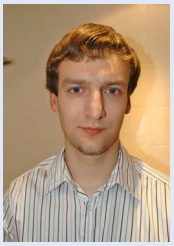
Колонка главного редактора



Несмотря на те или иные формальные требования, которые могут быть предъявлены к этой регулярной колонке, сложно идти против вещей вечных. Я имею в виду число 42, которое наконец-то настигло наше издание. Если кто-нибудь вдруг не в курсе, напомню, что согласно известной шутливой теории 42 – это «Ответ на главный вопрос жизни, Вселенной и всего такого». Данное явление пользуется особой популярностью, например, среди интернет-пользователей. Этим в свое время не преминули воспользоваться в Google, добавив «Ответ...» в свой онлайн-калькулятор, а не так давно и в «Яндексе», запустив 42.yandex.ru. Если же воспользоваться возможностями поисковых гигантов по их прямому назначению, не составит труда найти невероятное количество ссылок на загадку 42 и бесконечные совпадения, старательно отобранные энтузиастами со всего мира и связывающие это число с любыми аспектами жизни.

Так или иначе номер текущего выпуска «Open Source» явным образом свидетельствует о том, что в нем содержится абсолютный ответ на те вопросы, которые могут возникнуть у читателей. Поэтому если вам показалось иначе, для начала попробуйте изменить свои представления о жизни и только в крайнем случае присоединяйтесь к дискуссиям на нашем форуме (http://osa.samag.ru/forum).

Главный редактор
Дмитрий Шурупов
(osa@samag.ru)

«Open Source»
электронное приложение к журналу «Системный администратор»
№42, 13 апреля 2009 г.

РЕДАКЦИЯ
Исполнительный директор
Владимир Положевец
Главный редактор
Дмитрий Шурупов
Верстка и оформление
Владимир Лукин
Сайт электронного приложения:
http://osa.samag.ru

Новости мира Open Source

Python переходит на систему управления версиями Mercurial

Гвидо ван Россум (Guido van Rossum), автор языка программирования Python, объявил в почтовой рассылке python-dev о том, что определился с выбором децентрализованной системы управления версиями (DCVS) для своего проекта. Теперь разработка Python будет осуществляться с помощью Mercurial.

Mercurial – стремительно набирающая популярность распределенная система управления версиями, пользующаяся особым спросом среди разработчиков Python. Ее ближайший конкурент Bazaar поддерживается компанией Canonical. По словам Гвидо, в которых он подчеркивает субъективность суждений, Bazaar – менее предпочтительный для сообщества выбор. К тому же эта система более медлительна при выполнении большинства операций и сложнее для изучения бывшими пользователями Subversion.

До сих пор в Python использовали для разработки систему Subversion, однако последние тенденции мира крупных Open Source-проектов показывают, что разработчики тяготеют к переходу на децентрализованные системы вроде Git и уже упомянутых Mercurial и Bazaar. О точных сроках переноса репозитория Python на Mercurial пока не сообщается.

Debian теперь официально поддерживает ядро FreeBSD

Проект популярного дистрибутива Debian GNU/Linux объявил о появлении двух новых архитектур в своем официальном архиве: kfreebsd-i386 (GNU/kFreeBSD i386) и kfreebsd-amd64 (GNU/kFreeBSD amd64). Теперь Debian официально предлагает своим пользователям на выбор ядра двух разных операционных систем: Linux и FreeBSD.

Вообще, проект Debian GNU/kFreeBSD существует уже давно. Его цель – создать готовый дистрибутив, позволяющий запускать привычное для Debian окружение из приложений проекта GNU поверх ядра операционной системы FreeBSD. Фактически, получается «все тот же Debian GNU/Linux», но вместо «традиционного» ядра Linux в нем используется ядро другой популярной свободной ОС – FreeBSD. Приложения GNU, в свою очередь, хорошо известны пользователям обеих систем.

Теперь сборки Debian GNU/kFreeBSD для архитектур процессора i386 и amd64 признаны официальными, хотя пока считаются «нестабильными» и «экспериментальными».

Intel передает Linux Foundation управление проектом Moblin

Компания Intel передала некоммерческой организации Linux Foundation право руководить разработкой своего Linux-дистрибутива Moblin для портативных устройств.

Moblin (Mobile & Internet Linux Project) – это Open Source-инициатива Intel, запущенная в 2007 году и нацеленная на создание нового Linux-дистрибутива, ориентированного на использование в мобильных устройствах вроде нетбуков. До сих пор его координацией занимались сотрудники Intel, но теперь в компании решили, что развитие лучше продолжить в рамках мирового сообщества, передав управление Moblin некоммерческому консорциуму Linux Foundation.

Как отмечается в пресс-релизе Linux Foundation, «с технической поддержкой уважаемых разработчиков Linux-ядра и независимым, сторонним, руководством проект Moblin нацелен на то, чтобы стать самой продвинутой и открытой мобильной Linux-платформой».

Первая встреча разработчиков проекта Moblin, организованная уже усилиями Linux Foundation, пройдет 8 апреля на мероприятии Annual Collaboration Summit.

В Openmoko приостановили работы над следующим смартфоном

Стив Мошер (Steve Mosher), вице-президент по маркетингу Openmoko, поделился в почтовой рассылке ближайшими планами проекта. Согласно новым данным, работы по выпуску GTA03 заморожены.

Как пояснил Мошер, у Openmoko были два глобальных пути в 2009 году: выпуск нового Open Source-смартфона, известного под кодовым названием GTA03, или запуск так называемого project B. По ряду причин (преимущественно это нехватка ресурсов), компания выбрала второй путь, заморозив развитие GTA03.

Ожидается, что подробности о загадочном «проекте B» будут оглашены в течение ближайших месяцев. А пока Стив рекомендует энтузиастам, желающим поддержать Openmoko, продвигать свои наработки в основную кодовую базу, не останавливать работу над приложениями и не покидать проект.

В некоторых СМИ эти события оха-

рактеризовали как признание проектом Openmoko собственной несостоятельности и фактическую смерть столь интересной Open Source-инициативы. И, к сожалению, все последние новости от Openmoko больше подтверждают столь категоричные выводы, чем опровергают.

Linux Foundation будет распространять openSUSE Build Service

Компания Novell совместно с некоммерческой организацией Linux Foundation объявили о том, что решение openSUSE Build Service будет распространяться через сеть Linux Developer Network (LDN).

Решение openSUSE Build Service представляет собой открытую платформу, обеспечивающую инфраструктуру для разработки дистрибутивов openSUSE (автоматизирует процесс сборки пакетов с зависимостями для различного оборудования) и поддерживающую некоторые другие Linux-системы. Среди них – Debian и Ubuntu, Mandriva, а также Fedora, CentOS и Red Hat Enterprise Linux. Поддержка двух последних появилась в январе 2008 года, а исходный код openSUSE Build Service был открыт Novell под лицензией GNU GPL в январе 2007 года. Недавно также вышла новая версия openSUSE Build Service, 1.6, где появилась поддержка компиляции приложений для процессоров с архитектурой ARM.

Заявляется, что появление openSUSE Build Service в сети распространения LDN дополнит популярное приложение AppChecker, позволяющее разработчикам создавать портируемые приложения для Linux.

Portable Ubuntu упрощает запуск Linux в Windows

Авторы нового проекта Portable Ubuntu задались целью сделать максимально простым запуск операционной системы Ubuntu Linux в среде Windows. Им это удалось.

В Portable Ubuntu объединены такие Open Source-разработки, как ядро проекта coLinux (Cooperative Linux) для запуска Linux-ядра в Windows, графический X-сервер Xming (версия Xorg, собранная для Windows с помощью MinGW и Pthreads-Win32) и аудиосервер PulseAudio для Windows. Благодаря всему этому теперь любой желающий может запустить Ubuntu Linux прямо в Windows, не прилагая для этого никаких дополнительных действий вроде установки и настройки.

После запуска Portable Ubuntu в Windows стартует терминал и появляется привычное док-меню GNOME, через которое можно вызывать все стандартные Linux-приложения и работать с ними из Windows.

Как заявляют авторы проекта, «Portable Ubuntu для Windows – это полезный инструмент, когда вам нужно перейти к работе на машине, на которой установлена операционная система Windows».

Toshiba начинает продавать два ноутбука с OpenSolaris

Компания Toshiba выполнила свое обещание, которое было дано в конце прошлого года: в продаже появились два ноутбука с предварительно установленной операционной системой OpenSolaris.

Желающим купить лаптоп с OpenSolaris предлагаются две модели: субноутбук с 12-дюймовым дисплеем Portege R600 стоимостью в 1599 USD и ноутбук Tecra M10, стоимость которого в зависимости от комплектации колеблется от 1099 до 1399 USD. Характеристики Tecra M10 таковы: процессор Intel CORE 2 DUO P8400 2,26 ГГц/P8600 2,4 ГГц/T9400 2,53 ГГц, от 2 до 4 Гб оперативной памяти DDR2 (800 МГц), жесткий диск на 160/200/320 Гб, видеокарта NVIDIA Quadro NVS 150m (256 Мб), Wi-Fi, веб-камера.

На оба ноутбука предварительно установлен последний стабильный релиз ОС OpenSolaris – 2008.11. В его составе – OpenOffice.org 3.0, Adobe Flash Player, VirtualBox 2.0.6, Glassfish V2, Java SE Development Kit 6 Update 10, NetBeans 6.5, Sun Studio Express 11/08. Подробности доступны на http://www.shopopensolaris.com.

Linux Foundation назвала победителя конкурса рекламы

Некоммерческая организация Linux Foundation объявила победителя своего конкурса «Мы Linux» на лучший рекламный ролик, посвященный операционной системе GNU/Linux.

Обладателем гран-при рекламного конкурса стал 25-летний дизайнер из Израиля со своим роликом «Что значит быть свободным?». Победитель поедет в Токио (Япония) на организуемое Linux Foundation мероприятие Japanese Linux Symposium, которое пройдет в октябре 2009 года.

Конкурс на лучший видеоролик «Мы Linux» стартовал в декабре прошлого года. За это время на участие было заявлено 90 роликов со всего мира, которые в совокупности собрали более 100 тысяч просмотров. Победителя выбирал комитет, в который вошли такие заметные представители ИТ-сообщества, как Мэтт Эсей (Matt Asay) из CNET и Alfresco, Тим Орейли (Tim O'Reilly) из O'Reilly Media и Джо Брокмайер (Joe Brockmeier) из сообщества openSUSE.

Лучшие видеоролики с конкурса:

- ☑ победитель: http://video.linuxfoundation.org/video/1106.
- ☑ ближайшие преследователи: http://video.linuxfoundation.org/video/1262 и http://video.linuxfoundation.org/video/1057.

Дмитрий Шурупов, по материалам www.nixp.ru (osa@samag.ru)

Обзор Linux-игр Savage 2 и World of Goo

В прошлом выпуске «Open Source» (№041 от 27.03.2009) была подробно рассмотрена многоплатформенная Open Source-игра «Yo Frankie!». Надеюсь, многие успели скачать, установить и опробовать ее. На сей раз, как и было обещано, рассмотрены две закрытые разработки разных жанров, одна из которых – коммерческая. Итак, первая игра – «Savage 2: A Tortured Soul».

Savage 2: A Tortured Soul

Это бесплатная интернет-игра, жанр которой определить достаточно сложно. В ней сочетаются такие популярные направления, как FPS (first-person shooter), RTS (real-time strategy) и RPG (action role-playing game). О том, как все это умещается в одном флаконе, я расскажу дальше.

Игра доступна для различных распространенных платформ: Microsoft Windows, Mac OS X, GNU/Linux. Скачать ее можно несколькими способами, все подробности об этом доступны на http://www.savage2.com/en/download.php. Размер скачиваемой 32-битной версии для GNU/Linux составляет 784 Мб. Чтобы не скучать во время скачивания, можно ознакомиться с достаточно подробным руководством по интерфейсу игры на странице http://www.

savage2.com/tutorial. После того как игра будет скачана и установлена (в моем случае это была версия 1.5.0), потребуется зарегистрироваться, а затем скачать обновление, что займет еще около 80 Мб. Теперь можно приступить к игре!

Геймплей

Суть игры традиционна: противостояние двух сторон, коими здесь являются «Людской Легион» и «Звериная Орда». Обе стороны имеют уникальных юнитов со своим набором атак ближнего и дальнего боя, набором магических атак и способностей. Ярким примером может служить юнит Гаргулья, который, согласно легенде, умеет превращаться в камень и восстанавливать свою жизненную энергию. Где же пересекаются все перечисленные ранее жанры?

Играть в Savage 2 можно в двух режимах: режим командира и обычный. В первом игра приобретает вид обычной стратегии реального времени. Здесь доступны все присущие данному жанру возможности: добыча ресурсов, строительство зданий, точек респауна (мест, откуда павший юнит может начать игру снова), управление юнитами. Однако на последнем стоит остановиться поподробнее. Дело в том, что в режиме командира может играть только один – тот, кого выбрала команда перед началом игры (по ходу действия он может быть изменен). Остальные же играют в обычном режиме за простых юнитов. Естественно, в таком случае командир не может управлять вашим юнитом, но он может руководить атаками и общей стратегией стороны. Никто не запретит двигаться в любых направлениях, но тогда не стоит рассчитывать на победу и винить кого-либо в проигрыше.

Основной и единственный ресурс в игре – это золото, которое добывается на шахтах. В обычном режиме игрок зарабатывает золото, убивая соперников. К слову, «обычный» режим здесь не совсем обычен, поскольку соединяет два жанра: ролевую игру и шутер от первого лица. Если с шутером все вполне ясно, то о ролевой составляющей стоит сказать отдельно. Юнит обладает четырьмя характеристиками, среди которых сила, выносливость, интеллект, ловкость. Во время игры зарабатывается опыт, при достижении определенных значений которого текущий уровень повышается, а значит, можно улучшить характеристики. Золото же можно отдать команде либо потратить на зелья или магические предметы.

В игре существует несколько типов юнитов. Грубо их можно сгруппировать следующим образом: строительные, обычные боевые, платные боевые, против строений, Hellbourne. Первый тип может принимать участие в строительстве различных сооружений, а второй тип доступен всегда – вне зависимости от наличия средств. Доступ к третьему типу юнитов можно получить только при некоторых условиях, а юниты, которые идеально подходят для разрушения строений, могут оказаться бесполезными против боевых. Наконец, последний тип становится доступным только при наличии строения Hell Shrine и по достижению игроком достаточного количества душ. Как распорядиться одним из трех видов Hellbourne, решать игроку: этот юнит, как и любой другой, имеет свои слабые и сильные стороны.

Премиум-аккаунт

За счет чего живет игра? В отличие от популярных и раскрученных брендов, имя которых уже само по себе приносит деньги, серверы Savage 2 поддерживаются за счет платных аккаунтов. Никаких критичных и интересных функций платный аккаунт не представляет, но все же перечислю их:

- ☑ два дополнительные слота для вещей, с которыми общее число достигает шести;
- ☑ подробная статистика пользователя в сети и в игре;
- ☑ полный доступ ко всем сыгранным матчам;
- ☑ возможность создавать кланы;
- ☑ можно играть неограниченное количество раз за Hellbourne-юнитов.

На мой взгляд, самым важным достоинством является возможность неограниченно играть за Hellbourne-юнитов: пользователи с базовым аккаунтом могут играть только по разу за каждого из трех единиц, и это ограничение касается не одного матча, а всей игры вообще.

Кроме того, это способ помочь разработчикам, если вам понравилась игра. Цена достаточно демократична даже для наших реалий – 10 американских долларов, что в перерасчете на российские рубли (на момент написания этого материала) составляло примерно 336.

Сам я задумываюсь о покупке премиум-аккаунта, на что есть несколько причин:

- ☑ меня раздражают рекламные блоки;
- ☑ мне не нравится, когда приходится отвлекаться на сообщения вроде «информация доступна только для премиум-аккаунта»;
- ☑ люблю получать от всего максимум;
- ☑ хочу пересматривать матчи;
- ☑ и главное – ведь игра доступна в качестве родного приложения для GNU/Linux!

Вообще, «родная» поддержка – это огромный плюс. Интересно, что в то время как куда более крупные игровые проекты,

Типичное сражение в Savage 2. Скриншот с официального сайта

Вид от первого лица в Savage 2. Скриншот с официального сайта

Живой мир World of Goo. Скриншот с vgnetwork.it

имея ресурсы и возможности портировать игры под другие платформы, этого не делают, другие могут себе такое позволить. Именно «родная» поддержка GNU/Linux стала решающим фактором приобретения следующей игры.

World of Goo

«Копорация Гуу» – так звучит название в русской локализации от компании «Акелла», Windows-издание которой вышло не так давно – в феврале 2009 года. Сама же игра появилась осенью прошлого года и сразу произвела необычайный фурор. Это наглядно показывают награды, которые заслужил проект: «Инновационный дизайн» и «Лучшее техническое исполнение» на Фестивале независимых игр, а также десятое место в списке 50 лучших игр 2008 года по версии сайта Eurogamer (http://www.eurogamer.net/article.php?article_id=249671). На многих сайтах, посвященных играм, рейтинг этой игры не опускался ниже 9 из 10.

Итак, «World of Goo» разработана двумя бывшими сотрудниками Electronic Arts и доступна на таких платформах, как Wii, Windows, Mac OS X и GNU/Linux. Бесплатное демо можно скачать на сайте разработчиков: http://2dboy.com/games.php.

Геймплей

После запуска игры вы сразу попадаете в красочный и живой мир Гуу, который состоит из пяти частей. Каждая из них имеет свою неповторимую атмосферу, прочувствовать которую помогают музыка, графика и анимация (кстати, интересно, что музыку написал один из двух разработчиков, и несмотря на это, она представляется одним из самых атмосферных элементов в игре). Еще одним интересным фактом служит то, что разработчики активно использовали Open Source в своей игре: задействованы SDL (Simple DirectMedia Layer), ODE (Open Dynamic Engine), TinyXML и некоторые другие свободные разработки.

Суть игры достаточно проста: построить конструкцию таким образом, чтобы добраться до трубы, в которую должно засосать определенное количество Гуу – маленьких шариков. Простая задача имеет непростое решение – ведь конструкции имеют достоверную физику поведения: они раскачиваются, переламываются, разваливаются, подвергаются воздействию ветра, горят. Сами же конструкции строятся из разнообразных Гуу: одни могут быть использованы раз и навсегда, другие можно отсоединять от конструкции и повторно использовать в других местах. Чтобы добраться до заветной трубы, придется преодолевать пропасти, продвигаться между шипов, огибать различные препятствия, перелетать на шарах и все время строить, строить, строить.

Чтобы строить было интереснее, можно выполнять дополнительные критерии, такие как собрать большее число Гуу, закончить уровень за определенное время, выполнить задание за определенное количество движений. Если критерий для уровня будет выполнен, то на карте такой уровень будет отмечен флажком. Все Гуу, собранные сверх отведенной меры, отправляются в World of Goo Corporation – бонусный сетевой уровень. В этом уровне цель – построить самую высокую Гуу-башню, используя имеющиеся шарики Гуу. Сравниваться башня будет с башнями других участников. Башни игроков, которые строят свои строения одновременно с вами, обозначаются облачками.

Купить игру можно в системе PayPal. Для это достаточно перейти по ссылке buy now на странице сайта http://2dboy.com/games.php. Цена игры составляет 20 долларов США, что на момент покупки обошлось мне в 703 рубля.

Заключение

В этот раз были рассмотрены две игры, обе доступны под GNU/Linux без использования WINE и других средств эмуляции или виртуализации. Обе игры закрыты и представляют коммерческий интерес, однако они слишком разные, что, надеюсь, позволит найти каждой из них потенциального игрока. Любые комментарии можно отправлять ко мне на адрес электронной почты.

Никита Лялин
(tinman321@gmail.com)

Веб-сервер nginx. Часть 2: Другие возможности

В первой части статьи (см. «Open Source» 041) я рассказал о базовых и наиболее часто применяемых возможностях nginx. Но это – лишь малая часть того, что можно сделать с веб-сервером. Вторая часть посвящена некоторым более «продвинутым» возможностям, которые используются в крупных и высоконагруженных проектах.

Failover и балансировка

Крупные проекты редко состоят из одного сервера приложений: обычно их два или больше. Возникает задача балансировки клиентов по этим серверам, а также выполнения failover – необходимо, чтобы выход из строя одного из серверов не был заметен для клиентов.

Простейший способ решить эту задачу – техника Round-robin DNS, то есть назначение доменному имени нескольких IP-адресов. Но это решение имеет ряд недостатков, из-за которых лучше применять балансировку запросов по бэкендам на фронтенде nginx.

В конфигурационном файле это выглядит примерно так:

```
# Объявляем upstream – список бэкендов
upstream backend {
  # Перечисляем DNS-или IP-адреса серверов и их «вес»
  server   web1          weight=5;
  server   1.2.3.4:8080  weight=5;
  # Так можно подключаться к бэкенду через UNIX-сокет
  server   unix:/tmp/backend3   weight=1;
}
# Конфигурация виртуального сервера
server {
  listen <...>;
  server_name myserver.com;
  # Отправляем все запросы из / в upstream
  location / {
    proxy_pass  http://backend;
  }
}
```

Запросы, приходящие к nginx, распределяются по бэкендам соответственно указанному весу. Кроме того, можно сделать так, чтобы запросы с одних и тех же IP-адресов отправлялись на одни и те же серверы (для этого в upstream нужно указать директиву ip_hash). Так можно решить проблему с сессиями, но все же лучше найти какой-нибудь способ их репликации или (что еще лучше) использовать RESTful-подход (см. http://ru.wikipedia.org/wiki/REST).

В случае, если один из серверов откажется принимать соединения или соединение к нему оборвется по таймауту, он на некоторое время будет исключен из upstream.

Оптимизация nginx

Рассмотрим шаги оптимизации.

Увеличение количества и объема буферов

Для хранения принятых запросов и еще не отданных ответов nginx использует буферы в памяти, а если запрос или ответ не помещается в них – nginx записывает его во временный файл (и пишет при этом предупреждение в log-файл). Поэтому необходимо установить такие размеры, чтобы в большинстве случаев не требовалось обращаться к временному файлу, а с другой стороны – чтобы буферы не использовали слишком много памяти.

Для этого используются следующие параметры:

- ☑ **client_body_buffer_size** (по умолчанию: 8k/16k – в зависимости от архитектуры) – задает размер буфера для чтения тела запроса клиента. Обычно стандартного значения хватает: его требуется повышать, только если ваше приложение устанавливает огромные cookies.
- ☑ **proxy_buffer_size** (по умолчанию: 4k/8k) – задает размер буфера, в который будет читаться первая часть ответа, получаемого от проксируемого сервера. В этой части ответа находится, как правило, небольшой заголовок ответа. Стандартного значения обычно хватает.
- ☑ **proxy_buffers** (по умолчанию: 8 4k/8k) – задает число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от проксируемого сервера. Установите этот параметр так, чтобы большинство ответов от бэкенда помещалось в буферы.

Механизмы обработки соединений

Есть одна тонкость, касающаяся механизма обработки соединений, а именно – способ получения информации о событиях на сокетах. Существуют следующие методы:

- ☑ **select** – стандартный метод. На большой нагрузке сильно нагружает процессор.
- ☑ **poll** – стандартный метод. Также сильно нагружает процессор.
- ☑ **kqueue** – эффективный метод, используемый в операционных системах FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 и Mac OS X. На 2-процессорных машинах под управлением Mac OS X использование kqueue может привести к kernel panic.
- ☑ **epoll** – эффективный метод, используемый в Linux 2.6+. В некоторых старых дистрибутивах есть патчи для поддержки epoll ядром 2.4.
- ☑ **rtsig** – real time signals, эффективный метод, используемый в Linux 2.2.19+. При больших количествах одновременных соединений (более 1024) с ним могут быть проблемы (их можно обойти, но на мой взгляд, лучше с этим не связываться).
- ☑ **/dev/poll** – эффективный метод, используемый в Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ и Tru64 UNIX 5.1A+.

При компиляции nginx автоматически выбирается максимально эффективный найденный метод, однако скрипту configure можно насильно указать, какой метод использовать. Если вы решили сделать это, рекомендую использовать следующие методы:

- ☑ Linux 2.6: epoll;
- ☑ FreeBSD: kqueue;
- ☑ Solaris, HP/UX и другие: /dev/poll;
- ☑ Linux 2.4 и 2.2: rtsig, не рекомендуется при больших нагрузках.

Gzip

Включение gzip позволяет сжимать ответ, отправляемый клиенту, что положительно сказывается на удовлетворенности пользователя, но требует больше времени CPU. Gzip включается директивой gzip (on|off). Кроме того, стоит обратить на следующие важные директивы модуля gzip:

- ☑ **gzip_comp_level 1..9** – устанавливает уровень сжатия. Опытным путем выявлено, что оптимальные значения лежат в промежутке от 3 до 5: большие значения дают маленький выигрыш, но создают существенно большую нагрузку на процессор, меньшие – дают слишком маленький коэффициент сжатия.
- ☑ **gzip_min_length** (по умолчанию, 0) – минимальный размер ответа, который будет сжиматься. Имеет смысл поставить этот параметр в 1024, чтобы слишком маленькие файлы не сжимались (т.к. эффективность этого будет мала).
- ☑ **gzip_types mime-тип [mime-тип ...]** – разрешает сжатие ответа методом gzip для указанных MIME-типов в дополнение к text/html, который сжимается всегда. Имеет смысл добавить такие MIME-типы как text/css, text/javascript и подобные. Разумеется, сжимать файлы в форматах вроде gif и jpg не имеет смысла.

Кроме того, существует модуль gzip_static, который позволяет раздавать уже сжатые статические файлы. В конфигурационном файле это выглядит так:

```
location /files/ {
  gzip on;
  gzip_min_length 1024;
  gzip_types text/css text/javascript;
  gzip_comp_level 5;
  gzip_static on;
}
```

При использовании такой конфигурации в случае запроса «/files/test.html» nginx будет проверять наличие «/files/test.html.gz», и если этот файл существует и дата его последнего изменения больше, чем дата последнего изменения файла test.html, будет отдан уже сжатый файл, что сохранит ресурсы процессора, которые потребовались бы для сжатия оригинального файла.

Оптимизация приложений

Существует очень полезный трюк, который позволяет указать разработчикам приложений, какие страницы нужно оптимизировать в первую очередь. Для этого потребуется в конфиге nginx указать новый формат лога:

```
log_format  my_combined  '$remote_addr - $remote_user ↵
    [$time_local] '
    '"$request" $status $body_bytes_sent '
    '"$http_referer" "$http_user_agent" '
    '$upstream_response_time "$host"'

access_log /var/log/nginx/access_log my_combined;
```

Переменная $upstream_response_time содержит время ответа бэкенда, поэтому в лог попадает время обработки каждого запроса бэкендом. Далее понадобятся два скрипта:

Первый – /usr/local/bin/url_stats_report.sh:

```
#!/bin/sh

echo "=== Requests which took most of the time ===" > ↵
    /tmp/report.txt
echo "overall time - number of requests - average time - ↵
    url" >> /tmp/report.txt

cat /var/log/nginx/*access.log | ↵
    /usr/local/bin/url_stats.py >> /tmp/report.txt
cat /tmp/report.txt | mail -s "url performance report" root
```

Второй – /usr/local/bin/url_stats.py:

```
#!/usr/bin/env python

import sys

urls = {}

try:
    while 1:
        line = raw_input()
        line_arr = line.split(" ")
        try:
            host = line_arr[-1]
            host = host[1:]
            host = host[:-1]
            url = line_arr[6]
            t = float(line_arr[-2])
            #print host, url, t

            try:
                urls[host + url] = (urls[host + url][0] + t, ↵
                        urls[host + url][1] + 1)
            except KeyError, e:
                urls[host + url] = (t, 1)
        except ValueError, e:
            pass


except EOFError, e:
   pass

def sort_by_value(d):
    """ Returns the keys of dictionary d sorted by their values """
    items=d.items()
    backitems=[ [v[1],v[0]] for v in items]
    backitems.sort(reverse=True)
    return [backitems[i][1] for i in range(0,len(backitems))]

if (len(sys.argv) > 1):
    f = open(sys.argv[1], 'r')
    for k in f.readlines():
        k = k.strip()
        try:
            print urls[k][0], urls[k][1], urls[k][0] / urls[k][1], k
        except:
            print 0, 0, k
else:
    i = 0
    for k in sort_by_value(urls):
        print urls[k][0], urls[k][1], urls[k][0] / urls[k][1],  k
        i += 1
        if i > 100: break
```

Они не идеальны, но задачу выполняют: запуская /usr/local/bin/url_stats_report.sh (например, в postrotate утилиты logrotate), вы получаете наглядную картину, какие запросы занимают большую часть времени бэкенда.

Кеширование

Незадолго до выхода этой статьи Игорь Сысоев выпустил версию nginx 0.7.44 с экспериментальной поддержкой кеширования. Из-за большого количества багов за короткое время было выпущено еще несколько версий, и в настоящее время последний релиз, 0.7.50, уже достаточно хорошо (хотя и неидеально) работает с кешированием. Несмотря на то, что это все еще экспериментальная функция, я решил рассказать о ней, поскольку кеширования очень давно ждали многие администраторы.

Сейчас nginx умеет кешировать на диске ответы от HTTP- и FastCGI-запросов на бэкенды, указывать ключ для кеширования, учитывать заголовки X-Accel-Expires, Expires и Cache-Control и вручную устанавливать максимальное время жизни объекта в кеше. Обслуживанием кеша (очистка старых файлов, наблюдение за размером и тому подобное) занимается специальный процесс cache manager.

Положительной особенностью реализации является то, что при старте nginx cache manager начинает проверку кеша в фоне, благодаря чему nginx не делает то, что называется «дает сквида», то есть он не висит несколько минут, проверяя кеш перед стартом.

Я намеренно не указываю пример конфигурации, так как, во-первых, директивы могут еще измениться, а во-вторых, нужно глубокое понимание механизма кеширования, что требует вдумчивого чтения документации (http://sysoev.ru/nginx/docs/http/ngx_http_proxy_module.html#proxy_cache) и архивов рассылки nginx-ru.

За кадром

В статье освещена лишь часть возможностей nginx, которыми я пользуюсь чаще всего. За пределами рассказа остались такие вопросы, как поддержка SSI, работа с memcached, экспериментальный встроенный Perl и сторонние модули, реализующие дополнительную функциональность.

Владимир Русинов
(vladimir@greenmice.info)

Быстрое создание GUI с wxWidgets и wxFormBuilder

Сегодня перед начинающим программистом открыты большие возможности для создания приложений с графическим интерфейсом пользователя (далее – GUI, Graphical User Interface). Современные платформы для этой цели могут использовать как «родные», так и «сторонние» библиотеки, различающиеся в реализации отдельных компонентов, но сохраняющие общие принципы GUI: окна, кнопки, события клавиатуры, мыши и так далее. Например, на GNU/Linux-платформах программисты зачастую используют GTK+ и Qt, в ОС семейства Windows – WinApi, его ООП-обертку MFC или более «легкую» библиотеку WTL, а программисты Macintosh – Carbon API.

Стоит отметить, что большинство открытых GUI-библиотек можно использовать на всех популярных x86-платформах и не только. К таким универсальным инструментам, обеспечивающим максимальную переносимость с минимальными правками кода (или вообще без них), относятся:

- **wxWidgets** (http://www.wxwidgets.org). Главная особенность – способность использовать графические компоненты других GUI-библиотек, таких как MFC, GTK+, Carbon, Motif. Из наиболее популярных приложений, интерфейс которых создан на базе wxWidgets, можно выделить Amaya, aMule, Audacity и FileZilla. Написан на C++.
- **Qt** (http://www.qtsoftware.com). Прямой конкурент в плане поддержки множества платформ. Кроме того, намного более популярный и богатый по возможностям инструмент. Но ведь это не повод не пробовать альтернативы поскромнее и компактнее, не правда ли? Среди программ на базе Qt – графическая среда KDE и ее стандартные приложения. Вновь C++.
- **GTK+** (http://www.gtk.org). Этот фреймворк поддерживает меньшее количество платформ, но его оценят те, кто предпочитает программировать на Си. Среди известных программ – графическая среда GNOME и ее стандартные приложения.

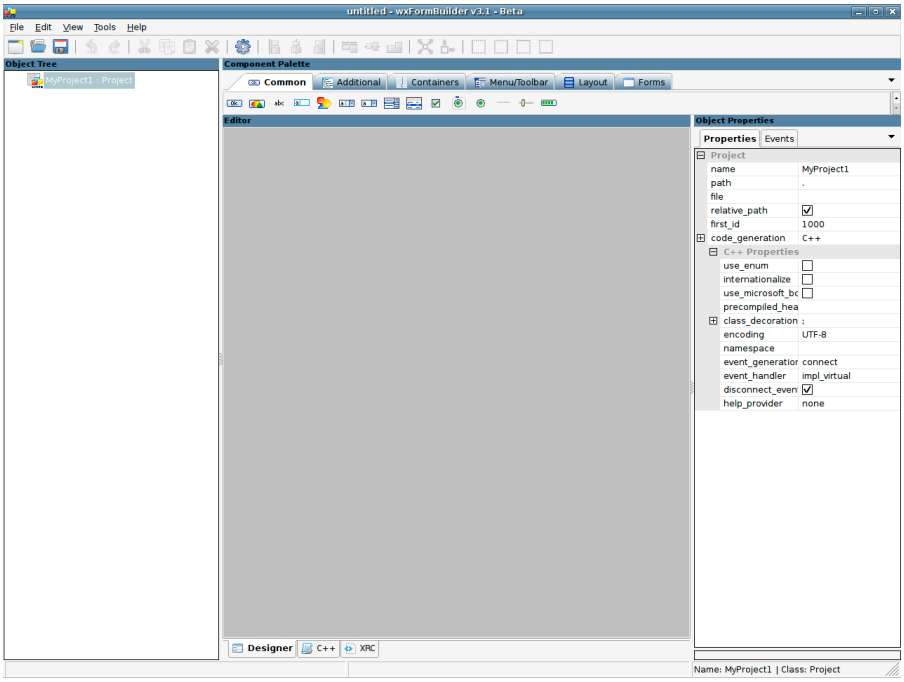


Рисунок 1. wxFormBuilder готов к работе

Почему же я выбрал wxWidgets?

- Открытый исходный код и легкая переносимость на другие ОС.
- Размер статически слинкованной библиотеки не превышает 1,2 Мб без архивации (и составляет всего 600-700 Кб в zip-архиве).
- Личные симпатии к структуре этого фреймворка.

Установка

В этой статье будет рассмотрена установка wxWidgets под ОС семейства GNU/Linux. Для получения информации по работе с другими платформами просмотрите wiki-страницу http://wiki.wxwidgets.org/Guides_%26_Tutorials.

В самом простом случае установить библиотеку можно через систему пакетов вашего Linux-дистрибутива, однако я настоятельно рекомендую получить последнюю версию исходных текстов с официального сайта и самостоятельно их собрать. Это даст возможность гибко сконфигурировать библиотеку. Процесс инсталляции довольно традиционный, но все же я его опишу по шагам:

Шаг 1. Выберите на странице загрузки исходных текстов (http://wxwidgets.org/downloads) подходящий для вашей платформы архив. Для Linux это обычно wxGTK или wxX11, но есть возможность загрузить все сразу (wxAll). Я рекомендую вариант wxGTK, поскольку он выглядит наиболее современно и занимает меньше места, чем wxAll.

Шаг 2. Установите компилятор и систему автосборки пакетов (если таковые отсутствуют). В Ubuntu это можно сделать следующей командой:

```
$ sudo apt-get install build-essential
```

Распакуйте содержимое архива в любую доступную вам директорию и перейдите в нее.

Шаг 3. Теперь необходимо сконфигурировать пакет для сборки. Для этого вызывается команда ./configure. Среди ее параметров:

- **--enable-unicode** – включает поддержку Unicode (потребуется для русского языка).
- **--disable-compat26** – отключает обратную совместимость с версией wxWidgets 2.6. Может быть полезно при статической линковке для уменьшения размера модулей.
- **--disable-shared** – делает модули статическими, а не динамическими.
- **--enable-monolithic** – включает компиляцию wxWidgets в одну библиотеку, которая может быть как статической, так и динамической.
- **--with-regex** – включает поддержку регулярных выражений через класс wxRegExp.

Я использую следующий набор параметров:

```
$ ./configure --enable-unicode --disable-compat –with-regex
```

Если в момент выполнения команды произойдет ошибка из-за отсутствия какого-либо пакета в системе, потребуется проинсталлировать его. Если источник пакетов – бинарный репозиторий, то обычно нужно установить соответствующий пакет с постфиксом -dev (т.е. заголовочные файлы пакета).

Шаг 4. После успешной конфигурации соберите библиотеку командой:

```
$ make
```

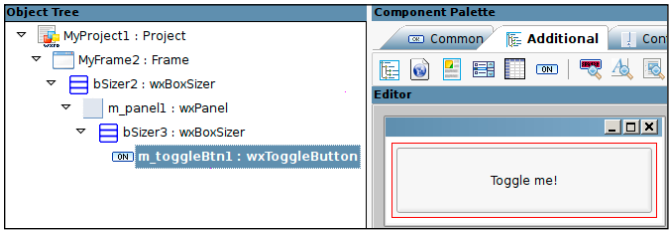


Рисунок 2. Степень вложения до кнопки

Рисунок 3. Панель инструментов wxFormBuilder

И установите ее в систему:

```
$ sudo make install
```

Шаг 5. Для проверки установки выполните в терминале команду:

```
$ wx-config --version
```

wxFormBuilder – средство быстрого проектирования UI

При разработке графических приложений описание интерфейса пользователя, особенно когда он многооконный или содержит вкладки, переходит в разряд монотонных занятий по описанию того или иного класса в заголовочном файле строчками вроде:

```
wxPanel *m_panel1;
wxStaticText *m_staticText1;
```

...и в файле кода C++ – чем-то вроде:

```
m_panel1 = new wxPanel( this, wxID_ANY, wxDefaultPosition, ↵
      wxDefaultSize, wxTAB_TRAVERSAL );
m_staticText1 = new wxStaticText( m_panel1, wxID_ANY, ↵
      _("Real path to a multimedia dir:"), wxPoint( -1,-1 ), ↵
      wxDefaultSize, 0 );
```

Логично автоматизировать данный процесс, сделав проектирование интерфейса более творческим и наглядным занятием. Для этого и созданы средства быстрого проектирования UI, одним из которых является wxFormBuilder (http://wxformbuilder.org). Остановимся на нем более подробно.

Рекомендую использовать последнюю версию этого инструмента, всегда доступную по адресу: http://wxformbuilder.org/?page_id=7. На момент написания статьи актуальной версией является 3.1. Для установки под Ubuntu воспользуйтесь deb-пакетами, которые доступны на странице загрузки.

Запустим среду (см. **рис. 1**). Слева, в Object Tree, будет отображено дерево задействованных классов и их экземпляров в соответствии с их вложенностью на общем виде посередине. Так, например, в обычной форме степень вложения объектов до кнопки, непосредственно расположенной на ней с точки зрения пользователя, имеет следующий вид (см. **рис. 2**).

Данную степень вложения можно сократить до 3 компонентов. Для этого необязательно использовать wxPanel и еще один wxBoxSizer. Вообще, wxPanel будет нужен только тогда, когда вы, например, захотите изменить цвет фона формы или будете использовать табы, но лично я привык его всегда включать в форме, когда использую wxFormBuilder.

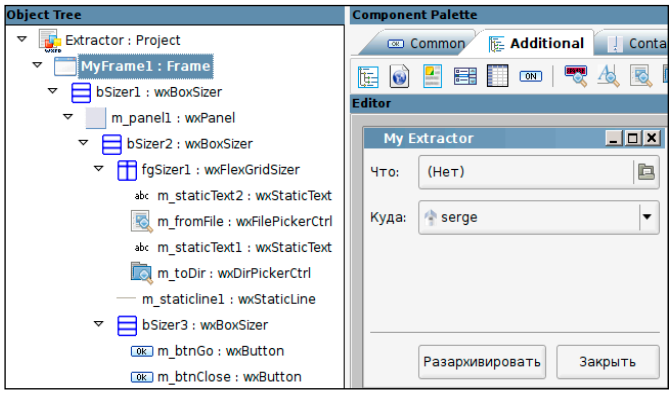


Рисунок 4. Форма будущей программы

Посередине сверху – Component Palette, на которой представлены все доступные виджеты библиотеки. Слева, в Object Properties, собраны все свойства и события того или иного класса. Чтобы назначить событие, нужно в соответствующем поле вкладки Events задать имя функции, которая будет обрабатывать то или иное действие. Она будет автоматически сформирована в Inherited Class, к которому вернемся позже. Стоит отметить несколько интересных свойств проекта (сделайте активным в Object Tree самый верхний элемент: «MyProject1: Project»):

Encoding и internationalize определяют кодировку и возможность перевода приложения на другие языки. В любом случае советую использовать включенную возможность перевода и UTF-8 в качестве кодировки.

Event_generator в положении table переключает режим генерации событий в таблицу подобно MFC, а в положении connect – в динамическую модель, подобную сигналам и слотам Qt.

Теперь – о главном меню wxFormBuilder. При помощи пункта «File → Generate Code» (горячая клавиша <F8>) можно сохранить сгенерированный код по пути, указанному в свойстве path проекта под именем файла file. Будут созданы два файла: <file>.cpp и <file>.h. Полученную пару не следует редактировать.

Для реализации функциональности следует воспользоваться пунктом «Tools → Generate Inherited Class» (<F6>), вызвав который нужно отметить формы, требующие генерации, и указать имена класса и файлов с кодом. В дальнейшем можно будет открыть полученный .cpp и внести необходимую функциональность в те или иные события, которые там будут расположены.

Все остальные пункты меню стандартны – за исключением «File → Import XRC», который импортирует XML-подобный файл проекта.

Рассмотрим элементы панели инструментов. Начнем с конца, так как они наиболее интересны:

Последние четыре кнопки включают/отключают границы того или иного виджета.

Следующая пара управляет максимальным растяжением виджета в ширину и использованием всего доступного пространства макета.

Последующие 6 кнопок управляют выравниванием виджета по горизонтали и вертикали.

Остальные, наверное, вам уже знакомы: генерация кода, работа с буфером обмена, функции отката и возврата, создание, открытие и сохранение файла проекта (см. **рис. 3**).

Создание простого проекта с помощью wxFormBuilder

В качестве простого примера работы с wxFormBuilder спроектируем несложный GUI для распаковывания файла типа *.tar.gz. Для этого создадим в редакторе форму, подобную той, что на **рис. 4**.

Выставим следующие параметры проекта (см. **рис. 5**).

Теперь надо добавить события OnButtonClick на две кнопки действий: «Разархивировать» и «Закрыть». Имена событий могут быть любыми: главное – чтобы они были разными. После того как все настроено и смакетировано, следует сохранить проект – желательно в том же каталоге, что указан в «path». Пора нажать <F8> и посмотреть на готовый код формы (он есть в листингах, приложенных к выпуску «Open Source»). Следующий шаг – <F6> – создаст Inherited Class. В открывшемся окне надо выбрать форму и ввести имена файлов и класса по усмотрению. Например, так (см. **рис. 6**).

Теперь можно закрыть окно wxFormBuilder (если вы не хотите редактировать форму) и перейти к написанию кода – для этого воспользуемся интегрированной средой разработки NetBeans. (Впрочем, ничто не мешает производить дальнейшие действия в другом IDE – в них нет ничего особенного. – **Прим. ред.**)

Взаимодействие с IDE NetBeans 6.5

Запустите NetBeans и откройте новый проект. Добавьте в него сгенерированные исходные файлы формы и создайте «Main C++ File» с таким содержимым:

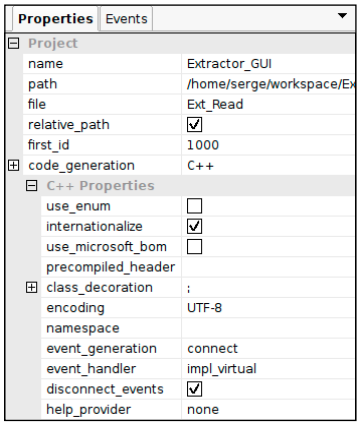


Рисунок 5. Параметры проекта

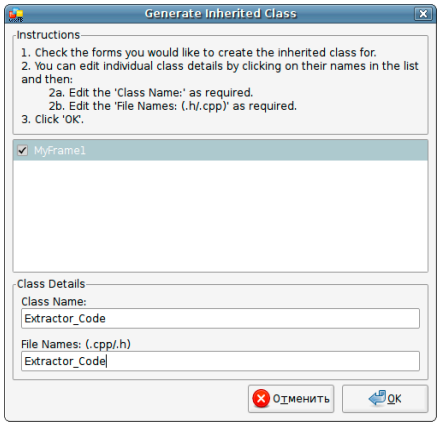


Рисунок 6. Параметры наследуемого класса

```
#include <wx/wx.h>
//Замените на имя *.h файла унаследованного класса
//(Inherited Class)
#include "Extractor_Code.h"
//Название этого класса может быть любым
class MyExtractor : public wxApp{
public:
      bool OnInit();};

IMPLEMENT_APP(MyExtractor)

//Название класса измените соответственно
bool MyExtractor::OnInit(){
      //Extractor_Code – имя вашего унаследованного класса
      //(Inherited Class)
      Extractor_Code* frame = new Extractor_Code(NULL);
      frame->Show();
return true;}
```

Теперь следует настроить компиляцию среды. Для этого откройте свойства проекта и во вкладке C Compiler в поле Additional Options добавьте строку:

```
`wx-config --cflags`
```

В такое же поле вкладки C++ Compiler:

```
`wx-config --cxxflags`
```

Перейдите во вкладку Linker, где следует открыть настройки поля Libraries и добавить в качестве опции строку:

```
`wx-config --libs`
```

Если среда говорит, что не может найти файлы include, то добавьте в настройках путей списка исходных файлов проекта:

```
/usr/local/include/wx-2.8
```

Осталось внести функциональность в события. Откройте файл *.cpp вашего наследованного класса и добавьте строки в соответствующие функции:

```
void Extractor_Code::UnArchive( wxCommandEvent& event )
{
    wxString tarComm;

    if(m_fromFile->GetPath() == wxT("")) return;
    if(m_toDir->GetPath() == wxT("")) return;

    tarComm = wxT("tar -zxvf ") + m_fromFile->GetPath() + ↵
      wxT(" -C ") + m_toDir->GetPath();
    wxExecute(tarComm);
}

void Extractor_Code::CloseApp( wxCommandEvent& event )
{
    Close();
}
```

Проверьте: у меня все работает!

Заключение

Подобным образом можно очень быстро написать GUI-обертку, а разработка полноценного GUI-приложения займет лишь время задания функциональности и проектирования интерфейса.

В заключение отмечу, что всю документацию по классам фреймворка всегда можно получить по адресу http://docs.wxwidgets.org.

Сергей Гулин
(sugia@yandex.ru)

Беседы о Qt. Часть 2: Строки

Поддержка Unicode и удобство работы со строками в этой кодировке – пожалуй, одно из основных требований к современным «основным» библиотекам. Например, в GTK+ 2 внутренняя кодировка – UTF-8, однако строки в GTK+ выглядят по-прежнему как указатели на char (а вернее, на переименованный тип – gchar), и для работы с ними используется два вида функций: «юникодные» и обычные. При этом к элементу строки, если она UTF-8, обратиться по индексу нельзя, поскольку неизвестно, сколько байт занимает каждый элемент. Поэтому в GTK для работы с отдельными символами таких строк надо заниматься их последовательным перебором по одному элементу (с помощью функций g_utf8_find_next_char() и g_utf8_find_prev_char()). Это я привел для сравнения.

В Qt строка представлена классом QString и внутренне хранится в особой структуре как поле-массив экземпляров класса QChar, каждый из которых инкапсулирует 16-битный код таблицы Unicode 4.0 (коды выше 65535 разбиваются на два элемента). Для пользователя-программиста этот внутренний массив скрыт и доступен через интерфейс. Оба класса: QString и QChar – отлично документированы, однако некоторым «тонким» моментам стоит уделить особое внимание. В частности вопросам скорости обработки строк, которые встают особо остро, когда со строками приходится работать в цикле.

Итак, вопрос первый – получение длины строки. Как вы знаете, обычная «сишная» функция вычисления длины строки поступает так: перебирает все символы строки, пока не находит ноль, и возвра-

щает увеличиваемый в каждой итерации цикла счетчик. В классе QString тоже есть функция length(), возвращающая длину строки. Но вычисляет ли она эту длину при своем вызове?.. Нет, функция возвращает заранее вычисленное значение, которое хранится в той же скрытой от внешнего мира структуре, что и массив символов. Если быть точным, то объявлена эта функция так:

```
inline int QString::length() const
{ return d->size; }
```

В результате, компилятор в месте её вызова просто будет вставлять, грубо говоря, «d->size» (d – указатель на «ту самую» структуру).

Теперь о доступе к отдельным символам. Например:

```
QString s ("hello");
if (s[0] == 'h')

;//делаем что-нибудь
if (s.at(0) == 'h')

;//делаем что-нибудь
```

Какой способ лучше: с использованием оператора [], как к массиву, или же с функцией at()? Последняя проверяет правильность индекса (чтобы он был не меньше нуля и не превышал длины строки) и возвращает, собственно, QChar-элемент из внутреннего массива со строкой. Причем возвращает его как const. А вот для [] есть несколько вариантов. Один из них действует так же, как at(). А вот другой возвращает экземпляр QCharRef, причем уже не const. Что это дает? Вы получаете возможность изменять полученный символ. Вот корявый с точки зрения английского языка пример:

```
QString s ("hello");
s[0] = 'y';
qDebug() << s; //будет выведено: yello
```

Итак, компилятор вызывает версию оператора [] в зависимости от того, что вы будете делать с возвращаемым значением. Если просто сравнивать используется обычный QChar, не QCharRef, так что будет вызвана та перегруженная версия [], которая выглядит точно так же, как функция at(). Но если вы будете изменять значение, то вызывается оператор [], возвращающий QCharRef.

Вот два примера, наглядно показывающие, как компилятор «переключает» вызовы оператора [].

В первом примере s[] вернет нам тип QChar, и изменение значения объекта ch не повлияет на строку s:

```
QString s ("hello");
QChar ch = s[0];
ch = 'y';
qDebug() << s;
```

В следующем примере s[0] возвращает QCharRef. Изменяя ch, мы уже оказываем влияние на s:

```
QString s ("hello");
QCharRef ch = s[0];
ch = 'y';
qDebug() << s;
```

И еще пара слов об изменении содержимого строки. Qt применяет технику «неявного разделения» (implicit sharing) для класса QString. Это означает, что при копировании строки A в строку B последняя будет содержать в себе указатель на текстовые данные из A и так до тех пор, пока одна из функций-членов B не соберется изменить эти текстовые данные. Только в такой момент данные будут скопированы отдельно, «лично» для B. Подобный подход называется еще copy-on-write – копирование при записи.

Перейдем к сложению содержимого строк. Есть несколько способов, но какой из них лучше? QString предоставляет нам операторы «+», «+=», а также функции append() и prepend(). Допустим, мы хотим прибавить слово world к строке A. Объявим строку A:

```
QString A ("hello ");
```

Варианты:

```
A = A + "world"; //плохо!
A += "world"; //хорошо
A.append ("world"); //тоже хорошо
```

Почему первый вариант плох? Сперва отмечу, что «плох» – понятие относительное, поскольку на современных компьютерах разница между этим и другим способами не будет заметна (если они реализованы вне ресурсоемких циклов). Однако если стремиться к совершенству, лучше использовать оператор «+=» или функцию append().

Разница вот в чем:

- ☑ оператор «+» создает новый экземпляр QString и возвращает его;
- ☑ оператор «+=» добавляет правый операнд к строке, не создавая промежуточного экземпляра;
- ☑ append() работает аналогично оператору «+=».

И еще одна тонкость с append(). Эта функция возвращает указатель на this – то есть, собственно, сам объект, к которому была добавлена новая строка. Это позволяет составлять цепочки вызовов append():

```
QString a;
a.append ("hello ").append ↵
      ("world").append ("!");
```

Для поиска подстроки в строке класс QString предоставляет функцию indexOf(). Остерегайтесь использовать ее в длинных циклах – это сильно замедлит скорость работы. Вот медленный код:

```
for (int i = 0; i < string.size(); ++i)
{
    //ищем с текущего индекса:
    int pos = string.indexOf ("href", i);
    if (pos != -1)
        ; //нашли!
}
```

Намного быстрее это будет выполняться, если «вручную» развернуть массив подстроки:

```
for (int i = 0; i < string.size(); ++i)
{
    if (string.at (j) == 'h' &&
         string.at (j + 1) == 'r' &&
         string.at (j + 2) == 'e' &&
         string.at (j + 3) == 'f')
          ; //делаем нечто
}
```

Выглядит весьма уродливо, но в некоторых случаях единственный способ не заставить пользователя скучать.

Как известно, почти все функции Qt там, где нужны строковые параметры, принимают их как QString. Между тем при обращении к функциям сторонних библиотек мы сталкиваемся с необходимостью передачи им обычных «сишных» строк, будь то 8-битные ASCII или многобайтные UTF-8. Как быть? Несколько примеров.

Пример 1:

```
QString a ("hello");
char *ascii = a.toAscii().data();
```

Пример 2:

```
char *utf8 = a.toUtf8().data();
```

Дело в том, что у QString есть ряд вспомогательных функций вроде toAscii(), toLocal8Bit(), toUtf8() и других, которые возвращают экземпляр класса QByteArray. QByteArray – удобная оболочка для работы с байтовым массивом. И чтобы получить непосредственно его значение, а вернее – указатель на байтовый массив, можно обратиться к функции QByteArray::data(). Для ускорения работы, если массив нужен вам только на чтение, используйте функцию constData():

```
QString a ("hello");
const char *ascii = ↵
      a.toAscii().constData();
```

Массив в QByteArray завершается нулевым символом, то есть ведет себя

как обычная «сишная» строка, но отмечу, что QByteArray может хранить символы «\0» и в других местах массива (не только в конце). Просто функции, которые работают с «сишными» строками, обычно полагают, что строка заканчивается нулевым символом. Обратите внимание, что функция QByteArray::size() возвращает размер без учета последнего нулевого символа. Работать с QByteArray можно по большому счету так же, как с QString: многие функции повторяются, то есть в обоих классах есть поиск, замена, разного рода преобразования и тому подобное. Итак, QByteArray – удобная надстройка над традиционной строкой char*.

Еще одна важная тема – интернационализация строк. Грубо говоря, их перевод. Qt для этого предоставляет свой механизм. Рассмотрим его. Первым делом надо заключить все строки, которые вы хотите перевести, в функцию tr(), объявление которой выглядит так:

```
QString QObject::tr (const char ↵
      *sourceText, const char ↵
      *comment = 0, int n = -1 ) ↵
      [static]
```

Вот пример ее использования:

```
QString s (tr ("hello"));
```

В состав Qt входит утилита lupdate, которая смотрит в исходники вашего проекта, ищет в них функцию tr(), вычленяет из нее текст и на основе этого обновляет либо создает файлы перевода. Эти файлы надо указать в файле проекта и желательно включить их бинарные версии в ресурсы программы. Как это сделать? Предположим, ваш файл проекта (расширение .pro) и исходники находятся в одном и том же каталоге. Создадим в этом каталоге подкаталог translations. Сюда будут помещаться файлы с переводами. Например, мы хотим добавить в программу русский перевод. В файл проекта добавляем строку:

```
TRANSLATIONS += translations/ru.ts
```

Итак, ru.ts – это будущий файл с исходником русского переводом. Будет и бинарный файл – сразу пропишем его в файле ресурсов. Напомню, что файл ресурсов имеет расширение qrc и XML-формат. В простейшем (т.е. нашем) случае этот файл будет таким (назовем его app.qrc):

```
<!DOCTYPE RCC><RCC version="1.0">
<qresource>

 <file>translations/ru.qm</file>
</qresource>
</RCC>
```

Заметьте, что ru.qm, а не ru.ts – о его происхождении я расскажу ниже. А пока – пропишем файл ресурса в файл проекта (.pro):

```
RESOURCES += app.qrc
```

Итак, в файле проекта появились две новые записи:

```
TRANSLATIONS += translations/ru.ts
RESOURCES += app.qrc
```

Теперь «пометим» в исходниках нужные строки функцией tr() и выполним из консоли команду:

```
lupdate файл-проекта.pro
```

Утилита lupdate прочитает из проекта список файлов-переводов и обновит их либо создаст новые. В нашем случае она создаст файл ru.ts. Внутри это XML-файл с записями такого вида:

```
<message>
  <location filename="mainwindow.cpp" ↵
      line="78"/>
  <source>hello</source>
    <translation type="unfinished"> ↵
      </translation>
</message>
```

Перевод должен быть вписан между парными тегами translation. Такие файлы удобнее всего править средствами утилиты от Qt под названием linguist. После правки и сохранения в linquist надо выбрать пункт меню «File → Release». Для этих же целей можно воспользоваться консольной утилитой lrelease:

```
lrelease файл-проекта.pro
```

Утилита создаст файлы перевода в другом, уже бинарном, формате с расширением qm. Так из ru.ts и получается ru.qm. Именно qm-файлы и надо «подключать» к программе при запуске, чтобы строки были переведены. Подключение выглядит следующим образом. В классе главного окна объявим переменные:

```
QTranslator myappTranslator;
QTranslator qtTranslator;
```

MyappTranslator будет «переводчиком», читающим созданные вами qm-файлы, а qtTranslator будет отвечать за чтение файлов перевода, встроенных в саму Qt. Теперь где-нибудь в конструкторе главного окна пишем следующее:

```
qtTranslator.load (QString ("qt_%1").arg ↵
      (QLocale::system().name()), ↵
      QLibraryInfo::location ↵
      (QLibraryInfo::TranslationsPath));
qApp->installTranslator (&qtTranslator);
myappTranslator.load (":/translations/" + ↵
      QLocale::system().name());
qApp->installTranslator ↵
      (&myappTranslator);
```

Для qtTranslator мы загружаем встроенные файлы перевода для текущей локали. Для myappTranslator – читаем файл перевода из ресурса, из каталога «translations». В качестве имени файла берется QLocale::system().name(). Расширение .qm можно не указывать – Qt сама его подставит. Обратите внимание на возможность Qt использовать несколько объектов-переводчиков, чем мы и пользуемся.

Переводы загружены: теперь после выполнения этого кода все строки, отмеченные tr(), будут представлены уже в переведенном виде – конечно же, если вы их перевели в ts-файле.

Примечания:

- ☑ Изменения не отслеживаются автоматически. Если вы хотите обновить перевод, надо выполнить всю цепочку действий: запуск lupdate, правка ts-файлов, запуск lrelease.
- ☑ Кодировка в ts-файлах – UTF-8.
- ☑ Файлы переводов могут быть не только в ресурсах, но и внешними. Просто из ресурса удобнее к ним обращаться: всегда знаешь, где лежит нужный файл.

Продолжение статей «Беседы о Qt» – в следующих выпусках «Open Source».

Петр Семилетов
(tea@list.ru)

«Open Source» приглашает к сотрудничеству!

Электронное приложение «Open Source» всегда открыто для сотрудничества с новыми авторами, с читателями и их конструктивными предложениями по улучшению издания, обоснованной критикой и любыми отзывами, с компаниями, занимающимися разработкой и продвижением программного обеспечения с открытым кодом. Приветствуются все энтузиасты, желающие опубликовать у нас свои статьи. Тематика нужных материалов очевидна из предназначения приложения, то есть FOSS (Free and Open Source Software): теория и практическое применение; исторические сведения, анализ сегодняшнего положения, прогнозы на будущее и другие аспекты, связанные с открытым ПО.

Среди наиболее интересных на данный момент общих тем можно выделить:

- ☑ общие обзоры новых и/или интересных проектов Open Source и конкретных приложений, свежих версий дистрибутивов Linux, *BSD и других систем;
- ☑ советы и рекомендации новичкам в GNU;
- ☑ истории успеха применения/распространения ПО с открытым кодом;
- ☑ философия и идеология Free Software;
- ☑ разработка приложений с применением средств Open Source.

Желательный объем статей: 6000 или 12000 символов (с пробелами). Примеры актуальных сейчас тем для статей публикуются на http://osa.samag.ru/todo. Но не стоит строго ограничиваться приведенными выше рамками!

Публичное обсуждение «Open Source» проводится на форуме сайта журнала «Системный администратор» по адресу: http://osa.samag.ru/forum. Связаться с редакцией можно по электронной почте osa@samag.ru.
