодовое имя |
|---|---|---|---|---|
| 1 | 2048 | 411647 | 200.0 MiB | EF00 Система EFI |
| 2 | 34 | 2047 | 1007.0 KiB | EF02 Раздел BIOS |
| 3 | 411648 | 821247 | 200.0 MiB | 8300 Файловая система /boot из Linux |
| 4 | 821248 | 20971486 | 200.0 MiB | 8300 Файловая система /root из Linux |

Мы здесь использовали деление на разделы GPT, но можно использовать вместо него и разделы MSDOS, если размер диска менее 2 TiB. В этом сценарии пропустите загрузочный раздел BIOS и используйте **fdisk**, чтобы изменить тип раздела с EFI System Partition на 0xEF. Прошивка UEFI *VirtualBox* работает как с разделами GPT, так и с разделами MSDOS, но другие прошивки могут поддерживать только GPT.

Создав файловые системы, подмонтируйте их:

```
# mkfs.vfat -F32 /dev/sda1
# mkfs.ext2 /dev/sda3
# mkfs.ext4 /dev/sda4
# mount /dev/sda4 /mnt
# mkdir /mnt/boot
# mount /dev/sda3 /mnt/boot
# mkdir /mnt/boot/efi
# mount /dev/sda1 /mnt/boot/efi
```

Используйте утилиту ArchLinux **pacstrap** для установки новой системы на подготовленные файловые системы (требуется соединение с Интернетом):

```
# pacstrap /mnt base
# genfstab -p -U /mnt | sed ‘s/cp437/437/’ >> /mnt/etc/fstab
# arch-chroot /mnt pacman -S grub-efi-x86_64
# modprobe efivars
# arch-chroot /mnt grub-install --target=x86_64-efi --efi-directory=/boot/efi --bootloaderid=arch_grub --recheck
# arch-chroot /mnt grub-mkconfig -o /boot/grub/grub.cfg
# umount /mnt/boot/efi /mnt/boot /mnt
```

Это установит системные файлы на файловую систему root, монтированную на **/mnt**. Она также добавляется к **/etc/fstab**, чтобы позаботиться дополнительной загрузке и файловых системах EFI, монтированных под **/mnt** (**sed** вставлено, чтобы обойти ошибку в **genfstab**). Затем мы установим пакет *grub* и проверим, загружен ли модуль ядра **efivars** (он обеспечивает доступ к переменным EFI в **/sys/firmware/efi**). Далее **grub-install** установит *Grub* в новую поддиректорию ESP – **arch-grub**, мы монтировали ее в **/boot/efi**. Если все пойдет хорошо, должно появиться сообщение: “Installation finished: No error reported [Установка завершена: ошибок не обнаружено]”. После этого мы генерируем файл настройки Grub и размонтируем наши файловые системы из **/mnt**. После перезапуска виртуальная машина должна предложить вам свеженастроенное меню *Grub* и загрузить нашу систему.

Часть Grub настраивает порядок загрузки EFI в NVRAM. Обычно при этом UEFI настраивается на автоматическую загрузку *Grub* при запуске

## Оболочка UEFI

Одно из особых приложений UEFI – это оболочка UEFI. Это базовая среда командной строки, которую можно применять для запуска приложений EFI или для базовой настройки и решения проблем.

При отсутствии правильной стартовой последовательности прошивка перенаправит вас в оболочку. Туда также можно попасть через интерфейс меню прошивки, если нажать на нужную клавишу. В VirtualBox это клавиша DEL, но у вас должна быть очень быстрая реакция! Можете также нажать **c** в меню *Grub*, чтобы попасть в окно приглашения *Grub*, и затем ввести **exit**. Две удобных команды оболочки – **help** и **bcfg**:

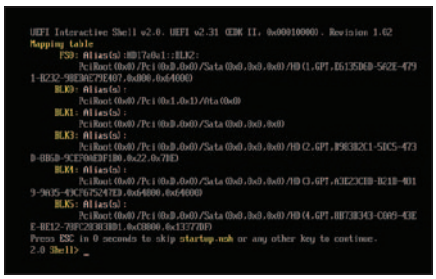
```
> bcfg boot dump -v
```

выведет список приложений EFI, перечисленных в менеджере загрузки. Вы можете добавлять/удалять опции или изменять их порядок. Чтобы добавить опцию загрузки:

```
> bcfg boot add 3 fs0:\EFI\arch_grub\grubx64.efi “ArchLinux Grub”
```

А чтобы переместить ее вверх в списке –

```
> bcfg boot mv 3 2
```



**Обращайтесь к оболочке, используя менеджер загрузки прошивки.**

Стоит запомнить еще одну команду оболочки – **reset**; она перезагружает компьютер, чтобы опять начать процесс загрузки с самого начала.

Опциями загрузки также управляет в Linux команда **efibootmgr** (от имени root). Если ваша прошивка не предоставляет оболочки, загрузите оболочку с открытым кодом с проекта Intel Tianocore, установите в подпапку внутри **/boot/EFI** и добавьте пункт меню менеджера загрузки для доступа к ней:

```
# mkdir /boot/efi/EFI/tiano
# wget https://edk2.svn.sourceforge.net/svnroot/edk2/trunk/edk2/ShellBinPkg/UefiShell/X64/Shell.efi
# efibootmgr --create --label ‘Tiano Shell’ --loader ‘\EFI\tiano\Shell.efi’
```

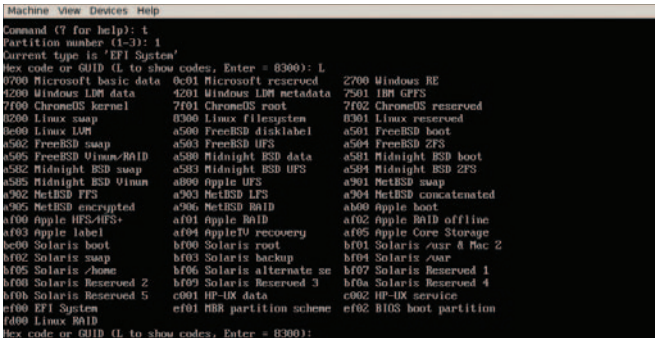
В оболочке есть не только это, но и многое другое. Имеет смысл стянуть копию спецификации оболочки с форума UEFI (www.uefi.org/specs) для более углубленного ее изучения.

## Типы разделов

GPT использует GUID для идентификации типа раздела так же, как это делает область типа раздела в разделе MSDOS. Можно использовать *gdisk*, чтобы изменить размер типа раздела GPT – вы можете ввести GUID или использовать короткий код, похожий на те, которые используются в *fdisk* для разделов MSDOS. Эти коды специально разработаны для *gdisk*.

Соответствующие GUID слишком массивны, чтобы мы перечислили их здесь, но некоторые из них приведены в Wikipedia (https://en.wikipedia.org/wiki/GUID_Partition_Table#Partition_type_GUIDs).

GUID, изначально используемый для файловых систем разделов Linux, тот же самый, что и используемый Microsoft для своих (он именуется базовый раздел данных). Во избежание путаницы, недавно был определен новый GUID для файловых систем Linux, но он, возможно, еще не используется существующими системами.




компьютера. Однако VirtualBox не сохраняет прошивки NVRAM после выключения виртуальной машины, и в результате настройки загрузки теряются.

Если это произойдет, запустится оболочка EFI, и вам нужно будет вручную запустить загрузчик:

```
2.0 Shell> fs0:\EFI\arch_grub\grubx64.efi
```

После загрузки ОС настройте загрузчик по умолчанию, чтобы обойти проблему отсутствия постоянной NVRAM:

```
# mkdir /boot/efi/EFI/BOOT
# cp /boot/efi/EFI/{arch_grub/grub,BOOT/BOOT}x64.efi
```

Если порядок загрузки не настроен, что и произойдет в том случае, если в NVRAM ничего не будет, прошивка вернется к **EFI\BOOT\BOOTx64.efi**. В результате это должно привести к успешной загрузке после запуска виртуальной машины. Теперь, чтобы заставить систему загружать BIOS, нужно просто настроить загрузку BIOS:

```
# dhcpcd
# pacman -S grub-bios
# grub-install --target=i386-pc --recheck /dev/sda
```

Начните с установки сетевого соединения; один из способов сделать это – команда **dhcpcd**. Затем установите пакет *grub-bios* и установите на диск версию *Grub* для BIOS. Это создаст код загрузки в MBR и раздел загрузки BIOS, который мы настроили раньше. Он использует наш готовый файл настройки *Grub*, **/boot/grub/grub.cfg**.

Выключите виртуальную машину и переключите ее настройки с EFI на BIOS, убрав отметку Enable EFI option на странице System. После перезапуска виртуальная машина будет загружаться через BIOS. Этот пример показывает, что на диск с разделами GPT или MSDOS легко установить систему, которая будет работать и с UEFI, и с BIOS. Стоит упомянуть, что в некоторых других дистрибутивах это пройдет не столь гладко.

Теперь займемся бегемотом в гостиной: безопасной загрузкой. Суть ее в том, что UEFI не загрузит ничего, не подписанного распознаваемым ключом. Новые ПК с предустановленным Windows 8 поставляются с ключом от Microsoft, который позволяет их новейшей операционной системе загружаться без проблем. Если в такой системе включена безопасная загрузка и в ней есть только этот ключ, она будет загружать только то, что подписано этим ключом. Большинство новых ПК настроено именно так, чтобы получить сертификацию Windows 8.

Компьютеры с предустановленным Windows 8 имеют безопасную загрузку по умолчанию. Однако ее можно отключить. Для пользователей Linux это на данном этапе самое практичное решение. Правда, некоторым безопасная загрузка будет важна – или они просто захотят воспользоваться ею – и для этого требуется, чтобы Linux был подписан ключом, который будет распознавать прошивка.

Для безопасной загрузки Linux есть две базовых опции. Первая – найти OEM, которые прилагали бы дополнительные ключи, которыми можно было бы подписать Linux. Вторая – подписать Linux ключом Microsoft. По ряду практических соображений обе эти опции малопривлекательны.

Подписывать ядро Linux непрактично: оно быстро меняется, такова его природа; к тому же многие создают собственное ядро с индивидуальной настройкой. Поэтому логично будет взять подписанный загрузчик, который затем сможет загрузить любой образ ядра.

> «Подписывать ядро Linux непрактично: оно быстро меняется.»

Среди примеров подписанных загрузчиков – предзагрузчик от Linux Foundation и решения, предлагаемые создателями дистрибутивов, например, Fedora Shim. Будучи подписаны ключом Microsoft, они будут работать на любой машине с Windows 8. Однако подписать их этим ключом оказывается проблематично (http://bit.ly/VFEAV9).

Чтобы избежать ситуаций, когда подобные решения могут быть использованы вредоносными программами для нанесения ущерба безопасным системам, требуется присутствие человека, который подтвердит намерение продолжать работу с неподписанной программой загрузки.

Суть предзагрузчика Linux Foundation в предоставлении подписанной программы загрузки, которая запустит по цепочке следующий загрузчик (тот же *Grub*), а тот затем и загрузит Linux. После предзагрузчика проверка безопасности прекращается, и такой подход не более безопасен, чем отключение безопасной загрузки UEFI. Однако это дает возможность Linux и другим системам с неподтвержденной безопасностью загрузиться в среду безопасной загрузки. Это решение предлагается фондом в качестве временного, на тот период, пока дистрибутивы не реализуют более безопасные решения.

Fedora Shim стремится пойти на шаг дальше, обеспечивая безопасную загрузку системы Linux. Он загружает специальную версию *Grub* с публичным ключом Fedora, и будет загружать в безопасном режиме ядра, обеспеченные этим ключом. Он также будет загружать и другие ядра, но это потребует ручного контроля во время загрузки. Безопасно загруженные ядра будут также ограничивать загрузку из командной строки и требовать, чтобы модули ядра имели подпись. Такой вариант дает возможность безопасно загрузить Linux с предопределенной конфигурацией, но если требуется индивидуальная настройка, то проблема не снимается: потребуется индивидуальная прослойка локально произведенных ключей, подписанных ключом, распознаваемым прошивкой. Пока что непонятно, как этого добиться. Подобные же подходы практикуются SUSE и Ubuntu.

Механизм подписи для безопасной настройки именуется Microsoft Authenticode и управляется Symantec (бывшей VeriSign). За годичную оплату в размере $99 вы можете подписать любое количество бинарников по своему усмотрению через Microsoft Developer Center по адресу https://sysdev.microsoft.com (для регистрации понадобится Windows Live ID).

Вопрос безопасной загрузки пока что не имеет однозначного и простого ответа. Есть методы обхода этой проблемы, но пока рано говорить о том, легко ли будет работать в Linux в среде безопасной загрузки. Если вас это возмущает, вы, возможно, заинтересуетесь кампанией, проводимой FSF: (http://bit.ly/nHYBRN). И всегда есть возможность отключить безопасную загрузку – вам всяко будет не хуже, чем сейчас. LXF

# Проект Tumbleweed

**Игорь Штомпель** заинтересовался проектом, название которого в переводе означает «перекати-поле». Куда же он катится?

Идея создания дистрибутивов Linux с поддержкой постоянного обновления получила свое воплощение в ряде проектов. Инициатива Tumbleweed, развивающаяся в рамках проекта openSUSE, осуществляет ее в виде самостоятельной ветки дистрибутива, использование которой не представляет особых сложностей.

## Предварительно

Прежде чем рассматривать проект openSUSE Tumbleweed, скажем пару вступительных слов. В основе Tumbleweed лежит подход под названием Rolling release. Проект Arch Linux, в русской версии FAQ, переводит данный термин как «плавающий релиз» [1]. Давайте рассмотрим эту концепцию подробнее.

Словосочетание «плавающий релиз» характеризует способ обновления ПО [2], в нашем случае, операционной системы. Выделяют несколько типов «плавающих релизов» – мы не будем приводить их здесь, отсылая интересующегося читателя к англоязычной Википедии [3]. Для нас важно, что при «плавающем релизе» понятия «версия», «переход на новую версию» и т. п. теряют смысл – в каждый момент времени дистрибутив содержит самые новые стабильные версии составляющих его пакетов. Пользователи, регулярно обновляющие такую систему, всегда будут иметь ее актуальной в широком смысле слова.

Конечно, плюсы такого подхода очевидны: это и новые версии ПО, и отсутствие необходимости переустановки операционной системы для обновления, и т. д. Но и минусы очевидны тоже: возможная нестабильность работы, ввиду использования новейших версий программ, хотя и «выпущенных официально»; повышенные требования к интернет-подключению (обновления бывают достаточно объемными).

Между прочим, согласно приведенной выше статье о «плавающем релизе», последнее понятие «наиболее часто употребляется относительно дистрибутивов Linux».

## Знакомимся

Проект Tumbleweed от openSUSE был представлен 30 ноября 2010 года, в списке рассылки opensuse-project, одним из ведущих разработчиков ядра Linux Грегом Кроа-Хартманом [4]. Более того, Грег Кроа-Хартман, как указано на официальном сайте проекта openSUSE, воплотил в жизнь саму идею «плавающего релиза» на базе операционной системы openSUSE. Кстати, об этом разработчике ядра Linux писали в **LXF81**, а также в электронном приложении к журналу Open Source (выпуск № 112) [5].

**› Колючий кустарник перекати-поле [*англ.* tumbleweed] разносится ветром на весьма далекие расстояния!**

Как известно, проект openSUSE имеет репозиторий под названием Factory. Данный репозиторий содержит нестабильное программное обеспечение, требующее дополнительной работы. Официальный сайт проекта openSUSE указывает: «...в Factory размыт контур между экспериментальной и стабильной версией» [6]. Отличием же ветки Tumbleweed от Factory является то, что она включает новые стабильные версии программного обеспечения, «готовые к ежедневному использованию» [7]. Вывод о достижении стабильного состояния пакета делается разработчиком, ответственным за его сопровождение (мейнтейнером). Что и подтверждает FAQ, посвященный Tumbleweed: «“Стабильная” версия пакета определяется человеком, ответственным за сопровождение пакета, именно эта версия и будет помещена в репозиторий Tumbleweed» [8].

В разделе, посвященном Tumbleweed на официальном сайте проекта openSUSE, разработчики заявляют: «...если вы будете использовать Tumbleweed, вам не нужно будет обновлять систему до более новой версии, поскольку у вас уже будет самая новая версия дистрибутива!». Выглядит это действительно интересно, так как ожидание выхода свежих версий любимых дистрибутивов в зависимости от релиз-цикла может продолжаться и полгода, и год; а пользователям openSUSE Tumbleweed не надо ждать выхода новой версии и тратить время на переустановку, перенос данных и т. д. Конечно, не обошлось и без «изъяна», но об этом чуть ниже, а сейчас приведем пример из уже накопившейся истории Tumbleweed. Gnome 3.0 стал доступен в Tumbleweed после выхода openSUSE 11.4, который поставлялся с Gnome 2.32; при этом сам репозиторий Tumbleweed впервые стало возможным подключить между выпусками openSUSE 11.4 и openSUSE 12.1. Таким образом, если в последнем стабильном релизе openSUSE 11.4 было доступно рабочее окружение Gnome 2, то появившийся репозиторий Tumbleweed позволял использовать Gnome 3.

Думается, читателю уже стало ясно, на кого ориентирован проект openSUSE Tumbleweed: в первую очередь, на пользователей, желающих иметь новейшие стабильные версии программного обеспечения, которые, в целом, недоступны из репозиториев последнего стабильного релиза openSUSE. Необходимость в установке новой версии операционной системы (часть пакетов которой может устареть уже к моменту выхода) для них отпадает.

Если же говорить об изъяне, то openSUSE Tumbleweed отличается некоторой небезопасной спецификой. В частности, при его использовании происходит частое обновление ядра Linux, что потребует от пользователя «самостоятельного обновления проприетарных драйверов» (например, для видеокарт от ATI или NVIDIA) [9]. Tumbleweed включает не только свободные компоненты, однако проприетарные драйверы зависят от ядра, а оно в Tumbleweed постоянно обновляется. В конечном итоге это может привести к неработоспособности системы. Поэтому для пользователей, незнакомых с процедурой «самостоятельного обновления проприетарных драйверов», применение репозитория Tumbleweed не рекомендуется; или, как точнее сказано в разделе Tumbleweed на официальном сайте, они «не должны» использовать этот дистрибутив.

## Запускаем

Перейдем к практической части. Здесь мы воспользуемся операционной системой openSUSE 12.2. Процесс ее установки описывать не будем. Скажем лишь, как получить загрузочный образ данной версии. Для этого достаточно перейти на официальный сайт проекта openSUSE, а затем перейти в раздел “Get it”, где можно выбрать соответствующий ISO-образ и скачать его [10].

Запуск openSUSE Tumbleweed заключается в удалении репозиториев установленной версии openSUSE, подключении необходимых репозиториев и в последующем обновлении.

Для начала взглянем на список уже подключенных репозиториев. Сделать это можно следующим образом: Меню запуска приложений kickoff (далее – Меню) > Приложения > Система > Центр управления (далее – Центр управления или YaST2) > Программное обеспечение > Репозитории программного обеспечения. После чего вы окажетесь в окне настройки репозиториев openSUSE.

Перечень репозиториев, которые необходимо подключить, помещен на странице официального сайта openSUSE, посвященной проекту Tumbleweed [11]. Здесь можно произвести подключение соответствующих репозиториев одним щелчком мыши – «за один клик». Но сперва необходимо ряд репозиториев удалить. Для этого перейдите в окно настройки репозиториев, как было описано выше. Далее, выберите репозиторий с именем openSUSE-12.2-Non-Oss, а затем нажмите кнопку Удалить. Затем осуществите эту же операцию для репозиториев с именами openSUSE-12.2-Oss, openSUSE-12.2-Update, openSUSE-12.2-Update-Non-Oss, после чего нажмите кнопку OK.

Теперь можно перейти и к установке «за один клик». Для этого в специальном разделе “How to try Tumbleweed?” страницы, посвященной проекту Tumbleweed, имеется ссылка one-click-install [12], которая позволяет загрузить файл **tumbleweed.ymp**; **.ymp** (YaST Metapackage File) – это формат пакетов в openSUSE [13].

Дождавшись загрузки файла **tumbleweed.ymp**, запустите его. Вы увидите окно, где будут предложены репозитории программного обеспечения из соответствующего перечня (см. рис. 1). Нажмите клавишу Далее. Затем появится окно с сообщением об изменениях (добавлении соответствующих репозиториев), которые произойдут, если нажать клавишу Далее. После нажатия этой клавиши появится окно со «стандартным» для таких ситуаций предупреждением:

```
Вы просмотрели изменения, которые будут сделаны в вашей системе?
Вредоносные пакеты могут повредить вашу систему.
```

Нажимайте клавишу Да.

Вам останется только ввести пароль пользователя root, и процесс добавления репозиториев продолжится. Кстати, в ходе этой процедуры, вам потребуется указать в соответствующем окне (Импортировать непроверенный ключ GnuPG), что вы доверяете GPG-ключу подключаемого репозитория Tumbleweed. После всех необходимых операций последует сообщение о том, что установка программного обеспечения прошла успешно, и последний шаг – нажать кнопку Завершить.

Из консоли аналогичные действия можно осуществить с помощью *zypper*. Выведем список репозиториев:

```
$ zypper lr
```

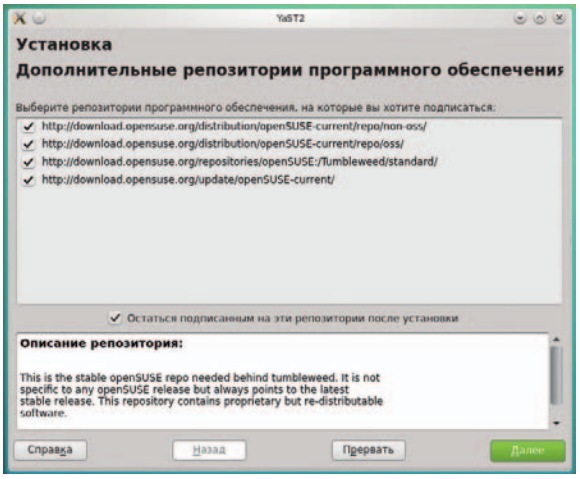
Как видно на рис. 2, у нас оказались подключенными четыре репозитория, а еще четыре – нет.

Далее, удалим ненужные репозитории –

```
$ sudo zypper rr 4 5 7 8
```

и подключим те, что нам требуются:

```
$ sudo zypper ar --refresh http://download.opensuse.org/repositories/openSUSE:/Tumbleweed/standard/ Tumbleweed
$ sudo zypper ar --refresh http://download.opensuse.org/distribution/openSUSE-current/repo/oss/ 'openSUSE Current OSS'
```




**› Рис. 1. Подключение репозиториев openSUSE Tumbleweed.**

```
$ sudo zypper ar --refresh http://download.opensuse.org/distribution/openSUSE-current/repo/non-oss/ 'openSUSE Current non-OSS'
$ sudo zypper ar --refresh http://download.opensuse.org/update/openSUSE-current/ 'openSUSE Current updates'
```

Последний шаг в базовом процессе перехода на использование openSUSE Tumbleweed – обновление (без подключения дополнительных репозиториев, например, Packman; о них – ниже). Обратите внимание, что, в отличие от добавления репозиториев с использованием графического интерфейса для *YaST*, при добавлении репозиториев с использованием *zypper* добавление GPG-ключа для репозитория Tumbleweed потребуется осуществить на этапе обновления (см. рис. 3):

```
$ sudo zypper dup
```

Кстати, использование команды **dup** рекомендуется при переходе на другой релиз дистрибутива (например, подключены репозитории новой версии дистрибутива, а старые удалены – теперь необходимо обновиться), в том числе и при обратном обновлении (со старшей версии на младшую) [14].

В нашем случае оказалось необходимо загрузить 712,2 МБ (358 пакетов для обновления, 35 новых). Среди новых пакетов: *amarok-lang*, *bundle-lang-gnome-extras-en*, *flash-player*, *flash-player-kde4*, *nepomuk-core*, *openSUSE-release-ftp*, *poppler-data* и др. Кроме того, среди пакетов для обновления имеется и *LibreOffice* (обновление с версии 3.5.4.7, поставляющейся вместе с openSUSE 12.2, до версии 3.5.4.13 из репозиториев Tumbleweed). Обновление прошло успешно, и стабильная работа дистрибутива не была нарушена.

## Дополнительные репозитории

Итак, openSUSE Tumbleweed запущен. Система работает, базовое программное обеспечение установлено. Но, как правило, его недостаточно для решения всех необходимых задач. Например, недоступны кодеки в полном объеме, нет «хороших» медиа-плейеров – *VLC*, *Mplayer* и многих-многих других полезных программ.

Чтобы помочь этому горю, разработчики подготовили специальные репозитории Packman, которые доступны и для пользователей Tumbleweed [15]. Вам предоставляется возможность

```
igor@examples:~> zypper lr
# | Псевдоним                | Имя                                | Включён | Обновление
--+--------------------------+------------------------------------+---------+-----------
1 | repo-debug               | openSUSE-12.2-Debug                | Нет     | Да
2 | repo-debug-update        | openSUSE-12.2-Update-Debug         | Нет     | Да
3 | repo-debug-update-non-oss | openSUSE-12.2-Update-Debug-Non-Oss | Нет     | Да
4 | repo-non-oss             | openSUSE-12.2-Non-Oss              | Да      | Да
5 | repo-oss                 | openSUSE-12.2-Oss                  | Да      | Да
6 | repo-source              | openSUSE-12.2-Source               | Нет     | Да
7 | repo-update              | openSUSE-12.2-Update               | Да      | Да
8 | repo-update-non-oss      | openSUSE-12.2-Update-Non-Oss       | Да      | Да
igor@examples:~>
```

**› Рис. 2. Вывод списка репозиториев с помощью *zypper*.**

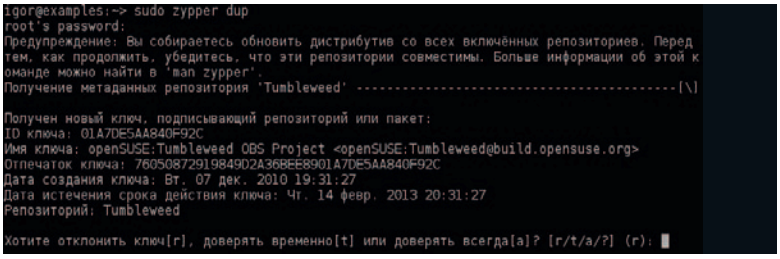



**Рис. 3. Добавление GPG-ключа на этапе обновления системы из репозиториев Tumbleweed.**

подключить такие репозитории Packman, как (в алфавитном порядке):

- **Games** – как видно из названия, репозиторий содержит игры, а всего в нем 43 пакета;
- **Essential** – кодеки, медиа-плейеры, ряд других программ (*faac*, *ffmpeg*, *flac123*, *gnome-mplayer*, различные плагины для *GStreamer*, *mpg123*, *mplayer2*, *mplayerplug-in*, *smplayer*, *xmms* и др.), всего в репозитории 243 пакета;
- **Extra** – различные приложения (не мультимедиа), многие из них – для работы в сети (*bareftp*, *dhcpdetector*, *rar*, *tea*, *di*, *fotowall*, *FreeCAD*, *nmapsi4*, *opencascade*, *polipo*, *putty*, *Q7Z*, *qputty*, *sipcalc*, *stunnel*, *qtsmbstatus* и др.), всего 155 пакетов;
- **Multimedia** – как и следует из названия, включает мультимедиа-приложения (*bino*, *DVDStyler*, *handbrake*, *imagination*, *isomaster*, *k9copy*, *kdvdcreator*, *kino*, *kmediafactory*, *LiVES*, *openshot*, *smplayer2*, *synfigstudio*, *tomahawk*, *xine-ui* и др.), всего 400 пакетов.

Добавить эти репозитории можно все сразу –

```
# sudo zypper ar http://packman.inode.at/suse/openSUSE_Tumbleweed packman
```

а можно и по отдельности. Например, для репозитория Essential это будет выглядеть следующим образом:

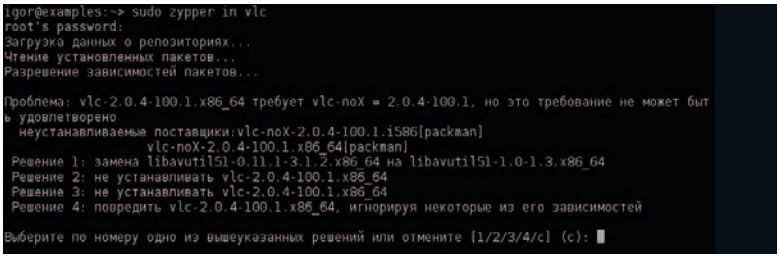



**Рис. 4. Проблема при установке *VLC*.**

```
# sudo zypper ar http://packman.inode.at/suse/openSUSE_Tumbleweed/Essentials packman-essentials
```

а для Multimedia –

```
# sudo zypper ar http://packman.inode.at/suse/openSUSE_12.2/Multimedia packman-multimedia
```

Но на самом деле, в одиночку подключать репозитории Essential и Multimedia не стоит, так как программы из второго репозитория имеют зависимости в первом. Например, при попытке установить *Imagination* из репозитория Multimedia без подключения Essential вы получите предупреждение о проблеме. В частности, *zypper* не сможет обнаружить ни в одном из репозиториев *ffmpeg*, который, как мы указали выше, поставляется в рамках репозитория Essential.

Далее обновим репозитории:

```
# sudo zypper up
```

Готово, все перечисленные репозитории Packman подключены.

Тепепрь займемся установкой необходимого программного обеспечения – например, медиа-плейера *Mplayer*:

```
# sudo zypper in mplayer
```

Если бы мы выполнили эту команду до подключения репозиториев Packman, мы бы увидели в ее выводе, среди прочего,

```
Не найдено поставщиков 'mplayer'
```

Сейчас же программное обеспечение будет успешно установлено (общий объем загружаемых пакетов – 20,6 МБ).

Учтите, что в процессе установки того или иного программного обеспечения могут возникать определенные конфликты версий. Например, при попытке установки медиа-плейера *VLC* –

```
# sudo zypper in vlc
```

возникла проблема, показанная на рис. 4.

Выбирайте для ее устранения тот вариант решения, который вам больше подходит.

Как видим, расширить базовые функциональные возможности Tumbleweed, подключив Packman, довольно просто.

## Заключение

Разработчикам openSUSE удалось создать операционную систему, которая вполне успешно поддерживает самое себя в актуальном состоянии, и ее установка не представляет особой сложности. Так почему бы не воспользоваться столь удобным шансом получать новейшие версии программ, почти не прикладая рук? Дерзайте! LXF

## Ссылки на источники


1 Проект ArchLinux о плавающем релизе (см. ответ на вопрос «Когда будет выпущен новый релиз?») – https://wiki.archlinux.org/index.php/FAQ_(Русский).

2 О «плавающем релизе» в русской версии Википедии – http://ru.wikipedia.org/wiki/Rolling_release.

3 О «плавающем релизе» в английской версии Википедии – http://en.wikipedia.org/wiki/Rolling_release.

4 Представление проекта openSUSE Tumbleweed Грегом Кроа-Хартманом [Greg Kroah-Hartman] – http://lists.opensuse.org/opensuse-project/2010-11/msg00206.html.

5 Статья «Разработчики ядра Linux. Часть 2: Грег Кроа-Хартман» Open Source № 112 (электронное приложение к журналу «Системный администратор») – http://osa.samag.ru/info/OpenSource112.

6 Раздел официального сайта проекта openSUSE, посвященный Tumbleweed – http://ru.opensuse.org/Portal:Tumbleweed.

7 Об отличии репозитория Tumbleweed от репозиториев Factory и Factory-tested на официальном сайте проекта openSUSE (в FAQ по Tumbleweed) – http://ru.opensuse.org/Portal:Tumbleweed/FAQ/1.

8 FAQ, посвященное Tumbleweed о принятии решения о стабильности того или иного пакета – http://ru.opensuse.org/Portal:Tumbleweed/FAQ/2.

9 Более подробно с информацией о работе с проприетарными драйверами от ATI в openSUSE можно – http://ru.opensuse.org/SDB:ATI_драйверы; то же для драйверов от NVIDIA – http://ru.opensuse.org/SDB:NVIDIA.

10 Официальный сайт проекта openSUSE – http://www.opensuse.org/ru/; Страница, на которой можно выбрать и получить необходимую версию загрузочного образа openSUSE – http://software.opensuse.org/122/ru.

11 Страница, посвященная openSUSE Tumbleweed на официальном сайте openSUSE – http://en.opensuse.org/Portal:Tumbleweed.

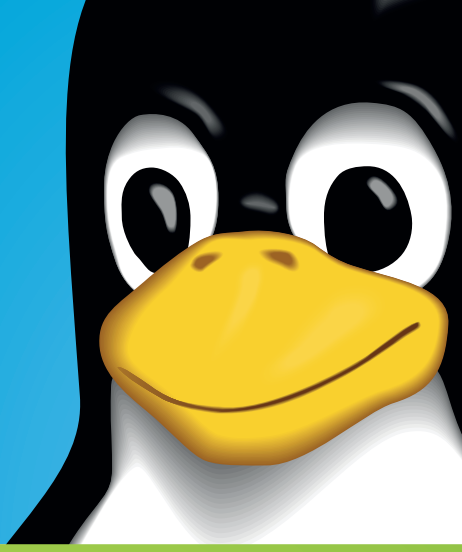
12 Удаление/подключение необходимых репозиториев для работы openSUSE Tumbleweed «за один клик» – http://dl.dropbox.com/u/29347181/tumbleweed.ymp; Спецификация для подготовки файла для установки «за один клик» – http://en.opensuse.org/openSUSE:One_Click_Install_specification.

13 Статья о метапакетах для *YaST* – http://en.opensuse.org/YaST_Metapackage_Handler.

14 Об использовании zypper на официальном сайте openSUSE – http://en.opensuse.org/SDB:Zypper_usage; тоже на русском языке – http://ru.opensuse.org/SDB:Zypper_использование_11.3.

15 Статья о дополнительных репозиториях на официальном сайте openSUSE – http://en.opensuse.org/Additional_package_repositories.

# Используйте свободное ПО — сэкономьте годовой бюджет!



Операционная система GNU/Linux поможет вам **с наименьшими затратами** решить проблему лицензирования программного обеспечения, навсегда избавиться от компьютерных вирусов и повысить надежность вашей компьютерной сети.

**С нашей помощью вы сможете**

**Сконцентрироваться на своем бизнесе,** не отвлекаясь на вопросы поддержки своей ИТ-инфраструктуры

**Забыть о вирусах, угрозах безопасности** и необходимости лицензирования программного обеспечения

**Оптимизировать затраты** на лицензирование ПО за счет максимально возможного использования свободного ПО

## ГНУ/Линуксцентр предлагает:

- внедрение наиболее дружественных вариантов ОС GNU/Linux и прикладных решений на базе свободного ПО;
- абонентскую поддержку вашей сети;
- обучение сотрудников вашей компании.

Наш опыт внедрения свободного программного обеспечения в организациях различного профиля поможет выбрать **оптимальное сочетание свободного и коммерческого программного обеспечения**, подходящее именно для вашей компании, а также поможет избежать технических и организационных проблем при внедрении свободного ПО.

## Решите проблемы лицензирования ПО и поддержки компьютерной сети с помощью профессионалов!

Москва +7 (499) 271-49-54

Санкт-Петербург +7 (812) 309-06-86

Linux-эксперт для вашего бизнеса. www.linuxcenter.ru

Linux center

# Из истории Иксов

**Алексей Федорчук** в очередной раз показывает, кто слову хозяин, возвращаясь к истории оконной системы X и всего, что с ней связано.

В очередной раз возвращаю сам себе некогда данное слово – больше не писать про историю Linux'а. В оправдание чего могу сказать, что в публиковавшемся цикле оказалась опущенной очень важная тема: история графических пользовательских интерфейсов, а конкретно – оконной системы X и работающих поверх нее оконных менеджеров и интегрированных графических сред рабочего стола (часто называемых десктопами). Им и будет посвящен настоящий цикл статей – или, скорее, субцикл внутри исторического цикла.

## Вступление

Обратиться к истории графических интерфейсов, используемых в UNIX и Linux, необходимо по ряду причин. Первая – та, что Linux в пользовательской своей ипостаси немыслим без современной инкарнации оконной среды X – Xorg. Немыслим настолько, что начинающие пользователи этой ОС часто ставят между ними знак равенства, не отдавая себе отчета в том, что, как говорится в народе, X – он и в Африке... Икс. Точнее, един во всех UNIX'ах и UNIX-подобных операционных системах.

В свое время немало было сломано копий о тему, как должна называться операционная система – главная героиня нашего цикла: Linux или GNU/Linux. И арьергардные бои на этом ристалище возникают временами и по сей день. Автор сих строк немало высказывался на эту тему – оснований повторяться не вижу. А в рамках нынешней темы замечу: если уж так нейметься прикрутить к имени Linux какой-либо префикс, то с куда большим основанием это может быть префикс X.

Ибо теоретически вполне можно представить дистрибутив этой системы без компонентов, разработанных в рамках проекта GNU – есть и практически близкие к этому реализации, например, Tiny Core Linux. А вот без оконной системы X, менеджеров ее окон, интегрированных десктопов и приложений графического режима говорить о каком-то десктопном Linux'е просто нелепо – как сказала поэт-линуксоид Алиса Деева,

*Это разговор не в пользу бедных –*
*В пользу продавцов клавиатур.*

Вторая же причина поговорить об истории Иксов – то, что разработчики проекта Wayland действительно обещают сделать Иксы достоянием истории. Которую надо рассказать, пока мы еще чего-то помним и нас еще не замочили.

## Доисторический период

Я не буду закапываться в глубокую праисторию и вспоминать о первых мышах, бегающих по закоулкам графического интерфейса в Исследовательском центре Hewlett-Packard в Пало-Альто – редкий лентяй не осветил этот вопрос всесторонне.

Не буду я рассматривать и детективную проблему – потибрили ли из Пало-Альто идею графического пользовательского интерфейса (по-нашему, по-бразильскому – GUI, именно этот термин и будет для краткости применяться в дальнейшем), или нет. И если потибрили – то кто же сделал это первым. Потому что по сему поводу исчерпывающе высказался лирический герой Венички Ерофеева в поэме «Москва-Петушки»:

«*Никто этого не знает, и никогда теперь не узнает. Не знаем же мы вот до сих пор: царь Борис убил царевича Димитрия или же наоборот?*»

А начну я доисторический период с первых графических интерфейсов UNIX. Которая, как известно, зародилась и долгое время развивалась в качестве чисто текстовой. Настолько чисто, что сама идея прикручивания к ней какого-либо GUI воспринималась как ересь. Но, однако, идея эта была реализована.

Первой на данное поприще вступила фирма Sun, о которой я говорил в «Берклиаде» (**LXF146**, июль 2011), изначально тесно связав свою SunOS со средствами поддержки GUI собственной разработки. Изначально таким GUI'ем была SunView (Sun Visual Integrated Environment for Workstations, изначально SunTools) – оконная система, возникшая едва ли не одновременно с самой SunOS в первой половине 80-х годов. От всех последующих оконных систем, ориентированных на UNIX, она отличалась тем, что большая ее часть поддерживалась ядром. Она включала набор стандартных пользовательских приложений – текстовый редактор, почтовый клиент, инструменты для настройки. И, наконец, она являлась частью стандартной поставки ОС. По-видимому, именно в SunOS впервые была реализована интеграция ОС, GUI и пользовательских приложений, получившая дальнейшее развитие не только в мире UNIX, а и в более иных операционках, доживших до наших дней (их же имена читатель и сам ведает).

На смену SunView в середине 80-х годов прошлого века пришла NeWS (Network extensible Window System) – более сложная система, основанная на PostScript. Однако получить широкого распространения она не сумела.

Наконец, в конце 80-х годов для SunOS была разработана оконная среда OpenWindows. Сначала она поставлялась как отдельное дополнение, а потом была включена в состав операционной системы. Однако и она не успела снискала популярности.

Ибо тогда же, в конце 80-х годов, начинается смена аппаратной платформы для SunOS: процессоры Motorola 68XXX в серверах и рабочих станциях заменяются на «камни» собственной разработки и производства, RISC-процессоры SPARC. Для которых разрабатывается и новая ОС, названная Solaris 2. А с ее приходом происходит и отказ от всех GUI собственной разработки. Они сменяются ставшими стандартными для UNIX-мира оконной системой X и рабочей средой CDE. О последней речь пойдет в статье про десктопы. А вот к истории первой обратимся сейчас.

## Рождение Иксов

Программные средства обеспечения работы в графическом режиме можно условно разделить на две части: ту, что обеспечивает взаимодействие с аппаратурой, и собственно интерфейс

### Кстати о названии

Как гласят легенды древности, некогда существовала операционная система V. Поверх нее работала оконная система W, созданная в Стенфордском университете Полом Асенте [Paul Asente] и Брайаном Рейдом [Brian Reid]. В дальнейшем она была портирована на UNIX, хотя и работала там, как говорят, очень медленно. Поэтому, когда под UNIX была разработана новая оконная система, ее просто маркировали следующей буквой латинского алфавита.

с пользователем. Разделение это не строгое – только что я говорил о SunView, в которой эти части были неразрывны. Однако магистраль развития GUI лежала в области разделения этих частей, что наиболее последовательно было реализовано в оконной системе X (X Window System).

Иксы начали разрабатываться в 1984 году сотрудниками MIT Робертом Шайфлером [Robert Scheifler] и Джимом Геттисом [Jim Gettys], к которым скоро присоединился Рон Ньюмен [Ron Newman). Идея разработки заключалась в том, чтобы создать графическую систему, полностью независимую как от аппаратной платформы, так и от ОС, даже родительской UNIX, которая бы обеспечивала студентам MIT доступ ко всему зоопарку компьютеров, проживавшему в стенах этого заведения. Работа производилась ударно-стахановскими темпами. Первая версия новой системы вышла в мае 1984 года, а вышедшая в январе 1985 года версия получила порядковый номер 6. Это привлекло к Иксам внимание фирмы DEC, и Иксы были портированы на ее VAX'ы.

Версии Иксов, именовавшиеся X#, сменялись с калейдоскопической быстротой: уже во второй половине 1985 года появляется X9 – первая версия, распространяемая свободно, на условиях так называемой лицензии MIT (до этого институт распространял ее за плату). А на рубеже 1985–1986 годов выходит версия X10 – подряд в нескольких вариантах, так называемых реализациях: X10 – декабрь 1985 года, X10R2 – январь и X10R3 – февраль 1986.

Версия X10R3 получает широкое распространение. Кроме VAX, она портируется на машины Hewlett-Packard, рабочие станции Apollo, Sun и ряд других. В связи с этим возникла необходимость сделать ее окончательно независимой от аппаратуры, что и было выполнено с привлечением сил DEC в рамках версии X11.

Разработка версии X11 заняла необычно много времени для этого проекта – более полугода: финальный вариант ее вышел аж в сентябре 1986 года. Ход работы над ней широко обсуждался в Сети благодаря открытым спискам рассылки. Это был пример первого в истории масштабного проекта, разрабатываемого распределенным коллективом программистов, связанных лишь Интернетом. Таким образом, разработка X11 послужила прообразом современных крупномасштабных проектов FOSS.

Длительность разработки X11, тщательность тестирования и широта обсуждения проекта привели к тому, что эта система стала уникальным для софтверной индустрии долгожителем: номер главной версии с тех пор не менялся ни разу, добавлялись только номера реализаций. С формальной точки зрения те Иксы, с которыми мы имеем дело сейчас (то есть X.org), также являются представителями ветки X11. Именно поэтому для X Window System широко распространилось сокращенное именование – X11.

Иксы в своей 11-й ипостаси продолжали триумфальное шествие по «железным» архитектурам и их операционным системам. Только что я говорил, что ради Иксов Sun прекратила собственные разработки в области графического интерфейса. Та же судьба постигла и все иные графические системы для UNIX – даже имена их ныне забылись. Иксы стали стандартным средством обеспечения работы в графическом режиме всех без исключения UNIX'ов и UNIX-подобных операционок.

Кроме того, Иксы были портированы на VAX/VMS фирмы DEC, OS/2 от IBM и даже, страшно сказать, на Windows. Хотя о последнем факте и не любят говорить вслух.

Проект приобрел такие масштабы, что для управления им в 1988 году потребовалось создать специальную организацию, названную X-Консорциум MIT [MIT X Consortium], которая объединила в своем составе как университетские круги, так и компании, разрабатывающие коммерческие решения. В 1993 году на смену ему пришел X Consortium, выпустивший в мае 1994 года X11R6, реализацию, которая просуществовала более 10 лет: следующей, X11R7, суждено было появиться только в конце года 2005! И различные минорные ее версии (последняя по времени – X11R7.7, появившаяся в июне 2012 года) используются в дистрибутивах Linux и BSD-системах по сей день.

## Кто рубил окно

Официальным названием новой системы было – X Window System, причем слово Window появилось в ней более чем за год до его внедрения в имя некой графической оболочки для DOS, которую тогда никто не воспринимал всерьез ввиду непригодности к практическому использованию. И для которой оно стало не чем иным, как торговой маркой, охраняемой законом.

Тем не менее позже, в кругах, далеких от UNIX, некоторое распространение получило словосочетание X Windows. Однако, говоря словами все того же героя Венички, «это позорно и преступно». И истинный линуксоид до такого никогда не опустится – и не из-за пиетета перед законами.

А вот более краткое X System, несколько жаргонное X11 (почему именно 11 – станет ясно со временем) и, особенно, просто X, не только допустимо, но и широко используется краткости для.

## XFree86 и другие

Сама по себе оконная система X версии X11 представляла собой набор открытых спецификаций [reference implementation], доступных под лицензией MIT, на основе которых предлагалось создавать конкретные реализации – X-сервера с сопутствующими компонентами. Которые уже могли распространяться на любых условиях, в том числе и коммерческих.

Этим предложением не замедлили воспользоваться разработчики программного обеспечения для UNIX-систем. В результате чего появился ряд проприетарных X-серверов, обеспечивавших графику в столь же проприетарных UNIX'ах. Однако наибольшую известность получила свободная реализация Иксовых референсов – XFree86.

Прототип проекта XFree86 зародился в 1991 году в рамках X11R5: соответствующий спецификациям этого релиза X-сервер был разработан Томасом Роэллом [Thomas Roell] и Марком Снайтили [Mark W. Snitily] для 32-битных платформ на процессорах Intel, получив вполне ожидаемое имя X386. Этот сервер распространялся под проприетарной лицензией фирмы SGCS (ныне SGI), сотрудником которой являлись его авторы.

Это, однако, не помешало Дэвиду Вексельблату [David Wexelblat], Гленну Лэю [Glenn Lai], Дэвиду Доуэсу [David Dawes] и Джиму Цилласу [Jim Tsillas] в 1992 году внести в его исходники коррективы столь существенные, что у них были основания выделить их в отдельную версию, названную X386 1.2E и распространявшуюся свободно. Такое положение создавало определенную путаницу. И потому после обсуждения и обмена мнениями стороны постановили: переименовать новый проект в XFree86. Что по созвучию намекало и его на связь с корнями (X-three-eighty-six), и на характер распространения (X-free-eighty-six).

Не пропало и дело исходного проекта: X386 был переименован в Accelerated-X, и Томас Роэлл создал фирму Xi Graphics, которая на протяжении многих лет занималась его продажами, а поддержку осуществляет чуть ли не по сей день. У нас еще будет повод вспомнить про этот X-сервер.

А пока вернемся к XFree86. Во вступлении я уже говорил, что своей популярностью среди широких пользовательских масс Linux во многом обязан Иксам, и конкретно – именно свободной их реализации. Но, с другой стороны, XFree86 повсеместным своим распространением обязана Linux'у в не меньшей степени. Ибо усилиями Ореста Зборовски [Orest Zborowski] она заработала на Linux'е чуть ли не со дня своего рождения, а именно с апреля 1992 года.

Вообще, взаимоотношение XFree86 с Linux'ом на том этапе развития обеих систем в источниках освещено довольно противоречиво. Остается не вполне ясным: создавалась ли свободная »

## Мистер X

На русский язык официальное название X Window System переводилось столь же официально – Оконная система X. Однако используется оно редко – в особо торжественной обстановке. Ибо практически сразу было вытеснено кратким словом «Иксы» – пусть и жаргонным, но адекватным. Его я использовал ранее и буду использовать впредь в этом цикле, когда официоз покажется уж совсем неуместным.

реализация Иксов уже с прицелом на юную, но активно развиваемую операционку? Или еще под абстрактный UNIX, а на Linux была Орестом лишь портирована? Прямых ответов на эти вопросы мне найти не удалось.

Так что в эти детали я вдаваться не буду. Тем более, что в рамках нашей темы сейчас важнее другое: в октябре 1992 года XFree86 была включена в комплект, который можно назвать первым в истории настоящим дистрибутивом Linux. Им стал SLS (Softlanding Linux System), разработанный Питером Мак-Дональдом (см. **LXF148**, сентябрь 2011). И с тех пор судьбы XFree86 и Linux'а были неразрывно связаны на протяжении 12 лет, вплоть до событий «Великого раскола», о которых я расскажу в следующей статье.

Однако Linux не была единственной операционкой, в которой реально использовалась XFree86. В недрах архивов проекта можно обнаружить списки издревле поддерживаемых им ОСей. В их числе мы увидим такие, ныне забытые, имена, как Amoeba (первая попытка Таненбаума [Andrew S. Tanenbaum] построить «настоящую» микроядерную ОС) и BSDi (коммерциализированная сестричка FreeBSD), Mach, ушедший в тень MacOS X, и «игрушечная» Minix, благоздравствующая, но «незнаменитая» NetBSD и скандально ославившаяся SCO, SVR3 и SVR4, давно ставшие абстракциями учебников по Computer Science.

А вот следов поддержки FreeBSD в ранних версиях XFree86 найти не удается: похоже, что в ее портах свободные Иксы появляются во второй половине 1994 года (собственно, одновременно с самой системой портов). Однако с этого момента XFree86 оказывается связанной с FreeBSD не менее тесно, чем с Linux'ом – и со временем мы увидим, что эта связь окажется особенно важной для нашей страны.

Как уже говорилось, изначально XFree86 была основана на X11R5, и так продолжалось во всех версиях 1-й и 2-й веток (нумерация их не имеет никакого отношения к версиям и релизам собственно Иксов). Последней представительницей этой линии была XFree86 2.1.1, вышедшая в мае 1994 года. Однако почти одновременно появляется и новая спецификация Иксов – X11R6, которая ложится в основу новых реализаций X-серверов. В их числе была и XFree86 3.0, увидевшая свет в конце августа 1994 года.

И это были те самые Иксы, с которыми впервые столкнулось большинство из ныне действующих применителей Linux первого и второго призывов. Те самые, которые появляются в портах FreeBSD. Иначе говоря, те самые, которые стали, как принято говорить, стандартом de facto для всех существовавших тогда свободных UNIX-подобных ОС. А со временем – и для несвободных тоже, но это будет не очень скоро.

## XFree86: немного о внутренностях

Говоря о XFree86 как об одной из реализаций X-сервера, я был не совсем точен: это была не одна из реализаций, а множество реализаций оного. Откуда их столько взялось? Ответить нетрудно.

Ныне, когда речь заходит о видеоподсистеме и ее поддержке в Linux'е, все обычно сводится к священной войне между приверженцами Nvidia и ATI/AMD. От которой дистанцируются резонные применители, любовно поглаживающие в сторонке свои встроенные Intel'я. И подчас вспоминающие даже о существовавших не так давно SiS и Matrox. Однако представить себе пестроту видеорешений, сосуществовавших каких-нибудь 15–20 лет назад, может только тот, кто видел ее своими глазами.

Ибо это была пестрота восточного базара. На котором можно было встретить видеокарты на любом чипе из выпускавшихся в те годы: и откровенно рабоче-крестьянском Trident, и плебейски-претенциозном Cirrus Logic, и S3 – символе мещанского благополучия, и аристократическом ATI, скромном трудяге – Tseng и блистательном Imagine128, 3DLabs, знаменующем высоты профессионализма, и многих других, имена которых стерлись в памяти даже ветеранов. И чуть ли не каждому производителю видеочипов в составе XFree86 полагался свой персональный X-сервер, а некоторым их доставалось по два (как S3), а то и по три (как ATI).

Впрочем, производители видеочипов пренебрежительно относились к оказанному им уважению. И мало того, что сами маркировали поддержкой своей продукции, но и не предоставляли информации о ней независимым разработчикам. Что само по себе было объяснимо: конкуренция среди «видеочипмейкеров» тогда была бешеная. Но не оправданно – и безвременная кончина почти всех «фигурантов дела» из предыдущего абзаца стала тому подтверждением. А выживших заставила в той или иной форме «делиться»...

Но это будет еще не скоро. А пока работа многих свободных X-серверов – результата «слепого» реинжиниринга фирменных решений – часто оставляла желать лучшего. Вплоть до того, что иногда графический режим просто не удавалось запустить с «родным» сервером. Правда, в составе XFree86 имелись «всечиповые» сервера VGA и SVGA, работающие на любых картах с поддержкой соответствующих стандартов. Однако наблюдать на дисплее, подключенном к крутейшей видеокарте, что-нибудь вроде 640×480 при 16 цветах, доставляло тогдашним пользователям не много радости. Конечно, не все было так мрачно, и большинство карт на типовых чипах (особенно не гипермодерновых) с типовыми же X-серверами из поставок XFree86 работали вполне справно. Однако вопрос поддержки видеосистемы свободными Иксами в 90-х годах стоял весьма остро. И попытки его кардинального решения тем или иным образом предпринимались постоянно.

Первым направлением таких попыток было, разумеется, совершенствование самих X-серверов. На этом поприще особенно прославилась фирма S.u.S. E. – создатель одноименного дистрибутива Linux. Благодаря тесным контактам с фирмой Elsie – производителем высококлассных видеокарт, в том числе профессиональных – ее разработчики очень быстро получали информацию о новинках «видеожелеза» и столь же оперативно вносили патчи в свои реализации X-серверов.

Вторым направлением было использование коммерческих X-серверов, наподобие упоминавшегося выше Accelerated-X. Сами по себе они не были ни свободными, ни открытыми, но на определенных условиях могли распространяться (почти) бесплатно. Благодаря этому их нередко включали в состав дисковых наборов Linux-дистрибутивов и софта для них, вроде упомянутых в статье «Linux на Руси» (**LXF151**, декабрь 2011) боксов от Walnut Creek и InfoMagic. Правда, тут возникали свои сложности – как лицензионные, так и чисто технические. А для Руси этот путь оказывался закрытым напрочь, потому что тот же Accelerated-X в принципе не поддавался кириллизации.

Наконец, третий путь продемонстрировала в 1998 году маленькая и неприметная тогда фирма Nvidia. Выпустив незадолго перед тем «народную» 3D-карту Riva 128, она вскоре сопроводила ее и фирменным драйвером для работы в Linux'е – драйвером, работавшим безукоризненно.

Я не буду выстраивать причинно-следственных связей. Однако то, что звездный час Nvidia начался с Riva 128 и ее Linux'ового драйвера, остается медицинским фактом. На констатации которого я и поставлю запятую в своей истории. LXF

# Red Hat Enterprise Linux

предоставляет вам **производительность**, **масштабируемость**, **безопасность** и **надежность**, ранее доступные только на очень дорогих платформах

Самая популярная в мире Linux платформа для бизнеса

Обеспечивает высокую производительность, надежность, масштабируемость и безопасность

Сертифицирована ведущими производителями оборудования и разработчиками ПО


Совместима с широким спектром оборудования от рабочих станций до серверов и мэйнфреймов

Обеспечивает одинаковые условия работы приложений при использовании в физической, виртуальной и облачной средах

## Пользователи RHEL экономят на оборудовании, лицензиях на программное обеспечение и эксплуатационных расходах

redhat
PREMIER BUSINESS PARTNER

### ГНУ/Линуксцентр — Linux-эксперт для вашего бизнеса

- Premier Business Partner компании Red Hat
- 12 специалистов по разработке и внедрению, сертифицированных компанией Red Hat
- Более 100 клиентов, использующих Red Hat
- 10 лет на рынке

### Red Hat — ведущий серверный дистрибутив Linux

- Более 15 лет промышленного использования
- Свыше 80% рынка корпоративного Linux по данным CIO Insight
- 5 лет среди лучших вендоров
- Выгодная совокупная стоимость владения (TCO)
- Поддержка в течение 10 лет


**Специальное предложение!**

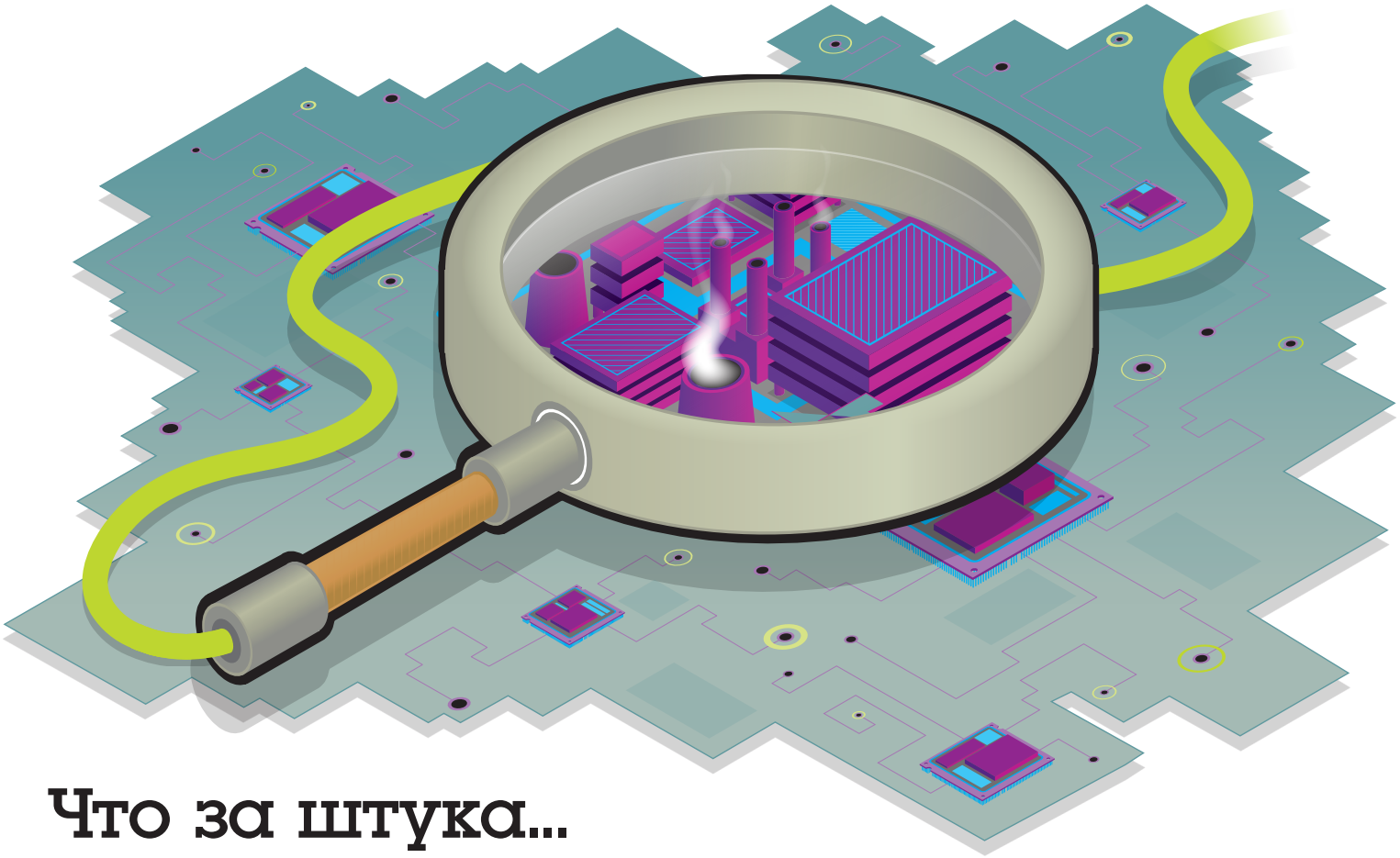
**Закажите Red Hat Enterprise Linux в ГНУ/Линуксцентре и получите в подарок книгу «Полное руководство пользователя Red Hat Enterprise Linux»**

Москва +7 (499) 271-49-54

Санкт-Петербург +7 (812) 309-06-86

Linux-эксперт для вашего бизнеса. www.linuxcenter.ru

Linux center



## Что за штука...

# System On A Chip

**Шашанк Шарма** расскажет о микроскопической технологии, лежащей в основе Raspberry Pi.

**В: А я-то думаю, где же я слышал этот термин. Конечно, Raspberry Pi!**

О: Точно. В основе Raspberry Pi – действительно System on a Chip, «система на кристалле», SoC, от Broadcom BCM2835.

**В: Ясно. Но что это, собственно, за система такая?**

О: В самом широком смысле, это микрочип со всеми компонентами, необходимыми для работы системы. В случае RPi, чип Broadcom включает процессор ARM, с частотой 700 МГц, и графический ускоритель Videocore 4.

**В: Минуточку! И что, все SoC настолько маломощные?**

О: Вовсе нет. Просто у RPi не те приоритеты, что у других подобных устройств. На самом деле, новые SoC поддерживают многоядерные процессоры.

**В: Как и моя материнская плата. Что же тогда особенного в SoC?**

О: Во-первых, на материнской плате компоненты находятся на разных чипах. Отдельно центральный процессор, отдельно графический, память и т.д. А в System on a Chip они совмещены в одном чипе, размером не превышающем обычный процессор.

**В: А! Значит, именно поэтому RPi получился таким миниатюрным?**

О: Совершенно верно! Системе требуется множество аппаратных компонентов, и в то время как в корпусе обычного ПК достаточно места, чтобы все их пристроить, на устройстве вроде RPi это было бы просто невозможно. С System on a Chip мы получаем мощные машины гораздо компактнее, не больше смартфона, да еще и с аккумулятором.

**В И это еще одно преимущество, да? Таким устройствам не нужна большая мощность.**

О Верно – благодаря очень плотной интеграции компонентов, что позволяет обойтись без лишних проводов, а значит, добиться большей эффективности. Возьмите тот же RPi, который потребляет всего 5 В через MicroUSB типа B. Собственно, я свой запитал через зарядник для смартфона!

**В А помимо RPi и смартфонов, в каких еще устройствах используется SoC?**

О Если устройство умещается в руке и работает на аккумуляторе, велики шансы, что это System on a Chip. Так что, помимо смартфонов, эта технология используется и в планшетах. По факту, большинство самых популярных наладонников Android работает на SoC – Nvidia Tegra 3 и Qualcom Snapdragon. Она же станет основой и для Microsoft Surface Tab.

> «Если устройство умещается в руке и работает на аккумуляторе, это System on a Chip.»

**В Потрясающе! Говоря о RPi и чипе Broadcom, вы упомянули только процессоры. А где же память?**

О Ну, во второй версии RPi имеется 512-МБ SDRAM, размещенное непосредственно на чипе Broadcom, посредством технологии пакет-на-пакете [package-on-package], или PoP.

**В PoP? Звучит как быстрая и дешевая заплатка для протекающей крыши.**

О Ваша догадка ближе к истине, чем вы думаете, по крайней мере, в плане быстроты и экономичности. Иногда создавать SoC специально для некоторых типов устройств просто нецелесообразно. На небольших системах, таких как RPi и других популярных открытых продуктах, вроде BeagleBoard, производители экономят место (и деньги), надстраивая множество чипов или «пакетов», как они предпочитают их называть, друг на друга.

**В Ну они же их не просто склеивают, я полагаю?**

О В чипах SoC используется технология поверхностного монтажа, известная как корпус BGA. Подсоединяются они при помощи маленьких шариков, расположенных снизу и сверху. Разработчики припаивают нижние, чтобы подсоединить чип к плате, а верхние используются для подключения памяти. Это дает больше возможностей, поскольку позволяет устанавливать пакеты разных производителей. Так, на некоторых платах RPi стоит память Hynix, а на других – Samsung.

**В OK. То есть в SoC нет своей собственной памяти?**

О Нет, этого я не говорил. Не все SoC устроены одинаково. Некоторые включают больше компонентов, за счет все той же технологии PoP. Все зависит от функционального назначения устройства.

**В Что же еще может обнаружиться в SoC?**

О Помимо центрального и графического процессора и памяти, SoC может включать также Northbridge, контроллер, осуществляющий взаимодействие между процессором и другими компонентами SoC. В некоторых есть еще и Southbridge, контроллер функций ввода-вывода. А в любой SoC, предназначенной для коммуникации, будет также сотовый и прочие приемники для 4G, Bluetooth или Wi-Fi подключений.

**В Продолжая тему: могут ли в SoC использоваться процессоры ARM?**

О Могут. Да в большинстве SoC процессор ARM будет по умолчанию. Это для SoC предпочтительный вариант, ведь их архитектура обеспечивает высокую производительность при низком энергопотреблении, что делает их идеальным решением для мобильных платформ. Архитектура x86, столь популярная в настольных системах, напротив, проигрывает в энергоэффективности.

**В Вы хотите сказать, что существуют еще и SoC на x86?**

О Единственным производителем мобильных устройств с такой технологией является Intel. Называется она Atom Medfield. И впервые была использована в смартфоне IntelAZ210, в Великобритании известном как Orange San Diego.

**В Если SoC более компактные и менее энергоемкие, почему они не применяются вообще везде?**

О Хороший вопрос. Как бы ни были достойны эти системы, их преимущества тоже имеют свою цену. Из-за своей плотной интеграции они лишены гибкости, необходимой на настольном ПК или ноутбуке. Ведь на ПК вы можете поставить другой процессор или графический ускоритель, можете увеличить память, а на смартфоне – нет.

**В Приехали. Значит, SoC не выйдет за пределы мобильной сферы?**

О Пару лет назад все так и думали, но уже перестали. Традиционные процессоры берут пример с SoC, совмещая контроллер памяти, шину PCI Express и графический процессор в одном чипе. Главные примеры – Llano от AMD и Valley View от Intel. А мобильные SoC становятся все мощнее, пример – Samsung Exynos 5, основа планшета Google Nexus 10, а также последнего поколения Samsung Chromebook.

**В Ого! А раз в Chromebook используется Chrome OS на основе Linux, значит, все системы SoC тоже его поддерживают?**

О Ну, этот вопрос чреват долгими пояснениями. Как я уже сказал, в большинстве SoC используются процессоры ARM, и многие из этих устройств, такие как смартфоны Android, планшеты и RPi, работают на Linux. Но это не какая-то универсальная версия Linux. На всех этих устройствах они немного отличаются. На самом деле, поддержка различных SoC на основе ARM – это для разработчиков Linux огромный труд. По некоторым данным, каждый релиз ядра имеет более 70000 новых строк ARM-кода, по сравнению с примерно 5000 для платформ x86! Тем не менее, начиная с Linux Kernel 3.7, различные ARM SoC платформы будут иметь единую версию ядра.

**В То есть в ближайшем будущем я смогу установить свой любимый дистрибутив Linux на любое устройство ARM, так?**

О По крайней мере, мы к этому стремимся. Наиболее популярные устройства ARM SoC на Linux – RPi и Chromebook. На последний некоторые умудряются даже установить Chrome OS, параллельно с полнофункциональной Ubuntu или Fedora.

**В Не устаю твердить друзьям, что мобильные устройства — это будущее компьютеров. По-видимому, SoC — тоже.**

О Без сомнения, но обычным процессорам тоже всегда будет место на рынке, в тех сферах, где энергопотребление и размеры устройства не столь важны. Скажем, высокопроизводительный сервер или суперкомпьютер. LXF

# По рецептам доктора Брауна

**Д-р Крис Браун**
Доктор обучает, пишет и консультирует по Linux. Ученая степень по физике элементарных частиц ему в этом совсем не помогает.

Эзотерическое системное администрирование из причудливых заворотов кишок серверной

## Слово дня

Некоторые слова и выражения надолго переживают предметы, к которым они изначально относились. Многие из более молодых читателей, наверное, никогда и не видели сапога с ушком [a boot with a bootstrap], хотя свободно говорят о перезагрузке [rebooting] своего компьютера (ну, если они пользуются Linux, то, наверное, не перезагружаются так часто, но это другая история).

Если вы когда-нибудь копировали простой текстовый файл из Windows в Linux, то видели, что Linux добавляет в конец каждой строки только символ LF, а вот в Windows нужен и символ CR (carriage return – возврат каретки). Но что такое каретка и почему ее нужно возвращать? Если вы никогда не видели старой печатной машинки, вы этого не поймете.

### Сдвиг языка

Печатным машинкам мы обязаны и клавишей Shift, которая физически сдвигала либо литерные рычаги, либо каретку так, чтобы один литерный рычаг мог напечатать два символа. Название клавиши TAB – сокращение от «табулятор [tabulator]», а символ BEL (ASCII-код 7) напоминает нам, что на некоторых терминалах был настоящий маленький колокольчик, который звонил при получении этого символа.

Это приводит меня к телетайпам (первым компьютерным терминалам, которыми я пользовался), которые давно канули в прошлое, но объясняют, почему текстовые терминалы в Linux называются tty. Но любимейший мой пример – сигнал SIGHUP. Что же он значит? Так вот, HUP – сокращение от “hang up [отсоединение]”, и изначально этот сигнал применялся для принудительного завершения оболочки при отключении пользователя на коммутируемой линии, т. е. когда тот «вешал [hung up]» трубку. Но прошло уже много-много времени с тех пор, когда у телефонов был рычаг, на который можно было что-то повесить.

chris.linuxformat@gmail.com

# OpenNebula

## Никогда не стеснявшийся смешанных метафор доктор создает облако в песочнице.

*Rightscale*, *scalr*, *cloudkick*, *enstratus* – утилит управления облаком больше, чем можно сосчитать. Многие из них стоят денег, но *OpenNebula* – нет. *OpenNebula* – открытый проект, который разрабатывает – цитирую его сайт – «решение уровня предприятия для построения виртуализированных дата-центров и облаков IaaS и управления ими». Недавно они выпустили несколько виртуальных образов, с которыми все это очень легко попробовать. В них используется CentOS 6.3 с настроенным клиентом *OpenNebula 3.8.1* и хостом виртуализации на базе *QEMU*. Эти образы можно запускать в *VirtualBox*, *VMWare* или *KVM*, есть даже образ Amazon EC2.

Я загрузил образ для *VirtualBox*, выполнил инструкции по установке, и все заработало. Проясним, что происходит: на родной ОС моего ноутбука запущена виртуальная машина (ВМ) с установленной программой *OpenNebula*, которая может создавать «внутренние» ВМ и управлять ими. От самой мысли о ВМ, работающей внутри другой ВМ, у меня болит голова, но свою задачу эта схема выполняет – в ней можно быстро начать пользоваться *OpenNebula* и почувствовать, что это такое.

Во-первых, в ней есть утилиты командной строки для создания ВМ и управления ими. Например, новый экземпляр можно создать так:

```
onevm create --name “tty3” --memory 128 --cpu 0.5 --disk ttylinux --network cloud
```

Можно создавать шаблоны и новые экземпляры на их основе.

В виртуальном образе есть один (маленький) образ Linux; его можно загрузить. Никто не ждет, что в этих системах будут реально работать, а если вы хотите запустить и поработать с несколькими образами, выделяйте память экономнее. Есть еще web-интерфейс *SunStone*, он приятнее на вид.

www.opennebula.org

## Каталог образов

Каталог образов OpenNebula (The Marketplace) – это онлайн-каталог, в который разработчики могут загружать свои виртуальные образы. На данный момент их 16 – CentOS, SUSE, Debian, Ubuntu (каждый в версиях для различных гипервизоров), виртуальный роутер и образы для песочницы, о которых я говорил. В *SunStone* даже есть специальная вкладка, с которой легко выбирать изображения и импортировать их в свою локальную инфраструктуру.


**› На панели управления утилиты *SunStone* – информация об использовании системы.**

# SOCIAL AND MOBILE COMMUNICATIONS*

*ВСЕ О СОЦИАЛЬНЫХ СЕТЯХ, ИНТЕРНЕТ-СМИ И МОБИЛЬНЫХ КОММУНИКАЦИЯХ

## 5-6 МАРТА 2013
DIGITAL OCTOBER

WWW.i-COMFERENCE.RU

theRunet
Бизнес. Медиа. Будущее.

i-COM FEREN CE'13

# Стек LAMP

Вторая статья серии, в которой мы увидим, как HTML и PHP играют друг другу на руку, чтобы создать web-приложение.

Быть web-разработчиком непросто, потому что нужно уметь многое. Нужно изучить как минимум четыре языка (для стека LAMP это PHP, HTML, CSS и SQL) и, пожалуй, немного JavaScript. Нужно понимать структуру реляционной базы данных и немного представлять человеческие факторы, окружающие разработку web-приложений. Если все это у вас есть, рынок труда для вас открыт. А есть или нет – это большой вопрос.

Но с чего-то начинать все же надо, и в этом месяце мы рассмотрим HTML и PHP и увидим, как они работают вместе при создании простого web-приложения. В прошлом месяце мы начали устанавливать стек LAMP, и если вы хотите следовать за моим повествованием (а я надеюсь, что да), у вас должны быть установлены по крайней мере *Apache* и PHP.

Начнем со старого доброго “Hello World” на HTML. Есть ли смысл это делать? Конечно, есть! Для соответствия стандартам HTML-документ должен иметь довольно специфическое обрамление, поэтому наш файл должен выглядеть так:

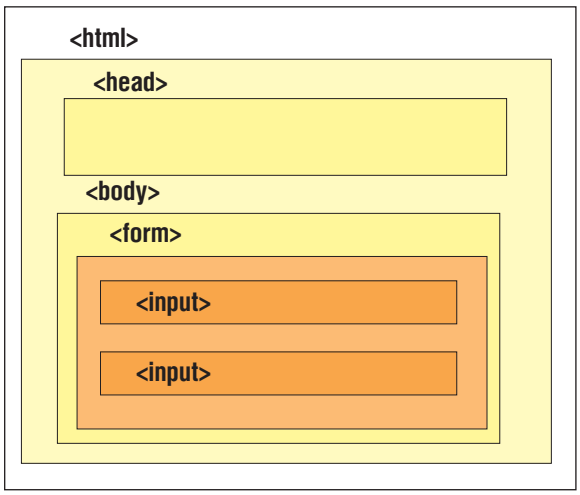
```
<!DOCTYPE html>
<html>
<head>
<title> Hello from HTML </title>
</head>
<h1> Hello World! </h1>
</body>
</html>
```

Строго говоря, мы должны говорить об XHTML, а не об HTML. XHTML строже HTML и появился в основном потому, что разные браузеры отображают HTML по-разному. Например, в XHTML открывающие тэги *должны* иметь закрывающие, и все тэги и атрибуты должны быть в нижнем регистре. На практике же браузеры очень толерантны и сделают все возможное, чтобы правильно обработать неправильный код; но это не является оправданием для неряшливого кода.

Если вы установили Apache с корневым каталогом документа **/var/www/html**, как мы сделали в прошлом месяце, поместите этот код в файл **/var/www/html/greet.html**, затем откройте его в браузере (т. е. откройте адрес http://localhost/greet.html). Тэг <h1> задает заголовок первого уровня, поэтому вы должны увидеть приятное большое “Hello World”.

Формы – это контейнеры, которые содержат элементы управления, с которыми взаимодействуют пользователи: текстовые поля, галочки, кнопки и т. д. Это ключевые компоненты любого web-приложения.

Наш первый пример – простой калькулятор с четырьмя операциями. Смотреть в нем особенно не на что, но с его помощью мы увидим, как пользователи вводят данные в web-приложение




**› HTML-тэги задают вложенную структуру документа. Здесь мы видим поля ввода внутри формы внутри...**

и как эти данные можно получить и обработать на стороне сервера с помощью PHP. Для начала создадим копию нашей страницы “hello world”:

```
$ cd /var/www/html
$ cp greet.html calculator.html
```

Теперь откройте файл и замените “hello world” разметкой формы, как показано ниже. Номера строк добавлены для удобства ссылок, они не являются частью файла:

```
1. <form action = “calculator.php” method = get>
2. Округлить до целого
3. <input type=”checkbox” name=”round”> <br>
4. <input type=”text” size=”10” name=”op1”>
5. <select size=”1” name=”operator”>
6. <option selected> +
7. <option> -
8. <option> *
9. <option> /
10. </select>
11. <input type=text size=10 name=”op2”>
12. <input type=submit value=”Calculate”>
13. </form>
```

Открыв http://localhost/calculator.html в браузере, вы должны увидеть форму. Немного разберем разметку: форма начинается в строке 1 и заканчивается в строке 13. Открывающий тэг в строке 1 определяет два важных атрибута: атрибут **action** задает web-страницу, которой передает данные форма, т. е. когда пользователь нажимает кнопку Submit [Отправить], браузер запросит эту страницу и включит в запрос информацию, введенную пользователем в форму. Атрибут **method** определяет, как эта информация включается в запрос. На форме пять элементов управления:


**› Наша первая форма. Нет, не видать ей призов за лучший дизайн.**

## Устранение ошибок

Интерпретатор PHP, находя синтаксические ошибки в коде, по умолчанию сообщает о них в журнал ошибок *Apache* (**/var/log/httpd/error_log**) и возвращает браузеру пустую страницу. Это поведение можно изменить, подправив настройки в **/etc/php.ini**. Откройте его и найдите строку с переменной **display_errors**. Установите ее в “On” и сохраните файл. Затем перезапустите *Apache*: **# service httpd restart**. Это заставит PHP сообщать об ошибках в браузер, а не загадочно молчать.

галочка (строка 3), два текстовых поля для операндов (строки 4 и 11), выпадающий список для выбора арифметического оператора (строки 5–10). В строке 12 создается кнопка, которая отправляет данные формы. Обратите внимание, что у всех элементов управления есть имена – например, текстовые поля называются **op1** и **op2**. Мы используем эти имена для получения содержимого элементов управления в нашем скрипте на PHP.

Теперь рассмотрим страницу, где обрабатываются данные формы. Создайте файл **/var/www/html/calculator.php** со следующим содержимым (но без номеров строк):

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title> Calculator Results </title>
5. </head>
6. <body>
7. <?php
8. $op1 = $_REQUEST[‘op1’];
9. $op2 = $_REQUEST[‘op2’];
10. $op = $_REQUEST[‘operator’];
11. if ($op == ‘+’) $res = $op1 + $op2;
12. if ($op == ‘-’) $res = $op1 - $op2;
13. if ($op == ‘*’) $res = $op1 * $op2;
14. if ($op == ‘/’) $res = $op1 / $op2;
15. if (isset($_REQUEST[‘round’]))
16. $res = round($res);
17. echo “result is $res”;
18. ?>
19. <br/><a href=”calculator.html”>
20. Return to calculator </a>
21. </body>
22. </html>
```

Разберем его подробно. В строках 1–6, 21 и 22 задается стандартный каркас HTML. Наш код PHP ограничивается тэгами в строках 7 и 18. Первое, что мы делаем – достаем значения элементов формы. Например, в строке 8 мы получаем строку, которую пользователь ввел в текстовое поле **op1**.

Как видите, переменные в PHP начинаются с **$**. **$_REQUEST** – пример ассоциативного массива; он похож на обычный массив, только в качестве индекса используется строка, а не число. PHP автоматически заполняет этот массив всей информацией, принятой как часть запроса методами **GET** или **POST** или через куки [cookie]. В строках 11–14 выполняется соответствующее вычисление в соответствии с оператором, выбранным в выпадающем списке. В строке 15 проверяется, стоит ли галочка “Round to integer [Округлить до целого]”; если да, в строке 16 мы выполняем округление встроенной функцией **round()**. В строке 17 мы возвращаем результат браузеру. В строках 19 и 20 (обратите внимание, что мы вышли за тэги PHP) создается ссылка (с технической точки зрения называемая якорем), которая возвращает нас обратно на главную форму.

Для проверки откройте адрес http://localhost/calculator.html, заполните форму, нажмите кнопку Calculate [Вычислить] и насладитесь результатом. Если вы понимаете, как эти два файла работают вместе, вы почти полностью понимаете, как работают web-приложения. Важная идея, которую вам нужно понять, состоит в том, что управление постоянно передается от клиента (браузера) к серверу и обратно. Браузер запрашивает исходную страницу, сервер отправляет ее, пользователь взаимодействует со страницей и отправляет введенные данные, попадая на ту же самую страницу или на другую и т.д.

## Велик ли процент?

Вот еще один пример, в котором мы вычисляем ежегодный сложный процент банковского счета за заданное количество лет.

### PHP в двух словах

Синтаксически PHP напоминает семейство языков C – те же точки с запятой, фигурные скобки, арифметические, битовые, реляционные и логические операторы, циклы и ветвления. И сходство библиотеки ввода/вывода со стандартной библиотекой C мало похоже на случайное совпадение. Но PHP – не компилируемый, а интерпретируемый язык, и, как и в других языках на “P” (Perl и Python), в нем используется динамическая типизация. У него широкие взгляды на “true” и “false”, и он невероятно толерантен в выражениях со смешанным типом, допуская вещи вроде **$x = 123 + “456”**, которые вызвали бы змеиный шип у большинства компиляторов.

Простоты ради, на наш счет будет нельзя класть и снимать деньги: там будет только первоначальный баланс. Как и в предыдущем примере, у нас будут простая форма для ввода данных на чистом HTML и вторая страница для обработки данных. Сначала рассмотрим форму ввода данных. Для краткости я привел только тело страницы, убрав внешний каркас.

```
<h2>Расчет банковского процента</h2>
<form action=”calc_interest.php” method=”POST”>
  Исходный баланс: <input type=”text” name=”startbal”>
<br/>
  Процент дохода (%): <input type=”text” name=”intrate”>
<br/>
  Начальный год: <input type=”text” name=”startyear”>
<br/>
  Конечный год: <input type=”text” name=”endyear”><br/>
  <input type=”submit” value=”Calculate”>
</form>
```

Здесь нет особых новшеств: просто четыре текстовых поля и кнопка Calculate. Теперь перейдем к PHP. Опять же, я привожу только тело формы:

```
<?php
// Получить параметры ввода из формы
$balance = $_REQUEST[‘startbal’];
$intrate = $_REQUEST[‘intrate’] / 100.0;
$startyear = $_REQUEST[‘startyear’];
$endyear = $_REQUEST[‘endyear’];
// Построить таблицу с заголовками
echo ‘<table cellpadding=6 ><tr><td>Year</td><td>Interest</
td><td>Balance</td></tr>’;
// Цикл по годам
for ($year = $startyear; $year <= $endyear; $year++)
{
$interest = $balance * $intrate;
$interest = round($interest, 2);
$balance += $interest;
echo “<tr><td>$year</td><td>$interest</td><td>$balance</
td></td>”;
}
echo ‘</table>’;
?>
```


**› Простая форма, используемая калькулятором процентов для ввода данных.**

## GET и POST

GET и POST – два метода, используемые для передачи на сервер данных, введенных в форму. В методе GET информация добавляется к концу URL. Преимущество этого подхода в том, что адрес можно добавить в закладки и потом повторно зайти на страницу с этими же параметрами. В методе POST данные формы незаметно включаются в тело HTTP-запроса, поэтому увидеть их сложнее. Для наших простых примеров прекрасно подойдет любой из методов.

Общая схема та же – выдергиваем поля ввода из формы, выполняем вычисления и возвращаем результаты. В этом примере мы видим простой цикл **for**, который вы узнаете, если знакомы с любым из языков семейства C. Обратите внимание на неуклюжую смесь HTML-тэгов и переменных PHP в операторах **echo**. Это типичная ситуация. Тэги в основном относятся к таблице для отображения результатов.

## Сохранение состояния

Одна из сложностей при написании web-приложений состоит в сохранении «состояния» в течение срока жизни приложения. Классический пример – корзина в электронном магазине: Шерон открывает каталог, выбирает товары для покупки. Каким-то образом где-то нужно запомнить, что она выбрала, и *Apache*, разумеется, не будет этого делать. Он и не подозревает, что последовательность обработанных им HTTP-запросов представляла выбор и покупку товаров Шерон.

Вот простейший пример, который я смог предложить для иллюстрации проблемы: у приложения есть страница с единственной кнопкой. При каждом нажатии кнопки пользователем счетчик увеличивается, и его значение отображается на странице. Звучит довольно просто, и если бы вы писали обычную программу, это и в самом деле было бы тривиально. Вы просто бы взяли статическую переменную и продолжали инкрементировать ее. Переменная находится там, инкрементируется, и ее значение запоминается. Это можно попробовать написать в виде web-приложения таким образом:

```
<body>
<form action="counter.php" method="get">
<input type="submit" value="Count">
</form>
<?php
$count = $count + 1;
echo "<br/> count is $count ";
?>
</body>
```

Обратите внимание, что эта страница открывает самое себя – очень распространенный подход в web-приложениях. Беда только в том, что работать она не будет. При каждом запросе страницы серверный код инициализируется заново, и счетчик никогда не вырастет больше единицы. Есть и другая проблема: предположим, что мы как-то сможем создать статическое хранилище на стороне сервера. Помня о том, что одновременно с программой могут работать несколько пользователей, хотим ли мы подсчитать общее количество их щелчков? Наверное, нет. Скорее мы захотим вести отдельные счетчики для каждого.

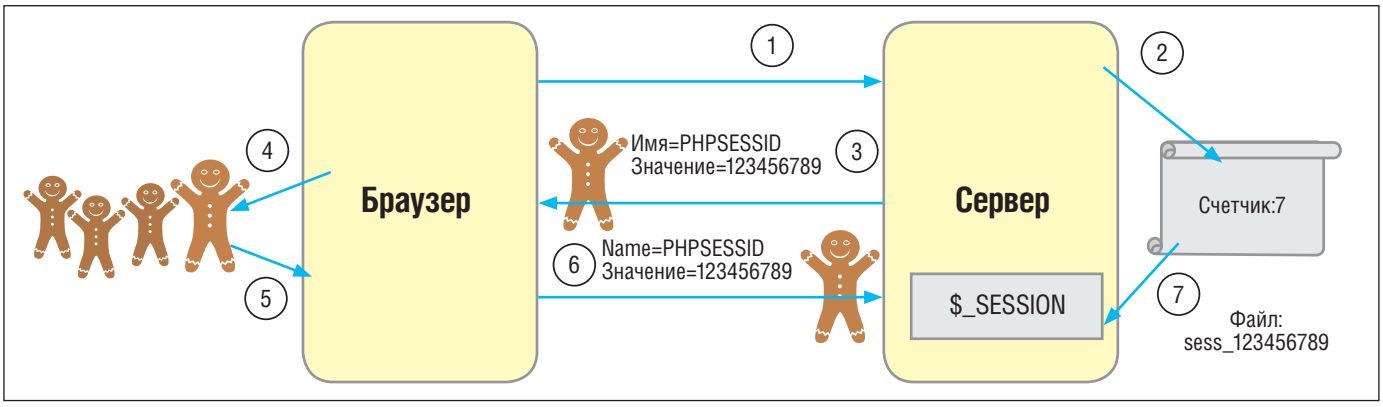
«Сохранение состояния», т. е. хранение информации о действиях каждого пользователя напрямую, не свойственно web-приложениям, так как web-серверы по сути не имеют состояния – они обрабатывают каждый запрос отдельно и сразу забывают о нем. Однако на помощь приходит PHP с ассоциативным массивов **$_SESSION**. Поместив данные в этот массив, приложение может сохранить их в течение последовательности взаимодействий, известной как сессия. Перепишем наше приложение со счетчиком:

```
1. <body>
2. <form action="counter2.php" method="get">
3. <input type="submit" value="Count">
4. </form>
5. <?php
6. session_start();
7. if (! isset($_SESSION['counter']))
8. $count = 0;
9. else
10. $count = $_SESSION['counter'];
11. $count = $count + 1;
12. $_SESSION['counter'] = $count;
13. echo "<br/> count is $count ";
14. ?>
```

Вот как это работает. При первом вызове **session_start()** в строке 6 создается новая сессия. Все данные, которые сохраняются в этой сессии, хранятся во временном файле на сервере. PHP формирует случайный идентификатор сессии, и он включается в имя файла. Чтобы получить корректную информацию о сессии при следующем подключении клиента, PHP возвращает браузеру куки. Имя куки – PHPSESSID, а значение – идентификатор сессии. Браузер возвращает куки обратно со следующим запросом, чтобы сервер мог получить идентификатор сессии и, следовательно, информацию сессии. В строке 7 мы проверяем, есть ли в сессии наш счетчик. Если нет, в строке 8 мы обнуляем счетчик – это происходит только при первом заходе на страницу. В противном случае мы получаем данные сессии (строка 10). После обновления данных сессии (строка 11) мы помещаем их обратно в сессию (строка 12). Конечно, наш простенький счетчик – лишь символическое изображение состояния, которое мы пытаемся хранить; в реальном мире это было бы нечто более масштабное.

Как вы, наверное, знаете, сейчас существует европейский закон о куки, и если вы пользуетесь данной технологией на публичном сайте, то перед отправкой куки вы обязаны получать на это информированное согласие посетителей сайта. LXF

**› При сохранении состояния сессии PHP возвращает браузеру куки с идентификатором сессии, чтобы восстановить состояние сессии при следующем запросе.**



LINUX FORMAT

# Учебники

## Наши эксперты помогут вам с любым приложением Linux!

**ЕВГЕНИЙ БАЛДИН**
**Подтвердивший свою квалификацию физик.**

## Мы разные, и это здорово!

Если вы заметили,
что вы на стороне большинства,
это верный признак того,
что пора меняться.
*Марк Твен Сэмюэл Лэнгхорн Клеменс*

Эта колонка – не столько о GNU/Linux, сколько о сообществе вокруг него. Когда пытаешься охарактеризовать это сообщество одной-двумя фразами, оно не очень-то получается. Ну да, эти люди называют GNU/Linux своим хобби. И что? У кого-то хобби футбол смотреть, что не характеризует однозначно эту разномастную толпу. Общаются в основном через Интернет? А что же делать, если мы пока меньшинство? Интерес к GNU/Linux не ограничивает нас в других предпочтениях и не вынуждает сбиваться в резервации.

Так вышло, что мне надо весной лететь в Аргентину. Не то что я туда не хочу, но для меня это буквально обратная сторона Земли, со всеми прелестями долгих рейсов и недешевых билетов. Случилась эта необходимость довольно внезапно. Мои знания об Аргентине почти исчерпываются сведениями из книги Жюля Верна «Дети капитана Гранта». Но я помнил, что там невдалеке Антарктида, и в неком сумеречном прозрении спросил на linux.org.ru: «Как попасть к пингвинам из Аргентины?» И мне ответили! Нашелся человек, побывавший в искомом месте с российской океанографической экспедицией, и он сообщил о пингвиньих толковищах и ценах на их посещение. Думаю, я не сильно ошибусь, сказав: фанаты GNU/Linux есть везде, и они помогут в любой задаче.

e.m.baldin@inp.nsk.su

## В этом месяце вы научитесь...















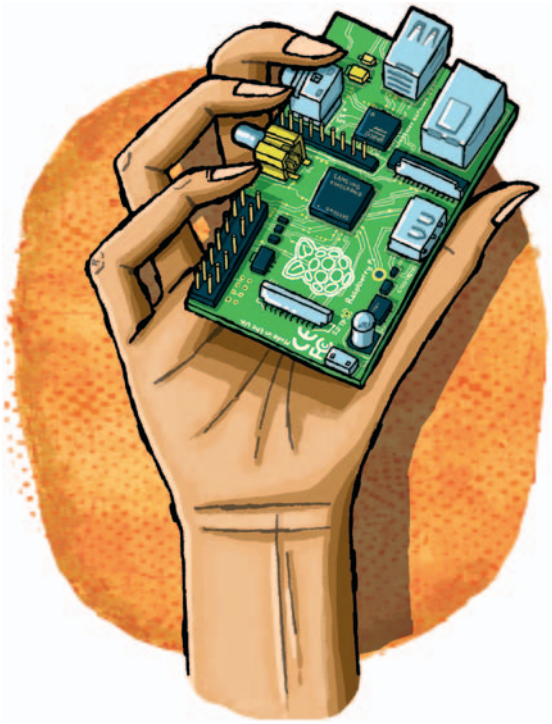


# Raspberry Pi:

**Бен Эверард** помогает освоить два стандартных языка, входящих в Raspbian, и написать пару простых программ.


## Наш эксперт

**Бен Эверард** оставил работу IT-консультанта и на два года отправился в Танзанию устанавливать системы на базе Ubuntu в местных школах. Теперь его знания находят применение в бурлящем котле открытий – редакции *Linux Format*.

На этом уроке мы вернемся к изначальной идее Raspberry Pi: обучению пользователей новым технологиям. На следующих четырех страницах мы кратко пройдемся по двум языкам программирования, входящим в Raspbian, рекомендуемый дистрибутив для Pi. Если у вас нет Pi, вы все равно можете следовать за нами – просто установите эти языки через менеджер пакетов. Если вы не программировали раньше, не переживайте: мы начнем буквально с нуля [*англ.* from scratch, – *прим. пер.*].

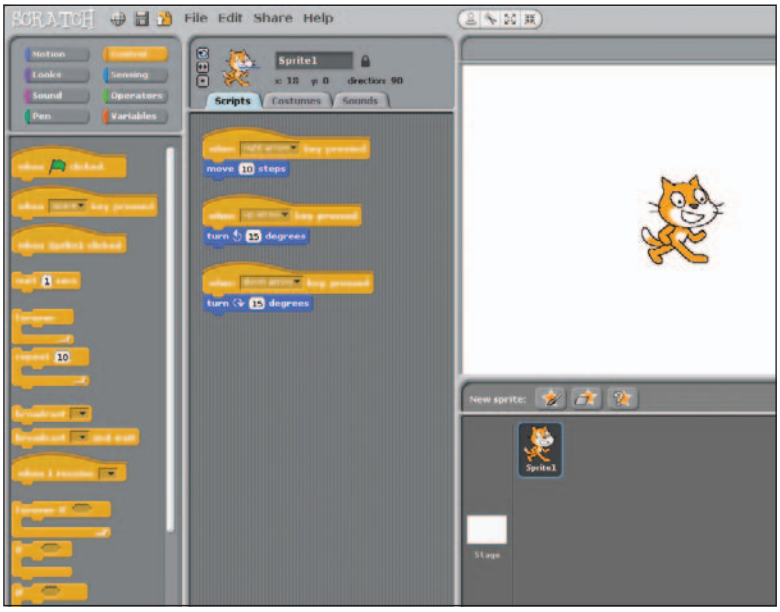



**Рис. 1. Блоки окрашены в разные цвета — чтобы было понятнее, из какого меню их выбрать.**

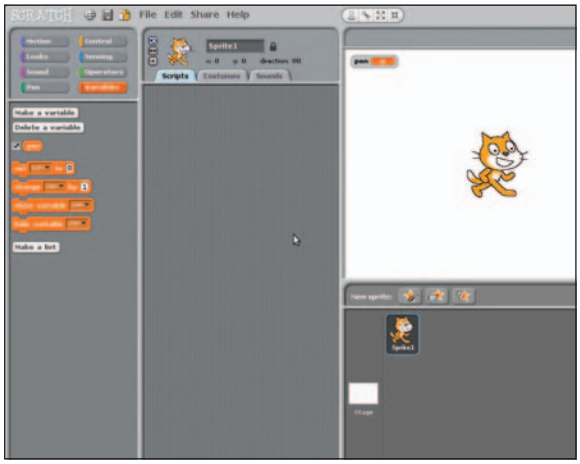



**Рис. 2. По умолчанию переменным присваивается значение 0, а за их текущим значением можно следить в области рисования.**

Scratch – прекрасный язык для изучения основ программирования: с ним не нужно беспокоиться о совершенстве кода. Так как программа пишется путем перетаскивания блоков в скрипт, не придется запоминать ни одной команды.

Мы напишем простую программу для рисования линий на экране с помощью клавиш управления курсором. Первое, что нам нужно – создать код, который позволит перемещать по экрану картинку с котом. Мы воспользуемся тремя отдельными блоками, каждый из которых будет запускаться по нажатию клавиши.

Нажмите Control [Управление], затем перетащите в скрипты блок **When Space Key Pressed** [При нажатии клавиши пробела]. Так создается скрипт, который запускается по нажатию пользователем этой клавиши. В выпадающем списке измените **Space** [Пробел] на **Right Arrow** [Стрелка вправо] и перетащите под него **Move 10 steps** [Переместить на 10 шагов]. Тогда при нажатии стрелки вправо кот будет перемещаться. Затем создайте похожие скрипты, с которыми кот будет сворачивать вниз при нажатии стрелки вниз и вверх при нажатии стрелки вверх. На рис. 1 показано, как это должно выглядеть. Теперь мы можем перемещаться во всех направлениях, и осталось добавить блок, отвечающий за прорисовку линий. Мы не будем рисовать постоянно, поэтому воспользуемся действиями Scratch **pen up** [поднять карандаш] и **pen down** [опустить карандаш]. Когда карандаш опущен, кот оставляет за собой след, когда поднят – не оставляет.

## Применим переменные

Чтобы переводить карандаш из одного состояния в другое, нам нужен код, который бы помнил состояние карандаша. В программах для этого используются переменные. Переменная – это участок памяти, куда можно записать и откуда можно считать данные.

# Программы

Прежде чем воспользоваться переменной, надо велеть компьютеру выделить под нее память. Мы также дадим ей имя, чтобы обращаться к ней в командах. Зайдите в Variables [Переменные], нажмите **Make a Variable** [Создать переменную] и укажите ее имя. После этого вы увидите набор команд для изменения или использования этой переменной. Теперь мы можем хранить данные, и нужно велеть компьютеру менять свое поведение в зависимости от значения переменной. Это делает блок **If...Else** [Если...То]. Он проверяет, истинно ли выражение. Если да, выполняется первый блок кода, если нет – второй. В нашей программе мы проверяем переменную **pen**. Если она равна 0, мы опускаем карандаш и устанавливаем ее в 1, а иначе поднимаем карандаш и устанавливаем ее в 0. Так можно переключаться между двумя состояниями с помощью пробела. Взгляните на рис. 3 – вот как должна выглядеть программа. Обратите внимание на оператор = в выражении **if**. Оно означает, что первый блок кода выполняется, только если переменная **pen** содержит (**equals**) 0, в противном случае (**else**) выполняется второй блок.

## Введение в циклы

Теперь можно перемещать кота и рисовать картину, но ведь здорово было бы добавлять готовые фигуры, например, кружки? Сейчас мы это сделаем. Хотя с технической точки зрения мы добавим 24-сторонний многоугольник – он очень похож на круг.

Это делается так: сначала **move forward 10** [переместиться вперед на 10 шагов], затем **rotate 15 degrees** [повернуть на 15 градусов], затем снова **forward 10** [переместиться вперед на 10 шагов], затем **rotate 15 degrees** [повернуть на 15 градусов] и так далее, пока не получится окружность. Можно поместить в код 24 пары строк, и код будет работать, но это не очень хорошо: и выглядит некрасиво, и требует много времени на создание, а если вы решите изменить размер окружности, придется это делать в 24 местах.

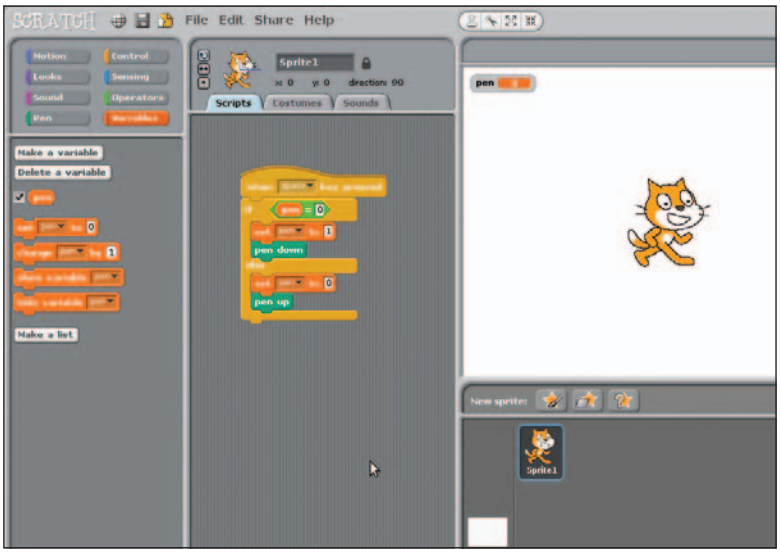
Взамен мы применим цикл. Это блок, который повторяет сам себя. Есть несколько видов циклов – некоторые повторяются, пока некое выражение не станет ложным (нечто вроде многократно повторяемой команды **if**); но мы воспользуемся тем, который повторяется заданное количество раз.

Внутри цикла нам нужны всего две команды: **move forward 10** и **rotate 15**. Подробности показаны на рис. 4. Ваша первая программа на Scratch готова! Файл проекта есть на DVD и на www.linuxformat.com/archives.

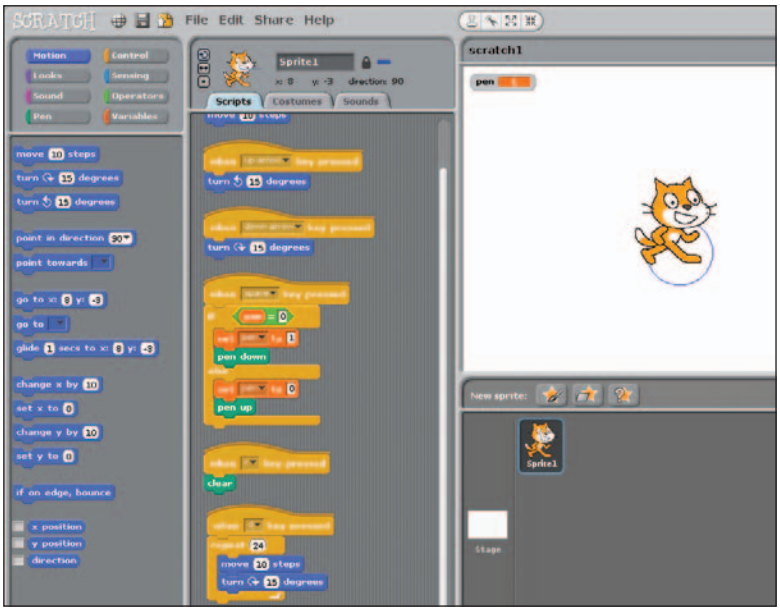
Программирование – не самоцель, а способ заставить компьютер делать то, что нам нужно, и теперь вас ограничивает только воображение. Можете написать стрелялку, рабочее приложение или нечто футуристическое, для чего и названия еще нет. Чтобы разбудить ваше воображение, вот несколько проектов из Интернета (если у вас установлен Flash, их можно запустить в браузере):

» **Super Mario Galaxy** Бегаете по свету, собирая звездочки. http://scratch.mit.edu/projects/Dolfus555/162167

» **Wipeout** На основе телешоу. Графика так себе, но играть весело. http://scratch.mit.edu/projects/awesomestickdude/1149306




› **Рис. 3. Блоки If...Then позволяют вашей программе принимать решение, и это краеугольный камень программирования.**




› **Рис. 4. Точно так же можно добавить фигуры в набор инструментов для рисования.**

» **Space War 4** Старая стрелялка космических кораблей с вертикальной прокруткой. http://scratch.mit.edu/projects/illusionist/879463

» **Snake Chamber** Познакомьтесь с генетикой и разведением змей! http://scratch.mit.edu/projects/DewleafWolf/2758178

» **Day Dream Scratch** – еще и прекрасный инструмент для создания анимации. http://scratch.mit.edu/projects/cremeglace/40150

# Python

Scratch отлично подходит для изучения основ программирования, но рано или поздно вы упретесь в его ограничения. Теперь мы рассмотрим популярный язык общего назначения Python. Первое, что о нем нужно знать – в отличие от Scratch, он полностью текстовый. Не то что на нем нельзя создавать графические программы – просто код программы представляет собой текст, а не перетаскиваемые блоки.

Поэтому, чтобы написать программу, откройте текстовый редактор. Подойдет любой; по умолчанию в Raspberry Pi есть *Leafpad*, им мы и воспользуемся; но если вы продолжите программировать, стоит поэкспериментировать с другими и найти самый подходящий для себя (у новичков в Python довольно популярен *Geany*). Обратите внимание, что текстовые процессоры, такие как *LibreOffice Write* или *Abiword*, не подойдут: они нарушат форматирование.

Создайте новый файл в *Leafpad* и добавьте в первую строку следующее:

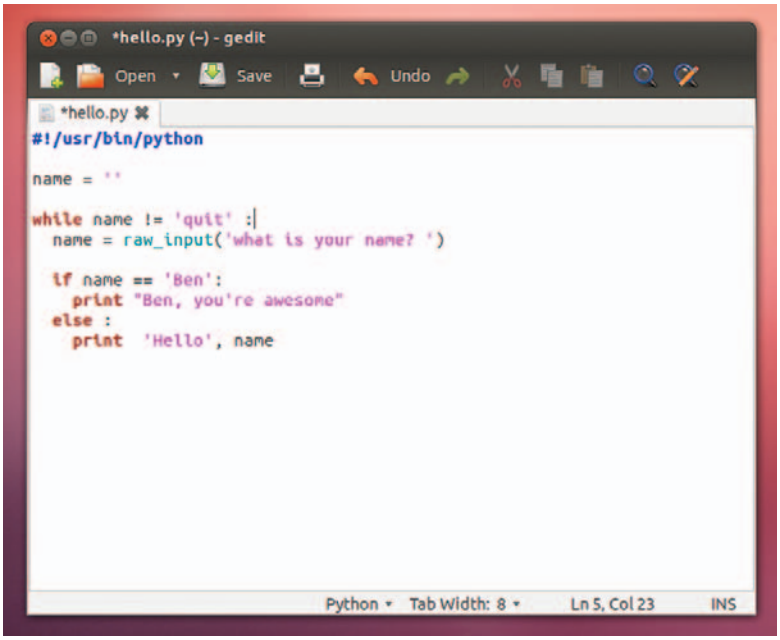
```
#!/usr/bin/python
```

Эта строка, несколько загадочно называемая «шапкой», велит системе запускать наш файл с помощью программы *python* в каталоге **/usr/bin/**. Эту строку нужно указывать в начале всех своих программ.

Теперь перейдем к внутренностям программирования. У программистов есть давняя традиция первой писать программу, выводящую на экран слова “Hello World!”, и мы не будем изменять ей. Оставьте вторую строку пустой (это не обязательно, но так код лучше читается), а в третьей напишите:

```
print “Hello World!”
```

и сохраните свой труд в файле **hello.py**. Для запуска программы откройте терминал и перейдите в каталог, где вы сохранили файл. Выполните команду **chmod a+x hello.py**, чтобы сказать системе, что файл исполняемый, и введите **./hello.py** для его запуска. На экране должно появиться **Hello World!**




**› Некоторые текстовые редакторы имеют дополнительные функции для программистов. В показанном здесь *Gedit* есть подсветка синтаксиса — она упрощает чтение кода.**

## Модули расширения

Одна из лучших особенностей Python – количество модулей расширения для него. Существуют способы добавить дополнительную функциональность, которой вы можете воспользоваться. Хотя, возможно, не все они подходят новичкам, все равно стоит посмотреть, что вы сможете сделать, когда лучше познакомитесь с языком. Вот наши пять главных модулей Python.

» **pyGames** Игры – это круто, программирование – тоже круто, а программировать игры, видимо, круче всего. Этот модуль поможет создать собственные игры для убивания времени.

» **pyGTK** Рано или поздно вы захотите уйти из командной строки и создавать графические программы. Этот модуль поможет создавать их для *GTK*.

» **pyQT** *GTK* – это прекрасно, если вы пользуетесь Gnome, но пользователям KDE больше понравится этот модуль.

» **RPi.GPIO** В Raspberry Pi есть набор входов и выходов общего назначения [General Purpose Input and Output – GPIO] для связи с внешним миром. Этот модуль поможет ими пользоваться.

» **NumPy** Для умников, которые любят математику! С ним легко манипулировать цифрами.

Программа показывает, что система работает правильно, но проку от нее немного. Как и в проекте Scratch, добавим немного пользовательского ввода. Однако с Python нужно добавить переменную для сохранения того, что ввел пользователь. Удалите строку **Hello World** (оставив только «шапку») и добавьте строку:

```
name = raw_input(‘What is your name [Как вас зовут]? ‘)
```

В этой строке создается переменная name, отображается сообщение **Как вас зовут?**, и введенные пользователем данные сохраняются в **name**. Сообщение мы заключили в обратные кавычки, чтобы компьютер знал, что это одна строка текста. Затем мы можем воспользоваться этой переменной, чтобы вывести чуть более персональное сообщение оператором **print**:

```
print ‘Hello’, name
```

Так как компьютер выполняет команды по очереди, эта команда должна быть ниже предыдущей. Если расположить их в обратном порядке, программа выдаст ошибку, так как мы пользуемся переменной до того, как она создана. Теперь сохраните файл и наберите **./hello.py** в командной строке для запуска программы.

## Решения, решения

Наша программа стала чуть более функциональной, но все равно довольно нежизненна. Она лишь выполняет те же два действия и завершается. Чтобы она стала полезна, нужно добавить действие с принятием решения, когда компьютер смотрит на то, что ввел пользователь, и в зависимости от этого выполняет разные действия.

Помните блок **if** в Scratch? То же самое есть и здесь. Базовая структура блока такова:

```
if <expression> :
  <indent> code block
```

Здесь **<expression>** нужно заменить выражением, которое может быть истинным или ложным. Например, **1 > 2**, или, что полезнее, **num > 2**, где **num** – переменная. В нашем случае мы проверим, что в качестве имени введено конкретное значение:

```
if name == ‘Бен’ :
```

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

Почему == ? Ну, компьютеры (да и программисты) не любят неопределенности. У каждого символа или слова должно быть только одно значение, иначе можно запутаться. = используется для присвоения переменной значения, поэтому для проверки на равенство нужно что-то еще. Опять же, «Бен» заключено в обратные кавычки, чтобы компьютер знал: это текст. Двоеточие означает, что выражение закончилось и сейчас мы скажем компьютеру, что делать.

Команде **if** может понадобиться выполнять больше одной строки кода – значит, нужен способ объединять код в блоки. В Python для этого используются отступы (Python в этом отношении более или менее уникален, и это кость в горле для ненавистников Python). В качестве отступов можно использовать пробелы или табуляцию, но в одном проекте важно пользоваться чем-то одним, потому что значение имеет не размер, а число отступов. Лично я использую отступ на два пробела, потому что меня учили так, и я ничего не хочу менять.

Итак, что должен сделать компьютер, если name == 'Бен'? Ну, разумеется, мы должны поприветствовать Бена соответственно:

```
if name == 'Бен' :
  print "Бен, you're awesome [ну ты крутой]"
```

Обратите внимание на два пробела в начале второй строки, а также на двойные кавычки. Двойные кавычки нужны потому, что внутри фразы есть апостроф. Другим людям мы тоже грубить не будем, поэтому добавим блок **else**, выполняемый во всех случаях, когда условие в **if** ложно:

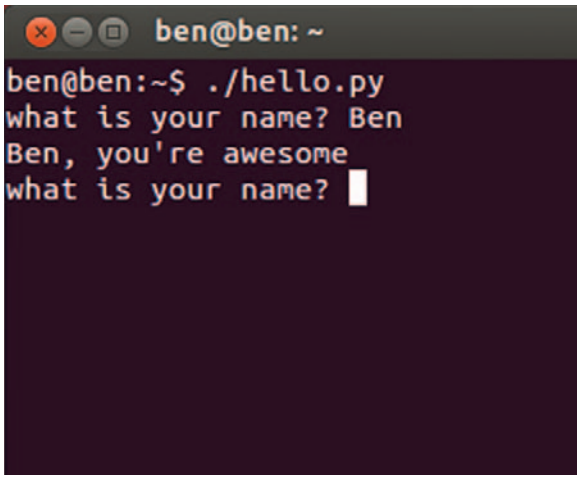
```
else :
  print 'Hello', name
```

Последняя возможность, которую мы добавим в программу – цикл. Он очень похож на цикл из программы на Scratch, только что выполняется не 24 раза, а до тех пор, пока мы не велим ему остановиться. Для этого воспользуемся циклом **while** и следующим синтаксисом:

```
while <expression> :
  <indent>code block
```

Мы можем остановить программу, введя имя **quit**. Поэтому наш цикл **while** будет таким:

```
while name != 'quit' :
```




**Увы, это единственный способ ублажить мое эго.**

## Решение проблем

Не спрашивайте меня – почему, но восклицательные знаки в программировании часто означают отрицание (**not**). Тем не менее, небольшая проблема все равно остается. Если поместить цикл перед присвоением **name = raw_input…**, появится сообщение об ошибке: дескать, неизвестно, что это за переменная **name**. А если поместить цикл после присвоения, имя будет запрошено всего однажды, а далее цикл продолжит бесконечно выплевывать приветствия.

Чтобы решить эту проблему, можно просто присвоить **name** пустую строку перед циклом **while**. Тогда сообщение об ошибке исчезнет, и цикл начнет работать. Теперь наша маленькая программа выглядит так:

```
#!/usr/bin/python
name = ''
while name != 'quit' :
  name = raw_input('Как вас зовут? ')
  If name == 'Бен' :
    print "Бен, ну ты крутой"
  else :
    print 'Hello', name
```

Обратите внимание: перед каждой строкой **print** уже четыре пробела – потому, что у каждой строки два отступа: один для цикла **while** и один для оператора **if**. Можете сохранить файл под именем **hello.py**, как и прежде, и запустить его командой **./hello.py**. LXF

## Интерактивный режим

Пока что скрипты на Python мы запускали, но это не единственный способ. С ним также можно работать в интерактивном режиме: вводим строку, Python ее обрабатывает, вводим следующую, и т. д. Это отличный способ быстро что-нибудь опробовать. Для входа в оболочку Python просто наберите **python** в командной строке. Закончив, наберите **exit()** для возврата в обычную оболочку.

**Интерактивный режим: вся мощь Python в командной строке.**

В интерактивном режиме нам было особенно удобно экспериментировать с новыми библиотеками и запускать отдельные команды, как в *Bash*.

После выхода из интерактивного интерпретатора вся ваша работа исчезнет, поэтому программы лучше писать в текстовом редакторе, как мы и делали на нашем уроке.

## Куда же теперь податься?

Если вы следовали за мной и вам понравилось писать программы, вы, наверное, задумались, что делать дальше. Что ж, и Scratch, и Python отлично подходят для начала, и прежде всего нужно выбрать язык себе по душе. Если это Scratch, лучше всего начать с http://scratch.mit.edu. Здесь вы найдете для изучения множество проектов других пользователей и видеоруководства по среде разработки.

Python гораздо популярнее в реальном мире, и вы сможете найти массу ресурсов по нему. На официальном сайте (www.python.org) есть руководство, которое хорошо объясняет язык, но слегка суховато.

Есть несколько отличных книг по этой теме (например, "Dive into Python [Ныряем в Python]", которая прилагается на DVD этого номера журнала и которую вы сможете прочитать бесплатно на www.diveintopython.net).

Подписчики бумажной версии *Linux Format* найдут все наши предыдущие руководства по Python в архивах на www.linuxformat.com и в нашей серии «Концепции программирования», где мы рассказываем о главных идеях, лежащих в основе программирования.

# Компьютерные

Сделайте первые шаги к пониманию сетей и объедините свои Linux-компьютеры – основные концепции сети вам растолкует **Джонатан Робертс**.

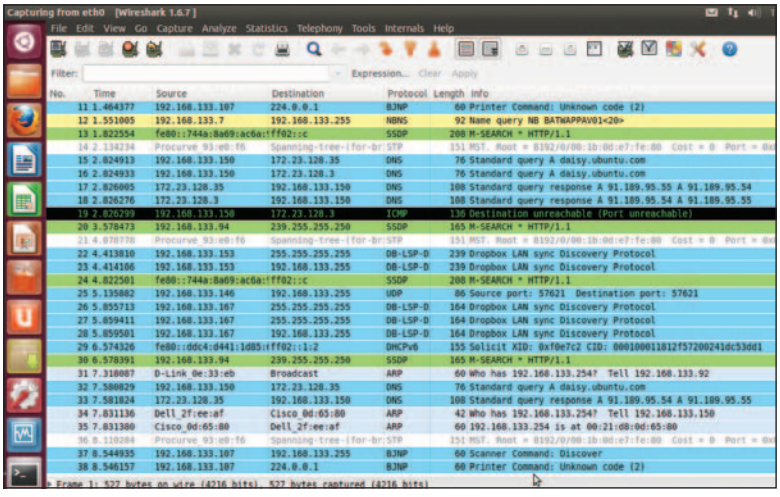
**Наш эксперт**

**Джонатан Робертс** сбежал из Башен *Linux Format*, чтобы поискать счастья в качестве сисадмина.


На страницах *Linux Format* мы извели массу времени на объяснения, как делать те или иные вещи – от создания анимаций и редактирования изображений до работы с командной строкой и шифрования данных. Причем большую часть этого времени мы не пользовались нашими компьютерами поодиночке, а соединили их друг с другом, разделив хранящиеся на них файлы и информацию еще и между нашими домами, а также через Интернет.

Чтобы помочь вам применить навыки работы в Linux в этой опутанной сетью среде, следующие два месяца мы потратим на знакомство с основами работы сети. Упор будет сделан на практическую работу: наряду с пояснением теории и терминологии мы научим вас пользоваться утилитами Linux для исследования и построения сетей.

На данном уроке мы подружим вас с тремя китами, на которых зиждется вся жизнь сетей – с пакетами, физическими соединениями и адресацией. А на следующем – создадим несколько виртуальных машин и построим из них несколько простых сетей.




**› Как и *tcpdump*, *WireShark* перехватывает сетевые пакеты, но ее приятный красочный графический интерфейс позволит нагляднее увидеть, что происходит.**

## Сети с пакетной коммутацией

Для начала поговорим о том, что же такое сеть: говоря попросту, это два или более компьютеров, связанных друг с другом через физическую среду, которой может быть медный проводник (Ethernet), оптоволоконный кабель или радиоволны (Wi-Fi). Отправляя сигналы через эту среду, компьютеры могут обмениваться информацией друг с другом.

Вроде ничего сложного: соединить два компьютера проводом, а они будут отправлять по нему электрические сигналы и делиться друг с другом информацией. Но вот беда: даже в такой простой модели (это мы еще не касаемся того, как электрические сигналы кодируются и декодируются!) быстро возникают проблемы.

Информация, которой должны обмениваться компьютеры, обычно довольно объемная и сложная, состоящая из многих миллионов бит. Ее можно отправлять по проводу последовательно – что занимает несколько минут, часов или даже дней. Когда компьютеров в сети всего два, печаль невелика: один может просто подождать, пока не получит сигнал об окончании, и затем отправит свою информацию в ответ.

Настоящие проблемы начинаются в больших сетях, с десятками или сотнями компьютеров, подключенных к общей среде. Пока один компьютер отправляет информацию, ни один из остальных не сможет воспользоваться передающей средой. Хуже того: если сигнал прервется, придется начать отправку заново – а если файл был большим, оно уж совсем обидно!

В компьютерных сетях эти две проблемы решаются с помощью пакетов. Большие объемы информации разбиваются на фрагменты меньшего размера (легче управляемые), которые называются пакетами. Наряду с передаваемой информацией пакеты содержат метаданные, которые, среди прочего, оговаривают местоположение пакета в потоке информации.

При потере пакета принимающий компьютер увидит, что части потока не хватает, и ему останется запросить повторную отправку только отсутствующего пакета, а не всего потока. Более того, так как пакеты разбивают данные на независимые фрагменты, все компьютеры, которые пользуются средой, могут передавать пакеты (а не целые файлы) по очереди, то есть смогут работать со средой более эффективно. Эти пакеты – не просто абстрактная идея: вы можете посмотреть, как отдельные пакеты передаются по сети. Для этого установите программу *tcpdump* с помощью менеджера пакетов своего дистрибутива Linux. После установки откройте терминал и найдите, какое устройство сейчас пре-

# сети: Основы

## Классическая модель сети

В этой статье мы заглядываем глубоко внутрь сети, от физической среды и пакетов до MAC-адресов и маршрутизируемых IP-сетей. Трудно даже понять, как все это сочетается друг с другом, а представьте себе, как трудно создавать системы, в которых все это работает! Это большая сложность, и для правильной работы сетевого подключения множество компонентов должны работать вместе.

Чтобы упростить создание сетевых программ и устройств, работающих друг с другом, несколько умных людей собрались вместе и придумали модель OSI (Open Systems Interconnection – взаимодействие открытых систем). В OSI определяется семь уровней, каждый из которых задает набор функций, имеющих первоочередное значение для правильной работы сети.

Каждый уровень предоставляет определенные сервисы, опираясь на нижележащий уровень, но не обязан знать, как реализованы прочие уровни над ним – он должен только уметь взаимодействовать с более низкими уровнями. Это означает, что уровни можно создавать и проверять отдельно и что можно иметь множество различных реализаций различных функций (например, применять оптоволокно или Ethernet в качестве физического уровня) без необходимости замены всех остальных компонентов.

На нашем уроке мы раскроем три первых уровня модели OSI:

» Физический уровень, определяющий, как электрические сигналы передаются по определенной физической среде.

» Канальный уровень, определяющий, как хосты физически идентифицируются в сети, т.е. с помощью MAC-адресов.

» Сетевой уровень, определяющий, как пакеты маршрутизируются между сетями и как назначаются логические адреса.

Четыре оставшихся уровня отвечают за поддержку надежных соединений между компьютерами, а также представление и использование переданных данных на уровне приложения. Сюда относятся аббревиатуры, которые вам наверняка знакомы: это протоколы TCP и UDP на уровне 4, методы кодирования файлов UTF, ASCII, JPG и т.д. на уровне 6, протоколы HTTP (работа с вебсайтами), FTP (передача файлов) и SMTP (передача почты) на уровне сетевых приложений 7.

доставляет сетевое соединение. Если активное соединение у вас всего одно, это можно сделать командой

```
ip a | grep “state UP”
```

Команда **ip** выводит информацию о сетевых подключениях – попозже мы познакомимся с ней ближе – а команда **grep** фильтрует результат, чтобы мы видели только подключения в состоянии 'up', то есть активные. Нас интересует фрагмент в начале строки – скорее всего это **eth0**, **wlan0** или **em1**, в зависимости от конфигурации. Его можно указать как параметр команде **tcpdump** и понаблюдать за сетевым общением. От имени суперпользователя-root выполните команду:

```
tcpdump -i wlan0
```

где **wlan0** – устройство, которое мы обнаружили выше. Затем откройте браузер и зайдите на какой-нибудь сайт. Вернувшись в терминал, вы увидите, как экран заполняется строками текста. Каждая строка представляет пакет, отправленный между вашим компьютером и сервером для загрузки web-страницы, и в каждом пакете есть информация о нем, включая время приема или отправки, компьютеры (или хосты), с которых и на которые он был отправлен, и многое другое.

Вывод команды и содержимое пакетов (в терминале отображаются только заголовки пакетов) можно сохранить в файле, приписав к команде параметр **-w** и имя файла, а потом проанализировать в графической утилите наподобие *Wireshark*.

Таким образом очень удобно диагностировать проблемы в сети. Например, если вам не удается заставить правильно работать DHCP, запустите отслеживание пакета на сервере DHCP, и узнаете, получает ли он запросы на адреса, отвечает ли он на запросы, блокируются ли пакеты брандмауэром – это поможет найти источник проблемы. »


› ***Nmap* — утилита командной строки для сканирования сети. Здесь мы интерпретируем результаты сканирования *nmap* и отображаем сетевую топологию с помощью *Zenmap*.**

## MAC-адреса

Зная, сколько компьютеров могут находиться в одной физической сети, вы можете поинтересоваться, как они идентифицируют друг друга в сети, т.е. как отправить информацию только на компьютер Боба, а не на компьютер Алисы?

На самом деле, компьютеры, подключенные к одной и той же физической среде (например, к медному проводу), видят все пакеты, отправляемые на все остальные компьютеры. От чтения всех пакетов подряд компьютер останавливает то, что его сетевая карта (Network Interface Card, NIC – это аппаратный компонент компьютера, который преобразует информацию от компьютера в сигнал, пригодный для передачи по физической среде, и наоборот) снабжена так называемым MAC-адресом [от Media Access Control – управление доступом к среде].

> «Логические адреса связываются с физическими MAC-адресами.»

Каждый пакет включает MAC-адрес получателя. Компьютер, получив пакет, проверяет, совпадает ли адрес его сетевой карты с адресом в пакете; если да, то из пакета извлекается содержащаяся в нем информация, если нет – пакет отбрасывается для экономии процессорного времени.

Адрес активной сетевой карты можно просмотреть опять же командой IP:

```
ip link show dev wlan0
```

где **wlan0** – имя устройства, которое мы выяснили ранее. Если набрать просто **ip link show**, будет показана информация обо всех сетевых картах, подключенных к компьютеру. MAC-адрес состоит из 12 знаков и следует за **link/ether** в выводе команды. Он выглядит примерно так: **ea:34:43:81:02:7c**.

Конечно, одним проводом сложно соединить больше двух компьютеров. Чтобы подключить несколько компьютеров к одной физической сети, используется устройство под названием «хаб [hub – *англ.* стыковочный узел]». У хаба есть несколько разъемов Ethernet, и по поступлении пакета на один из входов тот пересылается на все остальные интерфейсы.


**› Образчик сетевой карты — устройства, служащего посредником между компьютером и физической средой.**

Но даже в сетях с пакетной коммутацией могут возникнуть проблемы, если компьютеров слишком много – разные компьютеры будут пытаться передавать информацию в одно и то же время, в результате она будет повреждена, и эффективного обмена данными не состоится.

Это происходит потому, что компьютеры отправляют пакеты не по очереди, а когда попало, и используют сложные алгоритмы для выхода из коллизий.

Чтобы обойти данную проблему, применяют другое устройство – коммутатор, он же свитч [switch]. Коммутатор похож на хаб наличием нескольких интерфейсов Ethernet, но действует значительно умнее.

Вместо того, чтобы вслепую перенаправлять все получаемые пакеты на все остальные интерфейсы, коммутатор составляет таблицу MAC-адресов интерфейсов. Когда на коммутатор приходит пакет, он смотрит на MAC-адрес пакета и определяет, на какой именно порт пакет следует отправлять.

Таким образом уменьшается объем трафика, пересылаемого между интерфейсами, и снижается их конкуренция – то есть уменьшается количество коллизий, и обмен данными становится более надежным.

## Логические сети

Однако использование коммутаторов для расширения сетей и уменьшения коллизий – не конец истории, так как у MAC-адресов однородная [flat] структура. Это значит, что их трудно классифицировать или группировать.

При небольшой сети это не проблема, но с ростом сети таблицы MAC-адресов на коммутаторах, в которых производится поиск порта для отправки пакета, достигают огромного размера. Это замедляет работу коммутаторов и делает невозможным построение глобальной сети, такой как Интернет.

Чтобы решить эту проблему, можно разбить большие сети на множество мелких логически сгруппированных сетей и направлять трафик между ними с помощью технологий межсетевого взаимодействия.

Как это работает? Ну, для начала надо познакомить вас с адресами нового типа, которые называются адресами Протокола Интернета (Internet Protocol – IP). Возможно, вы уже видели IP-адреса – они обычно состоят из четырех чисел, разделенных точками, например, **192.168.1.218**.

В отличие от представления хоста в сети полным адресом, этот адрес разбит на две части – адрес сети и адрес хоста – с помощью так называемого префикса. Префикс – это число от 0 до 32, определяющее, какая часть IP-адреса является адресом сети, а какая – адресом хоста. Полный адрес записывается в следующем виде: **192.168.1.2/24**.

## Соединяем оба вместе

Эти логические адреса связываются с физическими MAC-адресами, о которых мы говорили ранее, при этом каждый IP-адрес связан с конкретным MAC-адресом. Для просмотра этой информации на своем компьютере обратитесь к **ip neighbour**:

```
ip neigh show
```

Каждая строка вывода соответствует одному хосту, достижимому с вашего компьютера. Первый параметр в строке – IP-адрес хоста; следующие два определяют, через какую сетевую карту он доступен; а **lladdr** содержит MAC-адрес устройства. С помощью

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

этой информации компьютер может формировать пакеты, адресованные тому или иному компьютеру.

А теперь кое-что интересное: компьютеры с одинаковыми адресами сети хранят эту информацию друг о друге и могут взаимодействовать напрямую. Но если адреса сети у них разные, они не хранят эту связь IP- и MAC-адресов и напрямую взаимодействовать не могут. Вместо этого они взаимодействуют через промежуточное устройство, известное как шлюз или маршрутизатор (он же роутер).

Роутеры отслеживают и сохраняют все связки IP- и MAC-адресов для своей сети, а также информацию о подключении к другим роутерам и сетям. Если хост в сети роутера хочет подключиться к хосту в другой сети, он отправляет сообщение на шлюз с полным IP-адресом другого хоста, и шлюз пересылает это сообщение в соответствующую сеть. Затем шлюз в этой сети отправляет пакет соответствующему хосту.

Если в этих других сетях сотни, тысячи или даже десятки тысяч компьютеров, вы поймете, как это снижает объем информации, которую должен хранить каждый роутер. Вместо MAC-адресов всех этих хостов он должен обеспечить хранение только одного адреса для каждой сети – адреса шлюза.

**› Коммутаторы-свитчи лежат в основе многих сетей, уменьшая количество коллизий в домене и позволяя многим хостам в одной сети надежно обмениваться данными.**

## Исследование информации о маршрутизации

Знание пути отправки пакетов – например, с Компьютера A на Роутер A и затем с Роутера A на Роутер B и, наконец, на Компьютер B – называется информацией о маршрутизации, так как это информация о пути, который должен пройти пакет.

Собственную таблицу маршрутизации своего компьютера можно просмотреть, указав аргумент **route** в команде **ip**:

```
ip route show
```

В первой строке вывода стоит “default”, затем идет IP-адрес, через который должны отправляться пакеты, и интерфейс, с которого они должны отправляться. Маршрут по умолчанию – оговорка для вашего компьютера: если для IP-адреса, с которым вы пытаетесь связаться, не задано другого маршрута, пакеты уйдут в этот шлюз.

Сравнив точку назначения маршрута по умолчанию с выводом команды **ip neighbour**, вы увидите, что MAC-адрес компьютера соответствует его шлюзу по умолчанию; вот так два компьютера могут общаться напрямую.

> «Знание пути отправки пакетов = информация о маршрутизации.»

## Разрешаем адреса

Таблицу соответствия MAC-адресов и IP-адресов, полученных от команды **ip neigh**, ваш компьютер строит с помощью протокола разрешения адресов (Address Resolution Protocol – ARP). В ARP используется специальный пакет, точно такой же, как показывала команда **tcpdump**. Желая выяснить MAC-адрес компьютера с заданным IP-адресом, ваш компьютер формирует пакет запроса ARP.

Тот содержит ваш MAC-адрес как отправителя и IP-адрес хоста, MAC-адрес которого вы хотите узнать, но не указывает MAC-адрес назначения. Затем этот пакет отправляется всем хостам, подключенным к той же физической сети, что и вы – т. е. на все порты коммутатора, но не с одного роутера на другой – и содержит, например, следующий запрос: «**Запрос компьютера с IP-адресом 192.168.1.2, передать ответ на 192.168.1.1 [Request who has 192.168.1.2 tell 192.168.1.1]**».

Если такой IP-адрес существует в этой сети, на MAC-адрес отправителя посылается arp-пакет REPLY с таким содержимым: «**Ответ 192.168.1.2 - ea:34:43:81:02:7c [Reply 192.168.1.2 is-at ea:34:43:81:02:7c]**». Теперь компьютеры могут общаться друг с другом напрямую. Это можно увидеть в действии, снова запустив команду **tcpdump**. Но вместо открытия web-страницы, зайдите в другой терминал и скомандуйте от имени root:

```
arping 192.168.1.1
```

с IP-адресом шлюза по умолчанию. Взглянув на вывод **tcpdump**, вы теперь должны увидеть запрос и ответ ARP. LXF

## Подробнее про IP-адреса

IP-адреса чуть сложнее, чем будет рассказано далее в основной статье. Хотя обычно IP-адрес записывается в виде четырех чисел, разделенных точками, на самом деле это одно 32-битное двоичное число. Каждое из четырех чисел представляет группу из восьми бит (октет) двоичного адреса, например:

**192.168.1.1**

**11000000.10101000.00000001.00000001**

Это означает, что каждое из четырех чисел IP-адреса может быть от 0 до 255.

Префикс, о котором будет говориться, определяет, сколько разрядов адреса (спереди) принадлежат к сетевой части адреса; его можно рассматривать как накладываемое на IP-адрес еще одно двоичное число, в котором в 1 установлены биты, соответствующие сетевой части адреса:

**11000000.10101000.00000001.00000001**

**11111111.11111111.11111111.00000000**

В данном случае, в IP-адресе **192.168.1.1** первые три октета (24 бита) представляют сетевую часть адреса, и хостам можно присваивать уникальные номера только из последнего октета. Поэтому в выводе команды **ip route** мы видим **.0/24**. Следовательно, в сети 192.168.1.0/24 есть всего 255 уникальных адресов, которые можно назначить хостам. По факту, их даже 253, так как первый и последний адреса резервируются для особых целей. Первый адрес – это адрес сети, и он обозначает не хост в сети, а саму сеть. Последний адрес используется для отправки широковещательных сообщений, адресованных всем хостам в сети.

Другой пример: в сети **192.168.1.0/28** всего 14 доступных адресов, так как максимальное количество адресов, представимых четырьмя двоичными цифрами – 16 (2^4), а первый и последний адреса зарезервированы.

# pdmenu: Меню

Нейл Ботвик рассказывает о простой системе, с помощью которой можно создавать простые меню, чтобы не запоминать сложные команды.

**Наш эксперт**

У **Нейла Ботвика** по компьютеру в каждой комнате, но по соображениям безопасности он ни за что не скажет вам, где находится центральный сервер.

Терминал – одно из главных достоинств Linux, но неопытным пользователям он может казаться пугающим, незнакомым и слегка недружелюбным. Как-то я работал в своем любимом терминале *Konsole*, и один из моих домашних спросил: «А откуда ты знаешь, какие кнопки нажимать?» И это ожидаемая реакция от людей, привыкших работать с кнопками и меню с всплывающими подсказками. Пускай графический интерфейс оставляет нам меньше выбора, но зато ясны варианты, из которых выбирать. Что же делать, когда новичку нужно выполнить несколько команд в терминале – потому, что эти действия нельзя сделать через графический интерфейс, или при работе с компьютером через SSH (а то и при подключаении с компьютера с Windows с помощью *PuTTY*)?

Один из вариантов – написать скрипт с помощью графических программ для создания скриптов, вроде *Zenity* или *Kommander*, но для этого нужна подходящая рабочая среда или, по меньшей мере, должны быть установлены нужные утилиты. Также есть программы типа *dialog*, с помощью которых в терминале можно создавать диалоговые окна *ncurses*; они тоже эффективно работают по SSH, даже из другой ОС.

И еще один вариант – дать пользователю способ запустить несколько команд, не погружая его во все возможности оболочки. Все эти проблемы можно решить с *pdmenu* (http://joeyh.name/code/pdmenu), причем вам даже не придется ничего программировать.

## Простой пример

У *pdmenu* один файл настройки. По умолчанию, если он не задан, программа будет искать его в **~/.pdmenurc**, а если там его нет – в **/etc/pdmenurc**. Чтобы задать другой файл, укажите его при запуске *pdmenu* как аргумент:

```
pdmenu /path/to/config.file
```

Это простой текстовый файл, следующий стандартным соглашениям. Комментарии помечаются #; для лучшей читаемости можно применять отступы, но программа их игнорирует. Вот простой пример, который позволяет монтировать, выводить список содержимого, проигрывать и выкатывать DVD-диск:

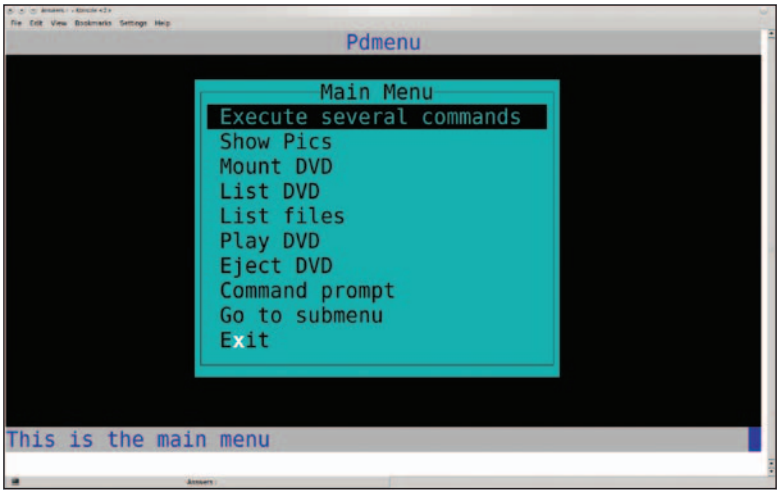
```
color:desktop:blue:blue
color:title:blue:white
color:base:blue:white
menu:main:Main Menu:This is the main menu
exec:Mount DVD::pmount /dev/dvd
exec:List DVD:display:lsdvd
exec:Play DVD::vlc dvd://
exec:Eject DVD::eject /dev/dvd
exit:Exit
```

Первые три строки – чистая косметика, в них задается цветовая схема; затем мы определяем одно меню. Строка, определяющая меню, состоит из трех элементов: названия меню, заголовка над меню и (необязательно) подсказки, которая отображается в нижней части окна. Затем определяются пункты меню. Отступы игнорируются, но упрощают читаемость пунктов меню в тексте. Команда **exec** запускает программу; у нее есть три параметра, опять же разделенных двоеточиями. Первый – название пункта меню, второй – набор флагов, влияющих на запуск команды, а третий – сама команда. Если флагов не указано, *pdmenu* запускает программу и возвращается в меню. Вывод команды на терминал не отображается, и вы увидите его только после выхода из *pdmenu*. Если нужно видеть этот вывод, как во втором пункте меню, где мы получаем список файлов DVD командой **lsdvd**, добавьте флаг **display**. Также можно добавить горячие клавиши для пунктов меню, поместив символ подчеркивания перед буквой, которая намечена в качестве горячей клавиши, например:

```
exec:_Play DVD::vlc dvd://
```

При нажатии горячей клавиши пункт меню не запускается, а только выделяется. Если одна горячая клавиша соответствует нескольким пунктам меню, выделение будет переходить с одного на другой. Выделение цветом работать не будет. Это происходит потому, что по умолчанию *pdmenu* отображает меню в стандартных цветах терминала – чтобы увидеть все цвета, нужно добавить параметр **-c** или **--color**. Другой способ это сделать – задать переменную окружения **COLORTERM**; при этом значение не играет роли, если переменная задана.

```
pdmenu -c pdmenurc
```




**› Создавайте простые меню оболочки с *pdmenu*.**

## Добавляем уровни

Одним меню, выполняющим отдельные команды, возможности *pdmenu* не ограничиваются – также поддерживается несколько меню или подменю. В первом примере у нас было одно выражение для меню с именем **main**. Имя на самом деле может быть любым – при запуске *pdmenu* загружает первое меню в файле (это можно подавить, указав параметр при запуске, но пока мы не будем этого касаться). Попробуйте добавить второе меню следующим образом:

```
menu:newsubmenu:Some more stuff:This is the first submenu
exec:Some Command::/usr/bin/somecommand
nop:This does nothing
exit:Main Menu
```

# оболочки

Так мы определяем новое меню – выражение **nop** ничего не делает. С его помощью можно добавить строку с описанием или пустую строку, если после выражения нет текста. В первом меню мы воспользовались **exit** для выхода из *pdmenu*. В подменю **exit** ведет себя иначе – она выходит из данного подменю, возвращаясь в то меню, которое вызвало подменю. Добавим в меню строку для перехода в подменю:

```
show:Go to submenu::newsubmenu
```

Ее набор параметров похож на параметры **exec**: текстовая надпись, несколько флагов и имя подменю, в которое нужно перейти. У команды **show** нет флажков, поэтому оставьте это поле пустым. Небольшое предупреждение: если два меню вызывают друг друга, *pdmenu* промолчит, но пользователь окажется в петле: **exit** будет постоянно возвращать его в другое меню вместо главного меню. Избегайте таких рекурсивных схем.

Мы уже рассматривали флаг **display** с командой **exec** – он показывает в окне вывод команды. Без него вывод пишется в окно терминала, но сразу же затаптывается при перерисовке меню. Другой способ сохранить вывод команды – воспользоваться флагом **pause**, который переключает пользователя в окно терминала/консоли, чтобы показать вывод команды, и после нажатия клавиши возвращает его обратно в меню. Еще один полезный флаг – **edit**, он позволяет пользователю под вашим контролем изменять аргументы команды. Проще всего пояснить это на примере:

```
exec:Play another DVD:edit:/usr/bin/vlc ~Which drive?:/dev/sr0~
```

Флаг **edit** велит *pdmenu* изменить команду перед запуском; изменяемая часть задается в формате **~title:default~**, где **title** – заголовок отображаемого окна, а **default** – значение по умолчанию (да-да!), которое пользователь может изменить до нажатия Enter. Значение по умолчанию можно оставить пустым. Флагов может быть несколько – например, **edit,display**.

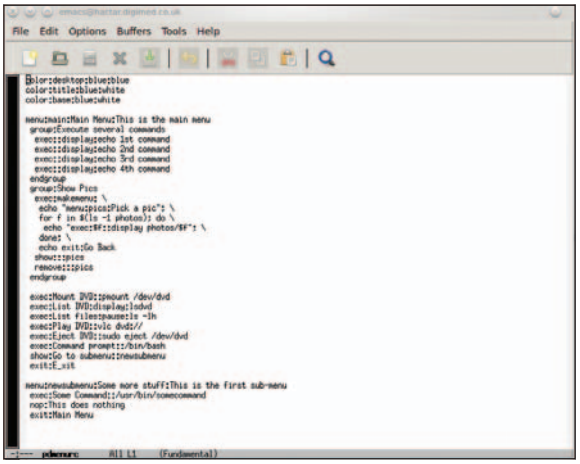
## Команды посложнее

*pdmenu* также может выполнять несколько команд в одном пункте меню, например, так:

```
group:Execute several commands
exec:::command1
exec:::command2
exec:::command3
endgroup
```

Подлинное хитроумие – использовать **exec** с параметром **makemenu**, который строит меню на лету. Он выполняет заданную команду почти так же, как обычно делает **exec**, но использует результат ее выполнения как определение меню. Например, построить меню для показа фотографий из каталога можно таким кодом:

```
group:Show Pics
exec:makemenu: \
echo “menu:pics:Pick a pic”; \
for f in $(ls -1 photos); do \
echo “exec:$f::display photos/$f”; \
done; \
echo exit:Go Back
```




› **Файл *pdmenurc* понятен даже с циклами для динамических меню.**

```
show:::pics
remove:::pics
endgroup
```

Выполняется все, что следует за **makemenu**; обычно это набор команд **echo**, пишущих меню. Обратные слэши нужны потому, что *pdmenu* считает все команды однострочными, а такие плохо читаются. Все прочее – стандартный скрипт оболочки. Он записывает выражение **menu**, затем проходит по списку файлов, выполняя команды **exec**, и заканчивает командой **exit**. Все это объединяется в группу, которая строит меню флагом **exec:makemenu**, отображает его флагом **show** и удаляет определение **menu**. Команды выполняются последовательно, поэтому удаление произойдет не раньше, чем пользователь выйдет из меню с фотографиями.

Вы можете спросить: «А зачем нужна система меню, цель которой – обойтись без ввода команд, но для ее запуска приходится команды-то и вводить?» Пока мы запускали *pdmenu* из оболочки, но ее можно запустить вместо оболочки. Для этого нужно изменить оболочку пользователя по умолчанию; поэтому либо создайте нового пользователя для проверки, либо добавьте в одно из меню возможность запуска стандартной оболочки – нам нужен запасной выход, если что-то пойдет не так. Этот пункт меню переведет вас в оболочку, даст пользователю выполнить все необходимые команды и вернуться в меню после завершения работы оболочки клавишами Ctrl+D или путем выхода из системы.

Если запустить **vipw** от имени root для изменения файла паролей, последний параметр в строке с каждым пользователем – это оболочка, запускаемая при входе пользователя в систему, обычно **/bin/bash**. Смените ее на **/usr/bin/pdmenu**. Также нужно добавить **/usr/bin/pdmenu** в список доступных оболочек в **/etc/shells**, если менеджер пакетов этого не сделал. Теперь при входе в систему пользователь сразу попадет в меню, а при выходе из главного меню он выйдет из системы. Это работает как для входа в систему на локальном компьютере, так и через SSH. Не обязательно даже пользоваться Linux: *PuTTY* в Windows – приличный SSH-клиент, который прекрасно ладит с *pdmenu*. LXF

# Shellinabox:

**Крис Нотли** покажет, как употребить *Shellinabox* с обратным прокси *Apache* для обеспечения защищенного доступа к Linux-компьютеру только через браузер.

**Наш эксперт**

**Крис Нотли** пользуется Linux и работает с ним более 12 лет, но реально подсел на него, открыв прелести *Asterisk* и *MythTV*.

Удаленный доступ к оболочке домашнего сервера стал для меня нормой жизни, и без него мне было бы трудно обойтись. Нужно ли проверить возможность подключения к удаленному сайту, изменить скрипт мониторинга или просто перезапустить сервис – мне больше незачем мчаться для этого домой.


**› Самый ПРОСТОЙ пример — *Shellinabox* может гораздо больше, чем просто показать оболочку!**

Полноценные VPN-решения с использованием IPSEC или PPTP дают гибкий доступ ко всей сети, но для их безопасной настройки придется потрудиться, а в корпоративных сетях такой трафик обычно запрещен брандмауэрами.

На другом конце шкалы – SSH-подключение напрямую к серверу через роутер или брандмауэр является и защищенным, и относительно простым, но SSH-доступ во многих сетях бывает невозможен. Самая частая причина проблем – стандартный порт SSH (TCP/22) запрещен брандмауэром (так нередко бывает в корпоративных сетях и в точках публичного доступа к Интернету). Иногда проблему можно решить, перенастроив SSH-сервер так, чтобы он слушал порт, связанный с web-сервисами (например, TCP/80 или TCP/443). Однако ничего не получится, если на том же хосте должен работать web-сервер. Это решение также встречает более серьезные препятствия в больших корпоративных сетях, где доступ к web-портам обычно осуществляется через прокси.

Еще одно потенциальное ограничение может возникнуть, если вы пользуетесь чужим компьютером для доступа. На заблокированных терминалах – например, на ограниченных в правах корпоративных компьютерах или киосках – может не быть SSH-клиента, и установить его нельзя. Программы для доступа к оболочке или к терминалу через web-браузер существуют довольно долго, но обычно используют Java со всеми проблемами совместимости, которые мы научились любить! С ростом возможностей современных браузеров (в частности, с развитием движков JavaScript в этих браузерах) стало возможным найти решение, не использующее плагины вроде Java.

## Shellinabox

*Shellinabox* – web-эмулятор терминала на Ajax, совместимый с большинством современных web-браузеров, поддерживающих CSS и JavaScript. Серверная часть программы представляет собой выделенный web-сервер, который при подключении клиента по умолчанию запускает **/bin/login**, выводя приглашение для ввода логина/пароля, и после успешной аутентификации запускает оболочку клиента по умолчанию.

*Shellinabox* запускается как демон и по умолчанию слушает защищенные соединения на порту TCP/4200. Порт можно заменить на TCP/443, то есть порт, зарезервированный для защищенных HTTP-соединений, но это приводит к двум большим проблемам. Первая – в том, что во многих случаях вы можете захотеть запустить другие web-сервисы через защищенные соединения с того же компьютера. Вторая – встроенный web-сервер предоставляет мало возможностей для тонкого контроля доступа, поэтому каждый, кто подключится к серверу через web-браузер, увидит приглашение для входа в систему. Внешний вид *Shellinabox* легко изменить, и для этого не понадобится ничего, кроме CSS.

В отличие от некоторых альтернатив, *Shellinabox* предоставляет полноценный эмулятор терминала на JavaScript, а значит, в нем можно выполнять самые разнообразные действия, обычно выполняемые в терминале. Прекрасный пример этого – домашняя страница *Shellinabox* (http://code.google.com/p/shellinabox/).

# Защитим доступ

## Web-прокси

Web-прокси – это сервис, который сидит между web-клиентом и сервером и существует в двух вариантах. Первый – прямой прокси, и его клиенты обычно должны знать о нем (как знают большинство браузеров). При возникновении запроса (например, при вводе адреса в браузер, в котором настроен прокси), клиент отправляет запрос прокси, который, в свою очередь, запрашивает содержимое непосредственно у запрашиваемого web-сервера. Затем прокси возвращает это содержимое клиенту. Прямой прокси также можно настроить в прозрачном режиме, тогда клиентам не нужно знать о прокси. В прямых прокси обычно есть некая форма для управления содержимым, на которой можно заблокировать доступ к нежелательным сайтом. Они также обычно кэшируют статическое содержимое, например, изображения, поэтому у многих провайдеров настроены прозрачные прямые прокси для экономии канала.

Второй вариант – обратный прокси, и на нашем уроке мы им и воспользуемся. Основное различие между обратным и прямым прокси в том, что обратный прикрывает определенные сайты или содержимое на стороне сервера, а прямой прокси обычно возвращает любое запрошенное содержимое. Web-клиентам не нужно знать, что сайт обслуживается обратным прокси, адрес будет указывать прямо на прокси, а клиент просто увидит ответ на свой запрос. С помощью обратного прокси можно применить к набору серверов постоянную конфигурацию безопасности вместо того, чтобы выполнять одни и те же настройки на каждом из них.

## Apache

Существует немало обратных прокси для Linux, но мы возьмем *Apache*. В модуле прокси *Apache* можно настроить пути (например, http://webserver/path1), ссылающиеся на другой сервер, а не на HTML-файлы на локальном диске. Это не обязательно физически отдельный сервер, это может быть и другой демон web-сервера на том же компьютере. Преимущество *Apache* здесь в том, что в нем легко управлять доступом к сервисам. Например, можно включить какую-либо форму аутентификации и/или явно разрешить или запретить подключение с определенных IP-адресов и т.д.

Некоторые web-приложения нужно немного доработать, прежде чем помещать их за обратный прокси. В частности, изменить абсолютные ссылки на относительные (например, **<link rel="stylesheet" type="text/css" href= "/css/default.css">** вместо **<link rel="stylesheet" type="text/css" href= "../css/default.css">** – заметьте две точки во втором примере). Чтобы обойти проблему, может потребоваться способность *Apache* переписывать содержимое. К счастью для нас, *Shellinabox* легко интегрируется с обычным прокси, и менять ничего не придется.

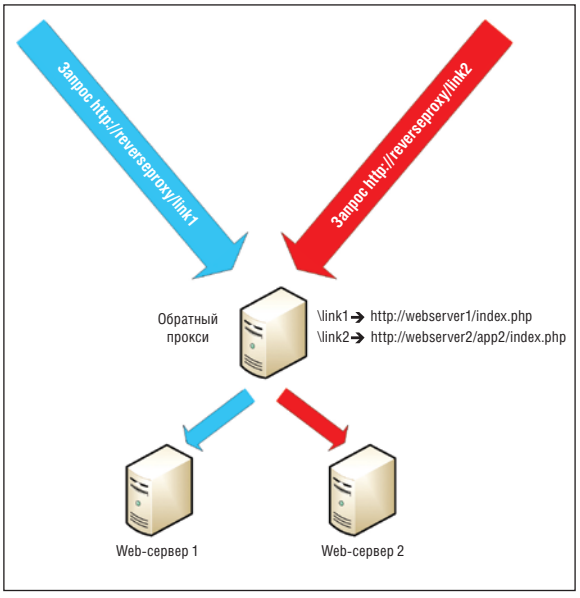
Еще одно большое преимущество *Apache* как обратного прокси в том, что можно продолжать обрабатывать другие web-запросы с его помощью. В моем случае *Apache* обслуживает *Shellinabox* вместе с *MythWeb*, *phpMyAdmin* и еще несколькими домодельными web-приложениями.

Соединив возможности *Shellinabox* с обратным прокси *Apache*, можно получить еще один метод удаленного доступа, совместимый почти со всеми операционными системами и конфигурациями брандмауэра, а также довольно эффектный при демонстрации другим пользователям! И *Shellinabox*, и *Apache* есть в стандартных репозиториях многих современных дистрибутивов Linux, и заставить обоих работать не так уж сложно. На нашем уроке мы установим *Shellinabox* и настроим его для работы с *Apache* на чистой установке Ubuntu 12.10 Server. Мы также рассмотрим два предыдущих релиза Ubuntu LTS (12.04 и 10.04). В других дистрибутивов местоположение файла настройки может быть другим. Однако большая часть этого руководства все равно подойдет.

## Погружаемся

Первый этап – установка чистой системы Ubuntu Server (12.10). Если вы хотите сделать это в *VirtualBox*, можно скачать готовый образ у отличных ребят из VirtualBoxImages (http://bit.ly/T0CDjH). При установке Ubuntu Server с нуля, единственное рекомендуемое изменение в конфигурации – включить сервер OpenSSH на этапе выбора программ [Software Selection] во время установки.

В случае готового образа с VirtualBoxImages надо учесть несколько моментов. Виртуальная машина попытается включить интерфейс WLAN1, поэтому обязательно задайте правильный »




**Типичный обратный прокси в действии.**

**Скорая помощь**

Когда у вас будет рабочий обратный прокси, примените его для безопасного удаленного web-доступа к такому оборудованию, как цифровые ресиверы, серверы NAS и т.д.

интерфейс в настройках. По умолчанию настроено получение IP-адреса по DHCP – можно либо изменить его на статический адрес, либо записать назначенный адрес (и далее до конца урока заменять SERVER_IP на этот адрес).

В образе *VirtualBox*, на который мы ссылались выше, по умолчанию установлена итальянская раскладка клавиатуры; если вам нужна другая, скомандуйте

```
sudo dpkg-reconfigure keyboard-configuration
```

Вам предложат изменить несколько параметров, включая страну – задайте их соответствующим образом. Если вы новичок в работе с сервисами, открытыми для доступа извне, то невероятно полезной станет привычка регулярно обновлять систему. Сервисы вроде *Apache* обновляются регулярно, и это позволит вам минимизировать уязвимость компьютера для внешних воздействий.

Сервер можно обновить с командной строки, запустив следующие команды:

```
sudo apt-get update
sudo apt-get -y dist-upgrade
sudo shutdown -r now
```

После перезагрузки сервера установите *Apache*, командой

```
sudo apt-get install -y apache2
```

Затем нужно установить *Shellinabox*, и способ установки зависит от вашей версии Ubuntu. В репозиториях Ubuntu 12.10 уже есть *Shellinabox*, и ее можно установить командой

```
sudo apt-get install -y shellinabox
```

В репозиториях Ubuntu 12.04 и ранее ее нет, и ее нужно загрузить и установить. Версия файла зависит от того, 32- или 64-разрядной версией Ubuntu вы пользуетесь:

**32-битная:**

```
wget http://shellinabox.googlecode.com/files/shellinabox_2.10-1_i386.deb
sudo dpkg -i shellinabox_2.10-1_i386.deb
```

**64-битная:**

```
wget http://shellinabox.googlecode.com/files/shellinabox_2.10-1_amd64.deb
sudo dpkg -i shellinabox_2.10-1_amd64.deb
```

Теперь можно проверить базовую функциональность *Shellinabox*, открыв браузер и набрав https://SERVER_IP:4200. Браузер пожалуется на сертификат SSL, который является самоподписанным. Нажав «Продолжить», вы увидите интерфейс *Shellinabox* по умолчанию со строкой входа в систему. Осмотрите интерфейс. Попробуйте все команды и меню, которыми обычно пользуетесь, перед тем, как мы настроим *Shellinabox* для работы с *Apache*.


**› При работе в локальной сети предупреждения можно спокойно игнорировать, но при удаленных подключениях проверяйте цифровую подпись сертификата.**

Чтобы пользоваться *Shellinabox* через *Apache*, в настройки по умолчанию нужно внести несколько изменений. Откройте файл **/etc/default/shellinabox** в любимом текстовом редакторе и выполните следующие изменения:

**До:**

```
SHELLINABOX_ARGS="--no-beep"
```

**После:**

```
SHELLINABOX_ARGS="--no-beep --localhost-only --disablessl"
```

Первый аргумент (**--localhost-only**) велит *Shellinabox* слушать соединения только с собственного сетевого адаптера. Эта важная мера предосторожности гарантирует, что подключаться к *Shellinabox* смогут только процессы, запущенные локально на сервере, и не позволит подключаться напрямую к сервису из-за неправильно настроенного брандмауэра или некорректного правила NAT.

Второй аргумент (**--disable-ssl**) разрешает *Shellinabox* принимать только незашифрованные подключения. Сначала это может показаться непонятным, так как мы строим защищенную систему. Однако подключения к демону *Shellinabox* всегда будут выполняться только *Apache* через сетевую карту сервера. Соединение должно быть незащищенным, потому что *Shellinabox* использует самоподписанный сертификат, которому *Apache* доверять не будет.

Следующий шаг – перезагрузить демон *Shellinabox*, чтобы он перечитал конфигурацию. Сделать это можно командой

```
sudo service shellinabox restart
```

*Shellinabox* настроен – можно перейти к *Apache*, и сначала настроим его на прослушивание защищенных соединений.

```
sudo a2ensite default-ssl
sudo a2enmod ssl
sudo service apache2 restart
```

Если *Apache* не используется для предоставления других web-сервисов, можно настроить его так, чтобы он принимал только защищенные соединения.

```
sudo a2dissite default
```

Также откройте файл **/etc/apache2/ports.conf** в своем любимом редакторе и измените следующее:

**До:**

```
NameVirtualHost *:80
Listen 80
```

## А нужен ли брандмауэр?

В Ubuntu брандмауэр по умолчанию не включен ни в настольной, ни в серверной версии. Для большинства домашних компьютеров это редко является проблемой, так как широкополосные роутеры обычно по умолчанию не перенаправляют входящие соединения к внутренним компьютерам.

Самый простой и безопасный способ развернуть сочетание *Shellinabox*/*Apache* – перенаправлять на сервер через роутер только защищенные web-запросы (TCP/443). Некоторые широкополосные роутеры способны перенаправить все входящие соединения на указанный внутренний компьютер (это часто называется «режим (или хост) демилитаризованной зоны, DMZ»). Это удобно, если на сервере есть набор разных сервисов, однако в этом случае я настоятельно рекомендовал бы запустить брандмауэр.

На сервере Ubuntu проще всего воспользоваться UFW (Uncomplicated Firewall – несложный брандмауэр). Следующий скрипт активирует UFW и разрешает соединения по SSH и защищенные web-соединения отовсюду:

```
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow https
```

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

**После:**

```
#NameVirtualHost *:80
#Listen 80
```

Следующий шаг – включить обратный прокси в *Apache*, чтобы он смог работать с *Shellinabox*. Это сделает команда

```
sudo a2enmod proxy proxy_http
sudo service apache2 restart
```

Теперь нужно разрешить доступ к *Shellinabox* в файле настройки *Apache*. Настройки можно поместить в конфигурации виртуального хоста или глобально. Первый подход удобен в том случае, если на одном и том же компьютере будут находиться и защищенные, и незащищенные web-сервисы, но обратный прокси будет работать только с защищенными соединениями. В этом случае нужно изменить файл сайта SSL по умолчанию, в Ubuntu это **/etc/apache2/sites-enabled/default-ssl**. Здесь мы уже отключили незашифрованный доступ к *Apache*, поэтому настройки безопасно поместить в глобальный раздел.

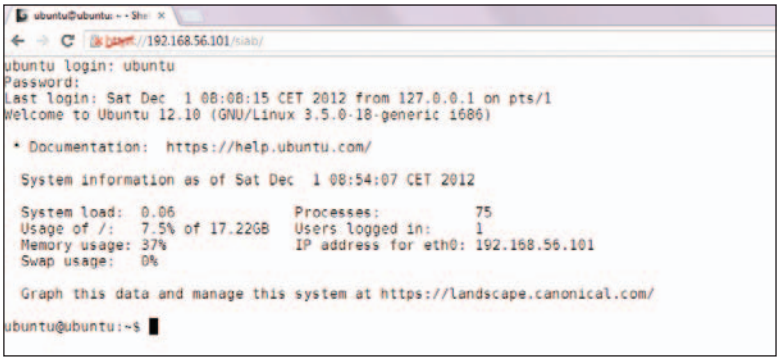
Откройте файл **/etc/apache2/mods-enabled/proxy.conf** в любимом текстовом редакторе и замените его содержимое на следующее:

```
<IfModule mod_proxy.c>
ProxyRequests Off
<Proxy *>
AuthType Basic
AuthName “The computer says no!”
AuthUserFile /etc/apache2/htpasswd.default
Require valid-user
Order allow,deny
Satisfy any
</Proxy>
ProxyPass /siab http://127.0.0.1:4200/
</IfModule>
```

Директива **ProxyRequests** определяет, будет ли *Apache* выступать в качестве прямого web-прокси. У нас *Apache* будет только обратным прокси, и ее можно безопасно отключить. Если вы хотите поэкспериментировать с ней, не включайте прямой прокси без авторизации, иначе ваш сервер могут моментально обидеть!

> «Важно не показывать, что находится за приглашением в систему.»

Следующий раздел определяет то, как предоставляется доступ для запросов прокси, и в данном случае мы пользуемся базовой аутентификацией HTTP. Директива **AuthName** задает сообщение, которые увидят пользователи, подключившиеся к серверу, и здесь по-настоящему важно не показывать, что находится за приглашением о входе в систему. Сообщение «Вход в оболочку!! [Login for shell access!!]» – исключительно неудачный выбор!




**Это оболочка, Джим, но не такая, какой мы ее знаем!**

Директива **AuthUserFile** ссылается на внешний файл с именами и паролями пользователей, имеющих доступ к файлу. Этот файл мы создадим на следующем этапе. Остальные директивы говорят *Apache*, что для доступа к сервису пользователь должен успешно аутентифицироваться.

Теперь надо создать файл с паролем, заданный в предыдущем файле настройки, с помощью утилиты *htpasswd*, командой

```
sudo htpasswd -c /etc/apache2/htpasswd.default siab
```

Утилита *htpasswd* создаст файл паролей и попросит ввести и подтвердить пароль для нового пользователя – в примере выше это пользователь siab. Имя этого пользователя должно обязательно отличаться от имени пользователя, используемого для входа на сервер (через консоль, по SSH и т.д.). Пароль тоже должен быть довольно надежным, так как с его помощью можно легко, но эффективно скрыть факт наличия доступа к оболочке.

Теперь мы готовы проверить подключение к серверу, но нужно в последний раз перезапустить Apache, чтобы он загрузил настройки обратного прокси:

```
sudo service apache2
restart
```

Теперь можно запустить браузер и открыть в нем адрес https://SERVER_IP/siab. Сайт SSL по умолчанию использует самоподписанный сертификат, поэтому появится предупреждение, но его можно спокойно игнорировать. Затем вам предложат ввести имя пользователя и пароль, созданные утилитой *htpasswd*. Должен появиться экран входа в систему *Shellinabox*, из которого можно войти в систему с обычными именем пользователя/паролем оболочки.

Теперь все работает, и можно спокойно включить перенаправление портов на роутере и удаленно подключиться к оболочке! LXF

**Скорая помощь**

С самоподписанным сертификатом вы привыкнете игнорировать ошибки браузера и с меньшей вероятностью обнаружите настоящую атаку. Чтобы этого избежать, бесплатно загрузите сертификат, которому доверяют большинство браузеров, с сайта startssl.com.

## Постоянные сеансы оболочки

Выполнять команды по беспроводной сети или широковещательному соединению, когда связь ведет себя своенравно, может быть сложно. Обрыв связи во время обновления программы или редактирования файла может оставить вашу систему в не лучшем виде.

Один очень простой способ этого избежать – привыкнуть пользоваться программой *Screen* для постоянных сеансов оболочки. *Screen* – полноэкранный оконный менеджер, позволяющий создавать виртуальные сеансы, которые могут сохраняться в разных подключениях. В Ubuntu Server он установлен по умолчанию, в других версиях Ubuntu, а также в большинстве других дистрибутивов Linux его легко установить из репозиториев.

Запустить новый сеанс *Screen* очень просто – наберите **screen**, и, нажав пробел, чтобы убрать информационное окно, вы попадете в свою оболочку по умолчанию.

Команды в *Screen* отправляются по нажатию Ctrl+A, а если затем нажать **?**, вы увидите список всех команд. Нам наиболее интересна команда отключения (Ctrl+A, d), которая отключит вас от сеанса *Screen*, оставив его работать в фоне. Повторно подключиться к этому сеансу можно, набрав **screen -r**. Если связь оборвется и *Screen* скажет, что вы уже подключены к сеансу, просто наберите **screen -d -r**, и подключитесь к сеансу принудительно.

Привычка пользоваться *Screen* при удаленном подключении позволит решать все задачи в оболочке, не опасаясь обрыва связи, и вы можете начать пользоваться ею даже в локальной сети!

# Виртуализируйте

**Дэвид Хейвард** показывает, как воскресить дух старого компьютера с Linux, запустив его виртуальный образ на новом компьютере.

Наш эксперт

**Дэвид Хейвард** Альпинист, фехтовальщик и писатель-романист. Известен также как любитель приврать о своих достижениях.


Виртуализация уже несколько лет является другом администратора серверной, так как преимуществ у нее масса. С появлением все более и более мощных компьютеров те полубоги, которые управляют серверами организаций по всему миру, обнаружили, что если виртуализировать некогда огромного зверя в углу кондиционируемой комнаты, это позволит сэкономить место и построить более эффективную модель управления сервером.

Но виртуализация под силу не одним обитателям Олимпа – мы тоже можем насладиться преимуществами виртуализации физического ПК. В нашем случае мы можем захотеть избавиться от старого компьютера, который включается лишь время от времени, но все еще нужен нам из-за каких-то древних программ, которые больше нигде не работают.

Место мы тоже хотим сэкономить – этот компьютер, как бы мы его ни любили, ужасно много его занимает. За то, что мы его уберем, супруги нас зауважают.

И в самом деле, хотя мы можем купить себе более мощный компьютер, с виртуализацией нам не придется уничтожать «дух» старого компьютера, а под духом мы имеем в виду то, что хотя само «железо» навсегда отправится на свалку, операционная система, приложения, настройки и все программы продолжат свою жизнь в виртуальной среде.

Сделать это на удивление просто – с помощью нескольких команд мы создадим образ старого компьютера и преобразуем его в виртуальную машину. Однако, как и с большинством задач в Linux и в компьютерах в целом, есть несколько способов добиться цели. Метод, который мы здесь рассмотрим – лишь один из многих, но также один из самых простых и наиболее эффективных; к тому же в нем используются встроенные команды Linux, а о работе системы нужно знать совсем немного. Если вы в целом представляете, как работает терминал и как в Linux идентифицируются диски, подключаемые к системе, то у вас все получится.

## Начинаем

По нашему сценарию, компьютеров два: на одном – Linux Mint 13, на другом – Ubuntu 12.04. Компьютер с Ubuntu PC немного новее, и на нем довольно много оперативной памяти, а именно 8 ГБ, а на компьютере с Mint всего 1 ГБ. Назовем компьютер с Mint 13 ПК1, а с Ubuntu 12.04 – ПК2. Наша задача – создать образ ПК1 и запустить его в виртуальной среде на ПК2. Звучит довольно просто, не правда ли?

Как мы уже говорили, сделать это можно разными способами. Мы возьмем внешний USB-диск размером 500 ГБ и подключим его к ПК1. Затем создадим образ ПК1 и сохраним его в файле на USB-диске для передачи на ПК2. Впрочем, вы можете просто отключить внутренний жесткий диск со своего ПК1, подключить его к ПК2 и запустить команду на одном компьютере, вместо того, чтобы переносить образ между двумя. Каждый из методов полностью нам подходит, хотя, если честно, установка жесткого диска ПК1 на ПК2 резко все ускорила бы; но сейчас мы не торопимся.

## Подготовка ПК1

Следующий этап – найти загрузочную копию дистрибутива Linux. Подойдет любой: вот у нас на флэшке есть копия Ubuntu 12.04. Загрузитесь с установочного диска и подключите внешний жесткий диск, затем запустите следующую команду, с целью разобраться в именовании съемных дисков:

```
lsblk
```

и нажмите Enter.

Выведется список всех блочных устройств (жестких дисков) в системе, включая разделы для каждого диска. В нашем случае,

```
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0    7:0    0 666.1M  1 loop /rofs
sda      8:0    0 232.9G  0 disk
├─sda1   8:1    0 229.1G  0 part
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0   3.8G  0 part [SWAP]
sr0     11:0    1  1024M  0 rom
sdb      8:16   1   1.9G  0 disk
└─sdb1   8:17   1   1.9G  0 part /cdrom
sdc      8:32   0 465.8G  0 disk
└─sdc1   8:33   0 465.8G  0 part /media/2C8069CF80699FD6
ubuntu@ubuntu:~$
```

**Рис. 1. Определите установленные диски и их текущие имена при помощи** lsblk.

# СВОЮ МАШИНУ

как показано на рис. 1, у нас есть жесткий диск объемом 232,9 ГБ с именем sda, о трех разделах, и внешний USB-диск sdc с одним разделом вместе с установкой Ubuntu. Запустив команду

```
sudo fdisk -l
```

мы получим подробную схему имен устройств и разделов. Но раз мы создаем образ всего жесткого диска, нам нужны только имя устройства диска, будь то hda или sda, и имя подключенного внешнего диска, hdb или sdb. На наших тестовых компьютерах, как показано на рис. 2, жесткий диск, образ которого мы делаем – sda, а внешний жесткий диск, на который запишем образ – sdc.

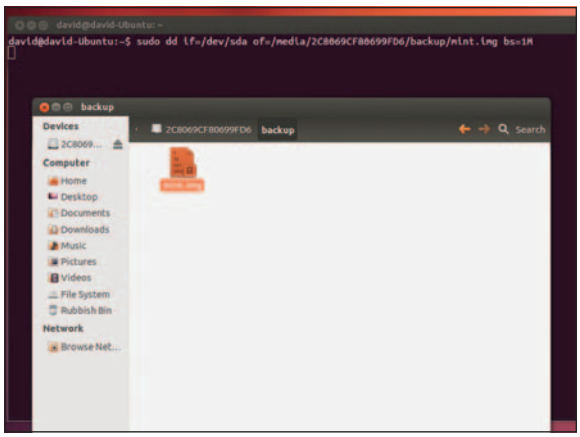
Наряду с именем устройства, команда **lsblk** также отобразит точку монтирования, на которой (или в которой) смонтирован внешний диск. На рис. 2 мы видим, что у единственного раздела на внешнем диске sdc точка монтирования /media/2C8069CF80699FD6. Этот путь нам пригодится в следующем разделе, когда мы перейдем к запуску команды создания образа.

> «Но виртуализация под силу не одним обитателям Олимпа.»

Вообще говоря, стоит также отметить, что sda – это главный жесткий диск компьютера, а sdb, sdc и т.д. – остальные физические диски. С другой стороны, hda используется для более старых устройств IDE, но принцип именования hdb, hdc и т.д. остается все тем же. Мы пользуемся установочным диском для того, чтобы операционная система на ПК1 не использовалась и, следовательно, все файлы, заблокированные во время обычной работы ОС, были доступны для создания образа. Образ, кстати, вполне можно сделать и с работающей системы хоста (ПК2), но в подобных случаях пользование установочным диском рекомендуется как правильный подход.

```
ubuntu@ubuntu: ~
sr0      11:0    1  1024M  0 rom
sdb       8:16   1   1.9G  0 disk
└─sdb1    8:17   1   1.9G  0 part /cdrom
sdc       8:32   0 465.8G  0 disk
└─sdc1    8:33   0 465.8G  0 part /media/2C8069CF80699FD6
ubuntu@ubuntu:~$ sudo fdisk -l

Disk /dev/sda: 250.1 GB, 250059350016 bytes
255 heads, 63 sectors/track, 30401 cylinders, total 488397168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00014c5f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048   480421887   240209920   83  Linux
/dev/sda2       480423934   488396799     3986433    5  Extended
/dev/sda5       480423936   488396799     3986432   82  Linux swap / Solaris

Disk /dev/sdb: 2002 MB, 2002747392 bytes
32 heads, 63 sectors/track, 1940 cylinders, total 3911616 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x4b287b9d

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1   *          63     3911039     1955488+   b  W95 FAT32

Disk /dev/sdc: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe0b2e182

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1              63   976768064   488384001    7  HPFS/NTFS/exFAT
ubuntu@ubuntu:~$
```

**› Рис. 2. Команда** lsblk **вместе с** sudo fdisk -l **даст вам подробную информацию о дисках.**




**› Рис. 3. Начинаем создавать образ ПК. Это может потребовать времени — возьмите чашечку кофе и развалитесь в кресле.**

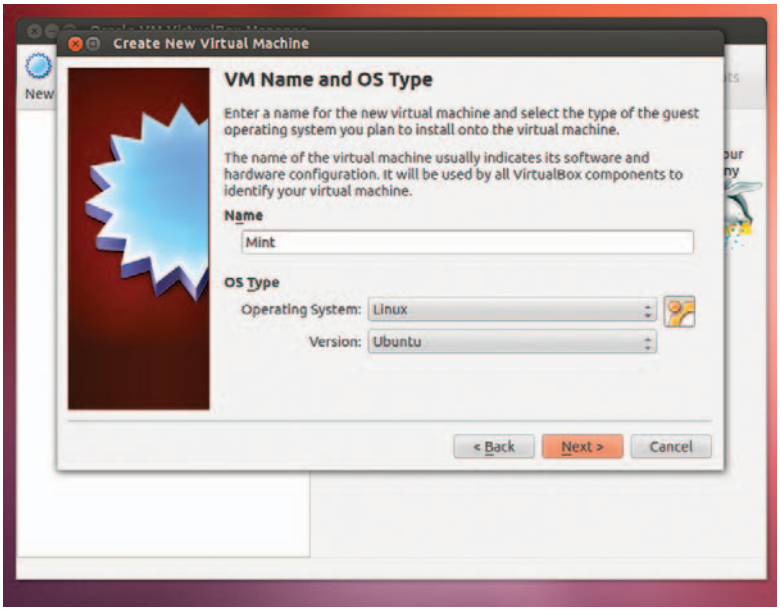
## Начинаем создавать образ ПК

Узнав, который диск какой, пора перейти собственно к созданию образа. Для этого мы воспользуемся распространенной и некоторыми редко применяемой командой **dd**. **dd** – команда очень мощная, и способна как на всевозможные волшебные преобразования и копирование данных, так и на затирание всего диска; будьте осторожны, пользуясь ею вне нашего урока!

Прежде чем заняться образом диска, создадим на внешнем диске каталог с креативным названием **backup**. Сюда мы запишем файл с образом.

После этого можно начать создавать образ ПК1, запустив в терминале команду:

»




**› Рис. 4. Создание новой виртуальной машины в *VirtualBox*, где будет жить наш образ. Выбирайте, естественно, Linux.**

```
sudo dd if=/dev/sda of=/media/2C8069CF80699FD6/backup/mint.img bs=1M
```

Нажмите Enter и введите свой пароль администратора для запуска команды. Здесь мы просим систему создать образ **/dev/sda**, жесткого диска ПК1, и скопировать его в файл **mint.img** в каталоге **backup** в точке монтирования **/media/2C8069CF80699FD6/**, которая, как мы выяснили, находится на внешнем диске sdc. Привесок **bs=1M** означает количество байт, которые считываются, записываются и конвертируются за один шаг.

Значение 1M значительно увеличит производительность при записи; другими словами, это займет меньше времени, но потребует гораздо больше памяти. Так как задача ПК1 – только запустить установочный диск, значения 1M с него хватит.


**› Рис. 5. Вуаля! Ваш бывший физический компьютер, а ныне — виртуальная машина, со всем прежним хозяйством.**

Как вы видите на рис. 3, процесс создания образа начинается с записи содержимого sda в файл **mint.img**. Учтите, что хотя мы и добавили параметр 1M, процесс потребует времени, в зависимости от размера диска, и приготовьтесь ждать результатов до нескольких часов.

Однажды мы решили создать образ диска в 2 ТБ, обойдясь без дополнительных ключей, и на это ушло 15 часов. Мы были не в восторге, но, как говорится, век живи – век учись.

Когда процесс создания образа закончится и у вас будут результаты работы процесса **dd**, отключите внешний диск и выключите ПК1 – его работа пока закончена.

Теперь подключите внешний диск с файлом образа к ПК2 и установите *VirtualBox* и *QEMU*, если еще не сделали этого; оба найдутся через менеджеры пакетов большинства дистрибутивов Linux.

Если не получается найти *QEMU*, поищите *AQEMU*, это графический интерфейс на *Qt4* для управления виртуальными машинами *QEMU* и *KVM*.

Следующий этап можно выполнить по-разному. Можно сконвертировать файл образа в виртуальный диск *VirtualBox* и пользоваться им в самом *VirtualBox*, а можно запустить этот файл в *QEMU*.

## Способ 1

Сначала сконвертируем образ ПК 1 в формат, который может читать и понимать *VirtualBox*, поэтому сперва скопируем свежесозданный образ **mint.img** на ПК2. Конечно, можно было бы работать прямо с внешнего жесткого диска, но с локальных дисков ПК2 все будет работать гораздо надежнее.

> «Значение 1M увеличит производительность при записи.»

Когда файл образа скопируется, зайдите в терминал и наберите следующую команду для преобразования файла **mint.img** в VDI-файл для *VirtualBox*:

```
vboxmanage convertfromraw mint.img –format VDI mint.vdi
```

Нажмите Enter, и *VirtualBox* примется конвертировать образ **mint.img** в формат VDI, используемый как формат виртуального жесткого диска в *VirtualBox*. Опять же в зависимости от размера образа это может занять некоторое время.

## Настройка виртуальной машины

После этого нужно открыть *VirtualBox* и создать новую виртуальную машину, щелкнув по кнопке New [Создать] в левом верхнем углу окна *VirtualBox*, затем нажать Next [Далее], указав имя новой виртуальной машины, в данном случае, Mint, и изменив Operating System [Операционную систему] и Version [Версию] на Linux и Ubuntu соответственно – см. рис. 4.

В следующем окне нужно выделить ОЗУ, достаточное для сконвертированного образа; изначально на ПК1 был 1 ГБ, поэтому все значения выше этого пойдут в плюс.

Когда вас попросят выбрать Virtual Hard Disk [Виртуальный жесткий диск], вместо варианта по умолчанию Create new hard disk [Создать новый жесткий диск] щелкните на Use existing hard disk [Использовать существующий жесткий диск] и выберите результат конверсии – файл **Mint.VDI**.

Наконец, нажмите Create [Создать] в окне со сводкой, и файл образа ПК1 будет использоваться как жесткий диск созданной виртуальной машины.

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

## Утилиты VMware

Если у вас есть экземпляр *VMware Workstation* и серверные утилиты, вам повезло. Установив Linux-версию *VMware vCentre Converter Standalone Client*, вы сможете создать образ своего компьютера прямо из работающей системы и переслать его на сервер *VMware*.

Разумеется, *VMware* стоит недешево. Поэтому либо идите нашим путем, либо копите денежки!

По сути, все, что мы сделали – дали виртуальной машине имя в *VirtualBox* и выделили достаточно оперативной памяти для ее запуска. Сама виртуальная машина уже существует в виде созданного и сконвертированного образа ПК1 в файле **Mint.VDI**.

После создания виртуальной машины подправьте все необходимые параметры, например, объем доступной видеопамяти, и нажмите Start [Запустить] для загрузки образа ПК1 в *VirtualBox*, как показано на рис. 5.

## Способ 2

Этот способ гораздо проще, так как конвертировать ничего не надо. Воспользовавшись ранее установленной *QEMU* или *AQEMU*, перейдите в терминал и наберите:

```
kvm mint.img -m 2047
```

Образ запустится в виртуальном окружении *QEMU*, так же, как это было при первом создании образа ПК1, что видно из рис. 6. Однако рабочий стол, соответствующий пользователю на ПК1, не загрузится. Несмотря на это, системой можно пользоваться так же, как и на ПК1, и все ваши приложения будут доступны, как получилось с нашими. Потерялись только параметры рабочего стола. Между прочим, параметр -M также выделяет образу 2 ГБ оперативной памяти; а так как на старом компьютере было всего 1 ГБ, это уже значительное улучшение.

Хотя этот способ быстрее, он довольно ненадежен по части запуска образа. Системы образов часто «падают» и работают очень медленно – но все-таки работают, а немного разобравшись в параметрах командной строки *QEMU*, вы сможете устранить большинство ошибок. У каждого метода есть свои преимущества, хотя новичкам в виртуализации больше подойдет *VirtualBox* с его плавным и удобным интерфейсом.

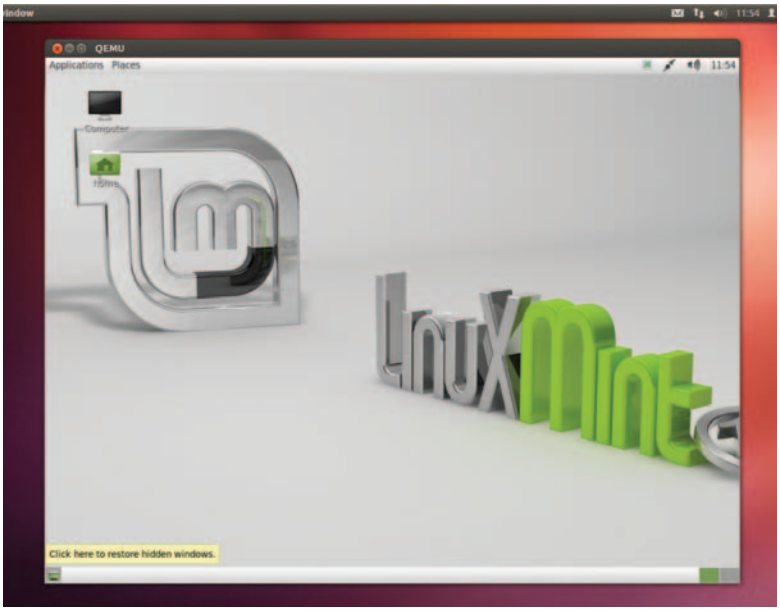



**Рис. 6. *QEMU* упрощает процесс, но здесь не хватает ряда возможностей *VirtualBox*.**

Последний этап – тщательно проверить виртуальный образ в *VirtualBox* или *QEMU*. Убедитесь, что все работает как полагается, и что приложения и все специализированные программы запускаются так, как запускались на исходном ПК1.

Если вы убедились, что образ ПК1 работает именно так, как должен, наверное, пора избавить от мучений бедного зверя? Или, как мы, сохранить старые компьютеры, хотя их образы мы уже сделали. Как известно, в глубине души мы скрытые плюшкины и ненавидим саму мысль о том, чтобы выбросить железку, работает она или нет! LXF


**QEMU — старая любимица племени виртуализаторов, но суровее к новичкам, чем *VirtualBox*.**

ЧАСТЬ 3

# Введение

**Джоно Бэкон** применяет *Twitter Bootstrap* для украшения проектов *Django*.

Наш эксперт

**Джоно Бэкон** управляет сообществом Ubuntu; он автор книги «Искусство сообщества» и основатель ежегодного саммита руководителей сообщества.

Два урока мы занимались созданием мощных сайтов с помощью популярного фреймворка *Django*. В предыдущих частях мы рассмотрели такие темы, как создание баз данных, добавление данных, использование форм и другие. Все это было очень... функционально, но, к сожалению, выглядело довольно уродливо.

На этом уроке мы узнаем, как немного приукрасить ваши приложения. *Django* добивается многого малым объемом кода. Мы создадим простую шаблонную программу и затем поэкспериментируем с набором различных возможностей, чтобы создать привлекательный сайт, совместимый с большинством браузеров. Приступим к делу.

## Современная эпоха

Скажу для новичков в web-программировании: для создания тем и стилей web-страниц используется специальный язык CSS (Cascading Style Sheets – каскадные таблицы стилей). С ним можно взять HTML-код страницы и изменить ее визуальное наполнение, например, цвета и контуры, а также расположение текста на странице. Увы, ветераны web-программирования хорошо знают, что CSS чертовски сложен, не с точки зрения языка, а из-за всех этих уловок и приемов, требуемых для корректного отображения содержимого страницы в различных браузерах, поскольку в каждом из них то там, то здесь есть отличия. По существу, понимание CSS – это отдельный навык, а написание красивого CSS – настоящее искусство.

В былые времена CSS для сайтов приходилось писать вручную, и в итоге наплодили безобразных сайтов. К счастью, времена изменились, и хотя вручную писать никто не запрещает, большинство людей пользуются клиентскими фреймворками, оборачивающими всю эту сложность в набор кирпичиков «Лего» для сборки дизайна страницы. Иначе говоря, вам достаточно знать, как применять кирпичики, а они сделают всю работу по отображению должного CSS в любом браузере, которым вы пользуетесь.

## Что умеет Bootstrap?

Иногда бывает трудно понять, на что способна та или иная программа. Вот краткий перечень возможностей *Bootstrap*:

- **Оформление текста** Заголовки, основной текст, выделение, жирный, курсив, аббревиатуры, адреса, цитаты, списки, код и т. д.
- **Таблицы** Множество различных способов форматирования таблиц.
- **Формы** Различные способы отображения форм и надписей на них.
- **Кнопки** Различные размеры, виды и состояния кнопок.
- **Изображения** Легко задаваемые стили для обрамления изображений, иконок и иконок-кнопок (например, для панелей инструментов).
- **Страницы** Навигационные цепочки, нумерация страниц и панели навигации.
- **Символы и оповещения** Легко настраиваемые символы и оповещения различными способами привлекают внимание пользователя.
- **Модальные диалоги** В *Bootstrap* легко отобразить в модальном диалоге формы и другое содержимое.

В *Bootstrap* имеется немалое количество и других возможностей; их полный список вы можете просмотреть на http://twitter.github.com.

Существует масса фреймворков, каждый со своими достоинствами и недостатками. В качестве фреймворка нам обычно нужно нечто гибкое (включающее разнообразные возможности для различных требований к интерфейсу пользователя), приятное на вид (соответствующее выбранному вами дизайну), небольшое по размеру (большие страницы увеличивают нагрузку на канал) и лицензированное так, как вам нужно (в данном случае, доступное свободно). Я исследовал множество различных фреймворков, и мне больше всех прочих понравился *Bootstrap*, созданный в Twitter.

*Bootstrap* обладает фантастическим набором возможностей. Он поддерживает различные гарнитуры шрифтов, виды кнопок, иконки, вкладки, модальные диалоги, гибкие структуры компонентов и другое. Он приятен на вид, настраиваем, снабжен прекрасную документацию и лицензирован по свободной лицензии Apache. К счастью, подключить его к *Django* просто, и это дает нам впечатляющие возможности по изменению внешнего вида сайта.

## Создаем шаблон

Начнем с создания простого шаблонного проекта *Django*, и затем мы сможем посмотреть, как работают различные компоненты *Bootstrap*. Я пробегусь по созданию проекта довольно быстро, так как почти все это должно быть вам знакомо.

Сначала создается проект *Django*:

```
django-admin.py startproject homepage
```

Затем зададим настройки в **settings.py**. Сперва зададим «шаблонный» поиск проекта в верхней части файла:

```
import os
ROOT_PROJECT = os.path.join(os.path.split(__file__)[0], "..")
```

Настроим базу данных:

```
'ENGINE': 'django.db.backends.sqlite3'
'NAME': 'homepage.db'
```

В *Django* есть понятие местоположения «статических» файлов. Сюда мы поместим файлы тем *Bootstrap*. По умолчанию это каталог **static** проекта. Настроим его – для этого укажите следующий параметр:

```
STATIC_URL = '/static/'
```

Свяжем с этим каталогом параметр **STATICFILES_DIRS**:

```
STATICFILES_DIRS = (
  os.path.join(ROOT_PROJECT, 'static'),
)
```

Как и в предыдущих проектах, мы помещаем шаблоны в каталог 'templates' проекта. Задайте следующий параметр:

```
TEMPLATE_DIRS = (
"templates",
)
```

Создадим в проекте приложение *mainapp*:

```
python manage.py startapp mainapp
```

Наконец, добавьте следующие параметры в раздел **INSTALLED_APPS** файла **settings.py**. В нашем приложении-шаблоне будет три страницы: домашняя, страница с информацией о программе и страница с контактами. В **urls.py** добавьте следующие параметры:

# в Django

```
url(r'^$', 'mainapp.views.index'),
url(r'^about/$', 'mainapp.views.about'),
url(r'^contact/$', 'mainapp.views.contact'),
```

Теперь добавьте представление в **views.py**:

```
from django.shortcuts import render
from mainapp.models import Contact
from mainapp.forms import ContactForm
from django.template import RequestContext
from django.shortcuts import render_to_response
def index(request):
    return render(request, 'mainapp/index.html')
def about(request):
    return render(request, 'mainapp/about.html')
def contact(request):
    form = ContactForm(request.POST or None)
    if form.is_valid():
        cmodel = form.save()
        cmodel.save()
    return redirect(index)
return render_to_response('mainapp/contact.html', {'contact_
form': form}, context_instance=RequestContext(request))
```

Этот код просто выводит страницы – кроме страницы с контактами, на которой выводится форма из модели **Contact data**. Добавим эту модель в **models.py**:

```
class Contact(models.Model):
    name = models.CharField(max_length=200)
    email = models.CharField(max_length=200)
    query = models.TextField(blank=True, null=True)
    def __unicode__(self):
        return self.name
```

Потом добавим файл **forms.py** для отображения **ModelForm**:

```
from django.forms import ModelForm
from mainapp.models import Contact
class ContactForm(ModelForm):
    class Meta:
        model = Contact
```

За исключением шаблонов (в которые мы будем добавлять информацию о теме), наш шаблонный проект готов. Теперь создадим базу данных:

```
python manage.py syncdb
```

Загрузим и добавим в проект тему *Bootstrap*. Для этого сначала создайте каталог **static** в корне проекта. Зайдите на http://twitter.github.com/bootstrap и загрузите последнюю версию *Bootstrap*. В загруженном архиве есть каталог **bootstrap** с тремя подкаталогами **css**, **img** и **js**. Распакуйте эти три подкаталога в свой каталог **static**. Теперь все необходимое для проекта готово, но мы пока не можем его запустить, поскольку нет шаблонов. Ну так мы их добавим!

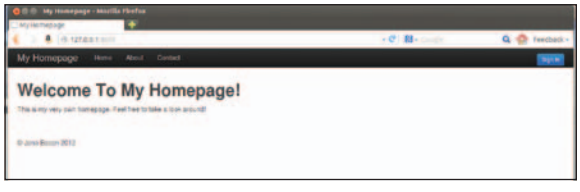



**Рис. 1. Наша раскладка по *Bootstrap* проста, но доходчива.**

## Памятка по Bootstrap: Кнопки

**Типы кнопок:**

- **btn** – стандартная серая кнопка с градиентом.
- **btn btn-primary** – выделяет кнопку визуально и задает первичное действие в наборе кнопок.
- **btn btn-info** – используется как альтернатива стилям по умолчанию.
- **btn btn-success** – означает успешное или положительное действие.
- **btn btn-warning** – означает предупреждение, связанное с выполняемым действием.
- **btn btn-danger** – означает опасное или потенциально негативное действие.
- **btn btn-inverse** – альтернативная темно-серая кнопка, не связанная с семантическим действием или назначением.
- **btn btn-link** – делает из кнопки ссылку, сохраняя поведение кнопки.

## Темы и шаблоны

Чтобы воспользоваться *Bootstrap*, нужно добавить в шаблоны код, который загружает библиотеку *bootstrap*; после этого в шаблоне можно будет пользоваться специальными командами для отображения содержимого различными способами. Раньше у нас было по одному шаблону на страницу, и все было просто. Для нашего сегодняшнего проекта в шаблонах нужно загружать существующий код, формирующий общую структуру сайта, а также код, загружающий библиотеку *bootstrap*.

Настроить расположение компонентов на странице с помощью *Bootstrap* можно по-разному, но для простоты мы ограничимся панелью навигации вверху и двумя колонками на странице – основным содержимым (col1) и боковой колонкой (col2). Сначала создадим базовый файл шаблона, который будут загружать все остальные. В этом файле мы загрузим тему Bootstrap и сформируем вышеупомянутую схему из двух колонок.

Хотя сейчас в нашем проекте *Django* всего одно приложение (*mainapp*), в будущем их может стать несколько, а мы хотим, чтобы в них во всех использовалась одна и та же тема. Итак, создадим наш базовый шаблон вне приложения *mainapp*, чтобы он мог совместно использоваться всеми приложениями.

Для начала создайте каталог **templates** в корневом каталоге проекта, и в нем – файл **base.html**. Начнем добавлять содержимое. Сначала добавим несколько мета-тэгов шаблона:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My Homepage [Моя домашняя страница]</title>
<meta name="viewport" content="width=devicewidth,
initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="">
```

Затем загрузим главный CSS-файл *Bootstrap*. Он живет в **static/css**, поэтому воспользуемся специальным тэгом шаблона *Django*, чтобы сослаться на каталог **static**:

```
<link href="{{ STATIC_URL }}/css/bootstrap.css" rel="stylesheet">
```

Теперь добавим дальнейшее содержимое – это несколько блоков CSS, которые введут небольшие зазоры вверху и внизу страницы:

```
<style type="text/css">
  body {
    padding-top: 60px;
    padding-bottom: 40px;
  }
  .sidebar-nav {
    padding: 9px 0;
  }
</style>
```

Далее добавим CSS-код Bootstrap, отвечающий за дизайн страницы. С ним страница будет правильно отображаться на планшетах, телефонах и других устройствах. Хотя на данном уроке мы этого не обсуждаем, код все равно добавим, чтобы при желании обратиться к нему в будущем:

```
  <link href="{{ STATIC_URL }}/css/bootstrapresponsive.
css"rel="stylesheet">
  <!--[if lt IE 9]>
    <script src="http://html5shim.googlecode.com/svn/trunk/
html5.js"></script>
  <![endif]-->
  <link rel="shortcut icon" href="../assets/ico/favicon.ico">
  <link rel="apple-touch-icon-precomposed"sizes="144x144"
href="../assets/ico/apple-touch-icon-144-precomposed.png">
<link rel="apple-touch-icon-precomposed"sizes="114x114"
href="../assets/ico/apple-touch-icon-114-precomposed.png">
  <link rel="apple-touch-icon-precomposed"sizes="72x72"
href="../assets/ico/apple-touch-icon-72-precomposed.png">
  <link rel="apple-touch-icon-precomposed"href="../assets/ico/
apple-touch-icon-57-precomposed.png">
  </head>
  <body>
```

Приступим к организации раскладки нашего сайта. *Bootstrap* предоставляет богатую коллекцию классов, применяемых к различным элементам HTML. Например, к тэгу ссылки **<a>** можно применить класс, который превратит ссылку в кнопку указанного типа. Аналогично, существует набор классов, применяемых к тэгу ** ** для настройки расположения компонентов страницы. Давайте сначала добавим панель навигации в верхней части страницы.

```
<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="navbar-inner">
    <div class="container-fluid">
      <a class="brand" href="/">My Homepage</a>
      <div class="nav-collapse collapse">
        <p class="navbar-text pull-right">
          <a href="" class="btn btn-small btn-primary">Sign In</a>
        </p>
        <ul class="nav">
          <li><a href="/">Home</a></li>
          <li><a href="/about">About</a></li>
          <li><a href="/contact">Contact</a></li>
        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </div>
</div>
```

В этом блоке мы сперва создаем внешнюю панель навигации (**navbar**), закрепленную в верхней части страницы (**navbar-fixed-top**), а затем добавляем компоненты к внутренней части панели (**navbar-inner**). Затем мы добавляем ссылку с названием сайта класса **brand**.

Далее добавляется кнопка Sign In [Войти] в виде ссылки с классами **btn**, **btn-small** и **btn-primary**. Это три различных класса; **btn** отображает ссылку в виде кнопки, **btn-small** уменьшает ее размер, а **btn-primary** выделяет ее синим, как важный элемент пользовательского интерфейса. Это хороший пример форматирования страниц с *Bootstrap*.

Наконец, с помощью тэга **nav** мы добавляем маркированный список с тремя элементами; они форматируются как кнопки навигационной панели. Теперь добавим код в тело страницы. Он создает две колонки:

```
    <div class="container-fluid">
      <div class="row-fluid">
        <div class="span9">
          {% block col1 %}
          {% endblock %}
        </div><!--/span-->
        <div class="span3">
          {% block col2 %}
          {% endblock %}
        </div><!--/span-->
      </div><!--/row-->
```

*Bootstrap* делит страницу на 12 колонок. Здесь мы создаем ** ** для главной колонки и используем в нем класс **span9** – он забирает 9 из 12 колонок. Затем мы устанавливаем второй ** ** в **span3** – это превращает три оставшихся колонки в узкую боковую колонку.

Внутри каждого ** ** определяем блок шаблона *Django* и задаем его имя (**col1** или **col2**). Именами мы воспользуемся в главных шаблонах, чтобы добавлять содержимое в эти колонки. Наконец, добавим нижний колонтитул и файлы JavaScript (они добавляются в конце файла, чтобы страница загружалась быстрее):

```
    <hr>
    <footer>
      <p>&copy; Jono Bacon 2012</p>
    </footer>
  </div><!--/.fluid-container-->
```

## Памятка по Bootstrap: Метки

Метки и значки позволят привлечь внимание к отдельным частям страницы. Непременно поиграйте с разнообразными вариантами.

**Метки:**

- **По умолчанию** – Default
- **Успешно** – Success
- **Предупреждение** – Warning
- **Важно** – Important
- **Информация** – Info
- **Инверсная** – Inverse

**Значки:**

- **По умолчанию** – 1
- **Успешно** – 2
- **Предупреждение** – 4
- **Важно** – 6
- **Информация** – 8
- **Инверсный** – 10

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.


table

| Name | Age | Location |
|---|---|---|
| Steve Harris | 55 | London, England |
| Nicko McBrain | 52 | Florida, USA |

table-striped

| Name | Age | Location |
|---|---|---|
| Steve Harris | 55 | London, England |
| Nicko McBrain | 52 | Florida, USA |

table-bordered

| Name | Age | Location |
|---|---|---|
| Steve Harris | 55 | London, England |
| Nicko McBrain | 52 | Florida, USA |

table-condensed

| Name | Age | Location |
|---|---|---|
| Steve Harris | 55 | London, England |
| Nicko McBrain | 52 | Florida, USA |

**› В *Bootstrap* есть варианты прорисовки таблиц для различных потребностей с небольшими различиями между ними.**

```
    <script src="../assets/js/jquery.js"></script>
    <script src="../assets/js/bootstrap-transition.js"></script>
    <script src="../assets/js/bootstrap-alert.js"></script>
    <script src="../assets/js/bootstrap-modal.js"></script>
    <script src="../assets/js/bootstrap-dropdown.js"></script>
    <script src="../assets/js/bootstrap-scrollspy.js"></script>
    <script src="../assets/js/bootstrap-tab.js"></script>
    <script src="../assets/js/bootstrap-tooltip.js"></script>
    <script src="../assets/js/bootstrap-popover.js"></script>
    <script src="../assets/js/bootstrap-button.js"></script>
    <script src="../assets/js/bootstrap-collapse.js"></script>
    <script src="../assets/js/bootstrap-carousel.js"></script>
    <script src="../assets/js/bootstrap-typeahead.js"></script>
  </body>
</html>
```

Наш базовый шаблон готов; теперь создадим шаблон страницы, чтобы проверить дизайн. Для этого загрузим шаблоны внутри приложения. Чтобы все шаблоны использовали одну и ту же тему, создайте каталог **mainapp** в каталоге **templates** и затем файл **base.html** в каталоге **mainapp**. Добавьте в него всего одну строку:

```
{% extends "base.html" %}
```

Она загружает наш главный файл **base.html** с кодом CSS, который применяется ко всем приложениям. Теперь создадим шаблон страницы. В каталоге **templates/mainapp** создайте файл **index.html** и добавьте в него следующий код:

```
{% extends "mainapp/base.html" %}
{% block col1 %}
  <h1>Welcome To My Homepage! [Добро пожаловать на мою домашнюю страницу!]</h1>
  <p>This is my very own homepage. Feel free to take a look around! [Я сам ее сделал. Смотрите, не стесняйтесь!]</p>
{% endblock %}
{% block col2 %}
{% endblock %}
```

В этом блоке мы загружаем базовый шаблон и добавляем два блока, которые мы определили в главной структуре. В первом блоке (col1), то есть в теле страницы, мы просто добавляем заголовок и немного текста; второй блок пока оставляем пустым. Теперь страница должна выглядеть так, как показано на рис. 1.

## Создаем раскладку

Теперь погрузимся в возможности *Bootstrap*, которые придают ему такую мощь. Во-первых, сейчас в боковой колонке ничего нет. Добавим во второй блок немного информации о нашей домашней странице:

```
{% block col2 %}
  <div>
    <div class="well">
      <h3>The Technology</h3>
      <p>Этот сайт создан при помощи Django и темы Bootstrap.</p>
    </div>
  </div>
{% endblock %}
```

Здесь с помощью класса well элемента  мы создаем у боковой колонки тонкий контур с закругленными краями. В том-то и прелесть Bootstrap – чтобы создать контур, не нужно писать никакого кода CSS. Теперь добавим несколько кнопок со ссылками на сайты *Django* и *Bootstrap*. Для этого достаточно добавить несколько ссылок с классом **btn**:

```
<a href="https://www.djangoproject.com/"class="btn">Django</a>
<a href="http://twitter.github.com/"class="btn">Bootstrap</a>
```

Ну вот, у нас есть красивые кнопки, вид которых изменяется при наведении курсора мыши. Существуют и другие классы, с помощью которых можно менять внешний вид кнопок. Например, чтобы изменить размер кнопок, воспользуйтесь классами **btn-large**, **btn-small** и **btn-mini**. Для этого просто добавьте их в код справа от первого класса. Например:

```
<a href="http://twitter.github.com/" class="btn btnsmall">Bootstrap</a>
```

Кнопки выглядят вполне мило, но лучше будет объединить их в группу, например, так:

```
<center>
  <div class="btn-group">
    <a href="https://www.djangoproject.com/" class="btn">Django</a>
    <a href="http://twitter.github.com/" class="btn">Bootstrap</a>
  </div>
</center>
```

Мы здесь воспользовались классом **btn-group** в тэге ** **; теперь кнопки сгруппированы.

## Таблицы

Таблицы в *Bootstrap* тоже форматируются просто. Те ужасные таблицы HTML, которые вы знали и любили (ненавидели), ушли в прошлое. Вот пример таблицы с приятным контуром, созданным с помощью класса table-bordered:

```
<table class="table table-bordered">
  <th>Name</th><th>Age</th><th>Location</th>
  <tr><td>Steve Harris</td><td>55</td><td>London, England</td></tr>
  <tr><td>Nicko McBrain</td><td>52</td><td>Florida, USA</td></tr>
</table>
```

Еще одна приятная возможность *Bootstrap* – набор иконок, которыми можно быстро и легко воспользоваться. Забыты дни, когда нужно было встраивать собственные иконки, чтобы придать привлекательность сайту.

С каждой иконкой связан собственный класс. Давайте добавим по иконке на каждую из добавленных кнопок:

```
<a href="https://www.djangoproject.com/"class="btn"><i class="icon-pencil"></i> Django</a>
<a href="http://twitter.github.com/" class="btn"><i class="icon-eye-open"></i> Bootstrap</a>
```

Здесь мы добавили пустой тэг **<i>** и просто указали класс для загрузки нужной иконки; при этом иконка будет показана на каждой кнопке с правильными отступами. Чтобы увидеть полный список доступных иконок, зайдите на домашнюю страницу *Bootstrap* – http://twitter.github.com/.

Есть и прекрасная документация. Беритесь за дело и экспериментируйте с различными возможностями оформления в собственных проектах. Удачи! LXF

# Erlang: Практика

**Андрей Ушаков** не поступился принципом единоличной ответственности. И вот куда это его завело...

Наш эксперт

**Андрей Ушаков** активно приближает тот день, когда функциональные языки станут мейнстримом.

Итак, мы продолжаем решать нашу большую задачу: создание многозадачных версий функций **map** и **reduce**. На этом уроке мы поговорим о том, как на основе функций, созданных в прошлый раз, реализовать распределенные (выполняющиеся на различных узлах) версии функций **map** и **reduce**. Также мы проанализируем допущения, принятые нами, когда мы реализовывали версии функций **map** и **reduce** на основе «многоразовых» процессов. Это позволит нам понять, куда идти дальше. Но сперва взглянем, где мы остановились в прошлый раз.

На прошлом уроке мы перешли к модели «многоразовых» рабочих процессов и ограничили их число. На основе этого подхода мы создали очередные многозадачные версии функций **map** и **reduce**: функции **parallel_map:limited_pmap/4** и **parallel_reduce:limited_reduce/5**. И, как обычно, при создании этих функций мы вынесли общую функциональность в ряд функций, располагающихся в модуле **parallel_limited_helper**. При реализации этой функциональности мы приняли два важных соглашения: во-первых, договорились, что ответственность за создание и уничтожение процессов ложится на вызывающую сторону. Во-вторых, все задания мы сразу распределяем между рабочими процессами, после чего только дожидаемся результатов их работы.

Еще раз рассмотрим эту общую функциональность. **parallel_limited_helper:limited_worker/1** является функцией, которую выполняет рабочий процесс во время своей жизни:

```
limited_worker(Fun) ->
  receive
    {task_request, MasterPid, Index, SourcePortion} ->
      Dest = Fun(SourcePortion),
      MasterPid ! {result, Index, Dest},
      limited_worker(Fun);
    _Other -> limited_worker(Fun)
  end.
```

В этой функции рабочие процессы выполняют задания на обработку порций данных, которые они получают в виде сообщений, посланных рабочим процессам главным процессом. Естественно, что результаты работы отсылаются обратно главному процессу. Мы экспортируем эту функцию из модуля **parallel_limited_helper**, т. к. мы договорились, что рабочие процессы создает внешний код. Пара функций **send_worker_tasks/2** и **send_worker_tasks/3** используется для распределения заданий между рабочими процессами (функций **send_worker_tasks/2** является интерфейсом, а функция **send_worker_tasks/3** – реализацией данной функциональности):

```
send_worker_tasks(PreparedData, WorkerList) ->
  send_worker_tasks(PreparedData, WorkerList, 1).
send_worker_tasks([], _WorkerList, _WorkerIndex) -> complete;
send_worker_tasks(PreparedData, WorkerList, WorkerIndex)
when WorkerIndex > length(WorkerList) ->
  send_worker_tasks(PreparedData, WorkerList, 1);
send_worker_tasks([{Index, Portion} | Rest], WorkerList,
WorkerIndex) ->
  Worker = lists:nth(WorkerIndex, WorkerList),
  Worker ! {task_request, self(), Index, Portion},
  send_worker_tasks(Rest, WorkerList, WorkerIndex + 1).
```

Эта пара функций всего лишь определена в модуле **parallel_limited_helper**, но не экспортируется из него, т. к. инкапсулирует один из внутренних шагов. И, наконец, функция **parallel_limited_helper:limited_core/4** является сердцем всех реализаций, основанных на этой функции:

```
limited_core(FinalAggrFun, SourceList, PortionSize, WorkerList) ->
  process_flag(trap_exit, true),
  PortionCount = parallel_common:calc_portion_
count(length(SourceList), PortionSize),
  PreparedData = parallel_common:prepare_data(PortionSize,
SourceList),
  send_worker_tasks(PreparedData, WorkerList),
  EmptyStorage = array:new([{size, PortionCount}, {fixed, true},
{default, none}]),
  FullStorage = parallel_common:collect_result(EmptyStorage,
PortionCount),
  process_flag(trap_exit, false),
  FinalAggrFun(array:to_list(FullStorage)).
```

В этой функции мы разбиваем исходные данные на порции, равномерно распределяем задания по обработке порций данных между созданными рабочими потоками, собираем результаты обработки порций данных рабочими потоками и объединяем результаты их работы в итоговый результат.

Рассмотрим соглашения, принятые нами при реализации функций из модуля **parallel_limited_helper**. Начнем с соглашения о том, что ответственность за создание и уничтожение рабочих процессов лежит на вызывающей стороне. С первого взгляда может показаться, что это решение – не из оптимальных: почему нельзя просто передать число рабочих процессов в функцию **parallel_limited_helper:limited_core/4** (вместо списка рабочих процессов)? Для понимания причин давайте поставим более общую задачу: нам необходимо создать распределенные версии функций **map** и **reduce**, т. е. версии функций, рабочие процессы которых выполнялись бы на заранее заданных узлах. При этом мы ограничиваем максимальное число рабочих процессов на каждом

## Я отвечаю за все? А вот и нет!

Термин Single Responsibility Principle (SRP) переводится как «принцип одной ответственности». Это принцип объектно-ориентированного программирования, и он означает, что каждый создаваемый класс должен отвечать за что-то одно и эта ответственность должна быть данным классом полностью инкапсулирована. Заменив термин «класс» на термин «сущность», то есть слегка обобщив (сущностью может быть и функция), этот принцип можно использовать и в функциональном программировании. Пусть для обработки некоторых данных необходимо сделать несколько подготовительных шагов. Если код, выполняющий эти шаги, и код по обработке данных будут находиться в одной функции, то это, очевидно, нарушение SRP-принципа. Наиболее логичным подходом (приводящим к понятному и легкому в поддержке результату) будет разнести все подготовительные шаги и обработку данных по отдельным функциям. После такого разделения также необходимо будет создать функцию, содержащую вызовы – как функций, инкапсулирующих подготовительные шаги, так и функции, которая выполняет обработку данных.

# многозадачности

## Что такое узлы и как их именуют

Узлом называется именованный экземпляр среды выполнения Erlang. Чтобы создать узел, достаточно при запуске среды выполнения Erlang указать ключ **"-sname"** или **"-name"** и имя создаваемого узла. При указании ключа **"-sname"** создастся узел с коротким именем, при указании ключа **"-name"** – узел с длинным именем. Если полное имя компьютера будет **CompName.DomainName**, то при создании узла с коротким именем **NodeName** его полное имя будет **NodeName@CompName**, а при создании узла с длинным именем **NodeName** его полное имя будет **NodeName@CompName.DomainName**. Если при создании узла задать имя в виде **Part1@Part2**, то это имя будет именем узла вне зависимости от типа создаваемого узла и имени компьютера. Функция **node/0** (BIF) позволяет получить имя текущего узла в виде атома. Если среда выполнения Erlang создана не именованной, то функция **node/0** в таком случае вернет атом **nonode@nohost** (этот атом можно использовать как обычное имя узла).

Желая использовать экземпляр среды выполнения Erlang для создания распределенной среды выполнения, мы должны его создать с длинным или коротким именем, т.е. сделать узлом. Отметим, что узлы с разным типом имен (т.е. узлы с длинными и короткими именами) не могут взаимодействовать друг с другом.

узле, что дает в результате ограничение на общее число рабочих процессов. Распределяя рабочие процессы по всем узлам равномерно, мы все же можем их создавать в нашей обобщенной функции (которая аналогична функции **parallel_limited_helper:limited_core/4**), передавая как параметры максимальное число рабочих процессов на каждом узле и список доступных узлов. А если мы хотим создавать на разных узлах разное число рабочих процессов, мы уже должны передавать список пар (кортежей из двух элементов) «узел – максимальное число процессов на узле».

В чем минусы такого подхода? Во-первых, тогда уменьшается количество сценариев использования данной функции. Действительно, при передаче в функцию списка рабочих процессов (которые создала нам вызывающая сторона) нам без разницы, созданы ли эти процессы на одном узле с главным процессом или же нет. Нам также без разницы, ограничено ли время жизни рабочих процессов многозадачной (или распределенной) версией функции **map** и **reduce** или же они являются долгоживущими (например, из некоторого пула процессов). С другой стороны, решив, что процессы должна создавать сама наша функция, мы получим ситуацию, когда, скажем, наша функция может создать рабочие процессы только на локальном узле (или с еще какими-либо ограничениями). Во-вторых, при таком подходе нарушается принцип SRP: функция содержит и реализацию многозадачной обработки списка, и функциональность по созданию рабочих процессов.

Конечно, когда мы использовали «одноразовые» рабочие процессы, их создание в нашей обобщенной функции было оправдано, т.к. было неотъемлемой частью алгоритма, и только наша функция знала об этих процессах. В случае же «многоразовых» рабочих процессов их создание неотъемлемой частью алгоритма не является. К тому же об этих процессах знает внешний код (т.к. он задает ограничения на их количество), и вполне логично, что именно он будет управлять временем жизни этих процессов. Если же нам необходимо, чтобы у нас на разных узлах было разное количество рабочих процессов, то при обсуждаемом подходе мы получим ситуацию, когда код для создания этих рабочих процессов находится как на вызывающей стороне, так и на вызываемой стороне. Действительно, на вызывающей стороне мы будем вычислять для каждого узла максимальное число рабочих процессов, и формировать список пар «узел – максимальное число рабочих процессов на этом узле». А на вызываемой стороне – создавать рабочие процессы в соответствии с переданным списком. Очевидно, что поддерживать и расширять подобную реализацию будет тяжело.

А теперь давайте решим поставленную выше задачу: создадим распределенные версии функций **map** и **reduce**. Как уже говорилось, функция **parallel_limited_helper:limited_core/4** передает ответственность за управление жизнью рабочих процессов вызывающей стороне. От вызывающей стороны функция **parallel_limited_helper:limited_core/4** ожидает список рабочих процессов, которые выполняют функцию **parallel_limited_helper:limited_worker/1** или ей подобную, т.е. с таким же протоколом взаимодействия. Это означает, что для создания распределенных вариантов функций **map** и **reduce** мы можем использовать функцию **parallel_limited_helper:limited_core/4**. Функции **parallel_map:distributed_pmap/5** и **parallel_reduce:distributed_reduce/6** будут реализациями распределенных версий функций **map** и **reduce**. В этих функциях мы создаем рабочие процессы на заданных узлах, используем созданные процессы для распределенной обработки исходного списка (при помощи вызова функции **parallel_limited_helper:limited_core/4**) и завершаем работу созданных рабочих процессов. Функция **parallel_map:distributed_pmap/5** имеет следующий вид:

```
distributed_pmap(_Fun, [], _PortionSize, _NodeList, _
WorkerCount) -> [];
distributed_pmap(Fun, SourceList, PortionSize, _NodeList,
_WorkerCount)
when length(SourceList) =< PortionSize ->
  lists:map(Fun, SourceList);
distributed_pmap(Fun, SourceList, PortionSize, NodeList,
WorkerCount) ->
  WorkerFun = fun(SourcePortion) -> lists:map(Fun,
SourcePortion) end,
  WorkerList = [spawn_link(Node, fun() -> parallel_limited_
helper:limited_worker(WorkerFun) end) || Node <- NodeList, _
WorkerIndex <- lists:seq(1, WorkerCount)],
  Result = parallel_limited_helper:limited_core(fun lists:append/1,
SourceList, PortionSize, WorkerList),
  lists:foldl(fun(Worker, _Aggr) -> exit(Worker, normal) end, true,
WorkerList),
  Result.
```

## Записи в языке Erlang

Записи в языке Erlang не являются отдельным типом данных: это всего лишь «синтаксический сахар» для доступа к полям кортежа по именам (заданным при определении записи), а не по индексам. Записи определяются при помощи директивы **"-record"**. Так, например, директива **-record(somerecord, {f1=[], f2=0})** определяет запись с именем **somerecord** и двумя полями: **f1** со значением по умолчанию **[]** (пустой список) и **f2** со значением по умолчанию **0**. Создать экземпляр записи можно следующим способом: **RecordInstance = #somerecord{f2=666}**. При этом в реальности создастся следующий кортеж: **{somerecord, [], 666}**. Получить значение поля записи по имени можно так: **RecordInstance#somerecord.f2**.

Определение записи, которое должно быть доступно нескольким модулям, выносят в отдельный файл (с расширением **.hrl**) и подключают директивой **"-include"** там, где это нужно.

Как и в предыдущих случаях, функция **parallel_map:distributed_pmap/5** содержит три варианта. Первый вариант предназначен для обработки ситуации, когда исходный список пуст, второй вариант – для обработки ситуации, когда размер исходных данных не превышает размера порции, а третий – для обработки всех остальных случаев. Функция **parallel_reduce:distributed_reduce/6** выглядит следующим образом:

```
distributed_reduce(_Fun, [], {Init, _PortionInit}, _PortionSize, _
NodeList, _WorkerCount) -> InitValue;
distributed_reduce(Fun, SourceList, {Init, _PortionInit},
PortionSize, _NodeList, _WorkerCount) when length(SourceList)
=< PortionSize ->
  lists:foldl(Fun, Init, SourceList);
distributed_reduce(Fun, SourceList, {Init, PortionInit}, PortionSize,
NodeList, WorkerCount) ->
  ReduceFun = fun(List) -> lists:foldl(Fun, Init, List) end,
  PortionReduceFun = fun(List) -> lists:foldl(Fun, PortionInit, List)
end,
  WorkerList = [spawn_link(Node, fun() -> parallel_limited_
helper:limited_worker(PortionReduceFun) end) || Node <-
NodeList, _WorkerIndex <- lists:seq(1, WorkerCount)],
  Result = parallel_limited_helper:limited_core(ReduceFun,
SourceList, PortionSize, WorkerList),
  lists:foldl(fun(Worker, _Aggr) -> exit(Worker, normal) end, true,
WorkerList),
  Result.
```

Функция **parallel_reduce:distributed_reduce/6**, как и функция **parallel_map:distributed_pmap/5**, содержит три варианта для обработки точно таких же ситуаций: пустого исходного списка, исходного списка малого размера (не больше размера порции) и для всех остальных случаев. В этих функциях мы задаем список узлов **NodeList**, на которых могут создаваться рабочие процессы, и максимальное количество рабочих процессов на каждом узле **WorkerCount**; тем самым мы ограничиваем общее количество рабочих процессов.

Внимательный читатель легко заметит, что в качестве параметров мы передаем в функции **parallel_map:distributed_pmap/5** и **parallel_reduce:distributed_reduce/6** список узлов **NodeList** и максимальное количество процессов на каждом узле **WorkerCount**. Может возникнуть вопрос, не противоречит ли такое решение всему вышесказанному. Если мы используем долгоживущие рабочие процессы (например, из некоторого пула процессов), такое решение будет просто неправильным. Во всех остальных случаях мы хотим просто выполнить распределенную операцию **map** (или **reduce**) на определенном наборе узлов, ограничив максимальное количество рабочих процессов на этих узлах. Если и тогда мы будет возлагать ответственность по созданию рабочих процессов на внешнюю сторону, то это не совсем то, что ожидает от нас вызывающая сторона (да и неудобно для вызывающей стороны). Конечно, можно было бы выделить операции по созданию рабочих процессов и завершению их работы в отдельные методы, что слегка повысило бы читаемость. Но это действие мы оставим для тех читателей, кому оно интересно.

Пора проверить, что наши распределенные реализации функций **map** и **reduce** (функции **parallel_map:distributed_pmap/5** и **parallel_reduce:distributed_reduce/6**) работают правильно. Для начала давайте создадим три узла: два узла с именами **slave1** и **slave2** для создания рабочих процессов и узел с именем **master** для главного процесса (и запуска функций **parallel_map:distributed_pmap/5** и **parallel_reduce:distributed_reduce/6** на выполнение). На компьютере автора (при создании узлов с ключом **-sname**) полные имена узлов будут следующими: **slave1@stdstring**, **slave2@stdstring** и **master@stdstring** (именно эти имена узлов мы будем использовать в нашем примере). Все действия в примере мы будем производить на узле **master@stdstring**.

Начнем с функции **parallel_map:distributed_pmap/5**: мы помним, что эта функция имеет три варианта. Вызов **parallel_map:distributed_pmap(fun(Item) -> 3*Item end, [], 2, ['slave1@stdstring', 'slave2@stdstring'], 2)** проверяет первый вариант (когда исходный список данных пуст) и возвращает пустой список, как и ожидается. Вызов **parallel_map:distributed_pmap(fun(Item) -> 3*Item end, [2, 3], 4, ['slave1@stdstring', 'slave2@stdstring'], 2)** возвращает следующий список **[6, 9]**. Так как размер исходного списка меньше размера порции, то мы проверяем второй случай. И, наконец, вызов **parallel_map:distributed_pmap(fun(Item) -> 3*Item end, [2, 3, 5, 6, 8, 1, 7, 2], 2, ['slave1@stdstring', 'slave2@stdstring'], 2)** возвращает следующий список: **[6, 9, 15, 18, 24, 3, 21, 6]**. Очевидно, что этот вызов проверяет третий вариант функции **parallel_map:distributed_pmap/5**, т. к. размер исходного списка больше размера порции. У нас выделено два узла под рабочие процессы, на каждом узле мы создаем по два процесса; в итоге у нас четыре рабочих процесса. Размер исходного списка – 8, размер порции данных для обработки – 2; легко видеть, что список будет разбит на четыре порции, и в результате все четыре рабочих процесса будут загружены.

А теперь проверим работу функции **parallel_reduce:distributed_reduce/6** (эта функция также имеет три варианта). Вызов **parallel_reduce:distributed_reduce(fun(Item, Agg) -> Item + Agg end, [], {1, 0}, 2, ['slave1@stdstring', 'slave2@stdstring'], 2)** проверяет первый вариант (когда исходный список данных пуст) и возвращает 1, т. е. начальное значение операциии свертки. Вызов **parallel_reduce:distributed_reduce(fun(Item, Agg) -> Item + Agg end, [1, 2], {1, 0}, 4, ['slave1@stdstring', 'slave2@stdstring'], 2)** возвращает значение 4. Размер исходного списка – 2, размер порции – 4; это означает, что данный вызов проверяет второй вариант функции **parallel_map:distributed_pmap/5**. И, наконец, вызов **parallel_reduce:distributed_reduce(fun(Item, Agg) -> Item + Agg end, [1, 2, 3, 4, 5, 6, 7, 8], {1, 0}, 2, ['slave1@stdstring', 'slave2@stdstring'], 2)** возвращает значение 37. Этот вызов проверяет третий вариант функции **parallel_map:distributed_pmap/5**, т. к. размер исходного списка больше размера порции. Как и в предыдущем случае, у нас четыре рабочих процесса на двух узлах (по два рабочих процесса

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

на узел). Размер исходного списка – 8, размер порции – 2, так что список будет разбит на четыре порции и все четыре рабочих процесса будут загружены. Проверку других сценариев работы этих функций (например, когда не для всех рабочих процессов будут созданы задания) мы оставляем читателям.

Займемся соглашением раздавать все задачи рабочим процессам сразу, при вызове функции **parallel_limited_helper:limited_core/4**. При вызове этой функции, мы разбиваем исходный список с данными на порции; порции являются списком пар (кортежей из двух элементов) «индекс – часть исходного списка с данными». Очевидно, что размер всех порций данных несколько больше размера исходных данных; действительно, размер всех порций равен размеру исходного списка, плюс размер на индексы всех порций, плюс накладные расходы на создание пары (кортежа из двух элементов) для каждой порции. И если исходный список очень велик, нам просто может не хватить памяти для создания всех порций. Например, на 32-разрядной системе размер адресного пространства процесса (в операционной системе Linux с обычным ядром) – 3 ГБ; если размер исходного списка больше 1,5 ГБ, то создать все порции данных для этого списка, очевидно, не получится. Даже если места для создания всех порций хватит, все равно их создание приводит к одномоментным накладным расходам на выделение памяти, создания объектов, копирования данных и к последующим накладным расходам на сборку мусора, когда все эти объекты станут не нужны. Создав все порции, мы сразу же распределяем их между рабочими процессами, т. е. отправляем рабочим процессам сообщения, содержащие данные для обработки порции для всех порций данных. При этом все порции данных в один момент времени оказываются в сети, вне зависимости от их объема. Понятно, что при большом объеме полученных порций (то есть большом объеме исходных данных) взаимодействие узлов по сети может стать узким местом, тормозящим всю систему (всю нашу многозадачную обработку).

На эту проблему стоит взглянуть несколько с другой стороны: так ли нам надо сразу разбивать все исходные данные на порции и отправлять их по сети рабочим процессам? Очевидно, что такой подход упрощает исходный код наших реализаций; более того, такой подход позволяет во всех наших реализациях (созданных до этого момента) использовать одну и ту же функцию для сборки результатов обработки: **parallel_common:collect_result/2**. Мы решили, что отказываемся от подхода, когда все делается сразу, и предпочли подход, когда порции формируются и отсылаются рабочему процессу по мере необходимости. Это решение подводит нас к тому, что мы не можем отделить стадию формирования и отсылки заданий рабочим процессам от стадии сбора результата их работы (как это было у нас раньше). Поэтому наше взаимодействие с рабочими процессами будет выглядеть следующим образом.

Мы получаем результат обработки какой-либо порции данных от какого-то рабочего процесса, сохраняем этот результат (в массиве, как мы это делали и раньше), после чего «отщипываем» от оставшихся необработанных данных порцию, формируем новое задание на обработку и отсылаем это задание рабочему процессу, с которым начали взаимодействие. И так до тех пор, пока мы не обработаем весь список исходных данных (пока список оставшихся необработанных данных не опустеет) и не получим все результаты обработки (понятно, что список оставшихся необработанных данных опустеет раньше, чем мы действительно обработаем все данные). Ну и, естественно, работу мы должны начать с того, чтобы каждому рабочему процессу раздать по заданию. Понятно, что такой подход несколько усложняет реализацию по сравнению с реализацией, в которой мы сразу разбиваем все данные на порции и отсылаем эти данные рабочим процессам.

А теперь давайте реализуем новый вариант многозадачных функций **map** и **reduce** на основе всего вышесказанного. Как и раньше, мы выделяем общую функциональность (на основе которой мы сможем реализовать многозадачные версии функций **map** и **reduce**) и помещаем ее в отдельный модуль; в нашем случае это будет модуль **parallel_smartmsg_helper**.

Чтобы получить более понятную и гибкую реализацию, введем несколько определений записей:

```
-record(tasks_descr, {created = 0, processed = 0, rest = []}).
-record(task_request, {master, index, portion}).
-record(task_result, {worker, index, result}).
```

Экземпляр записи **task_descr** хранит данные о процессе обработки исходного списка; поле **created** содержит количество созданных заданий на обработку; поле **processed** содержит количество заданий на обработку, выполнение которых закончилось; поле **rest** содержит необработанный остаток исходного списка. Экземпляр записи **task_request** содержит данные запроса на обработку (посылаемый главным процессом одному из рабочих процессов); поле **master** содержит идентификатор главного процесса; поле **index** содержит индекс порции исходных данных; поле **portion** содержит саму порцию исходных данных. Следует заметить, что вместо передачи идентификатора главного процесса в сообщении его можно было бы передать рабочему процессу одним из параметров функции, которую этот рабочий процесс выполняет. И, наконец, экземпляр записи **task_result** содержит данные с результатами обработки порции (посылаемые одним из рабочих процессов главному); поле **worker** содержит идентификатор рабочего процесса; поле **index** содержит индекс исходной порции данных; поле **result** содержит результат обработки этой исходной порции данных.

Следующий шаг, который мы сделаем в рамках нашей реализации – создадим функцию, которую должны выполнять рабочие процессы. Так как мы создаем рабочие процессы снаружи нашей обобщенной функции обработки списка данных (аналоге функции **parallel_limited_helper:limited_core/4**), на основе которой мы потом создадим очередные версии функций **map** и **reduce**, то очевидно, что эта функция должна быть экспортируемой. В нашей реализации это будет функция **parallel_smartmsg_helper:smartmsg_worker/1**:

```
smartmsg_worker(Fun) ->
  receive
    #task_request{master=MasterPid, index=Index,
portion=SourcePortion} ->
      Dest = Fun(SourcePortion),
      MasterPid ! #task_result{worker=self(), index=Index,
result=Dest},
      smartmsg_worker(Fun);
  _Other -> smartmsg_worker(Fun)
end.
```

В функции, которую выполняют рабочие процессы, мы обрабатываем два типа сообщений: во-первых, сообщения, являющиеся экземпляром записи **task_request** – это задания на обработку порции данных; во-вторых, все остальные сообщения, чтобы в очереди сообщений рабочего процесса не накапливались необработанные неизвестные сообщения. Следует сказать, что функция **parallel_smartmsg_helper:smartmsg_worker/1** практически идентична функции **parallel_limited_helper:limited_worker/1**, которую создали в предыдущей реализации. Мы не стали использовать в нашей новой реализации функцию **parallel_limited_helper:limited_worker/1**, чтобы не смешивать использование нескольких модулей и не смущать читателя этим.

На этом практикуме мы создали распределенные версии функций **map** и **reduce** и протестировали их. Мы также разобрались, что создание всех порций исходных данных и распределение их среди рабочих потоков в один момент времени – не лучшая идея, и начали реализовывать более сложный подход, в котором мы формируем задачи и отсылаем их рабочим процессам по мере необходимости. В следующий раз мы продолжим эту работу. LXF

# Введение в MPI

Параллельно поздоровавшись с миром, **Михаил Остапкевич** и **Евгений Балдин** призадумались о Жизни с большой буквы.

Наш эксперт

**Михаил Остапкевич**
Романтик, очарованный компьютерами и создаваемыми в них идеальными мирами; верит, что сложнейшие новые технологии могут и должны служить во благо человечеству.

Наш эксперт

**Евгений Балдин**
Физик, который действительно знает, что такое нехватка вычислительных ресурсов.

Времена больших векторных суперкомпьютеров прошли, ну или пока не настали. Что мы имеем взамен? Множество независимых «писишек», даже если они и установлены в стойки и управляются квалифицированными администраторами! Запустить на них пачку задач легко, но как подотчетные процессы будут общаться? Да с помощью MPI!

MPI или Message Passing Interface или, по-простому, интерфейс передачи сообщений – это стандартный программный интерфейс для передачи информации между процессами, выполняющими одну задачу. Стандарт поддерживается консорциумом MPI Forum, членами которого являются практически все крупные производители вычислительной техники и программного обеспечения. Первый стандарт MPI 1.0 принят в 1994 году. Формально разработка началась в 1991 году, когда небольшая группа ученых-информатиков собралась в уединенном австрийском горном пансионате и начала активно обмениваться мнениями. В сентябре 2012 года вышла спецификация MPI 3.0.

Технология MPI ориентирована в первую очередь на кластеры и на массивно-параллельные системы (MPP), но применяется также в системах SMP и NUMA. Это означает, что MPI-программы можно оптимизировать как для самых мощных решений из TOP500, так и для домашнего компьютера. Почти на каждом современном десктопе сейчас стоит более одного вычислительного ядра. Да что десктопы – телефоны тоже не отстают от этой моды. И эти ядра просто необходимо использовать!

Исполнение программы производится несколькими параллельно исполняемыми процессами, которые представляют собой разные экземпляры одной и той же программы. Каждый процесс имеет свой набор данных. Режим работы, когда одна и та же программа запущена на разных узлах или процессорах и обрабатывает разные наборы данных, называют SPMD (Single Program Multiple Data). Данные других процессов процессу не видны, поэтому для обмена данными между процессами требуется MPI, а именно посылка сообщений между ними. Механизмы MPI гарантируют доставку сообщений и соответствие порядка, в котором они посылались, порядку их приема. Существуют и другие библиотеки для обмена сообщениями между удаленными процессами (например, PVM). Однако среди всех библиотек, решающих похожие задачи, именно MPI получил наибольшее распространение. С технологической стороны, по-видимому, три причины сыграли в этом решающую роль:

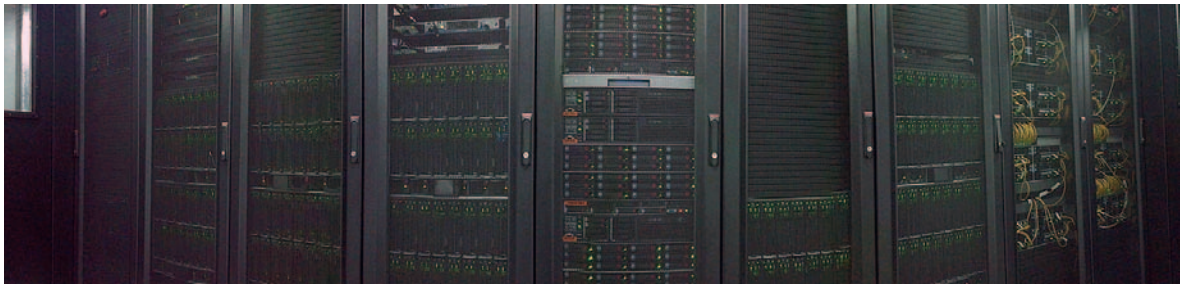
- высокая эффективность программ на MPI;
- высокая переносимость программ, написанных на MPI;
- наличие свободной реализации.

Эффективность является одним из важнейших свойств программ. Важнее только реализация необходимых функций и надежность работы и воспроизводимость результатов. В параллельном программировании, где программы отличаются большим объемом вычислений, эффективность приобретает особый смысл. Одна из самых главных причин, для чего создают параллельные программы – это уменьшение времени их работы, а оно напрямую зависит от эффективности.

Эффективность в MPI достигается в первую очередь за счет использования самого быстрого из доступных средств доставки сообщений. В мультипроцессоре используются окна разделяемой несколькими процессами памяти. В мультикомпьютерах используется самая быстрая сеть (Infiniband, Myrinet). Если она недоступна, то используется старый, добрый и надежный TCP/IP.

Обычный «сферический и в вакууме» прикладной программист, как правило, достаточно хорошо изолирован от специфики аппаратной платформы, на которой он пишет свои программы. Временами эта изолированность выходит боком, но такова жизнь. Прослойка из системного программного обеспечения и языков программирования высокого уровня позволяет ему игнорировать отличия между платформами. Следование общепринятым стандартам позволяет легко переносить программы с одной аппаратной платформы на другую.

В параллельном программировании языки и интерфейсы уже не скрывают новые, специфические для параллельных ЭВМ свойства архитектуры, которые связаны с координацией работы параллельно исполняющихся частей программы и обменом данных между ними. В результате получаются параллельные программы, ориентированные на конкретные машины. С одной стороны, свойства MPI позволяют строить его эффективные реализации на широком спектре параллельных ЭВМ. С другой стороны, набор операций удобен для написания самых разнообразных параллельных программ. В итоге, MPI стал самым распространенным стандартом для написания параллельных программ, для которого построены реализации на очень большом числе параллельных архитектур. Все это привело к тому, что параллельная программа, написанная с использованием MPI, обладает высокой переносимостью.



**Гибридный кластер Сибирского суперкомпьютерного центра — типичный вычислительный центр.**

Кроме переносимости и эффективности, другим существенным свойством MPI является языковая нейтральность. Реализации библиотеки MPI не привязаны к какому-либо одному языку. Вместо этого они описывают семантику операций, доступных MPI-программам через функциональный интерфейс. Как правило, MPI-программы пишут на C, Fortran или C++, но можно использовать и многие другие языки, например, C#, Java, Python или R.

Наличие свободной реализации позволяет легко установить MPI и начать обучать свои программы секретам взаимодействия:

```
> sudo aptitude install mpi-default-bin mpi-default-dev
```

Документация для вдумчивого изучения тоже не помешает:

```
> sudo aptitude install mpi-doc mpi-specs
```

*mpi-doc*, кроме html-справочника, добавляет в систему man-странички по многочисленным MPI-функциям, очень удобные для быстрого подглядывания.

## Здравствуй, Мир!

Предполагается, что вы представляете, что такое программирование на C, ну или хотя бы написали на этом языке свой личный HelloWorld. Вот так выглядит простейшая MPI-программа:

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char **argv) {
  MPI_Init(&argc, &argv);
  printf("Здравствуй, Мир!\n");
  MPI_Finalize();
  return 0;
}
```

В отличие от классического HelloWorld, для его MPI-реализации необходимо подключить **mpi.h** с определениями констант, типов и функций MPI. Также в коде появились две спецфункции:

- MPI_Init – инициализация MPI;
- MPI_Finalize – обязательное корректное завершение работы с MPI.

В этой простой программе для лаконичности мы не обрабатываем возвращаемые ошибки. В реальной же программе ошибки игнорировать не следует. После успешного вызова MPI_Init (когда возвращается значение MPI_SUCCESS) инфраструктура MPI приложения готова к работе, и мы можем пользоваться всеми интерфейсными функциями MPI.

Компиляция MPI-программы делается с помощью стандартизованной утилиты *mpicc*, которая входит в состав пакета *mpi-default-dev*. Программа *mpicc* представляет собой обертку поверх компилятора. Она скрывает от пользователя отличия между разными платформами. Компиляция MPI-модификации HelloWorld делается так:

```
> mpicc helloworld.c -o helloworld
```

Если эту программу теперь запустить, то она сделает то, что от нее ожидается – а именно, скажет

```
> ./helloworld
Здравствуй, Мир!
```

Нам же нужно больше, поэтому, воспользовавшись утилитой *mpirun* из пакета *mpi-default-bin*, выполним:

```
> mpirun -n 2 ./helloworld
Здравствуй, Мир!
Здравствуй, Мир!
```

Ключик **-n** задает число параллельно исполняемых процессов. В нашем случае с миром поздоровались две копии программы. Также вместо *mpirun* можно использовать *mpiexec* или *orterun*. Это синонимы.

Немного усложним учебный код:

```
#include <mpi.h>
#include <stdio.h>
main(int argc, char **argv){
  int rankNode, sizeCluster;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rankNode);
  MPI_Comm_size(MPI_COMM_WORLD, &sizeCluster);
  printf("Здравствуй, Мир! от процесса %d из %d\n", rankNode, sizeCluster);
  MPI_Finalize();
}
```

Функция MPI_Comm_size возвращает количество запущенных для исполнения MPI-программы процессов, функция MPI_Comm_rank возвращает номер процесса, вызвавшего функцию:

```
> mpicc helloworld2.c -o helloworld2
> mpirun -n 2 ./helloworld2
Здравствуй, Мир! от процесса 1 из 2
Здравствуй, Мир! от процесса 0 из 2
```

Если запустить больше процессов –

```
> mpirun -n 7 ./helloworld2
Здравствуй, Мир! от процесса 0 из 7
Здравствуй, Мир! от процесса 1 из 7
Здравствуй, Мир! от процесса 2 из 7
Здравствуй, Мир! от процесса 5 из 7
Здравствуй, Мир! от процесса 6 из 7
Здравствуй, Мир! от процесса 3 из 7
Здравствуй, Мир! от процесса 4 из 7
```

можно увидеть, что выполняются они не обязательно в порядке запуска (последним выполнился процесс 4, а не процесс 6).

Это проявление весьма неудобного свойства параллельных программ, а именно недетерминированного результата исполнения. Оно может приводить к редко воспроизводимым и трудно отлаживаемым ошибкам в параллельных программах. Возникает ситуация, подобная той, когда измерение физических величин влияет на состояние измеряемой системы. Отладочный код может повлиять на ход исполнения программы, и искомая ошибка перестанет проявляться.

В тех случаях, когда нужно установить некоторый четкий порядок, можно использовать операции MPI для синхронизации процессов. Платой за это является увеличение времени исполнения программ, так как при запуске операции синхронизации часть процессов попадает в состояние ожидания ее завершения.

Следует также учитывать, что, например, на процессоре Intel(R) Core(TM)2 Duo, как следует из названия, только два вычислительных ядра, поэтому довольно бессмысленно запускать больше двух процессов одновременно. Можно также потребовать, чтобы каждый процесс привязался к своему ядру:

```
> mpiexec -n 2 -bind-to-core -report-bindings ./helloworld2
fork binding child [[7067,1],0] to cpus 0001
fork binding child [[7067,1],1] to cpus 0002
Здравствуй, Мир! от процесса 1 из 2
```

```
Здравствуй, Мир! от процесса 0 из 2
> mpiexec -n 7 -bind-to-core -report-bindings ./helloworld2
Not enough processors were found on the local host to meet the
requested binding action
```

## Игра «Жизнь»

Клеточный автомат «Жизнь [Game of Life, или просто Life]» был предложен английским математиком Джоном Конвеем [John Horton Conway] в 1979 году. Вряд ли он стал бы так широко известен, если бы не американский математик и исключительно энергичный популяризатор науки Мартин Гарднер [Martin Gardner]. По игре «Жизнь» есть масса популярной литературы, в том числе и на русском языке.

Игра «Жизнь» проводится на бесконечной клеточной доске, где каждой из клеток присвоена 1 (живая клетка) или 0 (ничего нет). При переходе из нулевого в единичное состояние говорят, что клетка рождается, а из единичного в нулевое – умирает. Новое значение клетки определяется ее прежним значением и суммой значений ее восьми соседей. Клетка рождается при сумме три, а умирает при сумме менее двух – от одиночества и более трех – от перенаселенности. При всех иных конфигурациях значений и сумм клетка сохраняет свое состояние. Несмотря на простоту формулировки, эволюции разных комбинаций живых клеток могут весьма отличаться, и многие из них имеют свои имена. Например, на рисунке на следующей странице изображено глайдерное ружье – пример бесконечно растущей колонии.

Состояния всех клеток вместе образуют состояние всего клеточного автомата. Состояние клеточного автомата меняется по тактам, в дискретном времени. При смене состояния клеточного автомата состояния всех образующих его клеток меняются одновременно. Для каждой клетки новое состояние вычисляется по одинаковому для всех клеток правилу. Входные данные для правила – состояния самой клетки и ее восьми соседей на i-м такте. Результатом будет состояние этой клетки на (i+1)-м такте. Подобные алгоритмы замечательно параллелятся. Также интересна их особенность, состоящая в том, что время на одну итерацию не зависит от состояния автомата.

Следует понимать, что клеточный автомат – это абстрактная модель. В ней предполагается бесконечное число клеток. Реальные компьютеры обладают конечными ресурсами памяти и времени. Поэтому при моделировании клеточного автомата клетки размещаются в массиве конечных размеров. Для всех клеток, не лежащих на границах массива, работают правила моделируемого клеточного автомата без каких-либо изменений. Обработка же граничных клеток производится несколько иначе. Два наиболее распространенных способа обработки – замыкание граней (из прямоугольного массива получается трехмерный тор) и использование нулевых значений для тех соседних клеток из окрестности клетки на границе, которые лежат за пределами массива.

Последовательная программа для «Жизни» (без процедур задания начальных значений и печати или сохранения результата) выглядит примерно следующим образом:

```
#define QTY_STEPS  8192
#define SIZE_ARRAY 1024
char ar[2][SIZE_ARRAY][SIZE_ARRAY];
void computeCell(unsigned dir, unsigned x, unsigned y){
  unsigned sum;
  sum = ar[dir][y-1][x-1]+ar[dir][y-1][x]+ar[dir][y-1][x+1]+
    ar[dir][y][x-1]+ar[dir][y][x+1]+
    ar[dir][y+1][x-1]+ar[dir][y+1][x]+d ar[dir][y+1][x+1];
  if((sum<2)||(sum>3))
    ar[1-dir][y][x]=0;
  else
    ar[1-dir][y][x] = (sum==3) ? 1: ar[dir][y][x];
}
void simulateStep(unsigned layer){
  unsigned x,y;
  for(y=1;y<SIZE_ARRAY-1;y++)
  for(x=1;x<SIZE_ARRAY-1;x++)
    computeCell(layer,x,y);
}
void simulate(unsigned stepQty){
  unsigned step;
  for(step=0;step<stepQty;step++){
    simulateStep(step & 1);
  }
}
main(){
  simulate(QTY_STEPS);
}
```

## Классификация параллельных компьютеров

Современные параллельные компьютеры можно разделить на: параллельные векторные системы (PVP), симметричные мультипроцессорные системы (SMP), массивно-параллельные системы (MPP), системы с неоднородным доступом к памяти (NUMA) и кластеры.

Примером параллельной векторной системы (PVP) является первый суперкомпьютер CRAY-I, который был создан в 1975 году. Одно время понятия «параллельные векторные системы» и «суперкомпьютеры» были синонимами, но сейчас в TOP500 (http://www.top500.org/) классических PVP не осталось совсем.

Симметричные мультипроцессорные системы (SMP) содержат несколько одинаковых процессоров и общую память. Все процессоры имеют одинаковую скорость доступа к общей памяти. Имея последовательную реализацию некоторой программы, как правило, можно относительно легко построить ее параллельную реализацию для SMP. Основной недостаток SMP – плохая масштабируемость. Существуют доступные SMP-системы с десятками процессоров, но в списке TOP500 нет ни одной из них.

Компьютеры класса MPP (массивно-параллельные системы) состоят из узлов, объединенных высокоскоростным коммутатором. Узел содержит один или несколько процессоров и память. Узел не имеет доступа к памяти других узлов. Такие системы, в отличие от SMP, хорошо масштабируются. Число узлов в самых больших MPP-системах достигает сотен тысяч. За это приходится платить, так как параллельную программу для MPP-системы, в отличие от SMP, построить сложнее. Процессы, выполняющиеся на разных узлах, не имеют общей памяти, и все взаимодействие между ними делается с помощью посылки сообщений. На ноябрь 2012 г. в TOP500 находится 89 классических MPP-суперкомпьютеров, включая первые две позиции.

Системы с неоднородным доступом к памяти (NUMA) являются гибридом SMP и MPP. Физически вся память в NUMA распределена между узлами, как в MPP. Но при этом поддерживается единое пространство памяти, как в SMP. То есть из узла можно обратиться к памяти в любом другом узле. Время доступа к своей, локальной памяти меньше, чем к памяти другого узла.

Кластер – это дешевый аналог MPP-системы. В нем вместо скоростного коммутатора используется обычая сеть; таковые становятся все быстрее и быстрее. Если MPP пока еще выделяются мощью, то кластеры берут массовостью, ибо 411 суперкомпьютеров, остающиеся после вычитания MPP из TOP500, являются кластерными решениями.

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

Функция ComputeCell вычисляет новое состояние одной клетки. Функция SimulateStep вычисляет новое состояние всего клеточного автомата, а Simulate имитирует работу клеточного автомата указанное число шагов.

Обратите внимание, что в массиве ar введено 2 слоя. Это требуется для того, чтобы имитировать одновременность задания новых значений клеток. Чтобы исключить копирование одного слоя в другой, на четном шаге текущие значения клеток берутся из нулевого слоя, новые пишутся в первый; а на нечетных шагах, наоборот, текущие значения берутся из первого слоя, новые пишутся в нулевой.

Для MPI-версии программы возьмем за основу последовательную реализацию. Для простоты ограничимся двумя процессами. Разделим массив, хранящий состояния клеток, поровну.

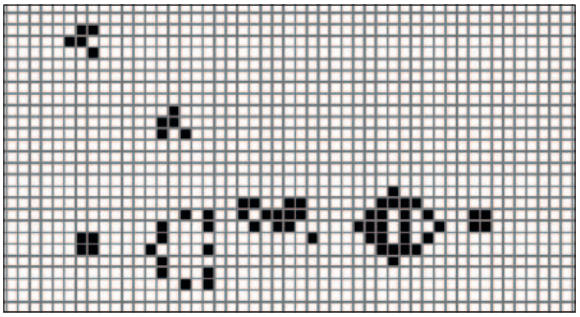
Разделение произведем по строкам. Пусть первая половина строк хранится в массиве ar первого процесса (rankNode равен нулю), вторая – в этом же массиве ar у второго процесса (rankNode равен 1).

При подсчете новых состояний клеток последней строки в нулевом процессе требуются текущие состояния трех клеток из строки снизу. Значения клеток этой строки подсчитываются в первом процессе. Аналогично, для первого процесса нужна строка сверху. Этой информацией о состоянии клеток на границе необходимо обменяться. Это единственное отличии MPI-программы от обычной последовательной. При росте числа одновременно выполняемых процессов также растет и число граничных элементов.

Строки с клетками, подсчитываемыми в одном процессе, но используемыми и в каком-либо другом процессе, называются теневыми гранями.

Функция ComputeCell не меняется, так что ниже мы ее не дублируем:

```
#define QTY_STEPS  8192
#define SIZE_ARRAY 1024
#define SIZE_HALFARRAY (SIZE_ARRAY/2+1)
char ar[2][SIZE_HALFARRAY][SIZE_ARRAY];
#define RES_OK         0
#define RES_ERROR      1
int mytag=99;
void simulateStep(unsigned layer){
  unsigned x,y;
  for(y=1;y<SIZE_HALFARRAY-1;y++)
  for(x=1;x<SIZE_ARRAY-1;x++)
  computeCell(layer,x,y);
}
void simulate(unsigned stepQty,int rankNode){
  unsigned step;
  MPI_Status status;
  for(step=0;step<stepQty;step++){
    simulateStep(step & 1);
    if(rankNode==0){
      MPI_Send(ar[1 - step & 1][SIZE_ARRAY/2 - 1], SIZE_ARRAY,
MPI_CHAR, rankNode + 1,  mytag, MPI_COMM_WORLD);
    MPI_Recv(ar[1 - step & 1][SIZE_ARRAY/2], SIZE_ARRAY, MPI_
CHAR, rankNode + 1, mytag,MPI_COMM_WORLD, &status);
    }
    else if(rankNode==1){
      MPI_Recv(ar[1 - step & 1][0], SIZE_ARRAY, MPI_CHAR,
rankNode - 1, mytag, MPI_COMM_WORLD, &status);
      MPI_Send(ar[1 - step & 1][1], SIZE_ARRAY, MPI_CHAR,
rankNode - 1, mytag, MPI_COMM_WORLD);
    }
  }
}
main(int argc, char **argv){
```



› **Глайдерное ружье.**

```
  int rankNode, sizeCluster;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rankNode);
  MPI_Comm_size(MPI_COMM_WORLD, &sizeCluster);
  if (sizeCluster!=2) {
    printf ("Только для двух ядер!");
    exit(1);
  }
  simulate(QTY_STEPS, rankNode);
  MPI_Finalize();
}
```

А вот оператор цикла в SimulateStep уже отличается от того, что был в последовательной версии, как и размер массива ar. Функция main уже содержит в себе команды инициализации и завершения MPI.

Наиболее сильным изменениям подверглась функция simulate. Теперь после каждого шага требуется передавать значения клеток всех теневых граней между соседними узлами.

Эту задачу выполняют операторы передачи MPI_Send и приема MPI_Recv. Они должны быть комплементарными, то есть каждый оператор передачи данных в одном процессе обязательно обязан иметь соответствующий ему оператор приема в другом процессе. В противном случае произойдет ненормальное завершение работы MPI программы. Подробности об использовании этих команд и описание их аргументов можно посмотреть с помощью команды man.

Собственно говоря, все. Кажется, ничего сложного, но если хочется увеличить число параллельных процессов, то придется аккуратно обрабатывать больше границ и следить за согласованием приема-передачи информации, то есть придется думать. Это и есть основная сложность! Но зато MPI-программа на двух процессах выполняется примерно за 110 секунд вместо 180 на сферическом в вакууме Intel(R) Core(TM)2 Duo одновременно с набором этого текста. Ускорение не в двойку, но достаточно значительное, чтобы за него побороться.

На примере программы моделирования игры «Жизнь» мы рассмотрели самые базовые возможности MPI. Из сотен функций, описанных в текущем стандарте MPI 3.0, мы использовали только шесть. Стандарт ориентирован на комфортную работу при распараллеливании широкого спектра задач, и для этого в нем присутствуют разнообразные группы функций для коммуникаций между двумя процессами, коммуникаций внутри группы процессов, работы с группами процессов, кэширования данных и многого другого. LXF

## Обратная связь

Приглашаем высказаться потенциальных авторов статей по параллельным вычислениям – ценные предложения, критику и советы присылайте по электронной почте: E.M.Baldin@inp.nsk.su, ostap@ssd.sscc.ru.

# Жить красиво без

Отложив *Compiz* в сторонку, **Павел Сёмин** размышляет, как украсить систему другими способами.

Наш эксперт

**Павел Сёмин**
На протяжении четырех лет Linux для него остается неисчерпаемым источником исследовательского вдохновения.

Разработчики по всему миру прилагают немалые усилия для того, чтобы эффектами *Compiz* любовались на как можно большем числе компьютеров. Однако остается немало причин, по которым пользователи вынуждены искать замену этому композитному оконному менеджеру.

Первая из них – проблемы с совместимостью. *Compiz* всегда конфликтует с самостоятельными оконными менеджерами вроде *Openbox*, *Fluxbox*, *Enlightenment*. В Gnome Shell и Cinnamon он вторгается как слон в посудную лавку, снося панель и меню. В *Xfce* и *LXDE* он не дружит с темами оформления, так что потребуется подбирать новые.

За несовместимостью в перечне проблем следует нехватка системных ресурсов. А некоторые пользователи всерьез боятся бритвы Оккама – установка *Compiz* иногда идет вразрез с идеалом максимальной простоты, который так дорог многим.

Остается искать альтернативные способы придания Рабочему столу Linux желаемой эффектности.

## Способ 1. Выжать максимум из своей рабочей среды

Известно, что в глубинах многих популярных графических окружений скрыта значительная композитная мощь. Поэтому одно очень простое решение проблемы приходит на ум само собой: естественно, попытаться пробудить эти дремлющие силы! Не потребуется ничего устанавливать – достаточно нескольких движений мыши, и система предстанет перед нами во всей красе.

Повезло, очень повезло любителям KDE. Графические возможности *Kwin*, оконного менеджера этой среды, начиная с версии 3.3 растут не переставая, и в настоящее время они поистине огромны. Не будь *Kwin* так сильно привязан к родному окружению, он, наверное, стал бы серьезным конкурентом *Compiz*.

Управление композитностью осуществляется в категории Эффекты Рабочего стола [Desktop Effects] Центра настроек KDE [System Settings]. Некоторые параметры, вроде теней, изменяются через диалог Настройка декораций [Configure Decoration] категории Внешний вид рабочего окружения [Workspace Appearance]. Писать подробное руководство не будем: интерфейс простой, любая опция включается одним щелчком мыши и подробно пояснена. При этом функциональность тоже не пострадала: для половины модулей доступны инструменты тонкой настройки. Эффекты, как уже говорилось выше, на любой вкус. Среди них отметим, например, знакомые пользователям *Compiz* «Вязкие окна», анимацию в виде джинна и, конечно, куб рабочего стола.

> «Несколько движений мыши, и система предстанет во всей красе.»

Gnome, исторический соперник KDE, в третьей версии поставляется с новым оконным менеджером *Mutter*, способным создавать немало эффектов. Вот только многие из них не вписались в концепцию планшетообразного интерфейса Gnome Shell. Из собственно оконных украшений – только тени. Заметим, что такой подход не сделал оболочку уродливой – наоборот, эффекты подобраны с умом, и оформление в целом стильное, модное. Но красота скорее планшетная, чем традиционная компьютерная. Настроек нет.

В основанной на Gnome 3 оболочке Cinnamon гораздо полнее раскрыт потенциал *Mutter* (здесь он называется *Muffin*, но суть та же). Окна вокруг пользователя проявляются из пустоты, гаснут и изящно скользят по Рабочему столу. Изменить параметры анимации можно через Настройки Cinnamon в группе Эффекты. Прозрачность, увы, коснулась только главного меню, тени нарисованы раз и навсегда. Зато разработчики сделали инструмент Обзор – красивый и функциональный переключатель окон и рабочих столов. Предварительно активируйте его в окне параметров Cinnamon. Обзор позволяет добавлять, убирать и переименовывать рабочие места, сохраняя сделанные изменения, перемещать окна и закрывать их простым перетаскиванием на значок корзины, расположенный внизу по центру. Похожий компонент Expo в *Compiz* умеет куда меньше. Некоторые эффекты, например, трехмерный переключатель окон, добавляются с помощью дополнений. Чтобы установить их, выберите в окне настроек пункт Расширения, щелкните по ссылке «Получить новые расширения» – откроется web-страница загрузки. Распакуйте скачанные архивы в папку **~./local/share/cinnamon/extensions** и активируйте модули в уже упомянутом пункте настроек. Иногда потребуются дополнительные настройки, так что не брезгуйте инструкцией к расширению. В целом графические возможности Cinnamon заслуживают высокой оценки.

О Unity, также являющейся веткой Gnome 3, говорить нет смысла, потому что в ней оконный менеджер – *Compiz*. А вот старый добрый Gnome 2 ни в коем случае нельзя оставить без внимания. Он еще не канул в Лету, к тому же некоторые пользователи перешли на MATE – вилку «второгнома» и его почти полный аналог.

Поэтому давайте вспомним, какие эффекты мы видели в этой среде. Никаких? А они есть! Композитность присутствует в *Metacity* версии 2.22 и выше, вот только по умолчанию она отключена и не настраивается графическим путем. Эффектами, прямо скажем, не балуют, их всего три: прозрачность, тени и предпросмотр при переключении окон по Alt + Tab. Чтобы заработала композитность, откройте в Редакторе конфигурации ветку /apps/metacity/general/ и поставьте галочку у пункта “compositing_manager”. Или же просто наберите команду

```
gconftool-2 -s --type bool /apps/metacity/general/compositing_manager true
```

Собственные возможности *Xfwm* (*Xfce*) в части украшений тоже не так широки, как хотелось бы. Композитность запускается первой галочкой на вкладке Эффекты категории настроек Диспетчер окон (дополнительно). Далее тремя щелчками мыши включаются тени под окнами разных типов. Перемещая ползунки, можно сделать прозрачными всплывающие, неактивные, перемещаемые и масштабируемые окна, а заодно их заголовки. Вот, пожалуй,

# Compiz

## Композитный оконный менеджер

Что подразумевается под загадочной «композитностью»?

На одной из страниц руководства Ubuntu Linux дан лаконичный ответ: «прозрачность и кубик». В действительности понятие «композитность» характеризует, в первую очередь, технические особенности отрисовки графического интерфейса. Композитный оконный менеджер сводит все потоки графической информации от приложений в промежуточный буфер, где собранные данные могут быть обработаны перед записью в видеопамять. Обработка расширяет возможности по управлению интерфейсом, а иногда и снижает производительность приложений, особенно связанных с графикой. Но с бытовой точки зрения шутливое определение тоже можно считать правильным, потому что для пользователей основная черта композитных оконных менеджеров – это именно предоставляемые ими разнообразные графические эффекты.

Композитный принцип применяется не только в Linux. В операционной системе от Microsoft, начиная с Windows Vista, за оформление окон отвечает композитный DWM (Desktop Window Manager, Диспетчер рабочего стола), реализующий знакомые почти всем эффекты Aero. Для Mac OS X создан Quartz Compositor. *Compiz* – наверное, лучший композитный менеджер окон для UNIX-подобных систем, и уж точно самый известный.

Wayland, новый протокол для организации графического сервера, который медленно, но верно набирает силы перед схваткой с X.org, значительно упрощает обработку графики. Поэтому переход на Wayland может стать хорошим стимулом для ускоренного развития композитных менеджеров, что кардинально улучшит графические интерфейсы Linux и упростит их создание.

и все. Ни трехмерных оконных переключателей, ни анимаций здесь не найдешь.

Наконец, *LXDE* вообще не имеет ничего общего с усладой для глаз: эффекты в *Openbox* полностью отсутствуют. Сходная ситуация сложилась с почти всеми другими легковесными рабочими окружениями и оконными менеджерами, например: *IceWM*, *AWM*, *Fluxbox*. В таких случаях без сторонних программ не обойтись.

Значит, мы подошли ко второму методу украшения системы.

## Способ 2. Применить добавочные композитные менеджеры

Предположим, что нас в целом устраивает оконный менеджер вроде *Openbox*, но душа просит пару несложных эффектов. Тогда наиболее логично первым делом попытаться найти вспомогательную программу, которая, не заменяя существующий диспетчер окон, превратит его в композитный и, соответственно, обеспечит желаемые эффекты. Таких программ создано несколько. Назовем их добавочными композитными менеджерами, или просто композитными менеджерами, если дословно переводить термин с английского.

### Transset

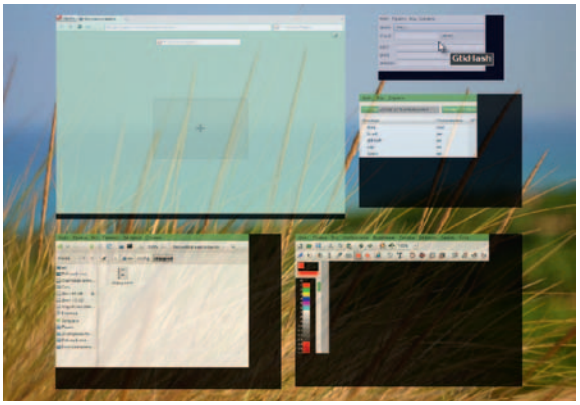
Эта крошечная программа, которую и композитным менеджером не назовешь, предназначена для изменения прозрачности окон. Устанавливается из основного репозитория. В работе утилита напоминает пушку. Сначала подходим к артиллерийской батарее – открываем консоль и пишем **transset**; далее выбираем орудие нужного калибра, то есть указываем необходимую прозрачность числом от 0,1 (почти прозрачно) до 1 (непрозрачно). Значение «по умолчанию» (если просто ввести **transset** без параметра) – 0,75. «Заряжаем пушку» нажатием Enter, прицеливаемся в нужное окно курсором и стреляем щелчком левой кнопки мыши. Через доли секунды окно станет полупрозрачным. Чтобы снять эффект, выбираем «снаряд полной видимости» (вводим **transset 1**) и пускаем его в полупрозрачное окно.

При частом использовании такое ручное управление неудобно. Можно «заряжать пушку» проще: свяжите команды **transset N** (N – любой уровень прозрачности) и **transset 1** с какими-либо комбинациями клавиш.

### Skippy-xd

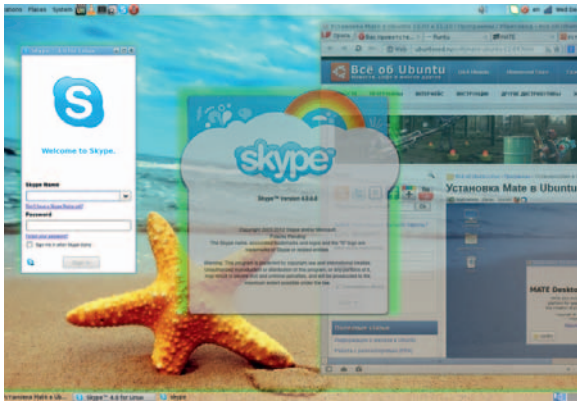
Это композитное дополнение показывает на экране миниатюры открытых окон. Его создавали как аналог инструмента *Expose* из Mac OS X. Пакеты в формате deb загружаем с официальной страницы программы: http://code.google.com/p/skippy-xd/downloads/list. После установки, чтобы программа запускалась без ошибок, необходимо проделать следующее. Открываем Deb-пакет обычным архиватором, ищем внутри файл **skippy-xd.rc-default**, распаковываем его. Далее находим в домашнем каталоге папку **.config**, создаем внутри нее директорию **/skippy-xd** и копируем туда только что извлеченный файл, который переименовываем в **skippy-xd.rc**. Теперь можно запустить программу командой **skippy-xd**, и если все работает хорошо, связать ее с комбинацией «горячих» клавиш, например, Win+W.

Если на заднем плане будут появляться полупрозрачные «призраки» открытых окон, мешающие разглядеть эскизы, то в файле **skippy-xd.rc** поменяйте значение useNetWMFullscreen на false. Более серьезный недостаток заключается в том, что кириллические заголовки окон в подсказках становятся нечитаемыми.



› ***Skippy-xd*** **полезен, когда на Рабочем столе скапливается много окон — увы, качество эскизов иногда огорчает.**

› **Вместе с *Compton* консервативный MATE в какой-то мере обретает «второе дыхание».**




### 3ddesktop

А вот и кубик. И в придачу к нему пять эффектов переключения рабочих столов, среди которых есть трехмерные. Программа прекратила развиваться в 2005 году, но подобных отдельных приложений больше нет. На сайте http://desk3d.sourceforge.net/ выложен исходный код и RPM-файлы. Из Ubuntu утилиту убрали несколько лет назад, но пакеты сохранились в архиве по адресу http://old-releases.ubuntu.com/ubuntu/pool/universe/3/3ddesktop/ – скачиваем версию 0.29 и устанавливаем.

Перед первым запуском скомандуйте в консоли **3ddesk --acquire** и подождите несколько секунд, пока перед вами промелькнут все рабочие места: программа считывает с них изображения. Далее возможны два варианта.

Проще всего выполнить команду **3ddesk**. Все Рабочие столы с красивым эффектом расположатся на экране. Можно будет пролистать их кнопками стрелок «вправо» и «влево» на клавиатуре и после этого перейти к выбранному рабочему месту нажатием Enter. Такой подход зрелищный, но не очень практичный. Гораздо чаще пользователю надо быстро переключиться на соседний рабочий стол. В *3ddesktop* эту функцию удастся реализовать через т. н. «виды». Виды отличаются настройками эффектов переключения. Все виды описываются в конфигурационном файле **/etc/3ddesktop/3ddesktop.conf**, причем среди них уже есть те, которые нужны в данном случае: goright и goleft (переход на рабочее место вправо и влево). Остается только соединить команды **3ddesk --view=goright** и **3ddesk --view=goleft** с какими-либо удобными комбинациями клавиш, вроде Win+<Вправо> и Win+<Влево>.

В этом же конфигурационном файле меняются общие параметры (например, разрешение текстур), выбирается эффект переключения, настраиваются анимация и масштабирование. Более подробную информацию о настройке утилиты можно найти в инструкции (**man 3ddesk**), в комментариях конфигурационного файла и в статье Сергея Яремчука, опубликованной в LXF82. В общем, читайте, дерзайте, и будет вам куб не хуже того, что в *Compiz*. Плохо только одно: иногда программа показывает устаревшие изображения рабочих столов или вовсе серые прямоугольники. Немного успокаивает, что при перемещении вправо или влево без масштабирования по команде **3ddesk --view=goright** изъян не сильно бросается в глаза.

Теперь расскажем о многофункциональных композитных инструментах.

### XCompMgr

*XCompMgr* – родоначальник дополняющих композитных менеджеров, его можно назвать классическим. Утилита входит в состав репозиториев многих дистрибутивов, в том числе Debian и Ubuntu.

Графического интерфейса нет. Для запуска программы открываем консоль, пишем команду **xcompmgr** и после нее указываем параметры-ключи, каждый из которых отвечает за какие-либо настройки работы. Так, тени и анимация угасания окон появляются при наличии ключей **-c** и **-f** – значит, вводим **xcompmgr -cf**. Если хотите еще и увеличить радиус тени (ключ **-r**) до 30 пикселей, то команда станет такой: **xcompmgr -cf -r 30**. Полный список ключей открывается командой **man xcompmgr**. Чтобы все эффекты стартовали автоматически при включении системы, добавьте составленную вами строку запуска *XCompMgr* в автозагрузку.

Разочаровывает, что на практике заработала только анимация окон, а тени так и не появились. Хорошо, что у *XcompMgr* есть очень достойный потомок.

### Compton

История *Compton* – наглядная иллюстрация принципов открытого программостроения. Сначала на базе исходного кода *XCompMgr* создали композитный менеджер *Dcompmgr*, а потом *Dcompmgr* переделали в *Compton*. Самое главное, что в ходе работы устранили ряд ошибок оригинального *XCompMgr* и добавили много новых функций, благодаря которым *Compton* по функциональности догнал встроенный композитный инструмент *Xfce*.

Расскажем о наиболее значимых новшествах. Одно из самых заметных – регулировка прозрачности заголовков окон, неактивных окон и меню, осуществляемая ключами **-e**, **-i** и **-m** соответственно. Значение, как обычно, от 0,1 (почти прозрачно) до 1 (непрозрачно). Кроме того, *Compton* делает прозрачными окна при перемещении, но этот эффект пока никак не настраивается. Опция **--inactive-dim** затемняет неактивные окна. Диапазон значений – от нуля (без затемнения) до единицы (кромешная тьма).

Предлагается менять цвет тени. Интенсивность каждого из основных цветов (красный, зеленый, синий) задается тремя ключами: **--shadow-red**, **--shadow-green**, **--shadow-blue**. Значение, правда, не от 0 до 255, как всегда, а от 0 до 1, так что предварительно придется каждое число делить на 255. Например, один из оттенков сиреневого цвета (r150, g56, b210) указывается так:

```
--shadow-red 0.59 --shadow-green 0.22 --shadow-blue 0.82 .
```

С ключом **-b** *Compton* запустится как фоновый процесс, тогда эффекты не пропадут, если закрыть окно консоли.

Сохранилась большая часть настроек *XcompMgr*:

- **-c** – базовый ключ, включающий композитность с тенями и прозрачностью.
- **-f** – запускает анимацию угасания для окон и меню, если активен параметр **-c**.
- **-r** – радиус тени, **-o** – степень прозрачности тени.
- **-l** и **-t** – сдвиг тени влево и вверх соответственно.
- **-I**, **-O**, **-D** – параметры эффекта угасания. Первые два ключа отвечают за шаг изменения прозрачности, а последний – за временной промежуток в миллисекундах между этими шагами.
- **-C** – запрет теней у панелей и доков, **-G** – отключение теней у перемещаемых окон.

Принцип управления прежний: команда **compton** с параметрами. Обязательны только ключи **-c** и **-f**, остальные – на ваше усмотрение.

Конечно, мы не могли рассмотреть здесь все варианты настройки. Полный список, как обычно, открывается вводом **man compton**. Набор параметров настолько велик, что использовать хотя бы половину из них в одной команде будет затруднительно. Понимая это, разработчики *Compton* добавили альтернативный метод настройки эффектов – через конфигурационный файл. Если поместить его в домашний каталог, то достаточно будет ввести команду

» **Пропустили номер?** Узнайте на с. 108, как получить его прямо сейчас.

```
compton --config ~/compton.conf
```

Образец файла можно найти на странице программы в *GitHub* (https://github.com/chjj/compton).

Если *Compton* вам понравился, добавьте команду запуска программы в автозагрузку.

Помимо всего перечисленного, *Compton* поставляется с собственным аналогом утилиты *transset*. Команда **compton-trans** при работающем *Compton* меняет прозрачность окон несколькими способами, каждый из которых очень подробно описан в инструкции **man compton-trans**. Есть обычное прицеливание курсором, «пушка» заряжается командой **compton-trans -s N**. А можно очень гибко увеличивать или уменьшать прозрачность с определенным шагом, выполняя последовательно команды **compton-trans -s -- -N** и **compton-trans -s +N**. Учтите только, что прозрачность здесь считается в процентах.

Получить этот маленький (60–70 КБ), но дорогой золотник в виде Deb-пакетов рекомендуем в одном из двух архивов на Launchpad, короткие строки для которых – **ppa:richardgv/compton** (сборка от текущего разработчика) и **ppa:mati75/evilwm**. Исходный код выложен на странице *Compton* в *GitHub*.

Программа активно разрабатывается, поэтому еще несколько месяцев назад все версии пакетов содержали большое количество ошибок. Доходило до исчезновения содержимого окон. Теперь *Compton* достаточно стабилен.

Не так давно началась работа по добавлению *Compton* в репозитории Debian. Будем надеяться, что проект действительно получит признание и поддержку, а автор не потеряет интерес к своему творению.

### Cairo-compmgr

Глядя на *Cairo-compmgr*, думаешь: «Хотел он стать *Compiz'ом*, но так и не дорос». Однако не с разочарованием из-за того, что «не дорос», а с гордостью за то, что «хотел», потому что благодаря этой смелой мечте получился наиболее функциональный из существующих добавочных композитных менеджеров, в самом деле «*мини-Compiz*». Хотя *Compton* уже дышит ему в затылок.

В репозиториях Ubuntu и Debian программы, однако, нет. Авторские Deb-пакеты *Cairo-compmgr* (см. http://cairo-compmgr.tuxfamily.org/) подойдут не ко всем системам. Пользователи новых версий Ubuntu могут установить композитный менеджер из следующего архива пакетов на Launchpad: **ppa:shnatsel/cairo-compmgr**.

В первую очередь *Cairo-compmgr* выгодно отличается наличием графического интерфейса. Запускаем программу через главное меню. Значок в виде паука появится в области уведомлений. Когда эффекты включены, он синего цвета. Чтобы временно выключить композитность, снимите галочку с пункта “Composite desktop” контекстного меню значка.

Открываем окно настроек через пункт Параметры. Между прочим, есть и файл настройки, который скрыт в папке **~/.config/cairo-compmgr**, но вручную его лучше не править, потому что эффекты станут сильно тормозить.

На первой вкладке лучше включить сразу все компоненты [Plugins], чтобы не упустить ни одну возможность. Базовые настройки композитной обработки данных тоже безопаснее не менять.

Параметры эффектов находятся на вкладках Window [Окно] и Effects [Эффекты]. Можно указать продолжительность [Duration] различных анимаций, задать значение прозрачности меню [Menu opacity] и заголовков (Decorations – Alpha; рекомендуем не меньше 0,5). Отметка Gradient включит плавный переход от цвета к прозрачности в заголовках. Позволяется подправить радиус и цвет тени. Эффект Freeze – затемнение окон зависших или тормозящих приложений – устраивает с настройками «по умолчанию».

Разнообразные дополнительные инструменты вновь напоминают о *Compiz*. Несмотря на то, что вкладка Accessibility [Доступность] неактивна, а функция выбора обоев не работает, *Cairo-compmgr* все равно на голову выше других композитных менеджеров по этому критерию. Средство Mosaic, поселившееся на вкладке Desktop, полезно, когда запущено много приложений: при нажатии определенной комбинации клавиш на экране отобразятся эскизы всех окон. Можно выбрать нужное, щелкнув по нему мышью и повторно нажав клавиатурное сокращение. Комбинацию по умолчанию Win+Tab легко заменить на более привычную Alt+Tab.

Дополнительных утилит две (последняя вкладка). Инструмент Set window opacity [Задать прозрачность окна] помогает гибко увеличивать или уменьшать прозрачность окон с заданным шагом (например, 5 %), причем, в отличие от *transset* и *compton-trans*, эта функция сразу связана с «горячими» клавишами. Наконец, модуль «снимка экрана» поможет запечатлеть полученное, будем надеяться, великолепие. Если, впрочем, заработает, потому что в целом стабильность и предсказуемость *Cairo-compmgr* далеки от идеала: может исчезать прозрачность, могут размываться заголовки, иногда не работает переключатель окон и встроенный «фотоаппарат».

Пожалуй, на этом все. Добавочный композитный менеджер *Unagi* лучше оставить за кадром, потому что пока все его «функционирование» сводится к выводу списка ошибок в консоли.

Помните, что разные композитные менеджеры нельзя включать одновременно. Обычно при попытке сделать это вы получите предупреждение о невозможности запуска, но иногда может зависнуть система. Убедитесь, что собственные эффекты оконного менеджера (например, *Xfwm*) выключены. Специализированные утилиты (*Transset*, *Skippy-xd*, *3ddesktop*), напротив, спокойно работают параллельно с любыми композитными инструментами.

Что касается совместимости добавочных композитных менеджеров, то в Gnome 3 и Cinnamon они не запустились. Впрочем, перед *Mutter* пасует сам *Compiz*. С целевыми диспетчерами окон и рабочими окружениями (*Openbox*, *Xfce*, *Gnome*, MATE) проблем не возникало.

## Подведем итоги

Среди программ, создающих эффекты рабочего стола, *Compiz*, безусловно, остается заслуженным лидером. На одну доску с ним можно поставить разве что *Kwin*. То, что предлагают встроенные оконные менеджеры остальных графических окружений и добавочные композитные менеджеры, смотрится куда скромнее.

Однако, как показывает практика, многих пользователей вполне устраивает набор из теней, прозрачности и простенького эффекта появления окон. В любом случае, не бойтесь: отсутствие *Compiz* не оставит вас лицом к лицу с бездвижной грубо-плоской картинкой. Благодаря разнообразным встроенным эффектам графических оболочек и альтернативным композитным инструментам все будет куда ярче и радужнее.

И, кроме того, вы узнаете чуть больше о графической подсистеме Linux. LXF

## Поиск пакетов

При описании добавочных композитных менеджеров мы сосредоточили внимание на системах, совместимых с Debian или Ubuntu. Пользователям других дистрибутивов, если необходимого пакета нет в репозиториях, а скомпилировать программу самостоятельно из исходного кода не получается, советуем обратиться к сервисам rpmfind.net (поиск RPM-пакетов) и slackfind.net (поиск пакетов для Slackware).

# Ответы

**Есть вопрос по открытому ПО? Пишите нам по адресу answers@linuxformat.ru, и мы найдем ответ.**

В этом месяце мы ответим на вопросы про...

1. Запуск компьютера без жесткого диска
2. TightVNC для просмотра DVD
3. Неперсистентный LXFDVD
4. Сетевые драйверы
5. Проблемы с принтером
6. Использование ТВ-тюнера в Linux

## 1 Бездисковые киоски

**В Я работаю волонтером в местном музее. У нас есть два публичных киоска для поиска и просмотра оцифрованных фотографий и видеопрезентаций в формате Flash. Они довольно старые, и компьютеры с Windows XP начинают доставлять нам неприятности. Хочу заменить их на другие и воспользоваться Linux в качестве операционной системы.**

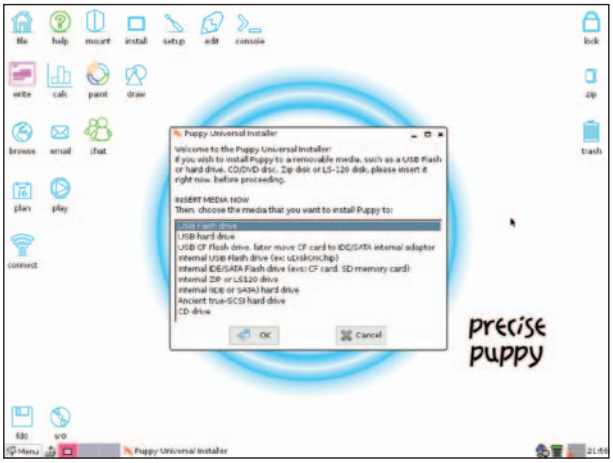
**На компьютерах запускается только web-браузер. В начале и в конце дня они включаются и выключаются автоматически. При запуске браузер открывает начальное окно с нашего web-сервера. Далее все запросы обрабатываются через web-сервер.**

**Нужен ли жесткий диск на таком компьютере? Можно ли загрузиться по сети с сервера и пользоваться только оперативной памятью? Я хочу, чтобы новые компьютеры были как можно более простыми и долговечными (и дешевыми).**

**Гренвилл Тэйлор [Grenville Taylor]**

О Бездисковые киоски можно загружать по сети, но Вам нужно простое решение, а настроить сетевую загрузку непросто. Если Вам нужен простой и дешевый вариант, я бы советовал запускать Live-дистрибутив с USB-флэшки. Компьютер загружается всего раз в день, и более медленная загрузка (но, тем не менее, быстрее сетевой) не критична. Для повышения производительности в большинстве Live-дистрибутивов можно сделать так, чтобы файловая система загружалась в оперативную память, и во многих из них есть возможность запускать дополнительные программы при загрузке системы – в Вашем случае, браузер. Хороший вариант для такой системы – Puppy Linux. Он компактный, легко устанавливается на USB-диск и расширяем. Загрузите ISO-образ с http://puppylinux.org, запишите его на CD и загрузитесь с него. Для установки дистрибутива на USB-флэшку щелкните по иконке Install [Установка] на рабочем столе и следуйте появляющимся указаниям. Когда закончите, перезагрузитесь, но настройки не сохраняйте. Загрузитесь с USB-флэшки, снова пройдите через первоначальную настройку и снова перезагрузитесь, но на сей раз сохраните настройки. На диске создастся область постоянного хранения. Теперь можно настроить все так, как Вам нужно.

В домашнем каталоге Вы найдете каталог Startup. Любая программа, скрипт или символическая ссылка на другую программу будут запущены при загрузке рабочего стола. Можно создать символическую ссылку на выбранный браузер или положить сюда скрипт, запускающий браузер в полноэкранном режиме со страницей приветствия – выбор за Вами.




**Puppy Linux — хороший выбор легкого дистрибутива для киоска, и он загружается с USB-брелка.**

## 2 DVD по сети

**В С помощью *TightVNC* я подключаюсь к домашнему настольному компьютеру с двух старых ноутбуков (не одновременно). Одна из нужных мне программ — *Handbrake*, с помощью проигрывателя DVD на настольном компьютере я конвертирую в ней фильмы для последующего просмотра на телефоне. Мне интересно, можно ли пользоваться проигрывателями DVD на ноутбуках (клиентах) вместо того, чтобы вставлять DVD в настольный компьютер (сервер).**

**Раньше я пробовал делать X-проброс с *Handbrake* и остановился на *TightVNC*, но даже тогда не мог понять, можно ли запускать клиентский DVD-проигрыватель на сервере. Я новичок в Linux (хотя с 2008 года пытаюсь «пользоваться» только им). Одно из «простых» решений — конвертировать все DVD в ISO, но тогда придется конвертировать весь диск, даже если мне нужны всего несколько треков. Кроме того, это лишний шаг в процессе, который мог быть гораздо проще, если бы DVD-проигрыватели клиента работали с сервером.**

**Whitbym, с форумов**

О Сделать это можно по-разному, в зависимости от того, как программа предпочитает обращаться к DVD. *Handbrake* требует смонтированного диска, и Вы сможете настроить к нему доступ по сети с *Samba* или NFS. *Samba* подходит лучше, к тому же тогда Вы сможете обращаться к диску с компьютеров с Windows. На ноутбуке добавьте точку монтирования DVD в **/etc/samba/smb.conf**, открыв в браузере адрес http://localhost:901 и воспользовавшись графическим интерфейсом или просто добавив в файл следующий текст:

```
[dvd]
comment = Shared DVD drive
writable = No
locking = No
guest ok = Yes
path = /mnt/dvd
```

Параметр path должен содержать каталог, в который монтируется DVD. Оповестите *Samba* об изменении настроек, перезапустив сервер.

```
sudo /etc/init.d/samba restart
```

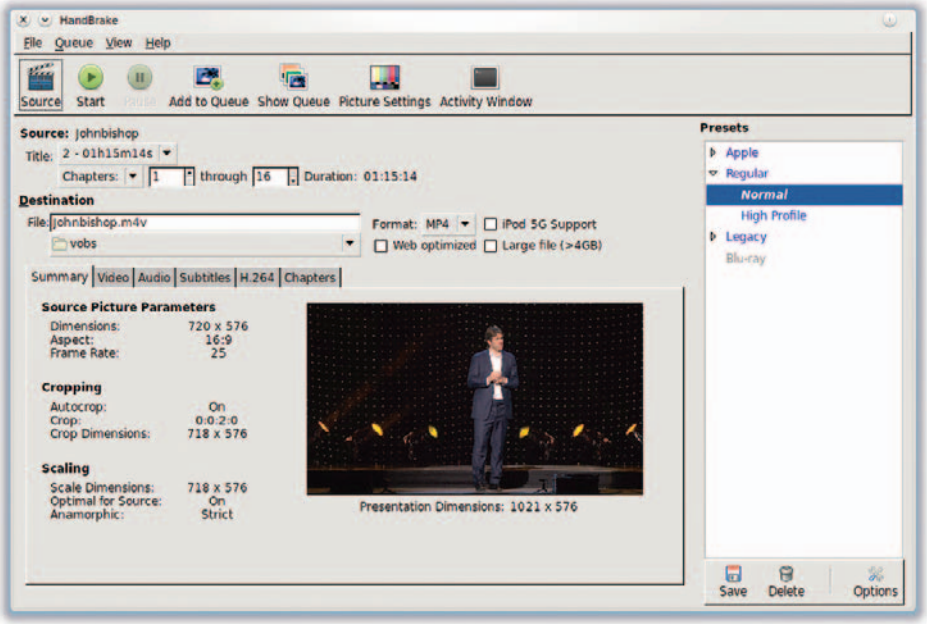
Вставьте диск в привод и убедитесь, что он смонтировался в нужный каталог, затем смонтируйте его на настольном компьютере следующей командой, которую можно запустить через VNC или SSH:

```
sudo mount -t cifs //laptop/dvd /mnt/laptop
```

Здесь **laptop** – имя хоста или IP-адрес ноутбука, а **/mnt/laptop** – каталог на настольном компьютере, в который нужно смонтировать DVD. Теперь на настольном компьютере можно запустить *Handbrake*, нажать кнопку Source (Источник) и указать точку монтирования.

Альтернативный вариант – экспортировать блочное устройство и затем смонтировать его на настольном компьютере. Преимущество этого подхода в том, что он также работает с программами, которые обращаются к диску напрямую, а не через смонтированную файловую систему. Сетевое блочное устройство (Network Block Device – **nbd**) – это функция ядра для доступа к удаленным устройствам или файлам, как если бы они были локальным блочным устройством. Запустите на ноутбуке следующую команду:

```
nbd-server 5555 /dev/sr0
```




**Сетевое блочное устройство позволяет обращаться с одного компьютера к DVD-приводу другого.**

Число – используемый порт; подойдет любой порт с номером больше 1024. За ним следует блочное устройство привода DVD, обычно **/dev/sr0** для первого или единственного привода оптических дисков. Затем на настольном компьютере выполните команду

```
sudo nbd-client laptop 5555 /dev/nbd0
```

Здесь laptop – имя хоста ноутбука, а номер порта – тот же, что использовался на ноутбуке. Если Вы видите сообщение об ошибке, загрузите модуль **nbd** вручную перед запуском клиента:

```
sudo modprobe -v nbd
```

Эта команда создает устройство **/dev/nbd0**, которое работает точно так же, как локальный привод DVD. Теперь можно либо смонтировать **/dev/nbd0**, либо воспользоваться любой программой, которая обращается к диску напрямую, например, **mplayer/mencoder**, точно так же, как если бы диск был в локальном приводе. Когда закончите, удалите блочное устройство командой:

```
sudo nbd-client -d /dev/nbd0
```

## 3 Персистентные LXFDVD

**С помощью скрипта *dvd2usb* я сделал загрузочный USB-брелок объемом 32 ГБ из дистрибутива на LXFDVD 166. Он работает нормально. Но я хочу пользоваться дистрибутивом на брелке как серьезной операционной системой, а он не поддерживает персистентность, поэтому бесполезен для меня. У меня есть десяток похожих брелков, на каждом свой дистрибутив, и почти всегда мне приходится ждать несколько часов, пока ISO-образ загрузится с Pendrive Linux. Я читаю *Linux Format* несколько лет, и очень немногие DVD пригодились мне, потому что ISO-образы на них бывают редко.**

**Был бы весьма признателен, если бы вы сказали, как сделать загрузочную флэшку с поддержкой персистентности из LXFDVD, не загружая огромные файлы. Я пробовал искать ответ на этот вопрос в Интернете, и везде мне советуют пользоваться Pendrive Linux или чем-то вроде.**

**Винсент Кингстон [Vincent Kingston]**

Наши DVD предназначены в основном для установки дистрибутивов, а скрипт *dvd2usb* – лишь способ воспользоваться DVD на компьютере без DVD-привода. Да, большая часть вариантов загрузки относится к Live-среде, но они предназначены лишь для того, чтобы попробовать дистрибутив, а не для постоянного использования. В ряде дистрибутивов, в частности, Ubuntu и его производных, есть утилита *Startup Disk Creator*, способная создать загрузочную флэшку с областью постоянного хранения данных, а *Unetbootin* умеет делать то же для других дистрибутивов, которые используют один ISO-образ. »

## Терминалы и суперпользователи

Мы часто предлагаем в качестве решения проблемы ввести те или иные команды в терминале. Хотя обычно то же самое можно сделать с помощью графических утилит дистрибутива, такие решения будут слишком конкретными (будут зависеть от дистрибутива). Команды в терминале более гибкие и – самое главное – ими можно пользоваться во всех дистрибутивах. Команды настройки системы часто нужно выполнять от имени суперпользователя, называемого также root. Существует два основных способа это делать, в зависимости от используемого дистрибутива. Во многих, особенно в Ubuntu и его производных, перед командой можно написать **sudo** – при этом будет запрошен пароль пользователя, и ему будут предоставлены привилегии root только на время выполнения команды. В других дистрибутивах используется **su**, для использования которой требуется ввести пароль root и которая предоставляет полный доступ root до того момента, пока вы не наберете **logout**. Если в вашем дистрибутиве используется **su**, запустите ее один раз и выполняйте любые заданные команды без предшествующей **sudo**.

## Коротко про...

# dmesg

В любой онлайн-дискуссии о проблемах с устройствами или драйверами недолго приходится ждать, пока кто-нибудь спросит: «А что говорит *dmesg*?». Вы набираете **dmesg** в терминале, и на экране пролетает несколько тысяч незамутненно-технарских сообщений. Это содержимое журнала ядра – список всех информационных сообщений ядра. Перенаправив вывод *dmesg* в программу постраничного просмотра, вы увидите сообщения об обнаружении различных устройств и загрузке соответствующих модулей ядра.

*Dmesg* выдает массу информации, и нам надо понять, как найти требуемые данные. Основных вариантов три: первый – перенаправить вывод в программу для постраничного просмотра:

```
dmesg | less
```

Затем вы сможете поискать название известного устройства или драйвера (в **less** нажмите /). Альтернатива **less** – фильтрация вывода с помощью *grep*, и если вы ищете информацию о USB-устройстве, то

```
dmesg | grep -i USB
```

будет хорошим началом. Также можно добавить параметры **-e** или **-T**. Оба они велят *dmesg* выводить время в удобном формате, но немного разными способами.

Третий способ доступен только в последних релизах *util-linux* (пакет, где содержится *dmesg*) с версии 2.22 и выше. Там добавлен новый параметр **dmesg**: **-w** или **--follow**. С **--follow** *dmesg*, как и прежде, отображает буфер сообщений ядра, но показывает и новые сообщения, если они появляются. Можно запустить **dmesg --follow**, подключить устройство и посмотреть, что ядро о нем думает, в реальном времени.

Если помещать дистрибутивы на DVD в виде ISO-образов, читателям будет менее удобно – им придется сначала прожечь дистрибутив на CD или DVD и только потом загрузиться с него. Я поэкспериментировал со скриптом для создания ISO-образов с DVD, но при создании загрузочных дисков используется столько различных параметров и методов, что не удалось найти какой-то постоянный метод, пригодный для всех, и образ удалось создать лишь для нескольких дистрибутивов. Если Вам нужна более постоянная установка на флэшке, проще загрузиться с DVD и установить дистрибутив на флэшку, как на жесткий диск. О том, чтобы загрузчик устанавливался именно на флэшку, а не на жесткий диск, нужно позаботиться, но в остальном все должно быть просто. Так у Вас появится портативный дистрибутив, который можно обновлять, и место для хранения своих файлов.

Основной недостаток в том, что запуск дистрибутива с флэшки сокращает срок ее службы – но флэшки дешевы и их легко заменить, так что этот недостаток не фатальный.

## 4 Нет сети

**У меня ноутбук Toshiba с предустановленной Windows 7, и на нем я установил Linux Mint 13 (64-битный). В Windows все работает отлично, а в Linux Mint не работает сеть — как проводная, так и беспроводная. Я предположил, что за проводное соединение отвечает Ethernet-контроллер Atheros. О контроллере Realtek я не подумал, считая, что он реализует только беспроводное соединение, а им я займусь, когда разберусь с проводным. Это правильно?**

**Несколько часов почитав Интернет на другом компьютере, я перепробовал множество других вариантов, в том числе переустановку *ndiswrapper* из исходников. Я знаю, что в файловой системе есть все компоненты *ndiswrapper*, но система не может их найти.**

**Потерпев неудачу с *ndiswrapper*, я решил попробовать установить драйвера Linux для Atheros. Загрузил Compat-wireless-2012-11-25-p с www.orbit-lab.org и выполнил команды** ./scripts/driverselect alx, make, sudo make install **и** modprobe alx **— и в результате получил сообщение "FATAL: module alx not found [ФАТАЛЬНАЯ ОШИБКА: модуль alx не найден]".**

**Хью Гэррioч [Hugh Garrioch]**

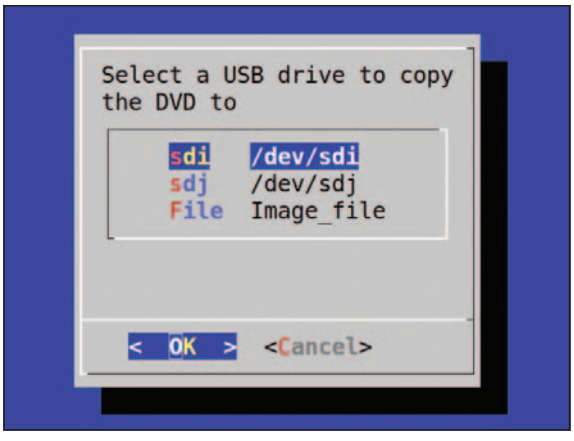
Для Вашего Ethernet-контроллера не нужен *ndiswrapper*: у него есть «родной» драйвер Linux. Так как чипсет довольно новый, драйвер еще не добавлен в основное ядро, но его легко установить. Использованная Вами процедура компиляции и установки модуля *alx* должна была работать, но нужно убедиться, что символическая ссылка **/usr/src/linux** указывает на версию ядра, для которой производится сборка (драйвера ядра «вне дерева» должны компилироваться отдельно для каждого ядра). Если ядро всего одно, проблем быть не должно. Еще один подводный камень – тот, что после установки любых модулей ядра нужно выполнить команду

```
sudo depmod -a
```

Она обновляет используемый **modprobe** список доступных модулей и их зависимостей. Эта команда выполняется при загрузке системы, но при самостоятельной компиляции модулей без перезагрузки ее нужно запустить отдельно.

Впрочем, есть и более простой вариант: воспользоваться пакетом для Ubuntu/Mint, который устанавливает этот модуль. С ним Вам не придется не только компилировать и устанавливать модуль, но и переустанавливать его после обновления ядра. Это особенно удобно с сетевыми драйверами, ведь если Вы забудете перекомпилировать их перед загрузкой в новое ядро, у Вас не будет сетевого соединения для загрузки новых. Откройте *Synaptic* и установите пакет, который Вы пытались установить вручную – *linux-backports-modules-cw-3.3-precisegeneric* (**cw** означает **compat-wireless**). Пакетов *linux-backports-modules-cw* несколько – выбирайте именно этот. Это пустой пакет, который просто вызывает версию, подходящую для Вашего ядра. Так мы гарантируем, что каждый раз при обновлении ядра будет устанавливаться нужный драйвер.

Ваша беспроводная карта довольно новая, ее официальный драйвер для Linux еще не выпущен. В принципе, драйвер есть, но его установка сейчас не очень удобна. Подробные указания и ссылки можно найти на http://askubuntu.com/questions/139632/wireless-card-realtek-rtl8723ae-bt-is-notrecognized. Однако Realtek любезно предоставила информацию и устройства, необходимые разработчикам Linux для написания драйвера, поэтому в новых версиях ядра эта карта будет поддерживаться.




**› Скрипт dvd2usb поможет воспользоваться LXFDVD на компьютере без DVD-привода.**

## 5 Брат мой — враг мой

**Я пытаюсь заставить работать в Ubuntu один из моих принтеров — Brother MFC-5890CN или Brother MFC-215C. Раньше у меня получалось, но теперь я не могу найти документ со всеми командами, который нужно скопировать и вставить в терминал. Мне нравится Ubuntu 12.10, но пока я не могу подключить и настроить принтер, приходится оставаться в Windows. Вводить команды в терминале меня вполне устраивает.**

**Билл Шеперд [Bill Shepherd]**

У Вас необычные принтеры – обоих нет в базе данных http://linuxprinting.org. Однако Brother любезно поставляет драйверы Linux, и необходимые пакеты можно найти на http://welcome.solutions.brother.com/bsc/public_s/id/linux/en/download_prn.html. Для каждого принтера загружаются два пакета: драйвер LPR и драйвер cupswrapper. Первый – драйвер, подходящий для использования принтера со старой системой печати lpr. Второй обеспечивает также работу с CUPS.

Выберите нужную версию пакетов – 32- (i386) или 64-битную (amd64), в зависимости от Вашего дистрибутива. Загрузите четыре пакета в отдельный каталог. Затем их можно установить по отдельности, дважды щелкнув по каждому пакету (при этом пакет загрузится в Ubuntu Software Centre), или все сразу с командой строки:

```
dkpg --install --recursive directory/where/you/put/them
```

После установки драйверов у Вас должна появиться возможность настройки принтера через утилиту в настройках системы [System Settings] или через браузер, открыв в нем адрес http://localhost:631. Некоторые пользователи сообщали о том, что принтеры не работали с этими драйверами из-за отсутствия необходимого каталога **spool**, что можно исправить следующей командой:

```
mkdir -p /var/spool/lpd
```

Если каталог существует, команда не делает ничего, и можно просто тупо запускать ее после установки пакетов, чтобы избежать проблем.

## 5 Телевизор молчит

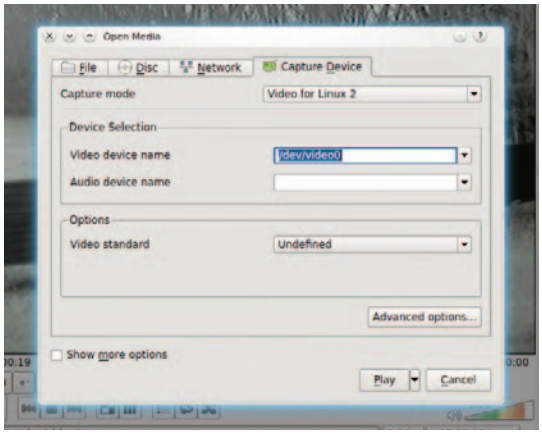
**Я только что перешел с Windows на Linux, установив 32-битную версию Ubuntu 12.04. Можете помочь настроить ТВ-тюнер PVR-TV7134? Какое ПО и драйверы необходимы?**

**Уильям Томас [William Thomas]**

Прежде всего, в Linux драйверы устанавливаются не так, как в Windows. На самом деле, драйвера большинства устройств уже встроены в систему в виде модулей ядра. Немногие остальные не нужно загружать из Интернета – обычно они доступны через менеджер пакетов дистрибутива. Модуль драйвера для Вашего ТВ-тюнера есть в ядре, он называется *saa7134*. Он должен загружаться автоматически при обнаружении карты во время загрузки. Чтобы это проверить, откройте терминал и наберите:

```
lsmod | grep saa7134
```

Команда выведет список всех используемых модулей, соответствующих запросу. Это как минимум *saa1734* и, возможно, некоторые другие,




› ***VLC* — хороший вариант для базовой работы с ТВ-тюнером, если вы не любитель KDE.**

например, *saa1734-dvb* для приемника Freeview. Если модуль не загружается автоматически, загрузите его командой

```
sudo modprobe -v saa7134
```

Она должна сообщить, что модуль загружен. Чтобы это происходило автоматически при каждой загрузке, можно добавить необходимые модули, по одному на строку, в файл **/etc/modules**. Для этого нужны права root, поэтому в Ubuntu это проще всего сделать командой

```
echo “saa7134” | sudo tee -a /etc/modules
```

При загрузке модуль создаст файлы устройств **/dev/video** или **/dev/dvb**, в зависимости от спецификации конкретной карты (здесь пригодился бы вывод **lspci**, как описано во врезке «Помогите нам помочь вам»). Есть разные программы для работы с ТВ-тюнером. Одна из наиболее мощных, пожалуй, *MythTV* – но, несомненно, и одна из самых сложных для освоения. Для обычного просмотра прекрасно подойдут *VLC*, *SMPlayer* или *Gnome MPlayer* – все они доступны в Ubuntu Software Centre. Запустив *VLC*, зайдите в Media > Open Capture Device [Мультимедиа > Открыть устройство захвата], выберите тип устройства – “Video 4 Linux 2” для аналогового тюнера или карты захвата и “TV (digital)” для тюнера DVB (Freeview). Затем выберите имя устройства, соответственно **/dev/video0** или **/dev/dvb/adapter0** для первой или единственной карты, и нажмите Play [Воспроизвести]. Для цифрового ТВ систему доставки [delivery system] нужно установить в DVB-T, это сервис бесплатного цифрового ТВ.

Если Вы пользуетесь DVB, нужно предоставить *VLC* список используемых каналов. Установите пакет *dvb-utils*, зайдите на www.digitaluk.co.uk/postcodechecker, чтобы найти свой локальный передатчик, и запустите в терминале

```
scan /usr/share/doc/dvb-utils/examples/scan/dvb-t/
YOUR-TRANSMITTER -o zap >~/channels.conf
```

Каналы просканируются и сохранятся в файле. Затем *VLC* и другие медиа-проигрыватели автоматически подхватят их оттуда. В порядке альтернативы можно воспользоваться *Kaffeine* – в нем есть все возможности, необходимые для цифрового ТВ, включая сканирование каналов. Его единственный возможный недостаток для Вас в том, что это программа для KDE. Запуститься-то на Unity она запустится, но программы KDE в Unity/Gnome, и наоборот, выглядят несколько чужеродными. LXF

## Помогите нам помочь вам

Ежемесячно мы получаем несколько писем, на которые не в состоянии ответить, так как проблема описана в них недостаточно полно. Чтобы дать вам наилучший ответ, нам нужно знать как можно больше.

Если у вас появляется сообщение об ошибке, приведите его точный текст и опишите конкретные условия, когда оно появляется. При возникновении проблемы с устройствами перечислите нам все установленные устройства.

Если Linux уже запущен, можете применить для этого отличную программу *Hardinfo* (http://hardinfo.berlios.de/) – она сохранит подробную информацию об устройствах и о состоянии системы в HTML-файл, который вы сможете приложить к своему письму.

Альтернативный и не менее удобный вариант – *lshw* (http://ezix.org/project/wiki/HardwareLiSter). Одна из указанных программ непременно должна быть включена в ваш дистрибутив (а иногда и обе).

Если вы не хотите или не можете их установить, выполните следующие команды в терминале от имени root и приложите файл **system.txt** к письму. Это здорово поможет диагностике.

```
uname -a >system.txt
lspci >>system.txt
lspci -vv >>system.txt
```

## Часто задаваемые вопросы

# Печать

**› Что такое CUPS?**

*CUPS – Common Unix Printing System* [Общая система печати Unix] – это набор драйверов и утилиты, предоставляющие полную поддержку принтеров и служащих для управления печатью в Linux и других операционных системах на базе UNIX.

**› Так это драйвер принтера?**

Да, но не только. *CUPS* предоставляет «портативный слой печати» между приложениями и устройствами печати. Конечно, в него входят и драйверы, но также и все остальное, что нужно программам печати.

**› То есть чтобы заставить принтер печатать, нужно редактировать файлы настройки через командную строку?**

Вовсе нет. У *CUPS* есть собственные графические утилиты настройки, они работают через web-браузер. Откройте адрес http://localhost:631 в своем любимом браузере. Может потребоваться ввод пароля вашего пользователя или root – и откроется домашняя страница *CUPS*. На ней можно просматривать список принтеров, добавлять и удалять их, а также управлять очередями печати и читать документацию.

**› Зачем пользоваться браузером вместо обычной графической программы?**

Для работы через web-интерфейс вам не нужны графические утилиты; на компьютере даже не нужен X-сервер. Вы можете пользоваться текстовым браузером, вроде *elinks*, или зайти в него браузером с другого компьютера.

**› Но это же небезопасно?**

Возможно, но с настройками по умолчанию к *CUPS* можно подключаться только с localhost. Их можно изменить, разрешив подключения с компьютеров локальной сети (разрешать доступ через Интернет обычно не рекомендуется), и управлять тем, каким пользователям доступны те или иные компоненты конфигурации.

**› Как со всем этим связан *gimp-print* и что у *GIMP* общего с печатью?**

*gimp-print* – набор драйверов печати, разработанных для применения с *GIMP*. Хотя *GIMP* отлично ладит с *CUPS*, для некоторых принтеров с *gimp-print* получаются лучшие результаты. Теперь эти драйверы также работают и с *CUPS*, так что *gimp-print* можно считать расширенным набором драйверов, которые работают со всеми программами, печатающими через *CUPS*, а не только с *GIMP*.

**› А что такое *gutenprint*?**

Это новое название *gimp-print*. Теперь, когда *gimp-print* теснее связан с печатью, чем с *GIMP*, новое название меньше вводит в заблуждение, хотя любое изменение имени ненадолго вызывает путаницу.

**› Как узнать, поддерживается ли мой принтер?**

Прежде всего зайдите на сайт www.linuxprinting.org: там есть обширная база данных с информацией о том, как поддерживается каждый принтер, и советы по тому, какими драйверами пользоваться.

# LXFHotPicks

**Fraqtive » Tiny BASIC » DeadBeef » Smem » Mundus » Glogg » GrafX2 » Pioneer » FBZX » PyCalendarGen » libexplain**

**Майк Сондерс**
Излазив самые недосягаемые и укромные уголки Интернета, Майк точно знает, где прячутся лучшие образчики открытого кода.

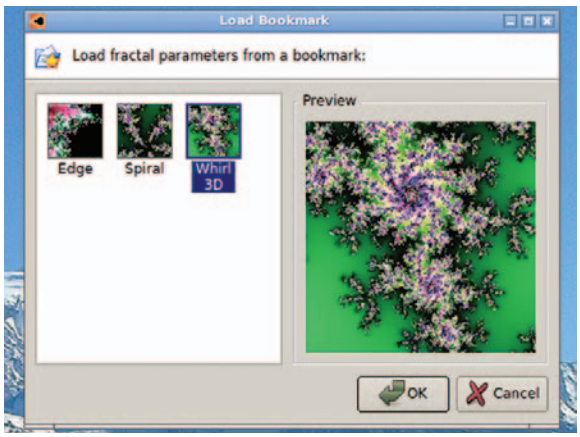
Генератор фрактальных изображений

## Fraqtive

**Версия** 0.4.6 **Сайт** http://fraqtive.mimec.org

Фрактальные изображения очаровывают даже тех, кто далек от математики. Генерируемые на основе уравнений, эти изображения отличаются потрясающей детализацией и красотой, завораживающей тем больше, чем дольше вы их наращиваете. Игра с фракталами удовлетворяет нашу врожденную страсть к исследованиям: мы видим нечто необычное или удивительное, и обнаруживаем в нем все больше деталей.

Программы генерирования фрактальных изображений существуют уже не одно десятилетие, практически на всех платформах, способных отображать графику, и наткнувшись на *Fraqtive*, особенных потрясений мы не ожидали. И – ошиблись. В общем и целом, *Fraqtive* не отличается от любой другой программы этого рода: вы можете сновать по изображению с помощью клавиш курсора и увеличивать и уменьшать масштаб колесом прокрутки мыши, чтобы разглядеть детали. Все это очень мило, и если вы раньше не имели дела с программой фрактальных изображений, вы сочтете ее более чем интересной.




**› Можно запомнить текущее положение и масштаб, а потом к ним вернуться.**

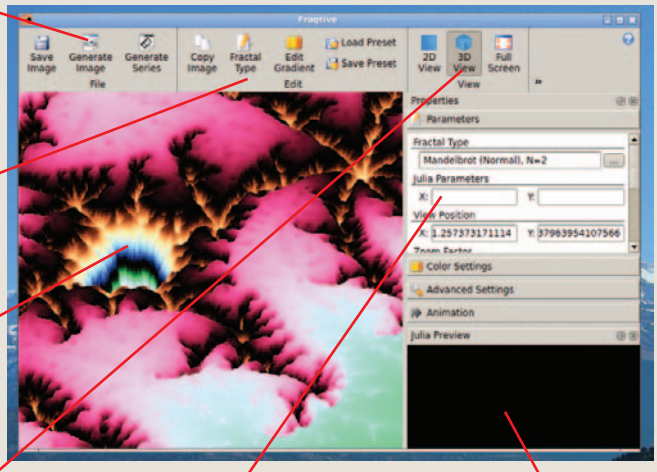
> «Эти изображения отличаются потрясающей красотой.»

Но во *Fraqtive* есть кое-что еще, с чем мы прежде не сталкивались во фрактальных программах для Linux, и это – режим 3D. Одним щелчком текущий вид преображается из достойного, хоть и плоского, изображения, в физический мир с горами и долами, который так и хочется потрогать пальцем через экран. Вы вдруг ощущаете себя исследователем реального мира, а не просто математической конструкции, и почти невозможно заставить себя прекратить поиски все новых подробностей.

А благодаря исключительной сложности фрактальных изображений вы можете обнаружить картину, которую до вас никто не видел – комбинацию углов, глубины и цвета, прежде не открывавшуюся ни одному человеку. На панели Animation справа можно перетаскивать бегунок, чтобы вращать 3D-изображение с разными скоростями, а кнопка Load Preset вверху задает цветовую схему.

*Fraqtive* позволяет создавать отдельные изображения текущего вида с высоким разрешением, а также создавать ряд изображений разных уровней отображения, которые можно объединить в анимацию. А если вам надоест предложенный по умолчанию фрактал Мандельброта, переключитесь на фрактал Джулия для разнообразия.

### Исследуем интерфейс Fraqtive




**Generate [Генерировать]**
Щелкните по этим значкам, чтобы сохранить изображение или ряд изображений с высоким разрешением.

**Settings [Настройки]**
Здесь настраивается тип фрактала и его цветовая схема.

**Main View [Основной вид]**
Используйте клавиши курсора для перемещения и колесо прокрутки мыши для изменения масштаба.

**3D View [Вид 3D]**
Нажмите сюда, чтобы переключиться с обычного (плоского) вида на 3D-версию.

**Parameters [Параметры]**
В этой панели можно детально настроить свойства фрактала.

**Julia preview [Предпросмотр Джулии]**
При просмотре плоского фрактала Мандельброта здесь отображается альтернатива – фрактал Джулия.

Язык программирования

# Tiny BASIC (для Curses)

**Версия** 0.6.8 **Сайт** http://tinybc.sourceforge.net

Да, да, мы знаем, BASIC – ужасный язык. Он стар, в нем нет полезных функций, а программы он выдает со структурой вороха макарон. И все же это один из самых доступных языков. Большинство из нас в Башнях **LXF** пробовали свои силы в 8-битных диалектах BASIC, и хотя сегодня мы не рекомендуем писать в нем большие приложения, когда-то он пробудил в нас интерес к программированию.

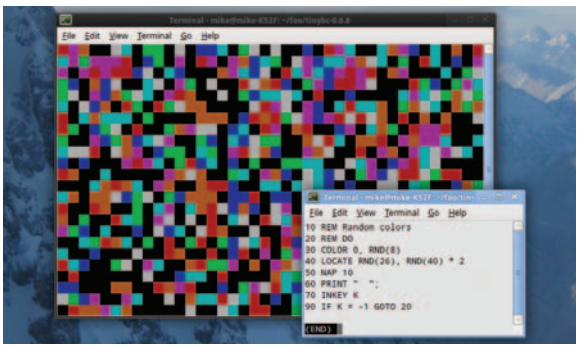
Диалекты BASIC печально известны тем, что при большой визуальной схожести они слегка несовместимы. Tiny BASIC пытается решить эту проблему, предоставляя спецификацию для крайне минималистской реализации языка – с обычными номерами строк и ключевыми словами, но без ряда функций, предполагаемых в полномасштабной реализации. Например, в Tiny BASIC нельзя задавать произвольные имена переменных: они просто идут от A до Z.

Работа в Tiny BASIC создает ностальгическую 8-битную атмосферу, когда вы выходите в пустую командную строку и начинаете там резвиться:

```
10 PRINT “Hello”
20 GOTO 10
```

(для команд нужно использовать номера строк и заглавные буквы). Есть очень полезный справочник на http://tinybc.sourceforge.net/tinybctut.txt, где объясняются функции и ограничения языка, и в Сети есть примеры программ на диалекте Tiny BASIC. Хотя вам и не создать игру уровня *Crysis* при столь скудном наборе команд, все же есть возможность поиграть с цветом и размещением на экране, так что в принципе вы сумеете сварганить скромный клон *Tetris*.

Tiny BASIC симпатичен нам по двум причинам: во-первых, благодаря получаемой от него отрадной дозе ностальгии – памяти о днях, когда к программированию компьютера приступали без возни со всякими IDE, библиотеками и тому подобной дрянью. А во-вторых, это хороший способ втянуть в программирование детей. Вы можете заспорить, что Python лучше, но Tiny BASIC уж так чист и невинен... возиться с ним – одна приятность.



**Если бы это еще и сохранялось на магнитную ленту, была бы полная иллюзия путешествия в 1980-е.**

> «Tiny BASIC чист и невинен... возиться с ним — одна приятность.»

Музыкальный плейер

# DeadBeef

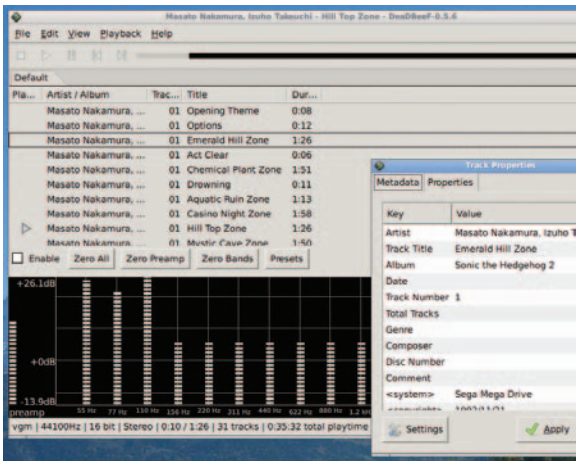
**Версия** 0.5.6 **Сайт** http://deadbeef.sourceforge.net

*DeadBeef* родом из мира языка hex [hexadecimal] – в нем шестнадцатеричные коды используются для создания человеко-читаемых слов. Вот вам простые примеры: 0xDEADBEEF, 0xBAADF00D и 0xCAFEBABE. Разработчики часто используют такие для обозначения особых мест кода, чтобы быстро находить их в шестнадцатеричном редакторе.

Но в любом случае, *DeadBeef* – один из самых многогранных музыкальных плейеров из нами виденных, и это впечатляет, учитывая конкуренцию и тот факт, что он пока что всего лишь на стадии версии 0.5.6. Разработчики называют его «идеальным музыкальным плейером», и мы бы не сказали, что они далеки от истины. *DeadBeef* поддерживает не менее 15 форматов музыкальных файлов (в зависимости от установленных плагинов), от MP3, Ogg и FLAC до файлов чиптюн [*англ.* cheap tune, музыка дешевых аудиосхем] для воспроизведения звуков из видеоигр (NSF, VGM, SPC, SID и пр.). Есть несколько достойных плейеров специально для воспроизведения чиптюн, но очень радует, что подобная поддержка встроена в «обычное» приложение. Это просто убойная функция.

Интерфейс *DeadBeef* прост в навигации и отлично настраивается: можно изменить способ сортировки песен или набор отображаемых столбцов и включить графический эквалайзер. Поддерживается отображение обложки и поиск файлов на Last.fm, а функциональность расширяют множество плагинов. Однако больше всего в *DeadBeef* нам понравилось то, что, несмотря на богатый набор функций, ему для работы не требуется ни огромных библиотек, ни инфраструктуры рабочего стола. Это голое приложение *GTK*, а значит, прекрасный выбор для старых компьютеров или при необходимости иметь минимальный менеджер окон.

Пользователи Android, возможно, видели музыкальный плейер с тем же названием в Google Play store. Он от тех же разработчиков и с похожим набором функций, но во всем прочем это другое приложение. Если решите его попробовать, поделитесь с нами своим опытом.



**По нашему мнению, музыка в *Sonic 2* была лучше, чем в *Sonic 1*, хотя игровой процесс и отставал.**

> «Интерфейс DeadBeef прост и отлично настраивается.»

Калькулятор потребления памяти

# Smem

**Версия** 1.2 **Сайт** www.selenic.com/smem

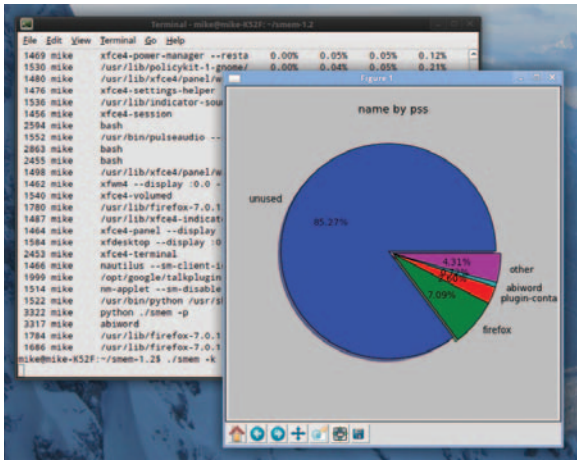
Потребление памяти – один из самых запутанных вопросов в компьютерных технологиях, и тому есть причина. Во времена, когда царили DOS и CP/M, ОЗУ либо использовалась программой, либо не использовалась. Вот так все было просто. В однозадачных операционных системах без разделяемых библиотек легко проследить, сколько памяти использует программа – но стоит перейти к современным ОС, и ваша задача намного усложняется. Например, если вы работаете с программой на *GTK* (да еще с динамическими ссылками), то в ОЗУ приходится загружать библиотеки *GTK*, что увеличивает требования к ОЗУ. Но все последующие программы на *GTK* будут использовать только загруженную в настоящий момент версию библиотеки.

Итак, все это довольно непросто. Один из лучших способов определить потребление ОЗУ программой – запустить *top*, а затем взглянуть на RES (размер, установленный для резидентной программы); он показывает, сколько физической памяти занимает программа (исключая виртуальную память). Но автору *Smem* этого недостаточно. Он говорит: «Поскольку большие порции физической памяти обычно делятся между несколькими приложениями, стандартное измерение потребления памяти, именуемое размером резидентного набора (RSS), будет значительной переоценкой». Итак, в определенных ситуациях колонка RES не особо помогает, и лучше было бы определить точную долю ОЗУ, которую схарчила та или иная программа.

Вывод *Smem* подобен колонкам использования памяти в *top*, но с дополнительной колонкой PSS (пропорциональный размер программы). Запустив ее с параметром **-k**, вы увидите вполне читаемые значения в килобайтах и мегабайтах. Для тяжеловесных приложений обнаружится значительная разница между показателями RSS и PSS – на нашей тестовой машине *Firefox* оказался в PSS на 25 МБ легче. Есть опция создания линейных и круговых диаграмм по полученным результатам – вот как она выглядит:

```
./smem -k --pie=name
```

Еще один удобный флаг – **-p**, он отображает использование памяти в процентах. Есть множество способов индивидуально настроить отображение результата – см. справочник.




**Здесь видно, что *Firefox* съел всего лишь 7,09% нашего ОЗУ — меньше, чем мы думали.**

> «Вывод Smem подобен колонкам использования памяти в top.»

---

Программа очистки домашней директории

# Mundus

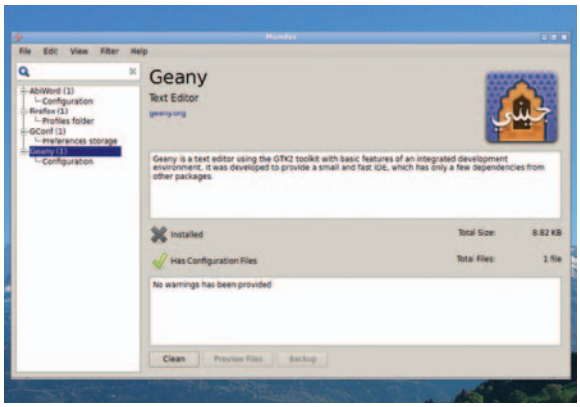
**Версия** 2.3.0 **Сайт** www.mundusproject.org

У нас в мире Linux это куда проще, чем в Windows. Взять, к примеру, настройки программы: работая в Windows, вы понятия не имеете, где ваша текущая программа хранит настройки. Они могут быть в папке Documents and Settings, или где-то еще, или внутри ужасного реестра Windows – или вообще разбросаны там и сям. Совершенно не так, как в Linux, где все весьма здраво: настройки конечного пользователя хранятся в домашней директории этого пользователя.

Но даже эта система не безупречна. Хоть программы и придерживаются одинаковой системы (то есть хранят данные настройки внутри папки с тем же именем, что и программа, но отмеченным точкой, например, **~/.thunderbird**), некоторые норовят отклониться от правил, используя другие места (например, **~/.config**). И при удалении программы из вашей системы Linux старые файлы настройки могут преспокойно уцелеть и сжирать место на диске. Введите **ls -a** на рабочем столе Linux годовалого возраста, и вы увидите, сколько у вас накопилось потенциально неиспользуемых директорий **config**...

*Mundus* борется с этим, сканируя вашу домашнюю директорию на предмет наличия следов ранее установленных программ, которых у вас больше нет, но данные настройки которых до сих пор сохраняются. У *Mundus* есть своя база данных, чтобы распознавать залежи файлов настройки, общим числом на 114 программ. Поддерживаются почти все основные приложения с открытым кодом для настольного Linux. Будучи запущен, Mundus сканирует вашу домашнюю директорию и выводит список обнаруженных настроек, заодно указывая, установлена ли соответствующая программа.

*Mundus* умеет создавать резервные копии данных перед их удалением, на случай возможной ошибки. Он прост в использовании и буквально творит чудеса с теми системами, где вы уже испробовали уйму приложений. Мы протестировали с ним нескольких наших машин и нашли-таки пожирателей диска, освободив около 150 МБ от старых файлов настройки и данных в кэше.




**У нас только один вопрос: если снести *Mundus*, есть ли другая программа для удаления его файлов настройки?**

> «У Mundus база данных, чтобы распознавать залежи файлов.»

Просмотрщик файлов журнала

# Glogg

**Версия** 0.9.1 **Сайт** http://glogg.bonnefon.org

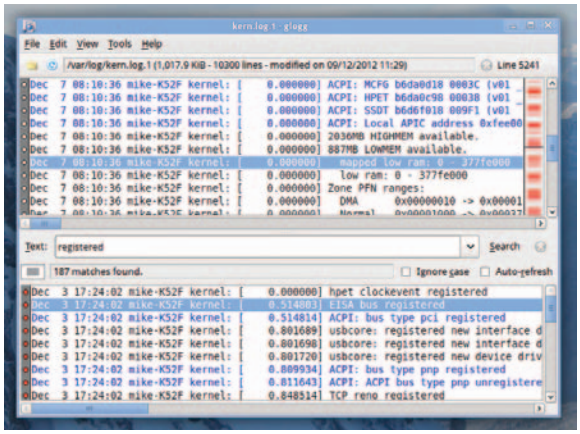
Вопрос на засыпку: что самое нечитабельное в мире Unix? Кто ответил «код Perl», тот угадал. Ну, а на пятки ему наступают файлы журналов. Да, они зависят от программы, которая создает их, и отдельные строки обычно не особо сложны для понимания, но пожалейте несчастных системных администраторов, которые по полчаса пялятся опухшими глазами в лог-файл ядра этак на 10 000 строк. Эти строки вскорости сольются у них в пятно, и легко будет прозевать нечто важное.

Правда, для поиска в журнале нужных кусков можно использовать *grep* и другие инструменты для просеивания текстов, но разве не здорово иметь для этого специально приспособленный инструмент? Им-то и является *Glogg*: он на 100 % фокусируется на просмотре и фильтрации информации лог-файла, работая в приятной манере *Qt*.

Чтобы его установить, не нужны все излишества KDE: просто распакуйте tarball и прочитайте README, чтобы познакомиться с инструкциями по компиляции.

Запустив приложение, щелкните по File > Open и найдите в **/var/log** то, что зацепит ваше внимание. Вы увидите содержимое файла в самой верхней панели; теперь введите что-нибудь в окно Text, и ниже появится новая панель, содержащая отфильтрованные результаты на основе введенного вами текста. Похоже на *grep*, но с гораздо более дружелюбным GUI.

Можно использовать в текстовом поиске регулярные выражения, и, нажав на кружочки рядом со строками в верхней панели, добавлять их к списку отфильтрованных результатов, даже если они не соответствуют исходному тексту. Еще одно приятное дополнение – цветные фильтры, доступные через меню Tools. Также имеется индикатор частоты, справа на верхней панели. Эти красные строки показывают, как часто искомый текст появляется в лог-файле – найдя при поиске слова “error” солидное скопление красного цвета, вы смекнете, что случилось нечто подозрительное. Если вы открыли лог-файл, обновляемый программой (такой, к имени которой вы обычно приписываете **tail -f**), можете поставить галочку в окне автоматического обновления, чтобы обеспечить обновление отфильтрованных результатов по мере роста файла.



**GUI хорош, но *Glogg* также поддерживает *Vi*-образные комбинации клавиш для навигации по журналу.**

> «Glogg на 100% фокусируется на просмотре и фильтрации.»

Пиксельный редактор

# GrafX2

**Версия** 2.4 **Сайт** http://code.google.com/p/grafx2/

Мир жесток, и в нем есть люди, так и не вкусившие радости обладания Amiga. Правда, в Сети до сих пор гнездятся ярые фанатики Amiga, готовые загрызть любого, кто намекнет, что дни компьютерной славы давно миновали, однако представим среднестатистического амиговладельца. В те дни, когда на ПК была убогая графика EGA и металлический монофонический звук, Amiga поражала воображение красочностью и высоким разрешением, хорошим звуком и многозадачной операционной системой, по сравнению с которой Windows той поры казался просто несерьезным.

Одной из программ, часто применяемых для демонстрации шикарных свойств Amiga, была *Deluxe Paint*, удивительно доступная графическая программа, которая позднее даже включила функции анимации. *Deluxe Paint* был одним из флагманов приложений Amiga, и часто шел вместе с машиной, так что при виде *GrafX2* мы невольно заухмылялись. Он разработан для пиксельной графики – то есть для поточечного редактирования созданных вручную изображений, а не преобразования векторов или ретуширования фото.

И он выглядит, ощущается и даже обоняется совсем как упомянутое приложение Amiga. У него собственный встроенный набор виджетов, и для его запуска нужно только установить SDL; и хотя интерфейс многим покажется ретроградным, он отличается чарующей простотой. Вы получаете ряд инструментов, размещенных в нижней панели, палитру цветов справа и доступ к дополнительным настройкам через кнопки в середине. При наведении мыши на кнопку внизу появится строка текста.

*GrafX2* использует двухцветный режим: одним цветом вы рисуете, используя левую кнопку мыши, а другим – используя правую. Здесь есть богатый выбор кистей, и когда вы добавляете текст, он тут же превращается в кисть, так что вы можете создавать подобным образом очень милые эффекты. Есть еще режим Grid (рисует в отдельных плитках мозаики, а затем объединяет их), поддержка анимации, режим двойного просмотра (с увеличением масштаба и без него), и прочая, и прочая.


***GrafX2* отлично работает как редактор спрайтов для космических стрелялок в стиле начала 1990-х.**

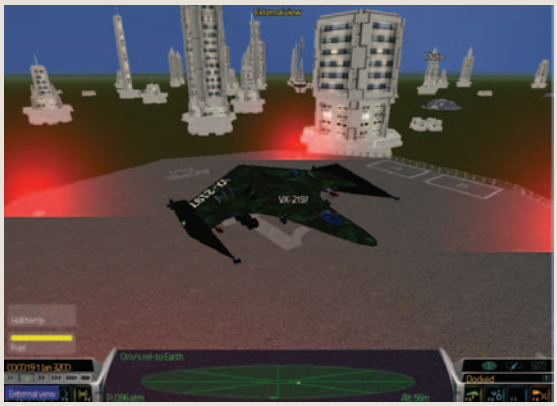
> «Он ощущается и даже обоняется, как приложение Amiga.»

## HotGames Развлекательные приложения

### Космический симулятор

# Pioneer

**Версия** alpha 28 **Сайт** http://pioneerspacesim.net



**Старый верный истребитель дальнего действия Eagle выглядит весьма привлекательно (хотя Mk3 был куда лучше).**

Мы большие фанаты *Elite* и *Frontier*, так что буквально истекли слюной, услышав о проекте Дэвида Брейбена [David Braben] на Kickstarter по возвращению франшизы к жизни. *Elite* был удивительно насыщенным космическим симулятором для 8-битной компьютерной поры: он радикально отличался от простых платформеров и паззлеров того времени, а *Frontier* зашел и дальше, обзаведясь гигантской галактикой с миллионами звездных систем и планет для пристального исследования. Но последняя игра из серии *Elite* появилась в середине 1990-х, так что мы ждали продолжения банкета много лет.

Тем временем группа фанатов *Frontier* начала работу над *Pioneer*, клоном с открытым кодом, сильно улучшенной графикой и возможностью запускаться без эмулятора на современных компьютерах (вы можете играть в PC-версию *Frontier* на *DOSBox*, но это несколько бредово). Поскольку *Pioneer* находится только на стадии альфа-28, мы не можем ожидать многого на данный момент, и, естественно, пока что в игре есть крупные недоработки. Но потихоньку начинает собираться единое целое: вы можете летать, заезжать на космодромы, забирать назначения с доски объявлений и видеть, какие интересные системы находятся в зоне видимости.

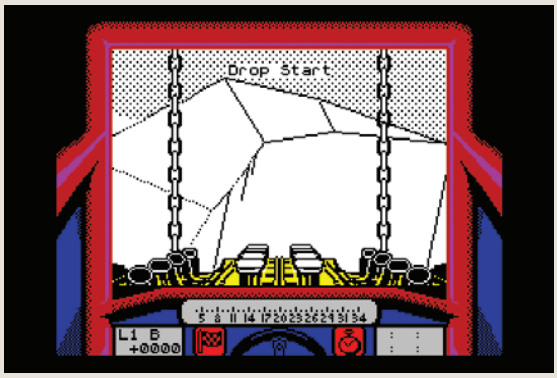
Это выглядит и ощущается очень похожим на *Frontier*, хотя разработчики старались избегать копирования Брейбеновской игры. Визуально она очень хороша, с красивыми световыми эффектами и текстурами планет, но некоторые элементы выглядят неуместно и требуют подновления. Но, что важнее всего, *Pioneer* охватывает просторы космоса, как это делал *Frontier*, и мы алчно ждем схождения с конвейера новых релизов. А вдвойне мы рады тому, что разработчики ставят ясную цель воссоздания *Frontier*: значит, они не будут отвлекаться на заманчивые идеи, способные потребовать несколько месяцев (или лет) на воплощение.

> «Вдвойне мы рады тому, что разработчики ставят ясную цель.»

### Эмулятор ZX Spectrum

# FBZX

**Версия** 2.10.0 **Сайт** www.rastersoft.com/fbzx.html




**Физика в *Stunt Car Racer* опережает свое время, да и играть в нее весьма приятно.**

Может, дело в нас, но, похоже, проблемы с эмуляторами и полноэкранным режимом стали хроническими. Мы отчетливо помним, как часами бились с эмулятором Sega Mega Drive (он же – Genesis), пытаясь получить достойное полноэкранное изображение, в итоге довольствуясь вместо него максимально развернутым окном. Все остальное или выглядело мерзко, или имело проблемы с рендерингом. Поэтому мы возликовали, увидев *FBZX*, эмулятор Speccy, большое внимание уделивший поддержке полноэкранного режима. Он несложен в установке и использовании: просто установите один пакет и введите **fbzx**, чтобы он заработал. Зависимостей немного, поскольку видео и звук обеспечивает SDL, и вам не придется устанавливать уйму программ, прежде чем приступить к работе.

Радует, что можно осуществить настройку изнутри эмулятора, нажав на клавишу F1, что вносит приятное разнообразие после изнурительной возни с файлами настройки. А можно запустить образ кассеты Spectrum или снимок памяти, вызвав соответствующий файл в командной строке – например, **fbzx stunt.tap**. В большинстве случаев игра будет запускаться автоматически, или выберите Tape Loader из собственного меню Speccy.

Изготовившись к полноэкранному действию, нажмите F9. Мы попробовали разные игры и нашли, что эмуляция отличная и отличается ровной скоростью, независимо от того, эмулирует ли *FBZX* модель 48K или +2A. А главное, прекрасно выглядит полноэкранный режим, без разрывов, странных соотношений сторон или каких-либо других проблем, с которыми мы сталкивались в наших пробах эмуляции. В общем, *FBZX* соответствует всем нашим требованиям к эмулятору: он прост в настройке, высокоточен и не путается под ногами. Другие эмуляторы Speccy могут ошеломить чрезмерным количеством настроек, выполняемых при первом запуске, но в *FBZX* этого нет, так что он получает от нас высший балл 8-битников.

> «Другие эмуляторы могут ошеломить, но в FBZX этого нет.»

Генератор календаря

# PyCalendarGen

**Версия** 0.9.4 **Сайт** http://bit.ly/W6WBJX

Сколько раз вам требовалось создать календарь для работы или для дома? Возможно, немного, но есть вероятность, что хоть раз в жизни вам такое понадобится, и вряд ли вы захотите тратить долгие часы на борьбу с *LibreOffice Writer* в попытках заставить его правильно разместить все эти крохотные квадратики и впихнуть в них текст. Нет, есть способ получше. И, в лучших традициях UNIX, есть и специальный инструментик для этой работы: неоригинально названный *PyCalendarGen*. Он написан на Python, и чтобы он заработал, надо установить модуль *mxDateTime* (в дистрибутивах на базе Ubuntu и Debian это пакет *python-egenix-mxdatetime*). Затем распакуйте файл **PyCalendarGen-0.9.4.tar.gz**, перейдите в образовавшуюся директорию и скомандуйте

```
./PyCalendarGen.py 2013 7 output.pdf
```

Учтите, что перед запуском нужно переключиться в его директорию, чтобы он смог найти свои шрифты. Здесь 2013 – это год, 7 – месяц, а **output.pdf** – файл, куда запишется результат. Как видно по экранному снимку, календари простые, но милые, и если вы не боитесь замарать руки кодом Python, цвет фона **framebgcolor** меняют в **PyCalendarGen.py**. Лучшая функция программы – настройка отдельных событий: редактируйте **days_enUS.txt**, и вы сможете добавлять текст к отдельным датам (месяц ставится спереди), например:

```
10.22 Международный день CAPS LOCK
```

**PyCalendarGen.py** допускает и другую индивидуальную настройку.


**› Включено много сугубо американских дат, но и вы легко сможете добавить важные для вас.**

Программа создания подробных сообщений об ошибках

# Libexplain

**Версия** 1.1 **Сайт** http://libexplain.sourceforge.net

Одно из любимейших нами сообщений об ошибках относится к славным дням *Netscape 4.x*. Колоссальный браузер вдруг застопорился, сотрясая диск и посылая CPU дикую нагрузку. Секунд через 20 безумия приложение умерло, оставив ценное сообщение: «Ошибка шины [Bus error]». Разработчиков явно не волновала обратная связь.

Функция **strerror()**, которая генерирует полезный текст, вам, возможно, знакома: скажем, вы пытаетесь открыть файл в своей программе C, а путь указали неверный – тогда **strerror()** выдаст нечто вроде

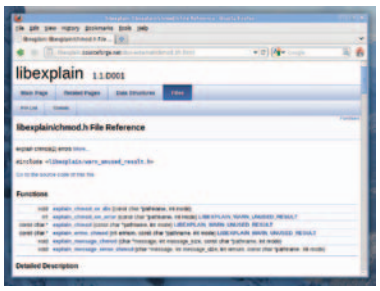
```
Такой директории нет [No such file or directory]
```

*Libexplain* идет дальше и добавляет в программы C более содержательные сообщения об ошибках, применяя **explain_open_or_die()**. Если запрашиваемый файл или директория не могут быть найдены, вы получаете полезную информацию:

```
open(pathname = “testdir/testfile”, flags = O_RDONLY) failed, No such file or directory (2, ENOENT) because there is no “testdir” directory in the current directory
```

[*Не удалось выполнить open(pathname = “testdir/testfile”, flags = O_RDONLY), такой директории нет (2, ENOENT), поскольку директории “testdir”нет в текущей директории*].

Это намного удобнее при отладке, потому что предоставляет конечным пользователям основания для выводов. *Libexplain* включает 185 функций, работающих в качестве оболочек для многих его стандартных библиотек. LXF




**› Арсенал средств *libexplain* по борьбе с чрезмерной лаконичностью очень хорошо задокументирован.**

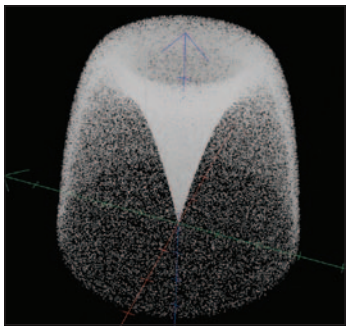
## Также вышли

**Новые и обновленные программы, тоже достойные внимания...**

**Kite 0.1.0**
Новый язык программирования, предназначенный для быстрой разработки приложений.
www.kite-language.org

**CLFSWM 1212**
Минималистский полноэкранный менеджер окон, написанный на Common Lisp.
http://bit.ly/a4evMb

**PyParticles 0.3.4**
Основанный на Python набор инструментов для моделирования частиц на OpenGL.
http://pyparticles.wordpress.com

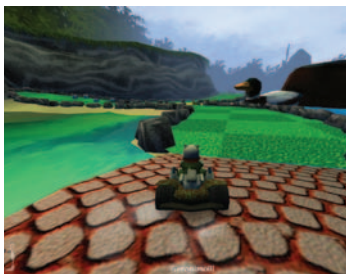


**› Взрывайте и разрушайте все, как безумный, прямо из-за стола.**

**BPM tools 0.2**
Наведите порядок в своей музыкальной коллекции, основываясь на тактах в минуту.
http://bit.ly/VEtPBv

**Likwid 3.0**
Инструменты для разработки супер-быстрых многопотоковых программ.
http://code.google.com/p/likwid/

**SuperTuxKart 0.8**
3D-гонки на картингах, теперь с режимом истории и усовершенствованным соперником-роботом.
supertuxkart.sourceforge.net



**› Всеобщий любимец мчится к версии 1.0 в стиле *Mario Kart*.**

# На диске

## Пробуйте новую операционную систему уже сегодня!

**DVD — это тонкая полоска металла между слоями поликарбоната. А металл изрыт крошечными ямками глубиной всего 120 нанометров. 120 нм — это четверть длины волны красного света, применяемого в лазерах DVD (в поликарбонате). Луч, который попадает в отверстие и отражается от его дна, пробегает на половину длины волны дальше, чем неотраженный. Поскольку лучи в лазерах сфокусированы, часть лучей оказывается в противофазе с остальными, и при суммировании они взаимно уничтожаются. Мы использовали этот трюк, когда свет+свет=тьма, свыше четырех с половиной миллиардов раз на DVD этого месяца. Наслаждайтесь!**

### Клем и его команда занялись KDE

## Linux Mint KDE

Да, у нас уже был Linux Mint в прошлом выпуске, но мы не удержались от соблазна капнуть Mint'олом в DVD этого месяца. Включенная в него версия KDE переносит подход Mint на рабочий стол, основанный на *Qt*. По умолчанию он имеет минимальную среду, а вы можете превратить ее в идеальную среду KDE.

Естественно, идеальный KDE у каждого свой. KDE, как никакая другая среда, дает колоссальные возможности индивидуальной настройки. Широкий выбор виджетов рабочего стола позволяет сделать рабочий стол интерактивным. У редактора данного диска есть виджеты для получения прогноза погоды, мониторинга устройств USB и отображения содержимого его папки Downloads.

KDE даже выходит за пределы компетенции рабочего стол, и включает пакет приложений *Qt*. Однако команда Mint не ограничилась исключительно приложениями KDE. Например, в качестве офисного пакета *LibreOffice* предпочли *Calligra*, а *Firefox* был выбран вместо *Reconq*.

Ну, а для пуристов KDE все родные программы находятся не далее **apt-get**.


**› Рабочий стол Mint KDE — это чистый холст для ваших творений.**

### KDE плюс web-приложения

## Netrunner

Если кто не собразил по названию, Netrunner – это дистрибутив для любителей работать в Сети. Облачные учетные записи и web-приложения представлены здесь весьма широко, хотя и не до такой степени, как в ChromeOS или Peppermint.

Выбор KDE вместо более легковесного рабочего стола может показаться странноватым, пока вы не узнаете, что дистрибутив собран Blue Systems, загадочной компанией, которая поддерживает Mint KDE и Kubuntu.

Настройки по умолчанию не сложнее, чем в Mint KDE, но мы нашли, что они прекрасно работают, хотя украшения окон довольно тяжеловесны. У него нет того уровня интеграции, как, скажем, у ChromeOS, но зато он и не привязан к одному производителю. Вероятно, самый необычный аспект этого сетевого подхода – менеджер пакетов. Хотя в нем по-прежнему остаются опции *Muon* на основе **apt**, появляется также и *jacknjoe.com*. Он использует протокол **apt**: для установки программ через ваш web-браузер.

**› Долгое время *Qt* был инструментарием для мобильного телефона, так что он отлично вписался в этот ориентированный на Сеть дистрибутив.**

## Посреди Fedora и Ubuntu

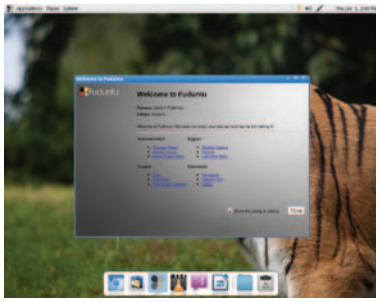
# Fuduntu

Fuduntu все никак не удавалось попасть на всеми желанное пространство **LXFDVD**, и поэтому мы решили, что пора уделить ему внимание. Это популярный дистрибутив, незаслуженно оказавшийся в тени. Его интерфейс – Gnome 2 старой школы, но это вовсе не значит, что система архаична. Просто разработчикам нравится ретро-стиль.

Несмотря на название, он не основан ни на Fedora, ни на Ubuntu. Когда-то давно его основали на Fedora (приняв RPM для управления пакетами), и предпринимались попытки его Ubuntu'зировать. Однако с тех пор он оторвался от своих родителей и стал самостоятельным дистрибутивом.

Fuduntu предназначен для среднего пользователя рабочего стола и не скрывает этого, и пакеты вроде *netflix-desktop* позволяют получить быстрый доступ к функциям, которые могут представлять некоторую сложность в настройке.

А если этого недостаточно, чтобы убедить вас его попробовать, то вот вам: на рабочем столе у него красуется картинка с тигром!



**Да здравствует Gnome2!**

## Важно ВНИМАНИЕ!

**Прежде чем вставить DVD в дисковод, пожалуйста, убедитесь, что вы прочитали, поняли и согласились с нижеследующим.**

Диски *Linux Format* DVD тщательно проверяются на предмет отсутствия на них всех известных вирусов. Тем не менее, мы рекомендуем вам всегда проверять любые новые программы надежным и современным антивирусом.

Хотя процесс отбора, тестирования и установки программ на DVD проводится со всем тщанием, редакция *Linux Format* не несет никакой ответственности за повреждение и/или утрату данных или системы, могущее произойти при использовании данного диска, программ или данных на нем. Настоятельно рекомендуем вам создавать своевременные и надежные резервные копии всех важных файлов.

Чтобы узнать об условиях использования, просим вас прочесть лицензии.

### Бракованные диски

В маловероятном случае обнаружения бракованного диска *Linux Format*, просим связаться с нашей группой поддержки по адресу disks@linuxformat.ru, для получения содействия.

## В жизни Linux не один. Есть еще BSD

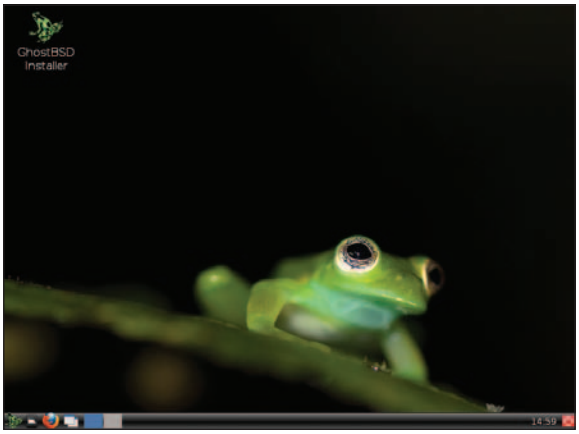
# GhostBSD

Какая разница между BSD и Linux? Ну, в зависимости от того, кого вы спросите, ответы будут: очень маленькая; огромная; или нечто промежуточное. На самом нижнем уровне, BSD использует другое ядро, и по этой причине не является формой Linux, и большинство системных программ здесь другие.

Но поскольку обе эти ОС – отпрыски UNIX, функционируют они очень похоже. Если вы знакомы с Linux, в BSD вы, вероятно, почувствуете себя как дома. Однако в настольных системах есть некоторые различия. Вероятно, самое большое из них кроется в структуре проекта. В Linux одна группа разработчиков трудится над ядром, другая – над *systemd*, третья – над... ну, вы поняли. Linux – это не столько проект, сколько коллекция самостоятельных проектов, которыми может воспользоваться каждый дистрибутив для создания операционной системы.

В BSD есть команда NetBSD, которая работает над ядром NetBSD и всеми программами NetBSD; есть команда FreeBSD, работающая над ядром и всеми системными программами FreeBSD, и т.д. Сплоченная система BSD куда лучше скоординирована, чем в Linux.

Однако Linux привлек намного больше разработчиков из разных организаций к совместной работе над системой, пусть




**Интерфейс GhostBSD LXDE почти неотличим от любого дистрибутива Linux на базе *LXDE*.**

даже это и выглядит сборной солянкой. Мы включили GhostBSD, поскольку она проста в использовании и загружает графическую среду.

## На диске

Помимо четырех симпатичных дистрибутивов, мы включили в состав диска весь код из руководств, все программы из Hotpicks и все программы, упомянутые в журнале, рядом с которыми вы видите значок **На диске**. Если вы вставите DVD в привод, он должен открыть страницу HTML по умолчанию, но если ваши настройки безопасности не позволяют этого, перейдите в **index.html**, чтобы открыть для себя все 4,4 ГБ прелестей Linux.

Чтобы познакомиться с дистрибутивами, установите диск в дисковод, затем перезагрузите компьютер. После этого загрузится экран, где вы сможете выбрать то, что вам нужно. Если ваш компьютер загружается в обычном режиме, а не с диска, вам придется изменить настройки BIOS на загрузку с DVD. Для тех, кому захочется перенести содержимое диска на устройство USB, мы включили удобный скрипт. Чтобы узнать, как им пользоваться, загляните в **dvd2usb.html** на диске. Цифровые подписчики при желании могут применить **dd** для перемещения загруженного ISO на устройство USB:

```
sudo dd if=lxf166.iso of=/dev/sdX
```

при необходимости включив путь к загруженному ISO, где X изменяется согласно вашему устройству USB. **Это уничтожит все данные на диске. Если вы установите не тот диск, он может стереть все данные с вашего жесткого диска.** LXF

# Информация о диске

Внимательно прочтите это перед тем, как использовать LXFDVD!

## Что-то потеряли?

Часто случается, что новые программы зависят от других программных продуктов, которые могут не входить в текущую версию вашего дистрибутива Linux.

Мы стараемся предоставить вам как можно больше важных вспомогательных файлов. В большинстве случаев, последние версии библиотек и другие пакеты мы включаем в каталог «Essentials [Главное]» на прилагаемом диске. Поэтому, если в вашей системе возникли проблемы с зависимостями, первым делом следует заглянуть именно туда.

## Форматы пакетов

Мы стараемся включать как можно больше различных типов установочных пакетов: RPM, Deb или любых других. Просим вас принять во внимание, что мы ограничены свободным пространством и доступными двоичными выпусками программ. По возможности, мы будем включать исходные тексты для любого пакета, чтобы вы могли собрать его самостоятельно.

## Документация

На диске вы сможете найти всю необходимую информацию о том, как устанавливать и использовать некоторые программы. Пожалуйста, не забывайте, что большинство программ поставляются вместе со своей документацией, поэтому дополнительные материалы и файлы находятся в соответствующих директориях.

## Что это за файлы?

Если вы новичок в Linux, вас может смутить изобилие различных файлов и расширений. Так как мы стараемся собрать как можно больше вариантов пакетов для обеспечения совместимости, в одном каталоге часто находятся два или три файла для различных версий Linux и различных архитектур, исходные тексты и откомпилированные пакеты. Чтобы определить, какой именно файл вам нужен, необходимо обратить внимание на его имя или расширение:

- **имя_программы-1.0.1.i386.rpm** – вероятно, это двоичный пакет RPM, предназначенный для работы на системах x86;
- **имя_программы-1.0.1.i386.deb** – такой же пакет, но уже для Debian;
- **имя_программы-1.0.1.tar.gz** – обычно это исходный код;
- **имя_программы-1.0.1.tgz** – тот же файл, что и выше этажом по списку: “tgz” – это сокращение от “tar.gz”;
- **имя_программы-1.0.1.tar.bz2** – тот же файл, но сжатый bzip2 вместо обычного gzip;
- **имя_программы-1.0.1.src.rpm** – также исходный код, но поставляемый как RPM-пакет для упрощения процесса установки;
- **имя_программы-1.0.1.i386.FC4.RPM** – двоичный пакет RPM для x86, предназначенный специально для операционной системы Fedora Core 4;
- **имя_программы-1.0.1.ppc.Suse9.rpm** – двоичный пакет RPM, предназначенный специально для операционной системы SUSE 9.x PPC;
- **имя_программы-devel-1.0.1.i386.rpm** – версия для разработчиков.

## Если диск не читается...

Это маловероятно, но если все же прилагаемый к журналу диск поврежден, пожалуйста, свяжитесь с нашей службой поддержки по электронной почте: disks@linuxformat.ru

# Содержание

LINUX FORMAT

## Сторона 1

### ДИСТРИБУТИВЫ

**GhostBSD 2.5** Дистрибутив семейства BSD UNIX с рабочим столом LXDE, 32-разрядная сборка (ISO-образ)
**Fuduntu 2012.4** Дистрибутив, представляющий собой нечто среднее между Ubuntu и Fedora, с рабочим столом Gnome 2, 32-разрядная сборка (загрузка с **LXFDVD**)
**Linux Mint 14** С рабочим столом KDE, 32-разрядная сборка (загрузка с **LXFDVD**)
**Netrunner 12.12** Дистрибутив на базе Ubuntu, предназначенный для работы с сетевыми сервисами, 32-разрядная сборка (загрузка с **LXFDVD**)

## Сторона 2

### ДИСТРИБУТИВЫ

**Linux Mint 14** Установочные DVD (ISO-образы)
**С рабочим столом KDE** – 32- и 64-разрядные сборки
**С рабочим столом Xfce** – 32- и 64-разрядные сборки

### ДОКУМЕНТАЦИЯ: 10 КНИГ О LINUX (НА АНГЛИЙСКОМ ЯЗЫКЕ)

**Bash Scripting** Подробное руководство по программированию на Bash
**Bourne Shell Scripting** Начальное руководство по программированию на Bash
**Cathedral Bazaar** Классический текст Эрика Раймонда [Eric S Raymond] «Собор и базар»
**The Debian Administrator's Handbook** Руководство администратора, написанное разработчиками Debian
**Dive Into Python** Учебник по программированию на Python
**Intro to Linux** Начальное руководство по Linux
**Linux Dictionary** Словарь Linux, объясняющий специфическую терминологию
**Linux Kernel in a Nutshell** Описание ядра Linux, созданное одним из его выдающихся разработчикоов – Грегом Кроа-Хартманом [Greg Kroah-Hartman]
**System Administrators Guide** Руководство по базовому администрированию Linux
**GNU Tools Summary** Руководство по работе в командной строке и обзор основных утилит GNU

### HOTPICKS

**Deadbeef 0.5.6** Проигрыватель музыки, написанный на GTK
**FBZX 2.10** Эмулятор ZX Spectrum, работающий в полноэкранном режиме
**Fraqtive 0.4.6** Программа для рисования фракталов
**Glogg 0.9** Программа для просмотра лог-файлов
**Grafx 2.4** Редактор пиксельной графики
**LibExplain 1.1** Программа для вывода подробных сообщений об ошибках
**Mundus 2.3** Утилита для удаления неиспользуемых файлов конфигурации
**Pioneer alpha 29** Игра, космическая стратегия XXXI века
**PyCalendarGen 0.9.5** Программа для печати календарей
**Smem 1.2** Средство мониторинга потребления оперативной памяти
**TinyBC 0.8** Интерпретатор BASIC, написанный на Curses

**Пожалуйста, перед использованием данного диска ознакомьтесь с опубликованной в журнале инструкцией!**

**КОММЕНТАРИИ?** Присылайте ваши пожелания и предложения по электронной почте: info@linuxformat.ru

**ДЕФЕКТНЫЕ ДИСКИ** В маловероятном случае обнаружения дефектов на данном диске, обращайтесь, пожалуйста, по адресу disks@linuxformat.ru



Тираж изготовлен ООО «Уральский электронный завод». 620137, Россия, г. Екатеринбург, Студенческая ул., д. 9. Лицензия МПТР ВАФ 77-15.

## Создание установочных дисков при помощи cdrecord

Самый быстрый способ записать ISO-образ на чистую матрицу – это через *cdrecord*. Для всех перечисленных ниже действий потребуются права root. Сначала определите путь к вашему устройству для записи дисков. Наберите следующую команду:

```
cdrecord -scanbus
```

После этого на экране терминала должен отобразиться список устройств, подключенных к вашей системе. SCSI-адрес каждого устройства представляет собой три числа в левой колонке – например, 0,3,0. Теперь вы можете с легкостью записать образ на диск:

```
cdrecord dev=0,3,0 -v /путь к образу/image.iso
```

Чтобы упростить дальнейшее использование *cdrecord*, сохраните некоторые настройки в файле **/etc/default/cdrecord**. Добавьте по одной строке для каждого устройства записи (вероятно, в вашей системе присутствует всего одно такое устройство):

```
Plextor= 0,3,0 12 16M
```

Первое слово в этой строке – метка; затем после адреса SCSI-устройства вы должны указать скорость и размер буфера. Теперь можете заменить SCSI-адрес в командной строке на выбранную вами метку. Все будет еще проще, если вы добавите следующее:

```
CDR_DEVICE=Plextor
```

Все, что вам теперь нужно для записи ISO-образа – это набрать команду

```
cdrecord -v /path/to/image.iso
```

Если вы не из числа любителей командной строки, в таком случае вам придет на помощь утилита *gcombust*. Запустите ее из-под root и выберите вкладку Burn и ISO 9660 Image в верхней части окна. Введите путь к образу, который вы хотите записать на диск, и смело нажимайте на Combust! Пока ваш образ пишется на диск, можете выпить чашечку кофе.

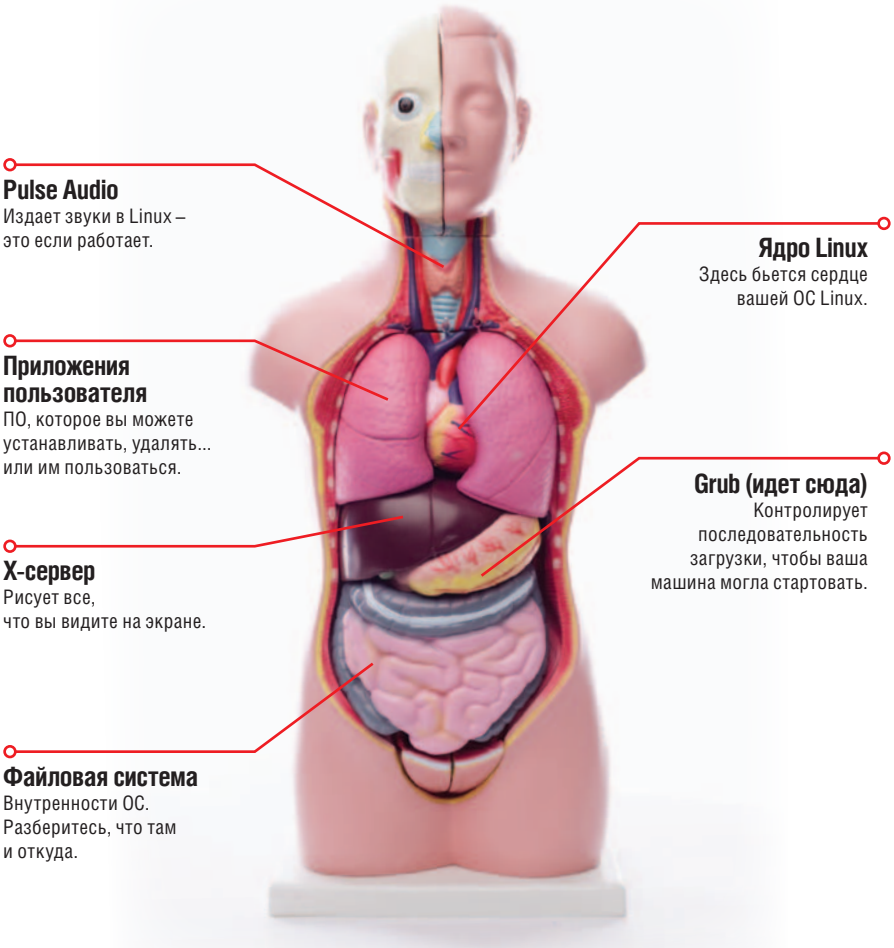
### Другая ОС?

Вам не обязательно использовать Linux для записи компакт-диска. Все необходимые файлы уже включены в ISO-образ. Программы вроде *cdrecord* просто переносят данные на чистую матрицу. Если у вас нет устройства для записи дисков, можно найти того, у кого оно есть, и записать диск на его компьютере с Windows, Mac OS X, AmigaOS или любой другой ОС.

### Нет устройства для записи дисков?

А что если у вас нет устройства, с помощью которого можно было бы записать образ на диск? Вы знаете кого-нибудь с таким устройством? Вам не обязательно использовать Linux для записи дисков: подойдет любая операционная система, способная распознать пишущий привод (см. выше).

Некоторые дистрибутивы умеют монтировать образы дисков и выполнять сетевую установку или даже установку с раздела жесткого диска. Конкретные методы, конечно, зависят от дистрибутива. За дополнительной информацией обращайтесь на web-сайт разработчика дистрибутива.




# В апрельском номере

## Анатомия Linux

Откройте таинства мистического возникновения вашей системы Linux, разберите ее на части и соберите снова. Все как в анатомии!

## Версии для Raspberry Pi

Попробуйте новые варианты Linux на своем Raspberry Pi, или сойдите с проторенной дорожки ради другой операционной системы.

## Steam для Linux

Игры высокого полета шагают в Linux! Срочно приведите свою матчасть и драйвера в порядок и будьте наготове.

## Hotpicks

Что может быть лучше шести страниц про блистательное свободное ПО, которые курирует блистательный сэр Майк Сондерс?

Содержание будущих выпусков может меняться – вдруг мы погрузимся в бесконечные вложения виртуализации...

LINUX FORMAT
Главное в мире Linux

Журнал зарегистрирован Федеральной службой по надзору за соблюдением законодательства в сфере массовых коммуникаций и охране культурного наследия
ПИ № ФС77-21973 от 14 сентября 2005 года
Выходит ежемесячно. Тираж 3000 экз.

РЕДАКЦИЯ РУССКОЯЗЫЧНОЙ ВЕРСИИ
**Главный редактор**
Кирилл Степанов info@linuxformat.ru
**Литературный и выпускающий редактор**
Елена Толстякова
**Переводчики**
Елена Ессяк, Светлана Кривошеина, Валентин Развозжаев, Елена Толстякова
**Редактор диска**
Кирилл Степанов
**Верстка, допечатная подготовка**
Сергей Рогожников
**Технический директор**
Денис Филиппов
**Директор по рекламе**
Владимир Савельев advert@linuxformat.ru
**Генеральный директор**
Павел Фролов
**Учредители**
Частные лица
**Издатели**
Виктор Федосеев, Павел Фролов

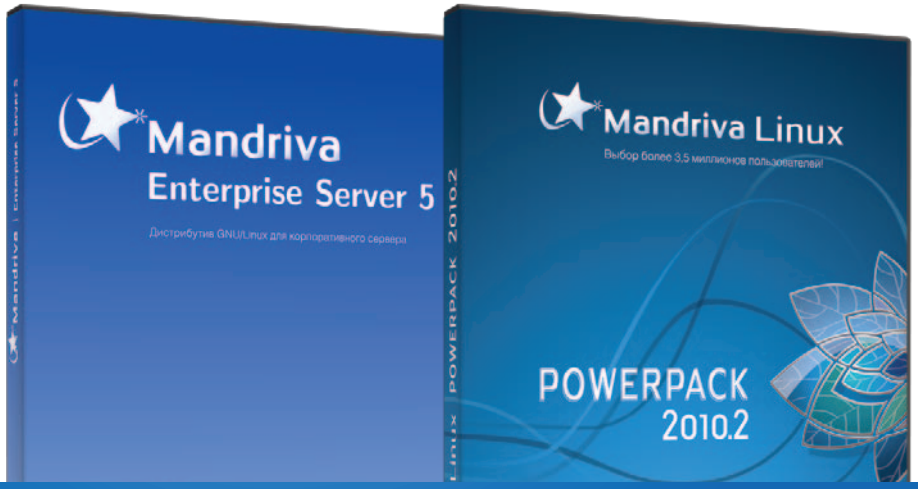
**Отпечатано в ООО «ЛапинЪ»**
188330, Ленинградская обл., Гатчинский р-н, пос. Сиверский, Вокзальная ул., 4
Заказ 7653

РЕДАКЦИЯ АНГЛОЯЗЫЧНОЙ ВЕРСИИ
**Редактор** Грэм Моррисон [Graham Morrison] graham.morrison@futurenet.com
**Заместитель редактора** Эндрю Грегори [Andrew Gregory] agregory@futurenet.com
**Редактор диска** Бен Эверард [Ben Everard] ben.everard@futurenet.com
**Художественный редактор** Эфраим Эрнандес-Мендоса [Efrain Hernandez-Mendoza] efrain.hernandez-mendoza@futurenet.com
**Выпускающий редактор** Гэри Уокер [Gary Walker] gary.walker@futurenet.com

ПОДГОТОВКА МАТЕРИАЛОВ
Джоно Бэкон [Jono Bacon], Нейл Ботвик [Neil Bothwick], Крис Браун [Chris Brown], Алекс Кокс [Alex Cox], Бен Эверард [Ben Everard], Дэвид Хейвард [David Hayward], Джон Лэйн [John Lane], Грэм Моррисон [Graham Morrison], Крис Нотли [Chris Notley], Адам Оксфорд [Adam Oxford], Джонатан Робертс [Jonathan Roberts], Майк Сондерс [Mike Saunders], Маянк Шарма [Mayank Sharma], Шашанк Шарма [Shashank Sharma], Ник Вейч [Nick Veitch], Евгений Балдин, Артем Зорин, Михаил Остапкевич, Павел Сёмин, Андрей Ушаков, Алексей Федорчук, Игорь Штомпель

**Художественные ассистенты** Стейси Блэк [Stacey Black], Джон Мак-Аллистер [John McAllister], Мэтт Ортон [Matt Orton]
**Иллюстрации** Шейн Коллиндж [Shane Collinge], Elly Walton Illustrations, iStockPhoto, Саймон Миддлвик [Simon Middleweek]
**Фото** Джейсон Каплан [Jason E.Kaplan]

КОНТАКТНАЯ ИНФОРМАЦИЯ
**UK:** Linux Format, 30 Monmouth Street, Bath BA1 2BW
Tel. +44 01225 442244 Email: linuxformat@futurenet.com
**РОССИЯ:**
**Санкт-Петербург (редакция):**
Лиговский пр., 50, корп. 15
Тел. +7 (812) 309-06-86
**Представительство в Москве:**
Красноказарменная ул., 17, м. «Авиамоторная» (в помещении АТС МЭИ)
Тел./факс +7 (499) 271-49-54
По вопросам сотрудничества, партнерства, оптовых закупок:
partner@linuxcenter.ru




Mandriva
www.mandriva.ru

Mandriva Linux — один из самых популярных дистрибутивов GNU/Linux в мире. Главные преимущества Mandriva — дружелюбный интерфейс, простота настройки, возможность быстрой адаптации пользователей, ранее не знакомых с этой ОС, совместимость с широким спектром программного и аппаратного обеспечения.

## Корпоративные продукты Mandriva Linux

### Mandriva 2010.2 Powerpack

Дистрибутив Mandriva 2010.2 Powerpack включает набор офисных и серверных приложений, и подходит для установки на офисной или домашней рабочей станции и на сервере. Дружелюбный интерфейс, простота настройки Mandriva Powerpack, совместимость с широким спектром аппаратного обеспечения и совместимость с «1С:Предприятие» обеспечивают корпоративным пользователям возможность легкого перехода с Windows на GNU/Linux.

### Mandriva Enterprise Server 5

Mandriva Enterprise Server 5 (MES 5) — это надежный и производительный дистрибутив GNU/Linux для корпоративного сервера. MES 5 поможет вам снизить текущие расходы и упростить инфраструктуру. В MES 5 интегрированы серверные разработки программистов Mandriva, в том числе сервер каталогов пользователей Mandriva Directory Server, а также ведущие свободные серверные приложения, которые помогут вам с минимумом затрат времени и энергии настроить и поддерживать необходимые вам серверы. Срок поддержки дистрибутива — 5 лет.

## Сертифицировано ФСТЭК

Дистрибутивы Mandriva Linux **сертифицированы по требованиям ФСТЭК по 5 классу для СВТ и 4 уровню контроля НДВ**, что дает возможность использовать их для обработки конфиденциальной информации в автоматизированных системах класса до 1Г включительно и обработки персональных данных в информационных системах класса до К2 включительно.

- **Mandriva 2008 Spring Powerpack** — дистрибутив для рабочей станции или небольшого сервера.
- **Mandriva Corporate Server 4 Update 3** — дистрибутив для создания корпоративного сервера.
- **Mandriva Flash** — дистрибутив GNU/Linux, загружающийся и работающий прямо с USB-носителя.

## EduMandriva — свободное ПО для образования

- Создано с участием российских преподавателей и методистов.
- Все ПО, необходимое для преподавания информатики.
- Методические материалы.

| Наименование | Стоимость, руб. |
|---|---|
| **Корпоративные продукты Mandriva** | |
| Mandriva Linux 2010.2 Powerpack (DVD-Box) | 1 300 |
| Услуга подписки на Mandriva Enterprise Server 5 на 1 год, базовый уровень (с физическим носителем) | 13 300 |
| Услуга подписки на Mandriva Enterprise Server 5 на 3 года, базовый уровень (с физическим носителем) | 34 800 |
| **Продукты Mandriva для образования** | |
| Комплект программного обеспечения Mandriva Linux и EduMandriva для школ | 3 500 |
| **Сертифицированные ФСТЭК продукты Mandriva** | |
| Сертифицированный ФСТЭК Mandriva 2008 Spring Powerpack на 10 рабочих мест | 28 500 |
| Сертифицированный ФСТЭК Mandriva 2008 Spring Powerpack на 5 рабочих мест | 14 500 |
| Сертифицированный ФСТЭК Mandriva 2008 Spring Powerpack на 1 рабочее место | 4 990 |
| Сертифицированный ФСТЭК Mandriva Corporate Server 4.0 Update 3 | 10 050 |

**MANDRIVA УЖЕ ИСПОЛЬЗУЮТ:**
МВД РФ, Минздравсоцразвития РФ, Минфин республики Саха (Якутия), Правительство Московской области, администрация Черниговского района Приморского края, ОАО «Морион», сеть магазинов «Компьютер-центр «КЕЙ», группа компаний «ИМАГ», компания «Азбука мебели», и многие другие.

С вопросами по продуктам Mandriva обращайтесь в **ГНУ/Линуксцентр**!

Москва
+7 (499) 271-49-54

Санкт-Петербург
+7 (812) 309-06-86

Linux-эксперт для вашего бизнеса. www.linuxcenter.ru

Linux center

# Новое поколение средств защиты

## Межсетевые экраны ССПТ, не имеющие IP-адреса

ССПТ-2 — это сертифицированное ФСТЭК, ФСБ и ГАЗПРОМСЕРТ средство защиты информации нового поколения, реализующее функции межсетевого экрана, но при этом остающееся «невидимым» для любых протоколов и тестовых воздействий, что достигается за счет отсутствия физических и логических адресов на его фильтрующих интерфейсах. **ССПТ-2 невозможно обнаружить никакими известными средствами удаленного мониторинга сети.**

Скрытность функционирования межсетевого экрана повышает надежность системы защиты в целом и существенно упрощает процедуру установки ССПТ-2 в компьютерные сети и функционирующие на их основе информационные и телематические системы.


## Назначение устройства

**Основное средство защиты для реализации различных политик информационной безопасности с помощью:**

- фильтрации пакетов на канальном, сетевом, транспортном и прикладном уровнях;
- управления транспортными соединениями между отдельными узлами ЛВС или виртуальной ЛВС (VLAN);
- контроля контента данных на прикладном уровне с учетом направления, времени и типа протоколов передачи трафика.

**Дополнительное устройство защиты для:**

- обеспечения безопасности функционирования ранее установленных в компьютерной сети средств защиты и устройств маршрутизации;
- мониторинга трафика с возможностью анализа данных регистрации пакетов по различным критериям и интеграции с IDS;
- обеспечения функционирования сетевых распределенных телематических приложений и GRID-ресурсов.