FoxTalk

Декабрь 2003
№ 12 (78)

русское издание

Solutions for Microsoft® FoxPro® and Visual FoxPro® Developers

Что получится, если скрестить лису с пингвином?

Уилл Хентцен (Whil Hentzen)

Откроются безграничные возможности для расширения вашего бизнеса, утверждает главный редактор журнала FoxTalk Уилл Хентцен.

Декабрь 2003

- From the Editor: **Что получится, если скрестить лису с пингвином?. . . 1**
  Уилл Хентцен
- **Работа Visual FoxPro под управлением ОС Linux 3**
  Пол Макнетт
- The Kit Box: **Составьте коллекцию 11**
  Энди Крамек и Марсиа Акинз
- **Лучшая альтернатива комбинированному списку. . . . 16**
  Эдвард Макдермотт
- Playing With the GUI in VFP 7: **Использование схемы «3Dots». 21**
  Предраг Боснич

материал имеет отношение к соответствующей версии

материал имеет отношение к соответствующей платформе

исходные тексты программ можно скачать из Интернета

Десять лет тому назад один гуру, известный в мире СУБД Clipper, постулировал, что дни независимого разработчика остались в прошлом, поскольку для того, чтобы оставаться независимым, слишком много всего надо знать. С тех пор мы выучили... язык структурных запросов SQL. Интерфейс WinAPI. Построитель экранов Screen Builder (сниппеты зло или благо?). Обработку данных в многопользовательском режиме. Операционную систему Novell. Инструментальный набор Connectivity Kit. Как использовать заключение программного кода в скобки (bracketing code) для организации кросс-платформенных приложений. Подключение к серверам SQL Server и Oracle. Технологию DDE, затем OLE, потом Automation для того, чтобы использовать в Fox преимущества, предоставляемые другими приложениями. Службы Web Services. Язык разметки HTML. Internet. Спецификацию XML. Взаимодействие с платформой .NET.

Думаю, мы убедились в том, что тот парень заблуждался — мы обрели почти безграничные способности к изучению нового, разве вы не согласны?

Когда я оглядываюсь на 2003 год, то явственно вижу, что положение «лисы» на Windows-рынке — это состояние застоя. В этой области трудно найти работу. Какова бы ни была причина (или причины) такого положения, конечным результатом является то, что для развития своего бизнеса вы должны вести поиски вне «арены», принадлежащей Visual FoxPro.

Есть несколько направлений развития информационных технологий, предоставляющих определенный шанс. За последние несколько лет мы расширили тематику журнала FoxTalk настолько, что теперь она включает рассмотрение не только VFP, но и других компонентов из набора Visual Studio, работу с серверной СУБД SQL Server, изучение технологии Automation, платформы .NET, наладонные ПК и Internet. Пора добавить еще одну возможность к постоянно пополняемому набору ваших умений и навыков. И

такой возможностью является операционная система Linux. Использование ОС Linux нарастает семимильными шагами. Эта операционная система все более широко применяется во всех отраслях компьютерной индустрии. А это означает, что через пару лет можно ждать открытия огромных возможностей для бизнес-приложений, работающих под управлением ОС Linux. Вы обладаете такими навыками, которых нет у большинства разработчиков: вам знакомы нормализация баз данных, разработка бизнес-приложений, ООП, шаблоны проектирования, спецификация UML и так далее, и так далее. Мы даже не осознаем, сколь многому мы научились за то время, пока занимались практической разработкой.

Позвольте мне рассказать вам одну историю. Несколько месяцев тому назад я присутствовал на встрече пользователей из Linux User Group, на которой обсуждалась СУБД PostgresSQL. Докладчик в течение пары лет использовал эту серверную (back-end) базу данных и объяснил, что вы могли бы организовать связь между двумя таблицами при наличии в каждой из этих таблиц поля, где хранятся одни и те же данные. Такие действия означают, что вам не надо хранить одну и ту же информацию в нескольких записях: вы могли бы иметь одну таблицу с записями о покупателях, а другую — с заказами, и организовать связь между этими двумя таблицами посредством поля, в котором хранится номер покупателя.

Докладчик пояснил, что назвал такие поля «объединительными полями» («joiner fields»), после чего я задал вопрос: «Вы имеете в виду первичный и внешний ключи, верно?» Он ответил: «Вы можете называть эти поля, как вам будет угодно, мне нравится называть их «joiner fields».

Понятное дело, при наличии таких «экспертов», тут кроются огромные возможности.

Как вам воспользоваться этими грядущими возможностями? Во-первых, истиной является то, что СУБД VFP не работает под ОС Linux, а последние заявления Fox-команды указывают на то, что мы могли бы «посинеть» от ожидания, когда же изменится эта ситуация. Так что тогда? Вы могли бы просто «нагрузить» Windows и выучить языки программирования сценариев Perl, Python и PHP. Или вы могли бы попытаться создавать приложения с помощью каких-то иных средств, вроде среды программирования Kylix фирмы Borland (грубо говоря, это эквивалент Delphi для ОС Linux). Но есть лучший путь. Не думаю, что к вам обратятся с просьбой разработать клиентские приложения для ОС Linux, по крайней мере, не сейчас. Напротив, я полагаю, первое, что произойдет в ближайшие 12 месяцев — это то, что многих из вас попросят организовать связь между приложением, созданным средствами Visual FoxPro, с серверной базой данных, работающей на Linux-сервере. Поэтому вы захотите освоить использование ОС Linux в качестве запасной «настольной» системы с тем, чтобы получить некоторое представление о том, как она работает. И тогда, если появится необходимость подключиться к серверу, вы к этому будете готовы.

Вот с чего следует начать. Обзаведитесь копией дистрибутива RedHat 8.0 — вы можете бесплатно загрузить образы дисков с сайта www.redhat.com или (это способ для тех из вас, кто не думает, что капитализм является чем-то потусторонним) приобретите компакт-диски в магазине. Если вам не подходит ни то, ни другое, напишите мне письмо по электронной почте: у меня осталось несколько копий компакт-дисков и я счастлив буду отослать их вам, если вы вышлете мне оплаченный почтовый конверт со своим обратным адресом.

Причина, по которой вам необходима именно версия RedHat 8.0 (а не Debian, или Mandrake, или какие-либо иные дистрибутивы), заключается в том, что эта версия является самой простой в инсталляции и в использовании. Это поистине невероятно. Вспомните то волнение, которое охватило вас, когда вы впервые загрузили версию FoxPro 2.0 с немодальными командным окном и панелями разработки? Или версию VFP 3.0 с таким большим количеством команд, функций и возможностей, что вам понадобилось пять лет на то, чтобы у вас дошли руки до всего этого изобилия? Такое же волнующее чувство охватывает вас и в данном случае. Вы получите не только безопасную, стабильно работающую операционную систему с графическим интерфейсом, который соперничает с интерфейсом ОС Windows XP, но также набор «офисных» приложений, включающий текстовый процессор, электронные таблицы и программное обеспечение для подготовки презентаций, все это совместимое на уровне файлов с пакетом Microsoft Office (начиная с лета прошлого года я использую офисный набор из состава версии RedHat для подготовки журнала FoxTalk), и клиентское приложение, которое предоставляет собой электронную почту/календарь/планировщик и составляет конкуренцию приложению Outlook — одним словом, вы получите все за исключением проблем с безопасностью.

Идем дальше. Найдите запасную машину. Это не обязательно должна быть очень «навороченная» система, я использовал пятилетней давности модель Dell 500 МГц с 256Мб оперативной памяти и парой «тяжеловесных» жестких дисков 7,200 RPM IDE 8Гб. Компакт-диск с дистрибутивом RedHat является загрузочным, если ваш дисковод в состоянии обеспе-

чить загрузку с CD-диска (если нет, вам потребуется немного потрудиться и потратить минуты две на то, чтобы создать загрузочную дискету из первого компакт-диска). Поместите первый компакт-диск в дисковод, с полдюжины раз щелкните мышью по кнопке OK, чтобы согласиться с теми настройками, которые предлагаются по умолчанию (я единственный раз отказался от предложенного значения, когда указал IP-адрес и имя машины вместо того, чтобы принять используемые дистрибутивом RedHat по умолчанию настройки DHCP и "localhost"), и вот вы инсталлировали операционную систему RedHat. Это и в самом деле так просто. ОС распознала всю мою аппаратуру (включая видеокарту circa-1996 и 17-дюймовый монитор, такой старый, что я, помнится, за новый заплатил за него 1200 долларов). При том условии, что в процессе инсталляции один раз потребовалось сменить компакт-диск, вся процедура заняла в общей сложности около 40 минут. Затем еще пять минут я потратил на то, чтобы получить обновления с сайта www.redhat.com. Мне не пришлось соглашаться с массой гнусных лицензионных соглашений EULA, которые позволяют кому-то еще иметь свободный доступ к моему компьютеру, или вынужденно в принудительном порядке активизировать продукт, за который я уже заплатил. А когда дело было сделано, я повернулся в своем кресле к другому столу и использовал те же самые компакт-диски для инсталляции системы на «детской» машине и потом еще раз для ее установки на другой машине, предназначенной для моей жены. И все это я проделал, не опасаясь того, что представители полицейского подразделения по борьбе с пиратством в области программного обеспечения собираются постучать в мою дверь.

Думаю, вы согласитесь: версия RedHat 8.0 — это настоящий мастер по части «удивить», и я предсказываю, что она намеревается завоевать обширные территории, включая предприятия ваших заказчиков. Следовательно, вы одолжите самих себя, поместив еще один инструмент в свой ящик с «подручными средствами», предназначенными для работы с Visual FoxPro; и, таким образом, когда поступит телефонный звонок от вашего старого или потенциального заказчика с просьбой связать созданное вами VFP-приложение с серверной базой данных, функционирующей на базе ОС Linux, вы сможете предложить этому клиенту некоторое решение, а не отсылать его к кому-то еще, кто предусмотрительно подготовился заранее и знает нечто такое, чего не знаете вы. Готовьтесь к расширению своего бизнеса сейчас.

❑❑❑

Работа Visual FoxPro под управлением ОС Linux

Пол Макнетт (Paul MacNett)

Хотя фирма Microsoft во всеуслышание заявляет об отказе выпускать версию Visual FoxPro для альтернативных платформ, надежда заполучить «многоплатформенную» VFP все-таки есть. Пол Макнетт демонстрирует, как мы можем овладеть ситуацией и самостоятельно поселить «лису» в новую среду обитания ОС Linux, используя для этого продукт с открытым кодом, который называется Wine.

Помните, в прошлом веке одним из самых существенных факторов, стимулирующих продажу FoxPro, была возможность исполнения и эксплуатации программного обеспечения этой СУБД на базе не одной, не двух и не трех, а четырех различных платформ?

Для меня наличие такой возможности стало мощным покупательным стимулом при выборе среды разработки: хотя я неизменно эксплуатировал свои программы под управлением ОС Windows, меня обнадеживало сознание того, что я не окажусь загнанным в угол, потребуй мои клиенты версий программного обеспечения, работающих в среде операционных систем Mac, Unix или DOS.

Сравните эту ситуацию с днем сегодняшним: с появлением очередной версии Visual FoxPro с ее новыми важными функциональными возможностями и исправлениями ошибок, я каждый раз ловлю себя на той мысли, что мы, как VFP-разработчики, находимся в довольно тяжелом положении, поскольку вынуждены вести разработку и осуществлять эксплуатацию на одной-единственной платформе.

Вы видели самые последние версии рабочих столов, реализованные в операционных системах Mac OSX или Linux? Тенденции в среде ваших клиентов таковы, что они стремятся использовать эти развитые операционные системы, и если вы похожи на

меня, вы захотите иметь возможность соответствовать пользовательским желаниям. К сожалению, если вы хотите эксплуатировать свое ПО на нескольких платформах, это намерение с большой вероятностью означает приобретение иных инструментальных наборов: Delphi/Kylix, Java или, может быть, C++. Вам придется научиться работать в этих инструментальных средах и при этом обходиться без встроенного в Visual FoxPro механизма курсоров. Не знаю как вы, но я не готов отказаться от мощи Visual FoxPro только потому, что фирма Microsoft отказывается обеспечить миграцию на другие платформы. В этой статье я продемонстрирую вам, как добиться работы интегрированной среды разработки (IDE) и исполняемых приложений Visual FoxPro под управлением операционной системы Linux. Надеюсь, если с VFP можно будет работать в ОС Linux, «лиса» вновь обретет «второе дыхание» и начнет новую жизнь, а разработчики, использующие Visual FoxPro, снова могут оказаться весьма востребованными на рынке труда.

Головоломка на тему, как заполучить VFP, работающую на рабочем столе Linux, складывается из трех фрагментов. Для начала вам необходима инсталляция Linux на рабочей станции вместе со всеми инструментальными средствами разработчика (gcc, make и так далее). Второе, вам необходима инсталляция VFP в среде ОС Windows, откуда вы будете копировать папки верхнего уровня и вложенные в них папки с программами VFP в файловую систему ОС Linux (чтобы избежать проблем с лицензированием, убедитесь в том, что вы не исполняете Windows- и Linux-инсталляции VFP одновременно). Третье, вам необходимо инсталлировать самую последнюю версию бесплатного продукта с открытым кодом, который называется Wine. Смотрите на Wine как на промежуточный слой, обеспечивающий совместимость с ОС Windows. Когда ваше VFP-приложение делает свои стандартные обращения к интерфейсу Windows API, эмулятор Wine перехватывает эти обращения и успешно имитирует интерфейс Windows API с помощью своего собственного программного кода: ваше FoxPro-приложение понятия не имеет о том, что в действительности оно исполняется не под управлением ОС Windows! На сегодняшний день довольно много Windows-приложений успешно работают в среде ОС Linux, используя эмулятор Wine. В данной статье внимание концентрируется на вопросах инсталляции и конфигурации Windows-эмулятора Wine, поскольку он является самым важным фрагментом упомянутой головоломки. В последующих статьях будет рассмотрено, что делать дальше.

Действительность (будьте реалистами)

Эмулятор Wine — это программное обеспечение, находящееся в состоянии alpha-версии. Раз так, не все в нем работает корректно. В среде Linux/Wine СУБД Visual FoxPro будет вести себя несколько иначе по сравнению с тем, к чему вы привыкли: в основном эти отличия касаются мелочей (но иногда окажутся затронутыми и важные вещи). Не следуйте, пожалуйста, описываемым в этой статье шагам с той мыслью, что все будет превосходно работать прямо «из коробки». Вам придется провести свое собственное тестирование и составить собственное представление относительно того, насколько это реально — эксплуатация вашего конкретного VFP-приложения в среде ОС Linux. Вы должны рассматривать данный опыт как эксперимент, и если в конечном итоге вы решите воспользоваться его результатами в производстве, самое важное — это организовать надлежащее резервное копирование вашего исходного программного кода. Со временем, когда продукт Wine будет полностью готов, не останется и ощутимой разницы между работой VFP под управлением ОС Windows по сравнению с ее работой в среде ОС Linux. До того времени, впрочем, это остается нашей задачей — отыскать различные проблемы и документировать их с тем, чтобы можно было найти и интегрировать в Wine-проекты средства их исправления.

Я планирую использовать Visual FoxPro под управлением ОС Linux в качестве своей основной среды разработки, поскольку, основываясь на тех испытаниях, которые я провел до сих пор, считаю, что эта среда достаточно надежна и достаточно функциональна для моих целей. Однако, за долгие годы я выработал у себя привычку стремиться к простоте: например, минимизировать зависимость от внешних DLL- и OCX-библиотек. Если вы зависите от элементов управления ActiveX (даже от тех из них, которые входят в поставку VFP), планируйте возникновение дополнительных конфигурационных проблем и возможно даже ситуаций, когда необходимо отказаться от использования таких элементов управления. Впрочем, эта статья была написана тогда, когда версия VFP 8 находилась на стадии beta-тестирования: трудно сказать, являются ли проблемы, с которыми я столкнулся, следствием тех вопросов, которые решаются в VFP, или тех, которые связаны с эмулятором Wine.

Ладно, теперь, когда вы прочли этот отказ от ответственности, пора запустить процесс, в результате которого Visual FoxPro будет работать под управлением инсталляции ОС Linux.

Выберите разновидность

ОС Linux и инсталлируйте ее

Если вы только приступаете к работе с ОС Linux, изобилие возможных вариантов выбора, которыми вы располагаете на каждом шаге в процессе инсталляции и конфигурации, может оказаться устрашающим. Вы не знаете, нужна ли вам инсталляция RedHat или SuSE, Gnome или KDE, vi или emacs, понятия не имеете о том, как разместить раздел подкачки (swap partition), и так далее, и тому подобное. Не слишком переживайте по этому поводу. Просто возьмите дистрибутив (подсказка — возьмите версию RedHat 8, поскольку в этой статье я использую именно ее), выберите вариант инсталляции на рабочую станцию, проверьте, чтобы были установлены инструментальные средства разработчика (они понадобятся вам позже для компиляции эмулятора Wine из исходного кода), и вы будете готовы без промедления приступить к делу. Не смешивайте свои занятия: инсталлируйте ОС Linux на второй компьютер, где имеется жесткий диск, содержимое которого вы не боитесь затереть. Есть несколько способов заполучить версию RedHat 8: можно купить ее в магазине, можно приобрести книгу в комплекте с инсталляционными компакт-дисками, загрузить файлы дистрибутива с сайта www.redhat.com или одолжить у приятеля. Описание процедуры инсталляции ОС Linux выходит за рамки данной статьи, но исходя из предположения, что у вас имеется достаточно новый компьютер со свободным жестким диском и приводом CD-ROM, обеспечивающим возможность загрузки с него системы, вы, вероятно, найдете этот процесс удивительно безболезненным, даже приятным.

Загрузка дистрибутива и инсталляция эмулятора Wine

Версия RedHat и другие дистрибутивы поставляются вместе с инсталляциями эмулятора Wine, но, учитывая с какой скоростью развиваются такие проекты, как Wine, эти инсталляции быстро устаревают. Возьмите себе за правило почаще заглядывать на сайт www.winehq.com с целью получить самую последнюю версию эмулятора. Имеются три различных варианта загрузки:

- 1. Двоичные пакеты (binary packages) для конкретных дистрибутивов ОС Linux (rpm, dpk).
- 2. Запакованный архив исходных текстов (tar.gz).
- 3. CVS-дерево.

Обычно, простым и безопасным путем является использование двоичных пакетов, особенно в том случае, если вы только приступаете к работе с ОС Linux. Считайте, что двоичные пакеты — это эквивалент инсталлятора Windows Installer. Все необходимое для исполнения программы находится в пакете программ, и после его инсталляции запись о данном приложении появится в главной базе данных управления пакетами программ. Диспетчер пакетов программ знает, куда следует поместить каждый файл, какие необходимо добавить связи и все прочее. В зависимости от того, что именно вы инсталлируете, вам даже может быть понадобится явно подтвердить свое согласие с условиями лицензионного соглашения или принять какие-либо простые решения относительно конфигурации.

Архивы исходных текстов — это запакованные архивы всего исходного программного кода проекта. Вы скачиваете и распаковываете такой архив и остаетесь один на один со структурой каталогов, в которой представлены исходный код и вспомогательные файлы проекта. При желании вы можете модифицировать этот исходный код, а затем выполнить компиляцию и инсталляцию программного обеспечения. Не переживайте, это не так трудно, как может показаться; компиляция вашего первого проекта с открытым кодом с успехом может обеспечить вам маленький глоток свободы и волнение, обусловленные чувством, что ваша машина снова подвластна вам.

В CVS–дереве (Concurrent Versioning System — система параллельного контроля версий) хранится весь исходный программный код с внесенными в него самыми последними изменениями. Как только Александр Джулльярд (Alexandre Julliard), осуществляющий поддержку проекта Wine, получает программную «заплатку», представленную одним из

Важные отличия

Между операционными системами Windows и Linux есть важные отличия, которые могли бы вызвать у вас проблемы, даже если вы уверены, что в точности следуете инструкциям.

1. Во-первых, в ОС Linux команды и имена файлов чувствительны к регистру, поэтому будьте внимательны, вводя их с клавиатуры. «Wine-2002.tar» — это не то же самое, что «wine-2002.TAR».
2. Вся работа, описываемая в данной статье, осуществляется в вашем домашнем каталоге, а именно /home/pmcnett. Вместо того, чтобы все время набирать строку «/home/pmcnett», вы можете использовать символ тильды. Так, строка “~/xxx” означает каталог xxx, находящийся в вашем домашнем каталоге, следовательно, для меня строка “~/xxx” означает “/home/pmcnett/xxx”.
3. Символ “.”, помещенный перед именем каталога, указывает на текущий каталог, так что вы уверены в том, что находитесь в надлежащем месте.

сотен разбросанных по всему миру разработчиков, он применяет эту «заплатку» к текущей версии программного кода, а затем передает внесенное изменение в CVS-дерево. Те, кто получил свое приложение Wine из CVS-дерева, ходят по краю пропасти и рискуют столкнуться с какой-либо проблемой в том случае, если самое последнее из внесенных изменений что-либо испортило. Wine-разработчики зависят от доступа к действующему CVS-дереву; пользователи же должны были бы вместо этого следовать двоичным или архивным маршрутом.

Для примеров, публикуемых в этой статье, я буду использовать вариант 2 — архивы исходного кода. (*Замечание редактора*: не позволяйте, чтобы ваш статус новичка в мире ОС Linux подтолкнул вас к попытке воспользоваться вариантом 1 — откомпилировать ваши собственные архивы исходного программного кода в самом деле очень просто! Попробуйте!) Загрузить и инсталлировать в вашу систему приложение Wine можно следующим образом:

- 1. Загрузите ОС Linux и графический интерфейс XWindows, зарегистрируйтесь в системе как обычный пользователь (не регистрируйтесь, пожалуйста, как корневой (root) пользователь), подключитесь к Internet, откройте у себя в браузере страницу www.winehq.com/download и перейдите в раздел «Wine Source from Tarball». Вам нужен файл, который называется Wine-20021219.tar.gz. Выберите загрузку этого файла по некоторому адресу, находящемуся в пределах вашего домашнего каталога. Предполагаю, что это будет каталог ~/wine. Размер файла Wine-20021219.tar.gz составляет приблизительно 7.7Мб: дожидаясь окончания загрузки, вы можете перейти к шагу 2. В приводимых мной здесь примерах используется файл Wine-20021219.tar.gz, который ко времени написания этой статьи содержал самую последнюю из имевшихся версий. Обратите внимание на то, что к тому времени, когда эта статья попадет к читателям, на сайте будут также представлены более поздние релизы, однако, пока еще останутся доступными и предыдущие версии. Даже несмотря на наличие более позднего релиза, я посоветовал бы загрузить версию, указанную мной, — по крайней мере, вначале — для гарантии того, что мои примеры работают, как предполагалось. Обратите внимание на то, что в имени файла требуется указать дату выпуска данной версии. На данном этапе, поскольку продукт пока еще находится на стадии alpha-тестирования, релизы приложения Wine не имеют номеров для отслеживания версий, но используют взамен дату выпуска. Включение версии продукта в имя файла является стандартной ситуацией при загрузке открытого программного кода, что, конечно, удобно, поскольку частые релизы являются в данном случае нормой.
- 2. Откройте окно командной оболочки shell. Почти все, что я делаю, чтобы сконфигурировать эмулятор Wine и ОС Linux, делается с помощью shell-окна, а не графических инструментов. По ходу дела вы оставите это shell-окно открытым и откроете другие окна, и будете переходить из одного окна в другое по мере проработки данного примера. Для работы в этом командном окне объявите себя суперпользователем (root) (вам потребуется знание корневого пароля):

```
su -
```

- 3. Теперь важно убедиться в том, что из вашей системы удалены все предыдущие инсталляции эмулятора Wine. В системах, где используется диспетчер пакетов RPM (RedHat), первой выполняется проверка базы данных пакетов с целью убедиться в отсутствии предыдущих RPM-инсталляций приложения Wine. Следующая команда выполнит для вас такую проверку:

```
rpm -q --all | grep -i "wine"
```

Если команда не возвращает никакой информации, значит у диспетчера RPM нет сведений о том, что приложение Wine когда-либо было установлено в данной системе. В противном случае вам придется деинсталлировать эмулятор Wine в соответствии со всеми записями, которые были возвращены по предыдущему запросу, используя следующую команду:

```
rpm -e --allmatches --nodeps <package name>
```

где имя пакета вводится явно в том виде, как оно было указано на выходе, после исполнения запроса rpm -q. Одно только то, что все следы эмулятора Wine были удалены из диспетчера RedHat Package Manager, не означает, что приложение Wine нигде больше не существует в вашей системе. Выполните поиск инсталляций, введя в своем командном окне суперпользователя следующую команду:

```
cd /
find -name wine
```

На этот поиск может уйти минута или две. Если какие-либо инсталляции найдены, удалите их вручную. Например, вы могли бы обнаружить вот такие файлы: /usr/local/bin/wine и /usr/local/bin/winedbg. Удалите их, набрав команду, вроде этой:

```
rm /usr/local/bin/wine*
```

Удалите также все библиотечные файлы, такие как libwine.*, в wine-каталоге /usr/local/lib и сам этот каталог. Если вы нашли wine-каталог, вы можете его удалить, исполнив команду:

```
rm -r --force /usr/local/lib/wine
```

- 4. Теперь, когда вы загрузили исходные тексты эмулятора Wine (предполагаю, что вы загрузили эти тексты в каталог ~/wine/Wine-20021219.tar.gz), пора их распаковать и разместить надлежащим образом. Для работы с вашей текущей командной оболочкой использована корневая учетная запись, но вы не должны инсталлировать эмулятор Wine в качестве корневого пользователя. Вы могли бы просто набрать команду exit в командной оболочке, чтобы выйти из режима суперпользователя, но почему бы вам вместо этого не запустить просто-напросто еще одну командную оболочку? Теперь у вас есть одна командная оболочка, которая готова и ждет выполнения команд в режиме superuser, и другая, предназначенная к использованию обычными задачами уровня пользователя. Обратите внимание на то, что командная строка дает вам визуальную подсказку относительно того, в каком качестве вы зарегистрировались, и если вы когда-либо запутаетесь в этом вопросе, то можете набрать в командной строке команду whoami. Так или иначе, переходите в окно новой командной оболочки (оболочки обычного пользователя) и введите следующие команды:

```
cd ~/wine                          ;сменить каталог
ls -al                             ;пролистать каталог
tar -xzvf Wine-20021219.tar.gz     ;после "W" нажмите TAB
ls -al
```

Теперь вы должны видеть, наряду с исходным сжатым архивом, каталог с именем ./wine-20021219. В этой структуре каталогов находится множество файлов, некоторые инструментальные средства и исходный программный код, а также сборочные make-файлы для эмулятора Wine. Если хотите, ознакомьтесь со структурой этого каталога. В частности, у вас может возникнуть необходимость внимательно прочесть README-файл:

```
cd wine-20021219   ;после "w" нажмите TAB
vi README
```

Для перемещения по этому README-файлу используйте клавиши pageup/pagedown и клавиши со стрелками, а когда закончите чтение, нажмите клавишу <ESC>:q.

- 5. Теперь, когда у вас есть разархивированный исходный код, почти уже пришло время сформировать и инсталлировать эмулятор Wine. Впрочем, есть одна важная «заплатка», которую мы должны прежде применить к исходному коду эмулятора Wine и которая позволит Visual FoxPro работать в среде ОС Linux. Загрузите файл с этой «заплаткой», который находится по адресу http://cvs.winehq.com/patch.py?id=7029 (сохраните результат в каталоге ~/wine/vfpwinepatch7029). Примените эту «заплатку», выполняя в консольном окне в режиме обычного пользователя следующие команды (кстати, последний аргумент — это буква «p» с последующей цифрой «один», а не буква «p» с последующей буквой «l»):

```
cd ~/wine/wine-20021219
cat ../vfpwinepatch7029 | patch -p1
```

По сути дела, вы осуществляете конвейерную пересылку (piping) выходного потока patch-файла в patch-команду, которая выполняет синтаксический разбор patch-файла и применяет «заплатку» к нужному исходному файлу, в данном случае к файлу ./windows/queue.c.

- 6. После установки «заплатки» исходный код готов для компиляции и инсталляции. Эмулятор Wine поставляется в комплекте с инструментальным средством, которое должно помочь вам осуществить эти действия, еще более упрощая процедуру. В консольном окне в режиме обычного пользователя (не в корневой консоли), наберите следующее:

```
cd ~/wine/wine-20021219
./tools/wineinstall
```

Сценарий wineinstall начинается с конфигурации вашей сборки (build). На экран будет выдан большой объем текста без малейшего шанса для вас прочесть его: не волнуйтесь, многословный выходной поток — это нормальное явление. Примерно минуту спустя вы увидите следующее сообщение:

```
We need to install wine as root user, do you want
us to build wine, 'su root' and install Wine?
Enter 'no' to continue without installing
(yes/no)
```

Это пример того, насколько недружественными по отношению к пользователю могут быть Linux-приложения. Не забудьте, эмулятор Wine пока еще находится на стадии alpha-тестирования. В сущности, если вы ответите “no”, приложение Wine все равно будет сформировано, но оно не будет инсталлировано — это не слишком полезно в нашем случае. Наберите “yes” <enter>, чтобы процесс продолжился, и отправляйтесь перекусить: сборка будет длиться целую вечность. (Ну ладно, мо-

жет быть недружественно, но не без индивидуальности, поскольку следующее сообщение гласит «Building... Go to lunch, grab a video, whatever...» — «Сборка... Отправляйтесь на ланч, посмотрите видеофильм, займитесь чем-нибудь...»).

По завершении процесса сборки вы получите приглашение ввести ваш корневой пароль. Введите его, чтобы процесс продолжался, убедившись со всем тщанием, что вы ввели этот пароль правильно, поскольку если вы ошибетесь, исполнение сценария прекратится и вам придется повторить этот шаг с самого начала. Ответьте "yes", чтобы создать локальный файл config (~/.wine/config). Обратите внимание на то, что инсталляционный сценарий создаст для вас новый каталог ~/.wine/. Это тот каталог, где хранятся различные файлы, которые будут влиять на поведение эмулятора Wine на этапе исполнения. Мне следовало бы на минуту вернуться назад и поговорить немного о следующем сообщении, которое вы увидите:

```
Searching for an existing Windows installation...
not found. (no matching /etc/fstab mount entry
found)

Windows was not found on your system, so I
assume you want a Wine-only installation.
Am I correct? (yes/no)
```

Эмулятор Wine можно настроить таким образом, что он будет использовать существующую инсталляцию ОС Windows. Скажем, у вас есть система с двойной загрузкой, когда вы можете загрузить ОС Linux или ОС Windows, что является распространенной практикой. Если вы корректно настроили инсталляцию ОС Linux для монтирования Windows-драйверов, тогда вы можете исполнять программное обеспечение, предназначенное для работы под управлением ОС Windows (те программы, которые работают с эмулятором Wine), без необходимости копировать и/или переустанавливать его в среде Wine. Есть множество вещей, которые при таком методе могут работать неправильно (при этом не последнее место принадлежит тому факту, что официально ОС Linux не поддерживает запись на NTFS-тома), по этой причине простоты ради ответьте, пожалуйста, «yes» на вопрос о том, хотите ли вы получить инсталляцию «только Wine». Согласитесь с теми условиями, которые предусмотрены по умолчанию в следующем диалоге, нажав клавишу <enter>:

```
Some fake Windows directories must be created,
to hold any .ini files, DLLs, start menu
entries, and other things your applications
may need to install. Where would you like your
fake C drive to be placed?
(default is /home/pmcnett/c)
```

Вы действительно можете выбрать каталог по своему усмотрению, но он обязательно должен быть подкаталогом вашего собственного домашнего каталога или же вы не будете иметь соответствующих прав доступа.

Тестирование инсталляции эмулятора Wine

Теперь, когда вы инсталлировали эмулятор Wine в своей системе, пора убедиться в том, что он работает. Эмулятор Wine должен был инсталлировать несколько знакомых утилит в каталог ~/c/windows/. Наберите следующую команду в консольном окне в режиме обычного пользователя:

```
cd ~/c/windows
ls -al
```

Увидели запись с именем notepad.exe? Как видите, в действительности это ссылка на библиотечный файл, который называется /usr/local/lib/wine/notepad.exe.so. Такова Wine-реализация блокнота Notepad, хорошо знакомого Windows-приложения для редактирования текста: попробуйте исполнить это приложение, чтобы убедиться в правильной работе Wine-инсталляции:

```
wine notepad.exe
```

Поскольку это первый запуск эмулятора Wine на исполнение, вы увидите как у вас перед глазами со свистом промелькнет масса текста, пока будут настраиваться атрибуты шрифта. Через минуту или около того вы должны будете увидеть окно программы Notepad, и если вы с ней поработаете, то обнаружите, что функциональное поведение этого приложения очень схоже с поведением его Windows-двойника. Выберите в меню команду Choose File | Exit, а затем снова запустите на исполнение wine-приложение notepad.exe (подсказка — нажмите клавишу «стрелка вверх», а потом клавишу enter) и обратите внимание на то, что запускается эта программа мгновенно. Однако, это не такое уж большое дело: в сущности, вы доказали, что можете загрузить Wine-версию приложения notepad.exe в среде ОС Linux, но вы пока еще не исполнили под управлением ОС Linux ни одного «родного» Windows-приложения.

Следующий тест заключается в том, чтобы взять несколько «родных» исполняемых Windows-модулей из каталога c:\windows, который находится в вашей Windows-системе. Я советую испробовать следующие Windows-программы в работе под управлением ОС Linux: sol.exe (Solitaire), calc.exe (Calculator) и regedit.exe (Registry Editor). Программу Regedit.exe можно использовать для редактирования реестра приложения Wine, который эмулирует ре-

естр операционной системы Windows. Если вы испытываете программу regedit.exe, переименуйте ее в winregedit.exe (в эмуляторе Wine существует собственная, работающая только в текстовом режиме реализация программы regedit, и в противном случае возникнет конфликт имен) и запустите ее на исполнение, набрав команду wine winregedit.exe. Тестируйте работу этих встроенных Windows-программ, скопировав их в свой каталог ~/wine/c/windows/, и затем немедленно их сотрите, чтобы избежать возможных лицензионных проблем.

Исполнение этих приложений производит большее впечатление, нежели работа блокнота Notepad, поскольку в данном случае вы запускаете под управлением ОС Linux исполняемые Windows-модули (бинарные файлы), хотя они и являются простыми автономными программами. Более сложные приложения имеют большее количество зависимостей по сравнению с этими небольшими приложениями-утилитами, поэтому они потребуют более сложных конфигурационных настроек. Приложения Microsoft Office, Internet Explorer и Visual FoxPro — все они могут работать в операционной среде Linux/Wine, но вы не сможете «инсталлировать» их в традиционном для Windows смысле этого слова. Вам придется скопировать все необходимые файлы и каталоги и внести корректные изменения в конфигурацию эмулятора Wine, чтобы заставить их работать. Следующие шаги продемонстрируют вам, как сконфигурировать эмулятор Wine для исполнения Visual FoxPro.

Копирование VFP-файлов и конфигурирование Wine для VFP

Вы почти готовы к загрузке интегрированной среды разработки Visual FoxPro в операционной среде Linux. Вам потребуется скопировать VFP-файлы в структуру каталогов операционной системы Linux и внести несколько изменений в конфигурацию Wine-инсталляции:

- 1. Создайте соответствующие каталоги в структуре каталогов эмулятора Wine. Для этой статьи я тестировал версии VFP с 5 по 8. В командном окне в режиме обычного пользователя наберите следующее (попробуйте нажать клавишу <tab> вслед за нажатием на клавишу “P”, чтобы раскрыть этот каталог, и попробуйте нажать клавишу «стрелка вверх» для строк 2-4, чтобы «сэкономить» на вводе с клавиатуры):

```
mkdir ~/c/Program\ Files/vfp5
mkdir ~/c/Program\ Files/vfp6
mkdir ~/c/Program\ Files/vfp7
mkdir ~/c/Program\ Files/vfp8
```

Вас смутил обратный слэш, разделяющий «Program» и «Files»? Не смущайтесь. Ответ прост — пробелы в именах файлов и каталогов не являются стандартным явлением в ОС Linux, и обратный слэш позволяет избавиться от них для нормальной работы командного интерпретатора. Это позволяет эмулятору Wine имитировать глупое Windows-соглашение относительно каталога c:\Program Files.

- 2. Скопируйте программные VFP-файлы из Windows-инсталляции. Используемые для этого методы будут различаться в зависимости от того, насколько доступны ваши Windows-тома из Linux-инсталляции. Моя рабочая станция, работающая под управлением ОС Windows NT, в локальной сети представлена как отдельный компьютер, поэтому я могу, например, копировать свою инсталляцию версии VFP 7, выполняя следующие команды:

```
cd ~/c/Program\ Files/vfp7
smbclient //nt_workstation/e$ -U Administrator
cd /prg/vfp7
recurse
prompt
mget *
```

В этом примере команда smbclient используется для подключения Windows-системы с сетевым именем “//nt_workstation” к разделяемому ресурсу с именем “e$” и входа в систему по административной учетной записи Administrator. Затем переходим в тот каталог, где инсталлирована VFP 7. Включается режим recurse, так что при копировании мы также получаем и все подкаталоги. Режим prompt отключается, следовательно, нам не надо подтверждать выполнение операции копирования для каждого файла и подкаталога. Команда mget * отдает smbclient распоряжение обработать все файлы. Смысл в том, что вам необходимо скопировать всю полностью VFP-инсталляцию на эмулированный приложением Wine C-драйвер. Если вам не удается скопировать VFP-инсталляцию таким способом, можно было бы прибегнуть к следующему методу: записать структуру каталогов VFP на компакт-диск с последующим монтажом этого CD-диска в ОС Linux (mount /mnt/cdrom) и скопировать файлы и подкаталоги оттуда.

- 3. Скопируйте пару файлов из системного Windows-каталога System32. Для копирования используйте тот же самый метод, что и на шаге 2, но в качестве целевого укажите каталог ~/c/windows/system. Вам необходимы файлы msvcr70.dll и oleaut32.dll. Кроме того, получите все исполняемые модули vfp (vfp*), которые могут находиться в каталоге Windows\System32 или в каталоге Program Files\Common Files\Microsoft Shared\VFP,

в зависимости от версии VFP, но в качестве целевого по-прежнему укажите Wine-каталог ~/c/windows/system. Поскольку все эти файлы инсталлированы или обновлены при инсталляции VFP, вы не должны столкнуться с лицензионными проблемами, копируя их теперь в ОС Linux. Не забудьте обзавестись отдельной лицензией для VFP, если вы будете эксплуатировать Windows-версию наряду с Linux-версией.

- 4. Теперь вам необходимо отредактировать конфигурационный файл эмулятора Wine (~/.wine/config), внеся в него, по крайней мере, одно изменение. Первое изменение касается того, как действует эмулятор с точки зрения операционной системы. Приложение Wine может имитировать все версии операционной системы Windows, по умолчанию это будет имитация версии win98, но VFP не будет работать под управлением эмулятора Wine в этом случае: нам необходимо заменить данную версию на версию NT (nt351, nt40, win2k или winxp). Проделайте это, вставив следующие строки в конец конфигурационного файла ~/.wine/config перед закрывающей строкой “# </wineconf>”:

```
;; Visual FoxPro Entries:

;; VFP5:
[AppDefaults\\vfp5.exe\\Version]
"Windows" = "nt351"

;; VFP6:
[AppDefaults\\vfp6.exe\\Version]
"Windows" = "nt351"

;; VFP7:
[AppDefaults\\vfp7.exe\\Version]
"Windows" = "nt40"

[AppDefaults\\vfp7.exe\\DllOverrides]
"oleaut32" = "native"

;; VFP8:
[AppDefaults\\vfp8.exe\\Version]
"Windows" = "win2k"

[AppDefaults\\vfp8.exe\\DllOverrides]
"oleaut32" = "native"
```

Если у вас есть собственные приложения, созданные с помощью VFP, которые вам необходимо исполнять, придется добавить соответствующие записи, аналогичные предыдущим, предназначенным для VFP. Например, у вас, скажем, имеется созданное в среде vfp7 приложение с именем dmp32.exe. Вам следовало бы добавить следующую запись:

```
;; DMP32:
[AppDefaults\\dmp32.exe\\Version]
"Windows" = "nt40"
```

Никакие DLL-библиотеки из исполняемых модулей перезаписывать не надо, только из интегрированной среды разработки VFP.

Работа Visual FoxPro под управлением ОС Linux

Теперь у вас все должно быть готово к запуску VFP. В Linux-консоли, предназначенной для работы обычного пользователя, наберите следующее:

```
cd ~/c/Program\ Files/vfp7
wine vfp7
```

На экране появится интегрированная среда разработки VFP вместе с командным окном. Прежде чем что-то делать, нажмите комбинацию клавиш Alt+Tab, чтобы переключиться на другое приложение, затем нажмите комбинацию клавиш Alt+Tab еще раз, чтобы вернуться в окно VFP. Вы сейчас заметите, что активным является командное окно с мерцающим курсором. Итак, поместите VFP на пустой рабочий стол. Щелкните мышью по полю управления, которое находится в верхнем левом углу окна, выберите в меню пункт To Desktop... и выберите рабочий стол desktop 4. VFP-окно исчезнет с вашего активного рабочего стола (предположительно, это был рабочий стол desktop 1) и появится на рабочем столе desktop 4. На панели задач (taskbar) выберите рабочий стол desktop 4. VFP должна исполняться на своем собственном рабочем столе, чтобы избежать проблем со странным переключением окон в момент появления дочерних окон, принадлежащих механизму IntelliSense.

Теперь поиграйте с VFP, работающей под управлением ОС Linux. Попробуйте создать и откомпилировать тестовый исполняемый модуль. Попытайтесь запустить на исполнение ваши готовые приложения. В следующий раз я подробнее расскажу о том, как работает VFP под управлением ОС Linux, и познакомлю вас с некоторыми предостережениями и обходными путями решения тех проблем, с которыми вы наверняка столкнетесь, а также более подробно расскажу об исполнении ваших откомпилированных VFP-приложений этапа исполнения под управлением ОС Linux. Как видите, VFP загружается и исполняется, но есть вещи, которые работают некорректно. Команда WAIT WINDOW, например, не показывает никакого текста. Имеется множество небольших проблем, но в основном VFP-ядро работает просто прекрасно.

Это начало нового приключения, и будущее покрыто мраком. Может быть, фирма Microsoft со временем поймет свою ошибку и обеспечит перенос VFP на платформу Linux. Или, может быть, она этого не сделает. В конце концов, благодаря проекту Wine, в сущности, это не имеет значения.

❑❑❑

Составьте коллекцию

Энди Крамек и Марсиа Акинз (Andy Kramek and Marcia Akins)

В версии 8.0 Visual FoxPro представлены несколько новых базовых классов, которые существенно расширяют функциональные возможности этого продукта. Однако, если рассматривать эти новые базовые классы с точки зрения готовых приложений, то мы должны провести испытания и на основе их результатов решить, в самом ли деле нам необходимо модифицировать уже существующие приложения. Один из таких новых классов — это базовый класс Collection. В своей статье Энди Крамек и Марсиа Акинз разбираются с тем, что же такое коллекция, как она работает и что она может сделать для нас в Visual FoxPro.

Энди: В новой версии реализована масса по-настоящему замечательных новых возможностей, но в ней есть также одна вещь, относительно которой я не вполне уверен в том, что могу с пользой ее применить — это коллекции.

Марсиа: Коллекции широко используются в большинстве продуктов фирмы Microsoft. По существу, один лишь язык программирования Visual FoxPro выделяется из общего ряда тем, что не имеет встроенных коллекций.

Энди: Да, но чем, собственно говоря, является коллекция? Может, это просто модное название для массива? В конце концов, у нас уже есть (в языке программирования Visual FoxPro) самая сложная насколько только можно себе вообразить обработка массивов, и если ты используешь тот класс, о котором мы говорили ранее (см. статью «Мы достойны массивов!» в декабрьском номере журнала FoxTalk за 2002 год), то легко можешь дополнить встроенные возможности для обработки массивов поиском в них элементов, строк и столбцов.

Марсиа: Хорошо, вот определение коллекции в том виде, как оно дается в справочном Help-файле, предусмотренном для языка программирования Visual Basic:

> *Коллекция — это упорядоченный набор элементов, на которые можно ссылаться по отдельности.*

и далее приводится замечание о том, что коллекция обеспечивает простой способ для того, чтобы можно было ссылаться на группу связанных элементов как на единый объект. Однако, как указывается, главное преимущество заключается в том, что эти элементы (или члены) коллекции не обязательно должны принадлежать к одному и тому же типу данных.

Энди: Мне понятно, когда такая возможность может стать важнейшим дополнением в языке программирования Visual Basic: поскольку единственный способ, с помощью которого ты можешь получить массивы со смешанными типами данных, — это объявить тип массива как Object, или, другими словами, как коллекцию. Мне хотелось бы знать, является ли это таким уж преимуществом в языке программирования Visual FoxPro, где мы и без того можем использовать массивы, в которых хранятся данные смешанных типов?

Марсиа: Ну, может быть и нет. Интересно, что в справочном Help-файле для Visual FoxPro дается совершенно иное определение коллекции:

> *Вы можете использовать коллекции для того, чтобы сгруппировать набор связанных элементов, обычно объектов, которые могут принадлежать к любому типу. Коллекции обеспечивают механизм для работы с объектами, хранящимися в контейнерах, и предоставляют стандартные способы организации доступа к объектам в коллекции и их итеративного перебора. Класс Collection функционирует как истинный контейнерный класс, несмотря даже на то, что он не обладает методом AddObject в отличие от классов Form и PageFrame.*

Энди: Хм-м, да, пожалуй, в этом определении значение придается иным вещам, верно? Поскольку справочный Help-файл у тебя уже открыт: какие свойства, события и методы есть у класса Collection в языке программирования Visual FoxPro?

Марсиа: Помимо стандартных свойств и методов, которые есть у всех объектов, — а именно, BaseClass, Class, Init(), Error() и Destroy() — у класса Collection очень мало «собственных» свойств и методов. Я перечислила их в таблице 1 вместе с кратким описанием того, что они делают.

Таблица 1. Свойства и методы класса Collection.

Имя	Тип	Описание
Count	P	Содержит количество элементов в коллекции и поддерживается автоматически. Если коллекция пуста, Count = 0. (Замечание: свойство Count используется в режиме «только-для-чтения».)
KeySort	P	Числовое значение, определяющее порядок, в котором VFP будет просматривать коллекцию в итеративном режиме при использовании конструкции FOR...EACH. Возможные значения: 0 (по умолчанию) — индексные номера элементов изменяются в порядке возрастания (1, 2, 3,...,n). 1 — индексные номера элементов изменяются в порядке убывания (n,...,3, 2, 1). 2 — ключи элементов упорядочены по возрастанию (Artichokes, Beans, Cauliflower,...,Zucchini). 3 — ключи элементов упорядочены по убыванию (Zucchini,...,Cauliflower, Beans, Artichokes).
Add	M	Этот метод добавляет элемент в коллекцию. По желанию можно указать имя ключа, ассоциируемого с данным элементом, а также местоположение добавляемого элемента в коллекции относительно существующего элемента («перед элементом» или «после элемента»). Увеличивает значение свойства коллекции Count.
Remove	M	Этот метод удаляет из коллекции элемент с указанным индексом. Уменьшает значение свойства Count. (Замечание: для того, чтобы вернуть номер индекса, вы можете воспользоваться методом GetKey() указанного элемента.)
GetKey	M	Этот метод возвращает соответствующий элементу коллекции ключ. Если в качестве параметра передано числовое значение, метод возвращает ключ (если указан) данного элемента или пустую строку. Если в качестве параметра передан символьный ключ, метод возвращает либо номер индекса соответствующего элемента, либо 0. Передача в качестве параметра каких-либо иных значений приводит к возникновению ошибки «Function argument value, type, or count is invalid» (номер ошибки 11).
Item	M	Метод возвращает указанный элемент коллекции. Может получать в качестве параметра числовое значение индекса элемента или его символьный ключ. Если указанный элемент не существует, возникает ошибочная ситуация «Index or expression does not match an existing member of the collection» (номер ошибки 2061).

Энди: Я тут обратил внимание на одну вещь: диапазон допустимых значений свойства Keysort начинается с нуля! Это довольно необычно для языка программирования Visual FoxPro, где мы в большей степени привыкли нумеровать четыре возможных варианта от 1 до 4, а не от 0 до 3. Я имею в виду тот факт, что у нас есть пять режимов буферизации (учитывая отсутствие буферизации), и они пронумерованы от 1 до 5. Аналогично, опции для таких свойств, как SourceType, UpdateType и WhereType, — все они нумеруются от 1 до n.

Марсиа: Верно, в контексте языка программирования Visual FoxPro такой подход несколько странен, но последовательности, начинающиеся с нуля, — это обычное явление в COM-компонентах и элементах управления ActiveX. Взгляни только с помощью браузера объектов на тип enums из любой библиотеки типов, и ты увидишь их там во множестве.

Энди: Логично (сказал он, не будучи в действительности убежден). Итак, в чем же заключаются преимущества этого нового класса Collection?

Марсиа: Во-первых, давай уточним одну вещь: в Visual FoxPro ты можешь реализовать коллекции самостоятельно, используя массивы в сочетании с методами access и assign. Так что здесь нет ничего действительно нового, просто вместо необходимости самому писать программный код для их реализации, механизм обработки коллекций теперь встроен в язык программирования. У коллекции по сравнению с массивом есть три главных преимущества:

- Свойство Count поддерживается автоматически. Это устраняет необходимость следить за длиной массива и тестировать каждый столбец, чтобы определить, не является ли строка на самом деле пустой.
- Ты можешь обращаться к элементам в коллекции либо по их позиционному индексному номеру, либо по смысловому ключу, и можешь по желанию переключаться между этими двумя режимами адресации.
- Есть встроенные методы для добавления элементов «в» и удаления их «из» коллекции.

Энди: Хорошо, тогда давай сравним эти два способа. Вот фрагмент программного кода для создания массива из названий четырех видов фруктов:

```
DIMENSION afruits[4,2]
 aFruits[1,1] = NEWOBJECT( 'Empty' )
 aFruits[1,2] = "Apples"
 aFruits[2,1] = NEWOBJECT( 'Empty' )
 aFruits[2,2] = "Bananas"
 aFruits[3,1] = NEWOBJECT( 'Empty' )
 aFruits[3,2] = "Cherries"
 aFruits[4,1] = NEWOBJECT( 'Empty' )
 aFruits[4,2] = "Damsons"
```

Марсиа: А вот эквивалентный код для создания коллекции фруктов (кстати, что такое «damson»?):

```
oFruit = NEWOBJECT( "collection" )
oFruit.Add( NEWOBJECT( 'Empty' ), 'Apples' )
oFruit.Add( NEWOBJECT( 'Empty' ), 'Bananas' )
oFruit.Add( NEWOBJECT( 'Empty' ), 'Cherries' )
oFruit.Add( NEWOBJECT( 'Empty' ), 'Damsons' )
```

Энди: Damson — это мелкая черная слива (по-моему, изначально она импортировалась из Дамаска, отсюда и ее название). Должен заметить, что этот

код гораздо компактнее. Теперь давай посмотрим: я могу получить ссылку на объект Cherries из моего массива двумя способами. Во-первых, я могу явно запросить элемент [3,1], исходя из того предположения, что мне известно: строка 3 столбец 1 — это объект Cherries.

```
loCherry = aFruits[3,1]
```

Марсиа: При всем должном уважении, такой способ не слишком практичен. Часто ли тебе действительно известны номера строки и столбца? Как правило, твои усилия направлены на то, чтобы получить именно эту информацию. Но, используя коллекцию, очень легко сделать и это:

```
loCherry = oFruit.Item(3)
```

Энди: Согласен, по этой причине я обычно использовал функцию ASCAN() для того, чтобы найти второй столбец с именем cherries и вернуть номер строки. Затем я использовал этот номер строки для того, чтобы получить объектную ссылку на первый столбец соответствующей строки, вот так:

```
lnRow = ASCAN( aFruits, 'cherries', -1, -1, 2, 15 )
loCherry = aFruits[ lnRow, 1]
```

(Замечание: если вы не используете версию VFP 8.0, вы не можете поступить указанным образом; но почему бы тогда вам не воспользоваться версией VFP 8.0?)

Марсиа: Очень ловко. Ты всего лишь должен помнить о необходимости передать в качестве параметров два раза “-1”, чтобы определить начальный элемент поиска и количество искомых элементов, один раз “2”, чтобы ограничить поиск вторым столбцом массива, и не забыть о том, что передача в качестве последнего параметра значения “15” означает выполнение поиска без учета регистра в режиме точного совпадения и в качестве результата возвращает номер строки. Я в таком случае явно отдаю предпочтение коллекции!

```
loCherry = oFruit.Item( 'Cherries' )
```

Энди: Ха! Несмотря на тот факт, что ключ является чувствительным к регистру, я начинаю соглашаться с тобой: до сих пор использовать коллекцию было намного проще.

Марсиа: Так, теперь давай удалим объект cherries из коллекции и из массива. Как бы ты это сделал?

Энди: Это легко. Просто воспользуйся функцией ADEL(). Конечно, нам необходим индекс удаляемого элемента, поэтому требуется пара строк программного кода, вроде вот этого:

```
lnRow = ASCAN( aFruits, 'cherries', -1, -1, 2, 15 )
ADEL( aFruits, lnRow )
```

Марсиа: Но ты удалил этот объект не полностью! Теперь в массиве есть пустая строка: тебе, помимо прочего, необходимо изменить размерность массива.

Энди: О да, конечно! Итак, необходимый код в полном объеме таков:

```
lnLen = ALEN( aFruits, 1 )
lnRow = ASCAN( aFruits, 'cherries', -1, -1, 2, 15 )
ADEL( aFruits, lnRow )

DIMENSION aFruits[ lnLen –1, 2 ]
```

Марсиа: А эквивалентный программный код применительно к коллекции вот такой:

```
oFruit.Remove( 'Cherries' )
```

Та-там!

Энди: Я убежден, работать с коллекцией намного проще, чем с массивом! Можем ли мы делать с коллекциями что-нибудь еще?

Марсиа: Да. Метод GetKey() вернет ключ, если ты передаешь индекс в качестве параметра (что позволяет тебе организовать выборочную обработку), вот так:

```
FOR lnCnt = 1 TO oFruit.Count
   lcKey = oFruit.GetKey( lnCnt )

   IF LOWER( lcKey ) = "damsons"
     *** Игнорировать damsons
   ELSE
     *** обрабатываем остальное
   ENDIF

NEXT
```

Энди: И я полагаю, что если тебе известен ключ, то можно использовать его значение, чтобы получить индекс (хотя теперь я начинаю недоумевать, зачем бы кому-то понадобились оба этих значения), вот так:

```
lnIndex = oFruit.GetKey( 'Damsons' )
```

Марсиа: Но мы еще не упомянули о самой важной возможности класса Collection: этот класс является базовым классом Visual FoxPro и его можно использовать для создания подклассов! Это означает, что мы можем создавать свои собственные классы Collection с дополнительными пользовательскими свойствами и методами.

Энди: Итак, ты утверждаешь, что где бы массив ни использовался, потенциально его можно было бы заменить коллекцией. Но еще важнее то, что ты могла бы заменить массив классом Collection, не только обрабатывающим хранилище данных вроде массива, но к тому же обладающим такими функциональными возможностями, которые в противном случае были бы реализованы в программном коде, внешнем по отношению к массиву. Короче говоря, ты можешь инкапсулировать массив.

Марсиа: Совершенно верно. На сопровождающей дискете есть пользовательский класс Collection, который именно это и делает применительно к различным типам файлов. В библиотеке классов colbase.vcx определены корневой класс (colfiles) и специализированные подклассы (colprgs, colforms и colvcxs) для работы с файлами программ, формами и библиотеками классов.

Энди: Здорово. А как это работает?

Марсиа: Ну, каждый подкласс имеет четыре пользовательских свойства, которые должны быть определены так, как показано в таблице 2.

Таблица 2. Свойства пользовательского класса Collection.

Свойство	Описание
cfiletype	Тип файлов в текущей коллекции. По умолчанию используется значение ALL («*»), определенное в корневом классе. Отдельные подклассы определяют соответствующее расширение (такое как «PRG», «SCX» или «VCX»).
copencmd	Команда, с помощью которой открывается файл из коллекции. По умолчанию команда не указана. Если ничего не указано, то для того, чтобы получить необходимую команду в первый раз, используется функция InputBox, вызываемая из метода FileModify().
cruncmd	Команда, используемая для исполнения файла из коллекции. По умолчанию команда не указана. Если ничего не указано, то для того, чтобы получить необходимую команду в первый раз, используется функция InputBox, вызываемая из метода FileRun().
laddquotes	Если это свойство имеет значение .T., то имя файла берется в кавычки как в команде Modify, так и в команде Run. По умолчанию это свойство имеет значение .F..

Энди: Понимаю. Итак, чтобы воспользоваться подклассом colprgs, необходимо просто указать значения для свойств: cFileType = “PRG”, cOpenCmd = “Modify Command”, cRunCmd = “Do”. Кавычки нам не нужны, поэтому для свойства lAddQuotes может быть оставлено значение, принятое по умолчанию. Какие методы есть у этих подклассов?

Марсиа: Опять-таки, необходимо лишь несколько методов, поскольку большая часть функциональных возможностей предоставляется самим классом коллекции. Все, что требуется, — это методы-оболочки для обработки необходимых нам конкретных функциональных возможностей. В таблице 3 представлены пользовательские методы.

Таблица 3. Методы пользовательского класса Collection.

Метод	Описание
filesadd	Находит файлы указанного типа в текущем рабочем каталоге и дочерних подкаталогах первого уровня и добавляет их в коллекцию. Использует файловую структуру, определенную в свойстве cFileType.
fileslist	Формирует список файлов в коллекции и выдает их на экран в виде «всплывающего» меню. Выбранные файлы запоминаются в пользовательском свойстве cSelFile для использования методами Run и Modify.
filemodify	Открывает файл, сохраненный в свойстве cSelFile, используя команду, указанную в свойстве cOpenCmd. Если свойство cSelFile пусто, вызывает метод FilesList().
filerun	Исполняет файл, сохраненный в свойстве cSelFile, используя команду, указанную в свойстве cRunCmd.
makefileobj	Защищенный метод, который возвращает объект file, добавляемый к коллекции. В качестве параметров получает путь доступа и имя файла.

Энди: Едва лишь взглянув на программный код метода FilesAdd(), я вижу, что это обычный VFP-код, в котором для формирования списков файлов из текущего каталога использованы функция ADIR() и локальные массивы. Метод MakeFileObj() просто использует базовый класс Empty и добавляет некоторые основные свойства файла: имя, путь доступа к этому файлу, тип файла и атрибут «runas» (который, как я понимаю, является простой конкатенацией пути доступа и имени файла) и возвращает ссылку на объект данного файла. Я понимаю так, что единственным специфичным для коллекции фрагментом программного кода является одна только вот эта строка:

```
This.Add( loFile, JUSTSTEM( laJunk[ lnFCnt, 1 ]) )
```

Марсиа: Все, что заключено в этой строке, это ссылка loFile (объект файла) и массив laJunk, где хранятся имена файлов, которые мы используем как ключ в коллекции, и этот ключ используется и выдается на экран с помощью метода FileList(). И этот метод также крайне прост. Большая часть его программного кода, как ты можешь видеть, относится к формированию «всплывающего» списка файлов. Мы используем свойство коллекции Count для того, чтобы управлять процессом формирования меню, а метод GetKey() для того, чтобы извлечь те

имена файлов, которые мы сохранили в качестве ключей:

```
LOCAL lnBar, lnCnt, lcName
 PRIVATE pcSelFile

 WITH This
   DEFINE POPUP shortcut SHORTCUT RELATIVE ;
     FROM MROW(),MCOL()
   lnBar = 0

   FOR lnCnt = 1 TO .Count
     lcName = PROPER( .GetKey( lnCnt ))
     lnBar = lnBar + 1
     DEFINE BAR (lnBar) OF shortcut PROMPT (lcName)
     ON SELECTION BAR (lnBar) OF shortcut ;
       pcSelFile = PROMPT()
     IF lnCnt < .Count
       *** Добавить разделяющую линию ко всем элементам,
       *** кроме последнего
       lnBar = lnBar + 1
       DEFINE BAR (lnBar) OF shortcut PROMPT "\-"
     ENDIF
   NEXT

   ACTIVATE POPUP shortcut
   *** Сохранить выбор в свойстве.
   *** (Замечание: для имен ключей
   *** используется верхний регистр)
   .cSelFile = UPPER( ALLTRIM( pcSelFile ))
 ENDWITH
```

Энди: Метод GetKey() действительно здесь полезен: использовать его намного легче, чем пытаться манипулировать массивом. Конечно, мы всего лишь получаем имя файла по ключу... Ты где-нибудь использовала объект файла на практике?

Марсиа: О да. В методах Run и Modify. Причиной создания объекта файла служит то, что в этом случае мы можем получить доступ непосредственно к его свойствам. В методах Modify и Run мы получаем имя того файла, с которым мы хотим работать и который определен в свойстве cSelFile, а затем для того, чтобы извлечь сам объект, мы используем метод коллекции Item(), указывая в качестве параметра имя ключа. Впоследствии мы можем обращаться к любому из свойств этого объекта файла:

```
*** Извлечь объект файла, используя имя ключа
 loFile = .Item( lcFile )
```

Энди: Что особенно замечательно, так это то, что ты даже можешь создать объект файла как визуальный класс, если тебе так хочется. Итак, чтобы использовать этот класс для модификации своих файлов программ, мне необходимо проделать следующее:

```
*** Создать коллекцию файлов
 oPrgs = NEWOBJECT( 'colprgs', 'colclass.vcx' )

 *** Добавить все программные файлы
 *** из текущего каталога и подкаталогов
 *** первого уровня
 oPrgs.FileAdd()

 *** Выдать на экран меню с перечнем файлов
 oPrgs.FileList()

 *** Модифицировать выбранный файл
 oPrgs.FileModify()

 *** Исполнить выбранный файл
 oPrgs.FileRun()

 *** Изменить выбранный файл
 oPrgs.FileList()
```

Марсиа: Да, такова последовательность необходимых действий в полном объеме, но на самом деле нет нужды обращаться к методу FileList() явно: при первом обращении к любому из двух методов, Modify или Run, тебе будет предложено выбрать рабочий файл.

Энди: Очень ловко. Ты, Марсиа, я полагаю, могла бы, вероятно, проделать все это и без класса Collection, но теперь я убедился в том, что использование этого класса очень здорово все упрощает! Будем надеяться, что и все остальные точно также в этом убедились и воспринимают наличие коллекций как еще одну вескую причину для обновления Visual FoxPro до последней версии.

Энди Крамек (Andy Kramek) — FoxPro-разработчик с большим стажем. Он носит звание FoxPro MVP и является консультантом и совладельцем фирмы Tightline Computers Inc., которая находится в г. Акроне, шт. Огайо. Энди постоянно выступает на различных конференциях, его работы широко публикуются в печати, а в режиме online его можно найти на форумах CompuServe (http://go.compuserve.com/msdevapps) и Virtual FoxPro Users Group (www.vfpug.com). andykr@tightlinecomputers.com.

Марсиа Акинз (Marcia Akins) — опытный разработчик, обладает статусом FoxPro MVP, является консультантом и совладельцем фирмы Tightline Computers Inc., которая находится в г. Акроне, шт. Огайо. «Заслуженный» спикер многих конференций, она часто публикует свои работы и хорошо известна как активный участник форумов CompuServe (http://go.compuserve.com/msdevapps) и Universal Thread (www.universalthread.com) marcia@tightlinecomputers.com

❑❑❑

Лучшая альтернатива комбинированному списку

Эдвард Макдермотт (Edward McDermott)

Не поймите Эдварда Макдермотта превратно. Ему нравятся элементы управления combobox и listbox. У него есть всего лишь две претензии к их манере обращения с табличными данными. Данные в поля этих элементов управления FoxPro загружает в момент создания соответствующих объектов. Это означает, что загрузка данных во все размещенные в форме комбинированные списки combobox ведется одновременно, замедляя процесс создания данной формы. Это также означает, что обновление данных в таблице не приведет к автоматическому обновлению содержимого элементов управления combobox в том случае, если вы загружаете данные списка с помощью SQL-запроса или путем указания соответствующего псевдонима.

Я использую элементы управления combobox для того, чтобы помочь пользователю. Как правило, я включаю в свое решение те средства, которые позволяют пользователю обновить список разрешенных для выбора значений. Не говоря о перечне штатов США (при условии, что Пуэрто Рико никогда не передумает), есть ли еще такой список предлагаемых для выбора вариантов, в который не требуется время от времени внести новое значение? Вот почему такое большое количество списков используют в качестве своего источника данных таблицы.

Вообразите форму с 10 полями, для представления каждого из которых использован элемент управления combobox. Затем представьте себе процесс открытия такой формы. «За кулисами» происходит следующее: FoxPro заполняет эти 10 комбинированных списков содержимым 10 таблиц, о которых упоминалось выше. Эти комбинированные списки заполняются данными даже в том случае, если форма открывается лишь для того, чтобы отобразить данные на экране в режиме «только-чтение». К этому моменту из базы данных «считывается» не только то значение, которое мы можем видеть в качестве выбранного, но и все допустимые значения, которые могут появиться в поле списка. Если предлагается сделать выбор из пары сотен значений, все эти данные перемещаются из базы данных на локальную машину, «съедая» пропускную способность сервера и сети.

Сформированное однажды, содержимое списков combo/listbox становится независимым от базовой таблицы. Это означает, что обновление данных в базовой таблице никак не скажется на элементе управления combo/listbox. Оператору пришлось бы закрыть и снова открыть форму, чтобы увидеть исправленное содержимое этих списков. (Один из способов обойти эту проблему, во всяком случае, для комбинированного списка, заключается в том, чтобы инициировать исполнение метода Requery данного элемента управления из его события DropDown, дабы список обновлялся данными из базовой таблицы каждый раз, когда пользователь его открывает.) В мире MDI-форм и незамедлительных обновлений такое функциональное поведение определенно представляется старомодным.

Пару лет тому назад я настолько разозлился, что нашел решение указанной проблемы. Мое решение включало создание пары классов, объекта для размещения в форме и объекта для выдачи на экран соответствующего списка и запоминания выбранного оператором значения.

Механика работы элемента управления Combobox

Как элемент управления listbox, так и элемент управления combobox, оба они используются для того, чтобы предоставить оператору возможность выбрать одно из списка имеющихся значений. (В данной статье не рассматривается ситуация, когда пользователь может выбирать несколько значений из поля listbox.) Список listbox используется в том случае, когда разработчику требуется, чтобы несколько возможных вариантов выбора оставались в поле зрения пользователя постоянно. Комбинированный список combobox используется для того, чтобы представить на экране только одно возможное значение и занять при этом минимум площади формы. Во всем остальном оба эти объекта весьма схожи. Собственно говоря, они схожи настолько, что используют один и тот же построитель. Из-за того, что он занимает минимум свободного пространства, комбинированный список combobox чаще используется при реализации поля для ввода данных, поэтому я решил имитировать характеристики именно этого элемента управления.

Объект комбинированного списка combobox состоит из области для отображения текста и кнопки. Определяя значения свойств этого объекта, я могу контролировать следующие его характеристики:

- Источник значений (row source) (тип источника значений (row source type) — это поля таблицы и, вероятно, их могло бы быть несколько).
- Автоматическую сортировку элементов.
- Источник данных, к которому привязан объект (control source).

Итак, мой объект-«заместитель» будет иметь текстовую область для представления на экране выбранного значения для оператора и кнопку для открытия списка, из которого будет выбираться другое значение, хранящееся в базовом источнике данных.

Впрочем, в действительности, многим элементам управления потребуются две области для представления текста: одна область для выдачи на экран подробного описания, а другая — для представления на экране кодового обозначения (например, «Georgia» и «GA»). В случае с элементом управления combobox вы просто выдали бы на экран какое-то одно из этих значений и показали на экране оба значения одновременно в списке в момент его открытия. Мне необходимо было добиться, чтобы эти два текстовых поля, ассоциированные с моим объектом, были универсальными. (Почему я остановился на двух полях? Я так и не нашел причину, оправдывающую существование третьего текстового поля.)

Теперь, когда у меня есть три вышеперечисленных объекта, необходимо решить, где следует разместить все свойства, ассоциированные с этими объектами? Испробовав три различных подхода, я решил, что большая часть свойств и методов должна содержаться в контейнере. Тем не менее, некоторые функциональные возможности все-таки должны принадлежать отдельным компонентам.

Когда оператор щелкает мышью по кнопке, метод должен открыть окно общего назначения и передать этому окну всю ту информацию, которая необходима для формирования отображаемого в нем списка допустимых значений. Следовательно, все эти данные должны быть определены пользователем при создании экземпляра объекта cnt_picklist_codedesc. По этой причине я предусмотрел свойства, перечисленные в нижеследующем списке. (*Замечание:* в качестве общего правила, те свойства и методы, которые добавляются в контейнер или компонент, я снабжаю префиксом, чтобы как-то отличать их. Префиксами служат строки UDP (User-Defined Property — свойство, определенное пользователем) и UDM (User-Defined Method — метод, определенный пользователем):

- udp_code_result (место для возврата выбранного значения из поля кодового обозначения);
- udp_code_source_field;
- udp_codesearchexpr;
- udp_databasename;
- udp_desc_result (место для возврата выбранного значения из поля подробного описания);
- udp_desc_source_field;
- udp_list_fields;
- udp_select_sqlwhere;
- udp_tablename;
- udp_window_to_use.

Событие Click создает экземпляр объекта окна, определенного в свойстве udp_window_to_use, а затем передает этому окну имя базы данных, имя таблицы, where-предложение, поля, предназначенные для отображения в списке, поля, возвращаемые в качестве кодового обозначения, и поля, возвращаемые в качестве подробного описания.

Для работы с этим контейнером я создал объект окна (названный «Standard Popup»), который мог бы служить значением, принятым «по умолчанию» для свойства udp_window_to_use. Этот объект представляет собой форму, в которой размещены четыре объекта:

- Элемент управления listbox;
- Текстовое поле для организации интерактивного поиска;
- Кнопка OK;
- Кнопка Cancel.

Форма «Standard Popup» на основе полученных ею данных формирует предложение SQL-SELECT и исполняет его, чтобы сформировать список listbox. Естественно, предложение SQL-SELECT сортирует результаты выборки по полям, указанным в свойстве udp_list_fields. Оператор затем может выбрать некоторый элемент из списка listbox и воспользоваться кнопкой OK, чтобы зафиксировать сделанный им выбор. К этому моменту данные из окна Standard Popup возвращаются обратно его родительскому объекту и там запоминаются, а всплывающее окно закрывается.

Хотя предложенный алгоритм представляется простым, может возникнуть путаница относительно правильного использования полей. Например, в свойстве udp_desc_source_field хранится имя того поля, чье содержимое должно быть возвращено из окна Standard Popup и сохранено в свойстве udp_desc_result. Есть два объекта, каждый со своими свойствами, в которых хранится либо имя поля, либо содержимое полей. Например, у контейнера есть свойство udp_list_fields, где хранится имя поля, содержимое которого выдается на экран в списке во

всплывающем окне. Это значение пересылается во всплывающее окно и запоминается в свойстве udp_field_to_view.

В качестве более сложного примера рассмотрим подробное описание, которое выдает на экран контейнер. Соответствующее поле определено в контейнере в свойстве udp_code_source_field. Это значение пересылается во всплывающее окно, где оно запоминается в свойстве udp_desc_field_to_return. (Не забудьте, оба эти свойства содержат имя того поля, где хранится описание, а не само описание.) Как только оператор сделал свой выбор, форма Standard_popup запоминает соответствующее подробное описание в свойстве udp_result_desc. В свою очередь контейнер запоминает это значение в свойстве udp_desc_result и, наконец, в свойстве this.txtdesc.Value. Мне иногда кажется, что я пытаюсь угадать, под каким наперстком спрятана горошина.

Когда оператор щелкает мышью по кнопке из контейнера cnt_picklist_codedesc, исполняется следующий программный код:

```
oWindow = CREATEOBJECT("Standard_popup")
oWindow.udm_Setup_window()
= oWindow.Show()
```

Событие Click кнопки контейнера cnt_picklist_codedesc создает форму, исполняет методы этой формы, а затем выдает эту форму на экран. Поскольку новая форма является модальной, едва появившись, она берет на себя управление экраном. (Эта форма получает управление сразу же после исполнения предложения oWindowShow().)

Затем пользователь делает свой выбор во всплывающем окне и нажимает кнопку OK. В этот момент всплывающее окно должно сохранить выбор пользователя и «спрятаться». (Всплывающее окно не занимается «самоуничтожением». Уничтожение окна выполняет контейнер после того, как «выкачает» из формы полученные результаты.) Эти действия возвращают управление первичной форме, а процесс развивается дальше и наступает событие Click всплывающего окна. Тут мы исполняем следующий программный код:

```
IF oWindow.udp_result_action = «OK»
     this.parent.udm_set_code_result( ;
     oWindow.udp_result_code, oWindow.udp_result_recno)
  this.Parent.udm_set_desc_result(oWindow.udp_result_desc)
ENDIF
= oWindow.release()
```

Вы можете видеть как этот программный код «выталкивает» данные из всплывающего окна, прежде чем удалить его из памяти. Запись в свойство udp_result_action осуществляется в том случае, если оператор воспользовался кнопками OK или Cancel.

Взаимодействие между окнами — это пример простого взаимодействия между MDI-окнами в FoxPro-приложении. Поскольку процесс остается неизменным при открытии второго окна, сначала откройте это окно, а затем закройте его, никаких более сложных функциональных возможностей не требуется.

Дополнения к базовому проекту

Разумеется, эта несложная версия продержалась не более пяти минут в общении с пользователями, прежде чем они возжаждали большего. Первой из дополнительных функциональных возможностей стала возможность интерактивного размещения списка listbox. Текстовое поле textbox, размещенное над списком listbox, позволяет оператору вводить необходимое значение. При каждом нажатии на клавишу в списке listbox ищется первое значение, которое начинается с введенной строки.

Следующее дополнение заключалось в предоставлении возможности иметь в выпадающем списке несколько столбцов. Чтобы такое было возможно применительно к форме общего назначения, этой форме следовало динамически рассчитывать ширину каждого столбца в списке listbox.

После этого поступил запрос на реализацию возможности иметь составное подробное описание, то есть такое описание, значение которого формируется из содержимого нескольких полей.

Наконец, был добавлен программный код для автоматического заполнения поля подробного описания во время инициализации в том случае, если было указано поле с кодовым обозначением. Чтобы обеспечить такое функционирование, я добавил еще одно свойство, udp_codesearchexpr, для хранения SQL-предложения, которое возвращало бы соответствующее подробное описание. Позже я добавил программный код для инициализации свойства udp_codesearchexpr в том случае, если это свойство оказывалось пустым. Задумайтесь об этом на секунду. Поскольку всю информацию для извлечения подробного описания передает контейнер, ему следовало бы обладать всей информацией, необходимой для формирования SQL-предложения, извлекающего это подробное описание.

Разрешить оператору вводить кодовое обозначение с клавиатуры

Некоторые пользователи любят вводить кодовые обозначения с клавиатуры. Некоторые из них обладают способностью помнить тысячи кодовых обозначений. В результате, контейнер cnt_picklist_codedesc был модифицирован с тем, чтобы позволить операто-

ру вводить кодовое обозначение с клавиатуры. Для этого потребовались еще два свойства:

- udp_allow_code_entry (позволить оператору вводить кодовое обозначение с клавиатуры).
- udp_in_error (флажок недопустимости кодового обозначения).

Метод LostFocus поля txtcode запрограммирован для выполнения проверки допустимости вводимого значения и там, где это необходимо, выставляет свойство-флажок udp_in_error, прежде чем обратиться к событию командной кнопки Click. Если оператор не ввел никакого допустимого значения с клавиатуры, открывается окно Standard Popup. Впрочем, в данной ситуации кнопка Cancel может быть блокирована (disabled). Вот почему я добавил свойство udp_in_error.

До последнего времени я упорно противился принятию концепции, согласно которой оператору разрешается вводить с клавиатуры подробное описание для проверки его допустимости. Теоретически, однако, это можно было бы сделать, по крайней мере, для тех ситуаций, когда подробное описание берется из некоторого поля базовой таблицы.

Доступ к объекту cnt_picklist_codedesc

Один из побочных эффектов упомянутого дополнения был связан с разрешением доступа и блокировкой полей. Пользователи хотели видеть все три поля заблокированными в момент их появления на экране в том случае, если контейнер располагался на экране, который был заблокирован для редактирования. С другой стороны, я не хотел, чтобы пользователь мог получить доступ к полю Txtdesc даже тогда, когда объект был разблокирован.

Решение сводилось к использованию двух различных методик: для контроля за процессом отображения полей на экране применялась одна методика, а для управления доступом к этим полям — другая.

Для управления внешним видом полей я просто добавил следующий программный код к методу refresh контейнера. Этот программный код обеспечил гарантию того, что все объекты в контейнере разделяют состояние блокировки данного контейнера.

```
this.txtdesc.enabled = this.enabled
this.txtcode.enabled = this.enabled
this.cmdLookup.enabled = this.enabled
```

Для управления доступом к полям txtcode и txtdesc я добавил программный код к событию When этих объектов, который автоматически пропускает вышеуказанные поля. Когда оператор по щелчку мышью или с помощью клавиши табуляции перемещается в поле txtdesc, фокус автоматически перемещался бы к соответствующей командной кнопке. (Программный код из метода When разблокирует или заблокирует поле кодового обозначения в зависимости от значения свойства udp_allow_code_entry.)

Пользовательская настройка всплывающего окна

Естественно, последующие запросы касались контейнера Standard Popup. Поскольку в некоторых ситуациях требовалось, чтобы характеристики всплывающего окна были определены пользователем, я добавил свойство для хранения имени формы — udp_window_to_use. Значением, принимаемым по умолчанию для этого свойства, оставалось Standard_popup. Однако, это значение могло быть перезаписано.

Чтобы создать пользовательское всплывающее окно, я должен был создать еще один экземпляр окна Standard Popup и соответственно его настроить. Для эффективного выполнения этой работы мне пришлось несколько модифицировать стандартное окно. Я добавил свойство udp_auto_retrieve_list. Когда это свойство имело значение .F. (по умолчанию его значением было .T.), окно не формировало и не исполняло автоматически предложение SQL-select для заполнения списка listbox. Это обеспечивает мне возможность задать пользователю дополнительные вопросы и сформировать более избирательное SQL-предложение. По сути дела, я объединил список listbox с механизмом поиска.

Некоторые несущественные вариации

Иногда пользователям необходимо было видеть только одно поле textbox. Это поле textbox могло бы быть подробным описанием, или оно могло бы быть кодовым значением. (Не спрашивайте меня, зачем это нужно. Я не всегда понимаю пользователей. Иногда, легче просто дать им то, чего они хотят.)

Дабы обеспечить существование такой альтернативы, я разработал объекты cnt_picklist_desc и cnt_picklist_code. Оба эти объекта являются подклассами исходного класса. Чтобы заставить эти новые подклассы работать, я должен был кое-где модифицировать исходный код.

Для реализации варианта «только кодовое обозначение» мне пришлось модифицировать исходный программный код таким образом, чтобы он работал в отсутствие подробного описания. Я должен был «подавить» тест на допустимость значения свойства

udp_desc_source_field. К счастью, работала следующая проверка:

```
IF this.Parent.Class <> "Cnt_picklist_code"
ENDIF
```

Это решение позволило иметь пустое свойство udp_desc_source_field, что привело к возникновению проблем с формой Standard Popup. Чтобы избавиться от них, я добавил следующий фрагмент программного кода в метод udm_setup_window:

```
IF VARTYPE(tcdesc_source_field) = "C"
    this.udp_desc_field_to_return  = tcdesc_source_field
ELSE
    this.udp_desc_field_to_return  = ""
ENDIF
```

Если форма Standard Popup открывалась из объекта cnt_picklist_code, она передавала бы свойство udp_desc_source_field с логическим значением .F.. Предшествующий программный код сохранял бы взамен пустую строку. Остальная часть формы Standard Popup была затем модифицирована таким образом, чтобы выполнялась проверка этого значения перед выполнением любой обработки свойства udp_desc_field_to_return.

Я понимаю, что при таком подходе складывается впечатление огромного объема работ. Первые два раза я просто копировал программный код и модифицировал его. Соответственно я должен был поддерживать три объекта вместо одного. Надеюсь, на этот раз мне не придется снова повторять одни и те же исправления в трех местах.

Если объект привязан к таблице

Создав эти контейнеры, я использовал их в формах. Некоторые программисты связывают размещенные в форме объекты с полями из таблицы базы данных. Другие предпочитают явно загружать значения в поля формы. Я принадлежу к первой группе. Чтобы связать контейнер с табличным полем из базы данных, я определил бы значение свойства ControlSource объекта txtcode контейнера. Тогда любое изменение значения этого объекта отображалось бы в поле соответствующей таблицы из базы данных (после ее обновления).

Хорошо, значение поля txtcode берется из записи таблицы. Измените указатель записи, и значение поля txtcode изменится. Мне пришлось написать обработчик данных для обновления подробного описания. Для этого потребовался следующий программный код в методе Refresh поля txtcode:

```
IF this.Parent.udp_save_txtcode_value <> this.value
     this.Parent.udm_get_desc_for_code()
     this.Parent.udp_save_txtcode_value = this.value
ENDIF
```

Я должен был убедиться в том, что подробное описание меняется всякий раз при обновлении значения поля txtcode. Как только вы связали объект с полем таблицы, любая команда, которая меняет указатель строки, может изменить значение этого поля. Однако изменение указателя строки не послужит сигналом ни к наступлению события ProgrammaticChange, ни к наступлению события InteractiveChange. Мое решение сводилось к обновлению подробного описания с помощью события txtcode.refresh.

Такое решение работало, но мне хотелось минимизировать загрузку системы, поскольку метод udm_get_descr_for_code исполнял предложение SQL-Select. По этой причине я добавил свойство udp_save_txtcode_value и использовал его для управления ситуацией. Если значение поля txtcode не менялось, подробное описание должно было оставаться в надлежащем состоянии.

Заключение

Классы, рассматриваемые в этой статье, обеспечивают разработчику серьезную альтернативу традиционным элементам управления combobox и listbox. Сопутствующие выбору объектов накладные расходы минимизированы, поскольку весь список целиком загружается только по запросу пользователя. Новые объекты могут быть дополнены отсутствующими у традиционных списков listbox и combobox функциональными возможностями, позволяющими учитывать дополнительные пользовательские критерии выбора.

Эдвард Макдермотт (Edward McDermott) проработал в компьютерной индустрии более 20 лет, последние 15 лет на уровне ПК с использованием таких языков программирования, как C++, Delphi, Powerbuilder, MS Access и Visual FoxPro. Сегодня он действует как консультант, чередуя техническую разработку в среде Visual FoxPro с управлением проектами.
edmcdermott@gosympatico.ca.

❑❑❑

Использование схемы «3Dots»

Предраг Боснич (Predrag Bosnic)

WIN 6.0 DOWNLOAD

Очень часто размеры элемента управления не позволяют вместить в него всю ту информацию, которую он содержит или отображает. Одним распространенным решением данной проблемы является использование многоточия (…) или, как его называет большинство, «3Dots», с целью продемонстрировать пользователю, что ему «доступно нечто большее». Каким образом могли бы мы включить «3Dots» в свои собственные интерфейсы? Этот вопрос изучает Предраг Боснич.

Давайте начнем с окна. Вертикальные и горизонтальные полосы прокрутки визуально оповещают нас о том, что данное окно меньше, чем та форма/область, которая в нем отображается, и что мы можем воспользоваться манипуляторами полос прокрутки дабы увидеть скрытую от нас часть формы/области. Такое решение используется почти во всех контейнерных элементах управления. Теперь посмотрите на элемент управления editbox. Если текст, содержащийся в поле редактирования, превышает фактические размеры этого элемента управления, появится полоса прокрутки, которая позволит нам «прокрутить» изображение, чтобы увидеть остальной текст. В Visual FoxPro такое функциональное поведение предусмотрено для многих элементов управления, но сетка grid применительно к ее столбцу не обладает данной возможностью. В то же время, grid — это такой элемент управления, в котором ширины столбца очень часто не хватает для того, чтобы вместить все его содержимое, и это именно то место, где может быть использована схема «3Dots».

Входящий в состав Visual FoxPro объект-контейнер grid — это превосходный элемент управления, и я не думаю, что отсутствие вышеупомянутой функциональности является преградой к его использованию. Однако, такая возможность, как схема «3Dots», может улучшить пользовательский интерфейс и произвести приятное впечатление на пользователя.

Анализ

Решение, которого я добиваюсь, аналогично решениям, используемым в таких приложениях, как MS Explorer или MS ListView. На рис. 1 показан проводник MS Explorer в ситуации, когда размер содержимого столбца больше размера самого столбца, и многоточие предупреждает: «В правой части столбца есть еще текст».

Рис. 1. Проводник MS Explorer демонстрирует схему «3Dots».

Ширины столбца может оказаться недостаточно не только для содержимого столбца, та же самая ситуация применима к его заголовку. Чтобы сузить проблему, оговорюсь, что я заинтересован в таких элементах управления grid, которые отображают только текстовые данные: текст, даты, числа и логические значения. Следовательно, схема «3Dots» будет применяться к текстовым данным. Сверх того, я буду следовать правилу, согласно которому мой элемент управления grid функционирует в режиме «только-для-чтения». На первый взгляд такое ограничение может показаться слишком суровым, но если вы над ним поразмыслите, то окажется, что в действительности это не так. Вообще говоря, я не считаю, что контейнер grid является хорошим решением для пользовательского интерфейса в том случае, если этот элемент управления доступен как для чтения, так и для ввода данных. Мне, конечно, известны пара хороших решений, когда доступность элемента управления grid не ограничивается только чтением данных, но, как правило, я предпочитаю упрощать себе жизнь и использовать объекты grid в режиме «только-для-чтения». Текст, числа, даты и логические значения — это наиболее часто используемые в элементе управления grid типы данных.

Точно так же ведет себя заголовок окна. Если размеры окна меняются таким образом, что длина заголовка превышает размеры окна, Windows помещает в конец заголовка многоточие (см. рис. 2).

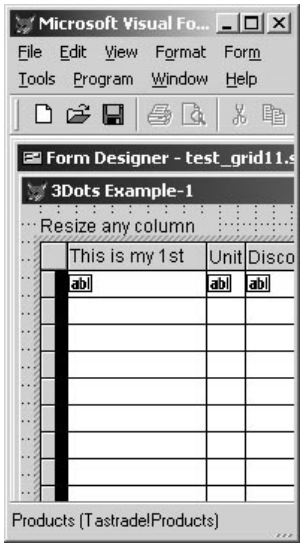

Рис. 2. Основное окно VFP с 3Dots-эффектом.

Решение

Прежде всего, одна маленькая, но полезная функция позволит мне конвертировать любой тип данных в строку. Эта функция называется xToStr(). Я не привожу здесь ее программный код, но вы можете найти его на сопровождающей дискете. Эта функция используется внутри класса для преобразования любых значений в строку.

Вторая необходимая мне функция — это способ «обрезать» строку, используя строковое значение и нужную длину; оба эти значения передаются как параметры. Это несколько более сложная ситуация, поскольку длина строки зависит от выбранного шрифта, по этой причине данная функция получает в качестве параметра также имя, размер и стиль шрифта. Функция называется xtDots() и отвечает за добавление многоточия в конец строки:

```
LPARAMETERS tcString, tnWidth, tcFont, tnSize, ;
   tcStyle

 * tcString – укорачиваемая строка
 * tnWidth – запрошенная длина строки
 * tcFont – имя шрифта строки (по умолчанию = 'Arial')
 * tnSize – размер шрифта строки (по умолчанию = 9)
 * tcStyle – стиль шрифта строки (по умолчанию = 'N')
 *
 * Возвращается: строка

 *--- шрифт, используемый по умолчанию ---------
 IF Empty(tcFont)
   tcFont = 'Arial'
 ENDIF

 IF Empty(tnSize)
   tnSize = 9
 ENDIF

 IF Empty(tcStyle)
   tcStyle = 'N'
 ENDIF

 *--------------------------
 LOCAL jnMax as Integer, jcDots as String,jn1 as ;
   Integer, jcTemp as String
 jnHeaderWidth = Txtwidth(tcString, tcFont, ;
   tnSize, tcStyle)*Fontmetric(6, tcFont, ;
   tnSize, tcStyle)

 IF jnHeaderWidth > tnWidth
   nMax = Len(tcString)
   jcDots = '...'

   FOR jn1 = 1 TO jnMax
     jcTmp = Left(tcString,jn1) + jcDots
     IF Txtwidth(jcTmp, tcFont, tnSize, tcStyle) ;
       *Fontmetric(6, tcFont, tnSize, tcStyle) ;
       > (tnWidth-10)
       RETURN jcTmp
     ENDIF
   NEXT
 ELSE
   RETURN tcString

 ENDIF
 RETURN jcTmp
```

По существу, обязательными параметрами являются строка и требуемая длина, поскольку для имени, размера и стиля шрифта функция будет использовать значения, принятые для этих данных по умолчанию. В ходе вычислений используются встроенные функции Visual FoxPro, такие как TxtWidth и Fontmetric. Как работает рассматриваемая функция? Очень просто, функция формирует строку, добавляя в нее символ за символом, взятые из той строки, которая была получена в качестве параметра. Когда длина формируемой строки сравняется с требуемым значением (уменьшенным на 10 пикселей для подстановки самого многоточия), функция вернет значение новой строки.

Располагая этими двумя функциями, пора приступить к работе с элементом управления grid. Мне хотелось бы сохранить рабочую процедуру в том же виде, в котором она пребывала до использования схемы «3Dots»: поместить элемент управления grid в форму (контейнер), указать источник данных для grid, а затем настроить все необходимые столбцы. В конечном итоге мне хотелось бы получить требуемую функциональность схемы «3Dots».

Первая трудность, которую я вижу, касается свойства ControlSource. Это свойство будет задаваться программистами, но, с другой стороны, я должен задать для этого свойства вычисленное мной значение. Как мне кажется, единственный способ разрешить проблему — это где-то запомнить исходное значение. В данном случае я использую для этого свойство Comment. Метод Init моего элемента управления grid имеет следующий программный код:

```
DoDefault()
LOCAL oCol as Object
```

```
FOR EACH oCol IN this.columns
    oCol.Comment = oCol.ControlSource
    oCol.header1.Comment = oCol.header1.Caption
    oCol.Resize()
NEXT
```

Этот программный код исполняется для всех столбцов элемента управления grid, и он запоминает значение свойства ControlSource в свойстве Comment. Точно так же обрабатываются заголовки столбцов. В конечном итоге вызывается метод Resize каждого столбца. В данном случае метод Resize является самым важным, поскольку он вызывает метод для настройки соответствующих свойств ControlSource и HeaderCaption.

Событие Resize каждого столбца должно содержать следующий код:

```
this.ControlSource = this.Parent. ;
   MakeSource(this,this.comment)
```

Объем программного кода, заключенного в этом методе, невелик, но что существенно — этот программный код вызывает метод MakeSource, который вычисляет значение свойства ControlSource. Метод MakeSource() имеет следующий программный код:

```
LPARAMETERS toColumn, tcFieldName

 * toColumn – ссылка на объект столбца
 * tcFieldName – имя поля (значение свойства
 *               ColumnSource)

 LOCAL jcSource as String, jcStyle as String
 jcSource = ''
 jcStyle  = ''

 *--------------------------------
 IF toColumn.FontBold = .t.
   jcStyle = jcStyle + 'B'
 ENDIF

 IF toColumn.FontItalic = .t.
   jcStyle = jcStyle + 'I'
 ENDIF

 IF Empty(jcStyle)
   jcStyle = 'N'
 ENDIF

 DO case
   CASE Type('&tcFieldName') = 'M' OR Type(;
     '&tcFieldName') = 'G'
     jcSource = tcFieldName

   OTHERWISE
     jcSource = "xtdots(xToStr(" + tcFieldName + ;
       ",.f.,.f.,.t.)," + ;
       Alltrim(Str(toColumn.Width)) + ",'" +;
       toColumn.FontName + "'," +;
       Alltrim(Str(toColumn.FontSize)) + ",'" +;
       jcStyle + "')"
 ENDCASE

 jcStyle = ''

 IF toColumn.Header1.FontBold = .t.
   jcStyle = jcStyle + 'B'
 ENDIF

 IF toColumn.Header1.FontItalic = .t.
   jcStyle = jcStyle + 'I'
 ENDIF
```

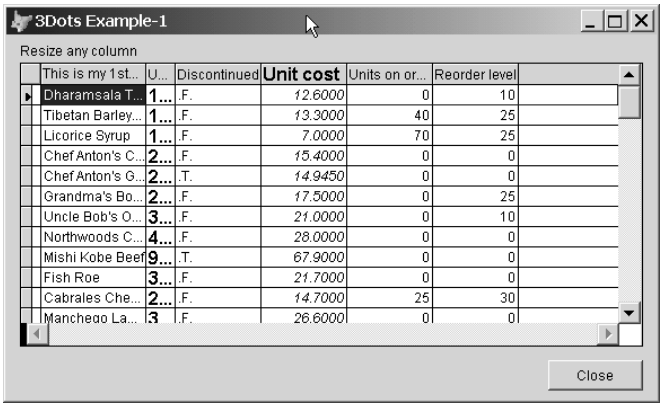

Рис. 3. Схема «3Dots» в действии — несколько заголовков и содержимое в первых двух столбцах были «обрезаны».

```
 IF Empty(jcStyle)
   jcStyle = 'N'
 ENDIF

 toColumn.Header1.Caption = Xtdots(toColumn.;
   Header1.comment, toColumn.width, ;
   toColumn.Header1.FontName,;
 toColumn.Header1.FontSize, jcStyle)

 RETURN jcSource
```

Этот метод получает два параметра — ссылку на столбец и имя поля, к которому привязан данный столбец. Имея эти данные, я могу сформировать новую строку, которая зависит от переданного поля, но которую при необходимости я могу обрезать и добавить к ней многоточие. Полученная строка становится новым значением для свойства ControlSource.

Чтобы протестировать предложенное решение, я создам форму с размещенными в ней элементом управления grid и кнопкой Close. Элемент управления grid связан с тестовой базой данных Tastrade, входящей в состав VFP, и таблицей Products.dbf из этой базы данных. Для демонстрации своего решения я использую первые шесть столбцов. Каждый столбец элемента управления grid имеет свойство ControlSource, указывающее на соответствующую колонку в таблице Products (см. рис. 3).

Для столбца номер два предусмотрено использование полужирного шрифта, для столбца номер четыре используется курсив и так далее. Разумеется, это сделано лишь в целях демонстрации, но тем самым доказывается, что функциональное поведение элемента управления grid/column соответствует ожидаемому (см. рис. 4).

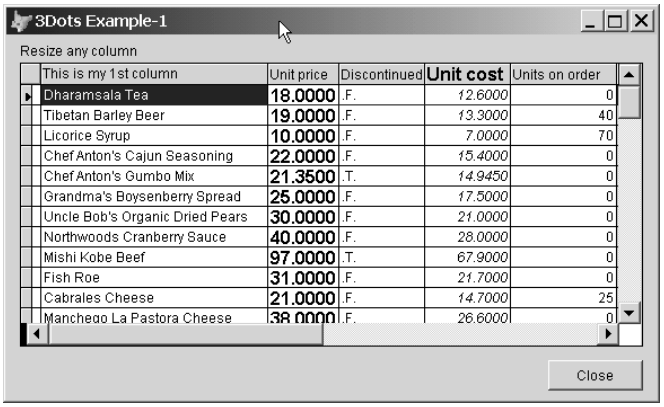

Рис. 4. Изменение размеров первых двух столбцов означает, что многоточие исчезнет.

Дополнения

Вероятно, наиболее важное направление усовершенствования предлагаемого решения — это попытка избежать внесения программного кода в событие Resize всех столбцов элемента управления grid. Кроме того, можно было бы написать универсальный программный код для проверки типа столбца и определения значения свойства Alignment для всех столбцов.

Заключение

Как минимум, данная статья доказывает тот факт, что «мелочи» могут сделать приложение более совершенным, обеспечивая простые, но интересные функциональные возможности пользовательского интерфейса. В очередной раз Visual FoxPro демонстрирует нам, что скорость обработки строк очень высока и можно воспользоваться этой скоростью там, где обычно мы говорим: «Нет! Это будет делаться очень медленно!»

Предраг Боснич (Predrag Bosnic) начал свое путешествие в области ИТ в 1979, используя компьютер UNIVAC 1100, язык программирования Fortran и Mapper. В течение последующих 20 лет он побывал в мире ПК, dBase, Clipper и Fox. На днях он обнаружил себя в Лондоне, работающим в фирме с названием Westwood Forster Ltd, где он является старшим разработчиком (senior developer) и вытворяет разные необыкновенные вещи с помощью Visual FoxPro и SQL Server. mbosnic@westwood-forster.co.uk.

❑❑❑

FoxTalk *русское издание*

Печатается ежемесячно

Учредитель и издатель:

ООО Эдэль.

Страничка в Интернете: http://newsletter.narod.ru или http://msnhomepages.talkcity.redmondave/dartemov/foxtalk.htm

(095) 325-5278
E-mail: foxtalk@online.ru
115304 Москва, а/я 208

Главный редактор: Д. Артемов
E-mail: dartemov@hotmail.com

Журнал зарегистрирован комитетом Российской Федерации по печати.

Регистрационное свидетельство № 015520 от 17.12.1996

FoxTalk (русское издание) индекс 72495

Объединенный каталог индекс 45007

Журнал для FoxPro-программистов.

FoxTalk (русское издание) индекс 72496

Журнал для FoxPro-программистов вместе с дискетой с исходными текстами программ.

FoxTalk (русское издание) индекс 72497

Подписка на старые номера журнала FoxTalk.

Библиотека программиста индексы 72769, 72490, 72491, 47771, 80375, 82841

Книги компьютерной тематики по последним версиям популярных программных продуктов.

Подписка в любом почтовом отделении связи по каталогу «Газеты. Журналы» Агентства Роспечать и «Объединенному каталогу».

Подписано в печать 25/12/03. Формат 60x90 1/8. Тираж 280 экз.