Предисловие

Qt 4: замечания последней минуты

Выход этой книги приурочен к появлению официального релиза Qt 4.0.0, и уже подготовленная информация была мгновенно адаптирована к новой версии. Весь материал отражает только последние данные, хотя для пользователей предыдущих версий, в частности Qt 3, везде есть соответствующие ссылки. Прошу относиться снисходительно к возможным неточностям, поскольку моим главным приоритетом была актуальность.

Автор

Время открытых систем пришло!

Сложно сказать, когда возник феномен "открытых систем" — если под открытыми системами подразумевать свободное обсуждение предметной области. Возможно, сама идея связана с возникновением научного метода, когда утвердилась практика свободного обмена информацией в целях установления истины. Впрочем, еще раньше возникла идея защиты и коммерческого использования информации.

По крайней мере, первые компьютеры не были ни открытыми, ни даже доступными для широких масс. История создания вычислительной техники связана с глобальными проблемами, военными ведомствами, крупным бизнесом и спецзаказом, что не способствовало открытости. Конечно, в те времена тоже был определенный круг посвященных, и даже существует мнение, что именно тогда и был "золотой век" вычислений, но факт остается фактом — только немногие могли получить доступ к полной и достоверной информации об архитектуре вычислительных систем и методах создания приложений.

Ситуация изменилась с возникновением системы Unix. Парадокс заключался в том, что компания AT&T, где была разработана система, в силу антимонопольного законодательства, не могла производить операционные системы, поэтому первые версии бесплатно распространялись в университетах и научных лабораториях. Условно Unix можно считать первой открытой системой, хотя долгое время она существовала только как поле для экспериментов ограниченной группы из Bell Labs. Для действительно свободного распространения идей и приложений (в обоих направлениях) не хватало важных составляющих: дешевых и массовых компьютеров, а также Большой Сети как носителя информации. Первоначально Unix была открытой в смысле кода, но не в смысле настоящей дискуссии идей. Затем последовала эра коммерциализации Unix, порождения различных версий, клонов и завуалированных переделок, в результате чего общая картина Unix стала выглядеть как вавилонское столпотворение. Это тоже можно назвать "войной идей" —

но, в основном, это было войной денег, адвокатов и корпоративных амбиций: многие хотели заработать на открытом коде. Где они сейчас? Скажем так: в разных местах, в основном, снова пытаются впрыгнуть в уходящий трамвай, на этот раз Linux. На этом фоне независимая инициатива и настойчивость Билла Гейтса выглядят даже своего рода благородно.

Продукты Microsoft тоже сделали свой вклад в открытые системы, и, возможно, больший, чем кажется: MS DOS, а после и MS Windows 3.11 внесли элемент массовости в само движение "компьюнити", благодаря чему развилась отдельная субкультура программистов-любителей. К тому же часто интересно наблюдать альтернативные подходы Microsoft, к, казалось бы, уже давно решенным проблемам — это привносит дух соперничества и эффект "игр с Большим Братом". Наконец, по сравнению с тогдашними расценками на другие системы, стоимость DOS и Windows на момент выхода была чисто символической: в сочетании с доступными персональными компьютерами это породило волну домашних хакеров (некоторые к тому времени уже успели взломать первые игровые приставки Commodore и Atari), что до этого было невозможным. Наконец-то кодированием, в том числе и системным, мог заниматься каждый желающий: я вспоминаю первую программу, взломанную утилитой `debug`, и признаюсь, что это был один из самых запоминающихся моментов в моей жизни. Впоследствии Linux много унаследовала от этого "персонального" стиля, в том числе ориентацию на доступную платформу IBM PC.

Сегодня система Windows тоже отчасти интересна, но времена меняются, и то, что раньше было секретом за семью печатями, а именно построение операционных систем, сегодня является общедоступной информацией. "Раскол" начался, фактически, с BSD[1], которая стала первой, официально некоммерческой, операционной UNIX-системой. Чуть позже появилось понятие *открытых систем* — и это положило неотвратимый конец всем остальным системам. Конечно, у них еще есть время стать открытыми или, по крайней мере, поделиться идеями.

Говоря об открытых системах, нельзя переоценить значение Linux, но здесь нужно внести ясность. Многие считают, что Linux является прямым потомком Unix — и это вводит путаницу. Да, действительно, многие базовые принципы были позаимствованы из этой системы. Но также многое было позаимствовано из других систем, в том числе — из MS Windows. Ядро Linux вообще не имеет прямых аналогов, в то время как программные интерфейсы соответствуют спецификациям POSIX с некоторыми расширениями. Ядро Linux может поддерживать почти все из известных файловых систем. То же относится и к протоколам, интерфейсам и т.д. Вообще, в силу известного "правила", если есть какая-то технология, то она либо реализована под Linux, либо кто-то сейчас этим занимается. Открытые системы предполагают свободный обмен идеями, в том числе открытыми стандартами, документами и кодом (что также является разновидностью документов).

---

[1] BSD (Berkeley Software Distribution) — программное изделие Калифорнийского университета, адаптированная для Internet реализация операционной системы Unix с комплектом ее утилит, разрабатываемых и распространяемых университетом. — *Примеч. ред.*

На самом деле более интересен не "диалог" Windows-Linux, поскольку это, в основном, пока системы для различных рынков. Приятно наблюдать открытость как раз тех "военных" систем, которые долгое время стояли на страже "серьезного" компьютинга. Под влиянием открытых систем значительно изменили (или меняют) свой статус такие "черные ящики", как Sun Solaris, QNX или HP-UX. Значительную активность в этом же секторе проявляют и Intel, IBM, Oracle и Silicon Graphics. Суперкорпорации спешат получить выгоду от быстрого и бесплатного аудита кода, который предоставляет сетевое сообщество Open Source. В результате новые технологии отрабатываются прямо на глазах у заинтересованных сторон, на что раньше уходили, без преувеличения, десятилетия.

Нельзя сказать, что Microsoft не заимствует открытых идей; при изучении вопроса можно убедиться, что 90% идей и технологий MS Windows позаимствованы в открытых источниках — от таких базовых вещей, как протокол IP или Internet Explorer, и до сотен деталей интерфейса, баз данных, алгоритмов и т.д. "Гений" Билла Гейтса заключается как раз в игнорировании этого явного факта.

Главная проблема сегодняшнего дня — это отсутствие у разработчиков культуры программирования под Linux, слабое понимание важности создания совместимого кода с помощью универсальных инструментов. Люди с трудом представляют, что несколько открытых интерфейсов значительно мощнее и проще, чем многочисленные интерфейсы WinAPI или "тулбоксы" MacOS. Притом не только мощнее и проще, но и дешевле — ведь за использование WinAPI нужно платить на каждом этапе, от самого Visual Studio до системного и прикладного ПО клиентских машин.

Но деньги — не единственный фактор. Создание совместимых, или, как сейчас модно говорить, консистентных, программ уже само по себе является достаточной ценностью — это признак зрелости разработчика, лояльности к другим и уважения интересов пользователя. Программы, созданные по принципу "потому что VBA (Visual Basic for Applications) сегодня удобнее", лишены этих качеств (хотя простое требование "лицензионного офиса" сразу бы поставило все на свои места: VBA оказался бы дорогим и не таким уж удобным, не говоря о его недостатках). Сколько миллионов человеко-лет тратится на переход от пакета VBA на что-то более стоящее, думаю, не знает никто — но даже по своему скромному опыту могу предположить, что большая половина офисов всего мира уже раскаиваются в своей доверчивости.

Хорошая новость: многие инструменты и библиотеки из мира Open Source[2], примыкающие на различных условиях к этой категории программных продуктов, переносимы на многие платформы, в том числе на MacOS X и MS Windows. Представьте себе, как было бы прекрасно, если бы тот же набор бухгалтерских инструментов "1С Бухгалтерия" и еще многие и многие другие программы (например, Adobe Photoshop или Macromedia Flash) были бы изначально написаны с расчетом на многоплатформенность с использованием той же библиотеки Qt, а в качестве хранилищ данных предполагалось бы задействовать, например, MySQL. Безусловно, эти инструменты также созданы коммерческими компаниями, и их коммерческое использование (точнее — использование без обнародования исходного кода;

[2] Имеются в виду программные продукты с открытым исходным текстом. — *Примеч. ред.*

согласно GPL[3], коммерческое использование программ с открытым кодом не требует лицензирования используемых библиотек и средств разработки) повлекло бы лицензионные отчисления — но эти отчисления на пару порядков ниже, чем при использовании лицензионного ПО от Microsoft. Надежность же таких программ, благодаря публичному тестированию, велика даже по сравнению с известными продуктами. Другое дело, что у нас по-прежнему предпочитают скорее пиратские копии Windows, чем условия "jewel box"[4] — но это вопросы экономические и политические, которые мы не будем здесь обсуждать. А может быть, мы уже живем в том светлом будущем, когда Microsoft откроет код и позволит загружать, распространять и модифицировать Windows бесплатно? Это было бы разумно, удобно и хорошо — и кто знает, возможно, наступление открытых систем поможет сделать это реальностью.

А пока те меры, которые принимает Microsoft, в частности в Windows XP SP2, для защиты от взлома собственного ПО и ограничения прав пользователей, могут серьезно повлиять на настроение любителей нового "софта" от Microsoft. Атмосфера дискомфорта при взломе очередной версии MS Windows должна оградить многих пользователей от этой системы — купите Windows один раз за полную цену, и вы поймете, что переплатили. Еще большим шоком будет покупка MS Office — проделайте мысленно такую сделку, и Visual Basic уже не будет казаться вам таким привлекательным.

Кроме стоимости, важен еще и факт, что Windows уже проигрывает почти по всем параметрам Linux. Существует, конечно, такая область, как компьютерные игры, — действительно, для многих пользователей (и для меня в том числе) это пока является преградой для полного удаления разделов FAT/NTFS со своих жестких дисков. Но, возможно, ситуация и здесь тоже скоро изменится.

В остальных же областях (таких, как пользовательский интерфейс, графика и возможности мультимедиа) Linux уже сравнялась практически с любой версией Windows или даже превосходит ее. Исправлена ситуация с драйверами — например, дистрибутивы Fedora Core 4 или Ubuntu легко поднимают всю периферию моего компьютера, в то время как Windows XP SP2 без диска с драйверами не нашла почти ничего "особенного". Что, возможно, важнее — сами производители устройств не без гордости отмечают поддержку Linux как стандартную опцию (это уже не говоря о серверных возможностях и безопасности любой системы семейства *nix).

Сегодня работать под Linux уже становится не только выгоднее, но и удобнее. Задача разработчиков и менеджеров IT-департаментов — разобраться в выгодах и применять на практике открытые технологии (рис. 1). Разумеется, любая технология служит конечным пользователям, и как раз библиотека Qt — прямой путь к их сердцам. Простая в освоении и использовании, хорошо документированная и

---

[3] GPL (General Public License) — общедоступная лицензия, т.е. право на получение и свободное распространение программного обеспечения и исходных файлов за право распространения на тех же условиях модификаций этого программного обеспечения. — *Примеч. ред.*

[4] Jewel box — с англ. футляр компакт-диска (имеется в виду покупка на условиях минимальной комплектации). — *Примеч. ред.*

компании Trolltech — основа популярной графической оболочки KDE (сокращение от “K Desktop Enviroment” — графический пользовательский интерфейс фирмы Corel) и еще нескольких сотен популярных приложений. В дополнение ко всему, эта библиотека так же хорошо интегрируется с MS Windows и MacOS X, как и с Linux или FreeBSD. Вам больше не нужно выбирать операционную систему для своих приложений — просто не думайте об этом.

Рис. 1. *Что отличает настоящее приложение Open Source? Разумеется, кнопка Show Source (Показать исходный код) на главной форме*

По духу, мощности, удобству и оптимальности кода библиотека и инструменты Qt похожи на Borland Delphi — с поправками на использование в случае Qt языка C++ вместо Object Pascal, а также на работу Qt на различных платформах. По мере того как вы будете изучать все большее число классов Qt, вы откроете для себя прекрасную, стройную архитектуру этой библиотеки. Что важно — чем больше вы будете знать о Qt, тем проще вам будет казаться программирование в этой среде. В любом случае вам будет доступен исходный код самой библиотеки и всех инструментов разработки, так что на любой вопрос из серии “как, в точности, это работает” вы сможете получить исчерпывающий ответ.

И, я надеюсь, даже если жизнь заставляет вас работать и кодировать под Windows, вы все равно сможете сделать это с расчетом на будущее и будете использовать Qt. Это скоро окупит себя — благодаря быстрому коду, отличному интерфейсу и мощным возможностям, а также средствам, повышающим скорость разработки. А главное — ваши приложения будут жить еще долго после того, как MS

Windows уже станет преданием. Не исключено, мы еще станем свидетелями этого знаменательного события.

Арсений Чеботарев, научный редактор и штатный автор журнала "Компьютеры+Программы" (PC World Ukraine), автор журналов "Сети и Коммуникации" и "Шпиль", а также газеты "Computer World Ukraine". В активе: 19 лет компьютерной практики, от оператора СМ2 и ЕС 1055 до... собственно, до автора этой книги, включая многие годы программирования и системного администрирования.

Вступление или вопросы, требующие ответа

Для кого написана эта книга

Эта книга написана для программистов, предпочитающих C++ всем остальным языкам программирования и желающих создавать производительные приложения для Linux, переносимые, но не в последнюю очередь, на другие платформы. Выбор вами как программистом именно библиотеки Qt может быть обусловлен несколькими причинами.

Если вы программировали в конце восьмидесятых–начале девяностых под MS DOS на Turbo Pascal или Turbo C и впоследствии утратили нить в освоении новых системных интерфейсов, то, возможно, вам будет лучше использовать Qt, чем пытаться угнаться за другими технологиями. С помощью Qt совершенно не обязательно осваивать системные вызовы. Вы можете программировать по-прежнему, при этом создавая графические приложения, задействовать сетевые интерфейсы и протоколы, работать с трехмерной графикой и базами данных. Для этого придется всего лишь освоить несколько (ладно, несколько десятков) новых классов, и никаких системных вызовов. Системные вызовы — против правил Qt, можете забыть о них надолго (или даже навсегда).

Еще один вариант — это переход на новую операционную систему, в частности — Linux. Для группы разработчиков, "застрявших" в Visual Basic, эта проблема может представлять непреодолимый рубикон. Большое количество приложений, существующих под Microsoft Windows, необходимо или желательно "портировать" в среду Linux (точнее — то и другое, особенно это касается популярных компьютерных игр), поскольку эта система стремительно набирает силу, особенно в области производства, но также давно проникла во многие научные лаборатории и кампусы, откуда Linux, собственно, и родом, и даже во многие офисы.

Например, в издательстве, в котором я работаю, около 50% сотрудников работают на X-терминалах с загрузкой по сети. В такой схеме, при гарантированной системе доступа и резервирования данных, вы практически лишены проблем с винчестером и другими физическими носителями. Любой сотрудник может поставить свою CD-"болванку" в очередь на прожиг на разделяемом "райтере" и в разумное время, около получаса, получить результат. Аналогично в разумных пределах выполняются и другие операции, вроде печати на дорогостоящих цветных лазерных принтерах.

Это тенденция, которую нельзя игнорировать. С другой стороны, многие профессиональные приложения созданы в Unix, Linux или FreeBSD — их доступность для платформы Windows также может принести выгоду создателям.

Еще одна категория потенциальных пользователей Qt — начинающие программисты, желающие получить как можно более прочный фундамент для будущей карьеры и профессионального роста. Знание Qt является абсолютной ценностью, как в академическом мире, так и в промышленности или офисном программировании. Вы сможете применить свои знания в любой сфере и в любой стране, особенно там, где сильна субкультура UNIX и Linux — страны Прибалтики, Голландия, Финляндия, Швеция, Германия, Корея, Китай, США, а в последнее время — и Россия. Для наших разработчиков быть в авангарде открытых систем — вообще, редкая, если не единственная возможность заявить о себе в мире программного обеспечения.

Наконец, эта книга может стать побудительным мотивом для менеджеров, руководителей программных проектов и руководства софтверных компаний. Возможно, ваш проект еще не поздно поставить на рельсы Qt, чтобы в дальнейшем застраховаться от проблем перехода на новую платформу? Или вы чувствуете пристрастие к стремительно быстрым и небольшим программам? Возможно, вы захотите использовать существующий потенциал опытных разработчиков и софтвер-архитекторов из старой Unix-гвардии? В таком случае использование Qt может сэкономить немало времени и средств для получения отличных результатов.

Что необходимо знать до того, как вы начнете читать эту книгу

Знание C++ в той или иной мере обязательно. Если вы испытываете проблемы в понимании классов и объектов — потратьте некоторое время и изучите одну из фундаментальных книг по этому прекрасному языку программирования (эту фразу я встречал минимум 20 раз в различных книгах, и было бы невежливо нарушать традицию).

Также будет нелишним знание устройства и архитектуры операционных систем, в частности Linux и MS Windows. В частности, вам придется ориентироваться в устройстве файловой системы, основах сокетов Беркли, графических элементах управления, и, возможно, пригодятся базовые знания в области баз данных. Естественно, что отдельное приложение не должно использовать все возможности Qt — но ведь вы хотите стать универсальным программистом, не так ли? С другой стороны, поскольку библиотека Qt в значительной степени не зависит от операционной системы, то глубокие знания в системных API не являются обязательными. Плюс заключается в том, что зная системные вызовы, вы сможете приблизительно оценить производительность и эффективность различных операций.

Наконец, это знание технологии программирования. В частности, вы должны понимать принципы сборки программ, их версий, значение компилятора, сборщика и отладчика, уметь работать с командной строкой и писать несложные командные файлы. Разумеется, вы всегда можете “спрятаться” от проблем за некоторой оболочкой вроде Visual Studio, но это будет только временное и ненадежное решение

проблемы. Кроме прочего, Visual Studio не доступна на платформах, отличных от Windows, так что лучше заранее отказаться от применения таких "добрых" помощников или, по крайней мере, настроить среду для компиляции проекта в пакетном режиме.

Что такое, собственно, Qt

На этот вопрос существует два ответа: короткий объясняет происхождение аббревиатуры Qt как сокращения от Quasar Technologies. Это имя некоторое время носила голландская компания Trolltech сразу после своего создания в 1994 году.

Если говорить о Qt как о продукте — то это универсальная всеобъемлющая библиотека, или программная оболочка (framework), для создания универсальных, переносимых приложений на языке C++. Переносимость включает использование исходного кода для получения приложений, работающих на платформах MS Windows, Unix, Linux, MacOS X и Embedded Linux. Нужно понимать, что, в отличие от Java, вы должны скомпилировать тексты программы для каждой из платформ, чтобы получить выполняемый файл нужного формата.

Насколько распространено использование Qt

Библиотека Qt распространена даже в большей степени, чем вы могли ожидать! Основа такой популярности — двойная лицензия на библиотеку Qt, предусматривающая возможность как коммерческого, так и некоммерческого использования в проектах Open Source. В результате Qt стала основой KDE, самой популярной на сегодня графической оболочки для Linux и FreeBSD. Насколько распространена оболочка KDE, надеюсь, объяснять не нужно, хотя приверженцы GNOME (GNU[5] Network Object Model Environment) могут с этим не согласиться, правда, без особого успеха (рис. 2).

В числе коммерческих партнеров Trolltech, использующих Qt в качестве инструментальной среды, можно сразу же привести таких авторитетов в мире бизнеса, как IBM, Siemens, Sharp, Adobe, Bosh, Boeing, Scania, NEC, HP, NASA, и еще более четырех тысяч не вошедших в этот список компаний из более чем шестидесяти стран. Версия Qt Embedded и построенная на ее основе оболочка Qtopia в данный момент активно продвигаются на рынок мобильных терминалов и PDA[6].

Какая версия Qt является текущей

На момент начала написания этой книги рабочей являлась версия 3.3.3-2. После появления на сайте Trolltech окончательной версии 4.0 книга была проверена и отредактирована для отражения изменений, которых оказалось достаточно много, и многие из них достаточно серьезны. Первое впечатление от нововведений — положительное, хотя пользователям версии 3.3.3 придется как следует переучиваться. Эта книга как раз и поможет сделать это в минимальные сроки.

---

[5] GNU — рекурсивное сокращение от англ. "GNU is Not Unix"; имеется в виду проект, предусматривающий свободное распространение программного обеспечения. — *Примеч. ред.*

[6] PDA (Personal Digital Assistant) — "карманный" компьютер, предназначенный для выполнения некоторых специальных функций . — *Примеч. ред*.

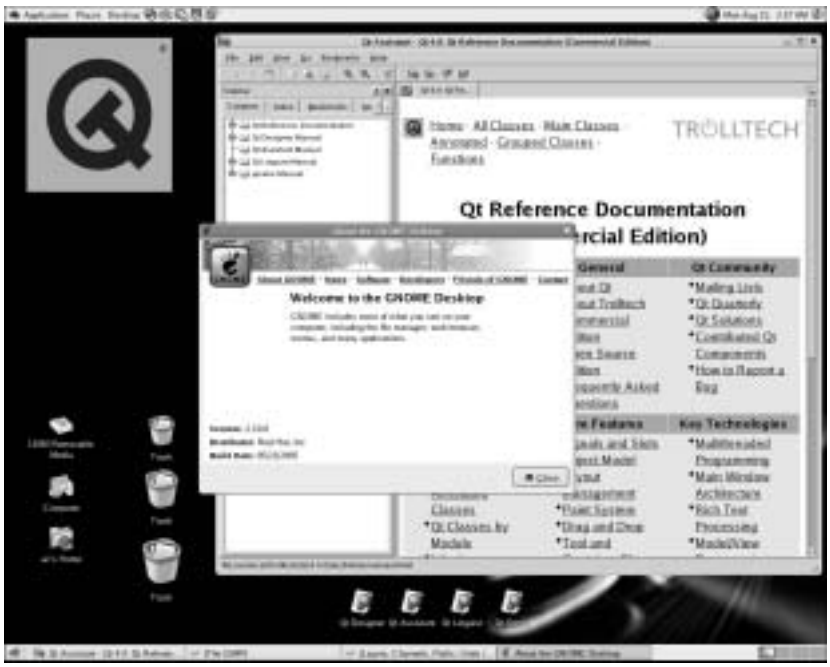

Рис. 2. *Хотя оболочка KDE написана на основе Qt, это не значит, что Qt не работает под управлением других оконных менеджеров*

В чем отличие Qt от Java

Действительно, говоря о сфере применения и целевых приложениях, библиотека Qt достаточно похожа на Java и является мостом между различными платформами. Различие в среде времени выполнения не так уж существенно: JRE (Java Runtime Engine) — тоже не более чем библиотека позднего связывания. Ко всему, наличие JIT-компилятора позволяет Java-приложениям реально конкурировать по производительности с C++.

Так что различие скорее в философии: Java предлагает экстенсивный подход, постоянно расширяемое дерево классов, предлагая многочисленные и не всегда совместимые расширения — сразу же всплывает в памяти сосуществование двух графических библиотек, AWE и Swing.

Библиотека Qt — скорее, интенсивный продукт; иерархия классов и сферы применения стабильны, расширения происходят как можно реже на основе накопленных исправлений и предложений пользователей. Основной акцент делается на повышении надежности и производительности существующего кода, а также более удобных инструментов разработки. Кроме прочего, это гарантирует надежность инвестиций в продукты, основанные на Qt.

Специалисты также выигрывают от такой стабильности: вы всегда сможете охватить Qt полностью и следить за всеми исправлениями и дополнениями. Для языка Java, как технологии, на современном этапе это вряд ли возможно.

В чем отличие Qt от Gtk

Действительно, многие, особенно любители сетевых дискуссий, часто поднимают вопрос "Qt vs[7] Gtk". Обе системы обладают приблизительно одинаковой мощностью и переносимостью. И у обеих систем достаточно адвокатов.

Gtk расшифровывается как GIMP Toolkit. Аббревиатура GIMP, в свою очередь, представляет собой сокращение от GNU Image Manipulation Program, т.е. программа, манипулирующая изображениями (подобная Photoshop). Конечно, впоследствии система Gtk приобрела свойства, далеко простирающиеся за обработку изображений (модуль, называемый GDK, GIMP Drawing Kit). В частности, были реализованы такие системные функции, как динамическая загрузка модулей, не зависимое от платформы многопоточное выполнение и глобальный цикл сообщений. Больше всего это напоминает реализацию MS Windows 3.11 поверх X-Windows, т.е. "операционную систему", лишенную файловой системы, доступа к устройствам и сетевых соединений.

К преимуществам Gtk следует отнести огромное количество "биндов" (привязок) к различным языкам программирования. Фактически, многие "малые" языки ставят своей первоочередной целью "бинд" к пакету Gtk, чтобы заполнить вакуум в собственных библиотеках. На сегодня Gtk-инструментарий "переведен" на 28 языков программирования.

Кроме того, пакет Gtk имеет некоторое преимущество в плане "действительно открытого способа разработки" — если для вас это является преимуществом. С другой стороны, Gtk не использует классы C++, а вместо этого, как и, например, WinAPI, реализует классы и систему сигналов вручную, поверх стандартного C.

По моему скромному мнению, программы и проекты, избегающие единого коммерческого начала и планомерного метода, страдают (и будут страдать!) синдромом "общественного мнения", поэтому лично мне ближе Qt. Хотя вам ничего не мешает использовать обе системы одновременно.

В чем отличие Qt от Tk/Tcl

Язык Tcl, с расширением для построения пользовательского интерфейса, называемым Tk (можно рассматривать Tk и как отдельный продукт, поскольку он иногда встречается без Tcl), является популярным средством для создания "фронтэндов" приложений, т.е. той части приложения, которая отвечает за взаимодействие с пользователем. Сам язык Tcl был задуман в качестве "клея" для приложений командной строки, с тем чтобы дополнить их графическим интерфейсом. Так что если ваше приложение уже существует в виде консольного приложения, то Tk/Tcl может оказаться лучшим способом "оживить" его пользовательский интерфейс.

В последние годы, однако, появилась тенденция к расширению Tcl до уровня универсального языка программирования, с тем чтобы полностью писать на нем свои приложения. Уже существуют приложения, содержащие тысячи строк кода на Tcl и выполняющие такие существенные операции, как доступ к сетевым ресурсам, базам данных и, конечно, взаимодействие с пользователем. Это стало возможно благодаря значительно возросшей мощности персональных компьютеров, а также с

[7] vs — от лат . "versus", означающего "против". — *Примеч. ред.*

тем, что приложения все больше находятся в состоянии ожидания или ввода пользователя, или ответа от сервера.

И все же Tcl имеет ограниченную область применения, связанную с производительностью. В том числе этот язык не подходит для вычислительных задач, таких как обработка видео в реальном времени или расчет трехмерных сцен. Также не подойдет он и для системного программирования. Не говоря уже о том, что интерпретируемый открытый текст нелегко оформить и защитить в качестве продукта.

С другой стороны, приложения Qt имеют производительность, сравнимую с производительностью оптимизированного ассемблерного кода, и способны на 100% (в рамках, отведенных операционной системой) занять процессор полезной нагрузкой. Также, работая в Qt, вы можете полностью сконцентрироваться на одном языке программирования, C++, в то время как приложения Tcl, как уже было сказано, часто являются "сборными" из написанных на C консольных приложений, "фронтэнда" на Tk и приложений Tcl, собирающих все вместе. В результате проектирование и поддержка приложений Tcl обычно требуют больше усилий, и результаты редко выглядят как коммерческие продукты.

Можно ли использовать Qt параллельно с системными вызовами

Безусловно, хотя лучше, пока существует такая возможность, избегать этого. Классы Qt созданы специально для того, чтобы скрыть от программиста системнозависимые участки кода и таким образом сделать ваш проект переносимым между платформами. Применяя системные вызовы, вы теряете переносимость кода — к чему тогда использование Qt?

Иногда, однако, системные вызовы — это единственная возможность достичь нужного результата. В таком случае вы обязательно должны так построить свой проект, чтобы системные вызовы были компактно представлены в отдельных модулях, хорошо задокументированы и их вызов был инкапсулирован в соответствующие классы.

С другой стороны, вы можете свободно использовать библиотеки, в переносимости которых уверены (например, libc или ATL).

Как читать эту книгу

Книга предназначена для последовательного прочтения, от начала и до конца. По крайней мере вначале прочтите первые главы, посвященные особенностям объектной модели Qt и классу `QWidget`. После освоения базиса любая интересная вам глава будет вполне доступной для понимания.

Перед запуском примеров я бы попросил вас хотя бы кратко ознакомиться с приложением А, в котором рассказывается о специальном препроцессоре `qmake`, обеспечивающем переносимость не только кода, но и процедуры сборки между различными платформами. Таким образом вы с самого начала сможете строить проекты в правильном ключе, по ходу практикуясь в синтаксисе `qmake`.

После того как вы в достаточной мере ознакомитесь с элементами управления и сможете бегло оперировать их свойствами и методами — вы, вероятно, сочтете более удобным пользоваться средством Qt Designer, описанным в приложении Б.

Я предостерегаю начинающих разработчиков от использования этого инструмента до тех пор, пока не сможете уверенно ориентироваться в сгенерированном им коде.

Чем эта книга отличается от документации

Я не хотел бы следовать авторам, которые вместо достоверных фактов и базовых данных о рассматриваемом предмете подают собственные домыслы. И хотя нет объективной точки зрения, но к этому все-таки нужно стремиться. В конце концов мы рассматриваем предметы антропогенного происхождения, так что было бы по меньшей мере невежливо игнорировать послания "богов" созданного ими мира.

Эта книга в значительной мере опирается на документацию и является ее свободным изложением — и я горжусь этим. Многие примеры взяты прямо из документации, чтобы вы могли прямо их компилировать или копировать из примеров в свои программы, не набирая новый текст в редакторе. Кроме прочего, я, как уже было сказано, рассчитываю на подготовленного читателя, который не нуждается в мотивации вроде "как много и быстро вы заработаете, используя данную технологию".

С другой стороны, эта книга значительно сэкономит ваше время, сконцентрировав ваше внимание на более важных вопросах программирования и почти проигнорировав те, которые не относятся к первоочередным или являются очевидными. Я нарочно не обращаю ваше внимание на простые вопросы вроде использования графических инструментов разработки для получения "быстрых" приложений, а также на такие изощренные вопросы, как, к примеру, создание собственных графических стилей, в чем должны быть скорее задействованы дизайнеры, чем программисты.

Поскольку я не планирую оставлять в покое Qt до конца своих дней (и я, и Qt, пожалуй, протянем еще немало тысяч дней), то, возможно, в следующих изданиях появится отдельная вступительная часть, описывающая создание простого приложения в стиле VBA, после чего последует настоящий материал, а также увеличится количество приложений, освещающих отдельные темы.

Чем эта книга отличается от других книг

Мне часто встречались компьютерные книги нескольких типов. Самыми тяжелыми из них, для моего восприятия, являются необъятные "библии", которые наполнены предложениями типа "нажмите на клавиатуре комбинацию клавиш <Alt+F> и в появившемся меню выберите команду Save". Эта книга имеет мало общего с "библиями", отчасти потому, что Qt — библиотека, а не приложение или среда разработки, и я просто лишен радости написать "нажмите клавишу <F1> для получения подсказки" (впрочем, один раз уже написал).

Еще одна распространенная категория литературы — наборы примеров, "поваренные книги" (cookbooks). В таких книгах вы можете на шестой странице узнать, как записать число римскими цифрами и в транскрипции на фарси, но тщетно будете искать азы техники программирования и основы данной иерархии классов. Эта книга — не набор рецептов, материал изложен полно и методически,

но это не сборник решений. Кстати, возможно, к моему стыду, я даже не знаю всех римских цифр, не говоря уже о фарси.

Также очень популярна в определенных кругах серия "для чайников". Хотя упомянутые книги порой остроумны, все же моя книга предназначена не для чайников, а скорее для обычных людей. Как уже было сказано, эта книга для тех, кто владеет или, по крайней мере, хотел бы овладеть языком C++ в совершенстве. Эти люди составляют элиту программистского сообщества и ни в коей мере не являются "чайниками". Я бы с удовольствием разместил здесь, как это принято в некоторых книгах "для чайников", рисунки моей старшей дочери, но, боюсь, меня не поймут.

Кроме прочего, я не включил в эту книгу привычные из американских переводов разделы "о чем рассказывается в этой главе". По-моему, это подсказка пользователю, какие из глав можно пропустить. Я нарочно не писал тех глав, которые вы должны пропустить. Как хороший современный компилятор, я исключил их еще на этапе компиляции — остальные написаны для вашего чтения.

В конце концов, это не справочник с характерными для него таблицами классов, свойств и функций-членов. Хотя тут описано большинство самых важных из более четырех сот классов Qt, не рассчитывайте на полное описание методов или списки свойств — для этого обращайтесь к документации.

Эта книга, скорее, представляет собой сборник конспектов по Qt или базовый курс. Кстати, курс Qt мог бы украсить любую учебную программу, и я рассмотрю любые предложения, если какой-нибудь из вузов решится на это. Главные вопросы, в том числе построение программ и основные классы, такие как `QWidget`, рассматриваются в самом начале, и им посвящено основное внимание. Другие классы упоминаются без детального рассмотрения или же им вообще не уделено внимания. К примеру, согласно алфавиту, класс `QApplication` находится далеко от начала списка классов — но было бы большой ошибкой рассматривать его после `QAccel`.

Примеры в этой книге носят сугубо иллюстративный характер. В отличие от книг, построенных вокруг одного большого проекта, каждый фрагмент демонстрирует только то, что должен демонстрировать. Можно сказать — в этой книге мало, т.е. недостаточно примеров для неподготовленного читателя. Опять-таки, возможно, следующее издание будет содержать большее количество авторских примеров.

Исходные тексты, по возможности, взяты из дистрибутива и "туториала" Qt, чтобы вы могли не набирать весь исходный текст. Я еще раз повторяюсь — примеры нарочно взяты из "туториала" Qt, чтобы вам было удобнее их компилировать, а не потому, что мне было лень заменить одни фрагменты исходных текстов другими. Обратите внимание: текст каждой программы-примера Qt начинается заголовком, позволяющим использовать его для любых целей.

Благодарности и прочее

Должен признаться, что за последние пять лет я, в основном, променял ремесло программиста на литературное поприще, поскольку вижу главную задачу не в создании отдельного приложения, а в популяризации и рекламе открытых систем. Если вы относитесь к активным программистам на Qt, то прошу относиться снисходительно к возможным ошибкам и опискам в данной книге.

В конце концов, эта книга полностью основана на интеллектуальной собственности Trolltech, документации и материалах сайта `trolltech.com` — и было бы странно, если бы было иначе. Я выражаю свою признательность всем создателям как самой библиотеки, так и сопровождающей ее документации за неоценимую услугу как лично мне, так и всему человечеству.

От издательства "Диалектика"

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: info@dialektika.com

WWW: http://www.dialektika.com

Наши почтовые адреса:

в России: 115419, Москва, а/я 783

в Украине: 03150, Киев, а/я 152