11•2002
http://radio-mir.com
Индексы: 48925, 45995

радиомир

В НОВЫЙ ВЕК —
С НОВЫМ НАЗВАНИЕМ!

ПОЗНАВАТЕЛЬНАЯ

Ваш компьютер

ДУХОВНЫЙ МИР ИНДИИ

профессиональному разработчику под Windows

СОФТ

От идеи до дистрибутива

- Разработка программного обеспечения
- Разработка справочных систем
- Системы создания установочных пакетов
- Утилиты для программистов
- Полезная информация

ALEASOFT

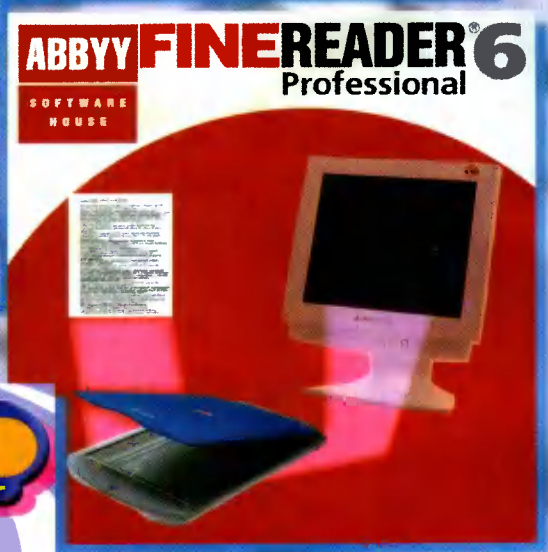
5600 ФОТО ОБЪЕКТОВ

БЕЛАРУСЬ 2002

ГИГАНТЫ МАШИННОГО ПЕРЕВОДА v.2.

ABBYY Lingvo 7.0 · Magic Gooddy 200
PROMT Internet 2000 · Сократ Интернет
Сократ Персональный 4.1 · Сборник тем
словарей для "Сократ Интернет 3.0 Базов
«ОРФО2000» · "Хамелеон 6.6" · Ра

ABBYY FINEREADER 6 Professional
SOFTWARE HOUSE



106.2 РАДИО

fm

Europa Plus

128 kbps

500 РУССКИХ УЧЕБНИКОВ

- ПРОГРАМИРОВАНИЕ
- OFFICE 2000
- НАСТРОЙКА
- ПЕРВЫЕ ШАГИ
- WEB-ДИЗАИН
- ИЗДАТЕЛЬСТВО

ORACLE 3cd

9i

Database Enterprise Edition 9.0.1

Новое поколение уникального продукта по созданию и обработке баз данных

Глоссарий современных 3D-терминов

Рис.10

Рис.7

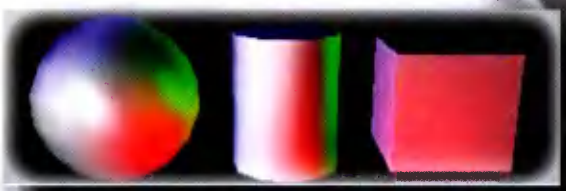


Рис.6

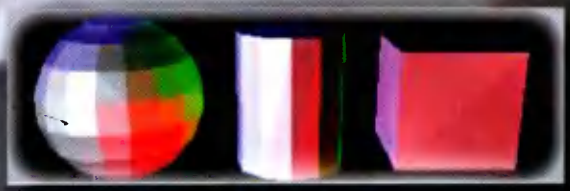


Рис.11

Рис.8

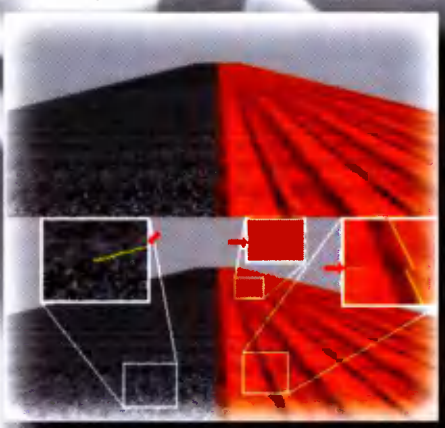


Рис.14

Рис.9

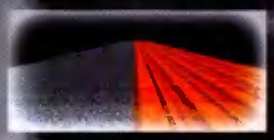


Рис.2

Рис.1

Рис.4

Рис.3

Рис.12

Рис.5

Рис.13

Учредитель и издатель:
ООО "Радиомир Пресс"
Свидетельство о регистрации
ПИ №77-9841 от 17.09.2001

Главный редактор
Ольга СТРЫЖАНКОВА

Зам. гл. редактора
Иван БЕЛЬСКИЙ

Редколлегия:
Сергей ДРОЗДОВСКИЙ,
Алексей КОНДРАШОВ,
Анатолий КОРБИТ,
Владимир КУЦЕНКО,
Елена ЛЕВИТМАН,
Николай МЕДВЕДЬ,
Геннадий ПЕЧЕНЬ,
Геннадий ШУЛЬГИН

Контактные телефоны:
в Москве (095) 105-99-89,
в Минске (017) 249-41-47

Компьютерная верстка —
Янина БЕЛЬСКАЯ

Техническая графика —
Наталья БЕЛЬСКАЯ,
Александр ГОРАЮТИН,
Мария КАЛАБУХОВА

Оформление обложки —
Наталья БЕЛЬСКАЯ,
Надежда БОГОМОЛОВА

Адреса для писем:
119454, РФ, г.Москва, а/я 37,
факс (095) 131-97-17;
220095, РБ, г.Минск-95, а/я 199,
фвкс (017) 221-01-10

E-mail: rl@radiopage.by
rm@radio-mir.com
WWW: http://radio-mir.com

Наши платежные реквизиты:
получатель: ООО "Радиомир Пресс",
ИНН 7729404741,
р/с 40702810900010000084
в ООО КБ "Агропромкредит"
ф-л Центральный, г.Москва,
корр. счет 30101810500000000109
БИК 044525109
Адрес банка: 113054, г.Москва,
ул.Зацепа, 43Б

Материалы для публикации принимаются в рукописном, печатном и электронном вариантах

Требования к графическим материалам рекламного характера в электронном виде: CorelDRAW до 9.0, все шрифты в кривых; bitmaps 300 dpi; TIFF 300 dpi; CMYK. Приложить печатную копию.

За достоверность рекламной и другой публикуемой информации несут ответственность рекламодатели и авторы. Мнение редакции не всегда совпадает с мнениями авторов

Адрес редакции: 119454, РФ, г.Москва, ул.Коштоянца, 6 — 233, тел. (095) 105-99-89

Подписано к печати 24.09.2002 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ № 2480.

радиомир
Ваш компьютер

ЕЖЕМЕСЯЧНЫЙ МАССОВЫЙ ЖУРНАЛ
С 1995 г. по 6/2001 г. издавался под названием "Радиолюбитель. Ваш компьютер"

№11/ноябрь/2002

ЧИТАЙТЕ В НОМЕРЕ

Полные листинги некоторых программ расположены по адресу
http://radio-mir.com

Н.ЛУТКОВСКАЯ,
E-mail: SRLSA@bsu.by

КВАНТОВЫЙ КОМПЬЮТЕР: от идеи к воплощению

Принципиальная возможность создания квантового компьютера [1, 2] привела к возникновению новых идей по его применению. Разрабатываются новые принципы работы, новые алгоритмы, новый вид передачи информации и другие возможные применения. Однако реализовать все это будет не так-то просто, поскольку квантовая физика имеет дело с миром отдельных атомов, молекул и прочих частиц, да еще вкупе с необычными особенностями квантового мира. Поэтому ученые "засучили рукава" и взялись за работу. Эти работы ведутся в нескольких направлениях. Сомнений в том, что эра квантовых компьютеров уже не за горами, остается все меньше и меньше.

Поскольку развитие электроники идет в сторону постоянного уменьшения размеров элементов, квантовые технологии, позволяющие свести габариты не только отдельных элементов или интегральных схем, но даже целых компьютеров до размеров отдельных атомов и молекул, являются ведущим направлением в компьютерной и электронной индустрии. Под эту область науки выделяется большое количество денег, например, такими крупными известными фирмами как IBM, Hewlett Packard и др. Большой интерес компьютерных фирм вызван тем, что квантовомеханические эффекты могут сильно изменить саму природу вычислений.

Проблемы создания квантового компьютера в настоящий период времени — это прежде всего физические проблемы. Основная из них — быстрый распад суперпозиционных состояний. Этот процесс называется декогеренцией. Она накладывает основное требование на физические элементы, предполагаемые к использованию в квантовых компьютерах — время сохранения когерентности состояний должно быть больше времени вычисления. Отсюда следуют два способа избежать распада когерентности — найти квантовую систему, максимально изолированную от внешней среды, или увеличивать время сохранения когерентности. В таблице указаны основные способы локализации одиночных квантовых систем. Как следует из **таблицы**, пути поиска хорошо изолированных квантовых систем можно суммировать.

Поле	Вещество
Микрорезонаторы: оптические; микроволновые	**Пучки**
	Ионные ловушки
	Лазерные ловушки
Резонаторы мод "шепчущих галерей"	**Естественно изолированные системы:** Молекулы в аморфном и поликристаллическом окружении; Примеси в кристаллах; Молекулы в биоструктурах (метод исследования — спектроскопия отдельных молекул)
Фотонные кристаллы	**Квантовые точки**
	Ядерные спины молекул

Изоляция полевых квантовых систем — мод поля — возможна в ***высокодобротных резонаторах*** оптического и микроволнового диапазонов, в которых достигаются параметры, обеспечивающие время сохранения оптических состояний от секунд до микросекунд при изменении числа фотонов от единиц до 100 для микроволнового диапазона.

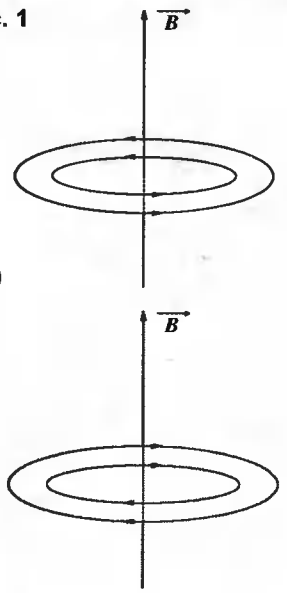
Другим перспективным методом полевой изоляции является использование поверхностных мод типа ***"мод шепчущих галерей"*** на поверхности микросфер из синтетического кремния, где возможно достижение добротности порядка $10^9...10^{10}$. Принципиально новым методом является использование трехмерных периодических диэлектрических структур — фотонных кристаллов, идеально "запирающих" фотоны определенной полосы частот внутри кристалла. Наиболее перспективным материалом для такой структуры является синтетический опал [3].

Изоляция отдельных массивных частиц (атомов, молекул, ионов) исторически начиналась с одномерной изоляции в пучках. Однако созданные природой объекты — молекулы — могут рассматриваться как отдельный, уже существующий элементарный квантовый компьютер. Поэтому первый большой прорыв ученых, разрабатывающих реальный квантовый компьютер, произошел в начале 90-х, когда они придумали, как проводить измерения, используя ***ядерный магнитный резонанс*** (ЯМР).

Основная идея ЯМР заключается в следующем. Отдельная молекула может себя вести как маленький компьютер. Информация хранится в ориентации ядерных спинов в молекуле, причем каждое ядро образует один кубит (напомню, что кубит, квантовый бит — это единица квантовой информации). Взаимодействие между ядерными спинами, известное как спиновое объединение, служит в качестве промежуточных логических операций. В сильном магнитном поле эти ядра вращаются вокруг линий магнитного поля с частотой, зависящей от их химической природы. Например, в поле 9,3 Тл ядро углерода-13 в молекуле хлороформа ($CHCl_3$) вращается с частотой 100 кГц. Действуя на молекулу радиоволнами с частотой, совпадающей с резонансной частотой ядра, можно управлять каждым отдельным ядром и, таким образом, заставить его выполнять логические операции. Представить это нам помогут понятия, используемые в квантовой физике [3]. На **рис.1** показано представление спина: *а)* — "вверх", *б)* — "вниз".

Как видно из рис.1, кубит имеет два базовых состояния — $|0\rangle$ и $|1\rangle$, которым могут соответствовать, например, направления "вверх" и "вниз" спина атомного ядра. Спин аналогичен направлению оси вращения прецессирующе-

Рис. 1



го волчка. Ось вращения может указывать вверх или вниз и, таким образом, соответствовать спину "вверх" или спину "вниз" — состояниям $|0\rangle$ и $|1\rangle$. Таким образом, воздействие на ядро и, соответственно, на ядерный спин может представлять собой обращение $|1\rangle$ в $|0\rangle$ или наоборот, т.е. так называемую однокубитовую операцию, или двухкубитовую — над двумя связанными ядрами, в которой состояние одного ядра зависит от состояния другого.

В работе Исаака Л.Чуанга (Isaac L.Chuang) [4] сообщается о демонстрации на ядрах трихлорметана (хлороформа) первого квантового алгоритма, позволяющего выполнить за одно действие процедуру, аналогичную идентификации за одну попытку изображения на каждой стороне монеты: орел, решка или две стороны одинаковы. В игре в фанты в руке у партнера находится монета. Надо определить, имеет ли данная монета две стороны разные (орел и решка), или они одинаковые. Очевидно, сделав два взгляда на разные стороны монеты, можно дать ответ. Но можно ли дать правильный ответ, взглянув только один раз? Авторы [4] смоделировали эту ситуацию на спиновых состояниях ядер молекулы хлороформа и дали ответ за один "прогон" квантового процессора. При этом спиновые состояния ядер водорода использовались в качестве аналога индикатора, на какую сторону монеты сделан взгляд (спин "вверх" или "вниз"), а состояние спина ядра углерода — для индикации результата наблюдения.

Хлороформ с изотопом углерод-13 является хорошим примером молекул, которые могут выступать в роли двухкубитовых квантовых компьютеров, потому что водород и ядро углерода-13 могут быть индивидуально адресованы с помощью радиоволн. Квантовые вычисления тогда сводятся к кодирующей программе — последовательности одно- и двухкубитовых операций в виде серии радиочастотных импульсов. Результаты затем считываются с сигнала магнитной индукции, который в конце вычисления генерируется вращающимся ядром. Этот сигнал является индикатором ядерного спина.

Ядерный магнитный резонанс похож на желаемое решение сложнейшей задачи построения реального квантового компьютера. Ведь ядра естественным образом изолированы от внешних шумов и, таким образом, могут находиться в когерентном состоянии в течение многих секунд, времени, достаточного для сотен логических операций. К тому же, ЯМР — это проверенная технология, которая уже много лет применяется для химического анализа.

Но у ЯМР есть несколько существенных ограничений. Отдельные молекулы не могут выдавать сигнал, достаточно сильный для того, чтобы его можно было наблюдать. Поэтому ЯМР-эксперимент должен подключать огромное количество молекул (порядка 10^{23}), чтобы их результирующий сигнал магнитной индукции мог быть измерен. Надо отметить, что эти молекулы обычно помещаются в растворитель, так что у первых квантовых компьютеров "жидкое сердце".

Чтобы начать вычисление, необходимо знать начальное состояние компьютера. Но при комнатной температуре во всех веществах число разноориентированных спинов молекул уравновешено, а распределены они хаотично. Другими словами, состояние каждого из компьютеров в растворителе не может быть известным, и проведение последующих вычислений будет бессмысленным.

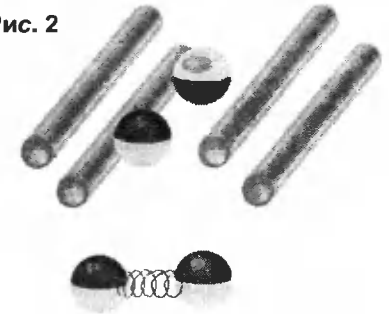
Но никогда не надо сдаваться прежде времени. В 1997 г. две группы ученых спасли судьбу квантового компьютера, независимо друг от друга обнаружив выход из тупика. Эти исследовательские команды возглавляют уже упомянутый выше И.Чуанг, который теперь работает в Алмаденской лаборатории IBM в Калифорнии, и Нейл Гершенфельд (Neil Gershenfeld) из Массачусетского института технологий (МИТ). Они смогли создать небольшой естественный барьер перед ориентированными, по желанию, спином "вверх" или спином "вниз" относительно магнитного поля ядрами некоторых молекул. Этот барьер используется для создания начального состояния (00 для двухкубитовых систем), с которого можно начинать вычисление. В это же самое время, Дэвид Кори (David Cory) из МИТ, а также Эйм Фами (Ame Fahmy) и Тимоти Хэйвел (Timothy Havel) из Гарвардского университета обнаружили, что при бомбардировке образцов радиоимпульсами можно эффективно собирать сигнал всех состояний, кроме начального.

Для проведения компьютером вычислений необходимо, чтобы он мог выполнять какие-либо логические операции. Для квантовых компьютеров существуют две логические операции, аналогичные И и НЕ. Первая представляет собой сдвиг одного кубита. Вторая проводится с двумя кубитами и называется ***контролируемая схема НЕ***, которая транспонирует или не транспонирует один кубит в зависимости от того, в каком состоянии находится второй кубит, парный первому. Обе эти операции являются простыми — надо просто бомбардировать "жидкие" образцы соответствующей последовательностью радиоимпульсов. С 1997 г. уже упомянутые две группы, особенно в университете Лос-Аламоса, построили жидкий ЯМР-компьютер, в основе которого лежит до 7 кубитов, выполняющих простые алгоритмы. Один из алгоритмов даже является одним из упрощенных вариантов алгоритма Шора для взламывания кодов.

К сожалению, квантовые компьютеры, построенные на принципе ЯМР в жидкостях, не будут мощнее созданных а Лос-Аламосе. Считываемый сигнал, который они производят, экспоненциально убывает с увеличением числа атомов, задействованных в вычислении, потому что увеличивается число молекул в начальном состоянии. Поэтому, по прогнозам ученых, в таких компьютерах число кубитов не превысит 10...12, потому что большее число кубитов не позволит отличить сигнал от фонового шума. Попытки построить машины более чем на 10 кубитах продолжаются, но для того чтобы квантовые компьютеры получили повсеместное распространение, надо все же искать какие-то другие принципы и направления работы.

Замороженные ионы. Существует и другая технология, которая меньше афишируется, но уже привела к хорошим результатам. В 1995 г. Игнасио Цирак (Ignacio Cirac) и Петер Цоллер (Peter Zoller) из Инсбрукского университета (Австрия) предложили использовать ионные ловушки, чтобы построить логические схемы. Ионные ловушки уже использовались в спектрографии и для уточнения стандартов времени и частоты. Но для того чтобы этот принцип мог использоваться в квантовых компьютерах, необходимо произвести доработки. Идея состоит в том, чтобы какое-то число охлажденных до низких температур ионов удерживать с помощью прибора, который называется "линейная радиочастотная ловушка Пола" (Paul trap). Это устройство устанавливает высокочастотное поле, которое удерживает ионы: жестко — в двух измерениях, и слабо — в третьем измерении. На **рис.2** показана структурная схема ионной ловушки. Так как ионы одноименно заряжены, то они отталкивают друг друга и стремятся построиться в одну линию на одинаковом расстоянии друг от друга, подобно бусинам на гибкой нити. Такое взаимное расположение позволяет им вести себя как целостная группа, что и необходимо для их использования в квантовых компьютерах.

Рис. 2

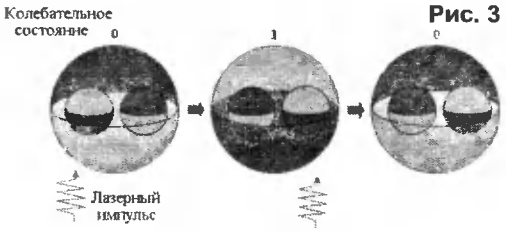


Кубиты первоначально хранятся во внутренних спиновых состояниях относительно внешнего магнитного поля. Они сообщаются ионам с помощью пульсирующего переменного магнитного поля, которое инвертирует биты или создает суперпозицию спина "вверх" и спина "вниз" в зависимости от времени воздействия. Преимущество ионных ловушек состоит в том, что в них суперпозиция очень устойчива и длится столько же, сколько сохраняются кубиты в ЯМР. Так можно получить отрезок времени, достаточный для выполнения логических операций.

Чтобы распределить кубиты между ионами, ученые обратились к ионным колебаниям. Цель заключается в том, чтобы охлаждать ионы до тех пор, пока они как группа не станут неподвижными. Это начальное состояние системы. Если сообщить небольшое количество энергии, ионы начнут колебаться. Но так как ионы — квантовые частицы, то они могут находиться в суперпозиции начального и колеблющегося состояний, и, таким образом, колебания ионов могут служить для хранения кубитов. Так как все ионы участвуют в колебаниях, кубит распределяется между ними. Это коллективное движение позволяет всем ионам одновременно разделять информацию и перепутываться. Такое разделение делает возможными операции типа ЕСЛИ — ТО, чтобы строить блоки компьютерных логических элементов. Вид алгоритма может быть, например, таким: *ЕСЛИ колебательное состояние — 1, ТО перевести кубит в первое внутреннее спиновое состояние.* Исследователи из Национального института науки и технологий (НИНТ) уже показали, что цепочка из четырех ионов может перепутываться, и сказали, что возможно создание перепутывания и большего числа частиц.

По крайней мере пять исследовательских групп со всего мира работают над квантовыми компьютерами с ионными ловушками, но команда Дэвида Уайнлэнда (David Wineland) в НИНТ всеми признается лидирующей. Его группа построила 2-кубитовый логический элемент на основе охлажденного до начального состояния иона бериллия. Используя луч лазера, направленный на ион, группа наложила на внешнее магнитное поле еще одно поле, величина которого изменяется в зависимости от положения иона. Колебания иона порождают переменное магнитное поле, и когда частота его колебаний становится равной разности энергий между двумя спиновыми состояниями, энергия преобразуется из спиновой в колебательную, переводя квантовую информацию из спинового в колебательное состояние. Такова основа контролируемой схемы НЕ (**рис.3**). Она была обнаружена в 1995 г., спустя несколько месяцев после того как сделали заявление Цирак и Цоллер.

Рис. 3



Считывание информации включает в себя рассеивание света ионом, причем можно сделать так, чтобы ион со спином "вверх" рассеивал свет очень сильно, а со спином "вниз" почти не рассеивал.

Ионные ловушки имеют свои ограничения. Одно из них — короткое время декогеренции кубита после перехода его в колеблющееся состояние. Т.к. ионы заряжены, то колебания сильно подвержены влиянию паразитных электрических полей, вызывая декогеренцию. Тем не менее, группа уверена, что это явление можно преодолеть путем лучшей изоляции иона от внешней среды. Ионные ловушки также страдают проблемой считываемости. Чем больше ионов находится в ловушке, тем больше риск того, что создадутся неконтролируемые колеблющиеся состояния, и невозможно будет провести вычисление. Следующим шагом в разработках будет создание регулируемых ловушек, каждая из которых будет содержать всего лишь несколько ионов. Информация от одной ловушки к другой будет передаваться путем физического перемещения ионов или с использованием телепортации. Для телепортации с исходного объекта считывается информация (используют перепутанные состояния квантовых частиц — эффект ЭПР), которая затем передается к месту назначения и используется для построения точной копии объекта, причем не обязательно непосредственно из материала самого оригинала, а можно просто из атомов тех же самых видов, из которых устроен оригинал. Оригинал будет уничтожаться в процессе его сканирования.

Альтернативные решения. Пока жидкий ЯМР находится в дебрях решения проблемы работы при комнатных температурах, несколько групп исследуют проведение операций ЯМР-типа над отдельными ионами в твердом состоянии. Особое внимание привлекла идея Брюса Кейна (Bruce Kane) из Мэрилэндского университета. Его идея заключается в том, чтобы поместить массив атомов фосфора внутрь кремния и покрыть его слоем изолятора, к которому сверху приделан подобный массив электродов, каждый из которых дает возможность приложить напряжение к подсоединенному к нему атому. Однако тут возникает проблема, как контролировать спин каждого ядра. Так же как и в ЯМР, спин ядра может быть изменен на противоположный с помощью радиоволн с нужной энергией. На внешней оболочке атома фосфора находится электрон, который взаимодействует сложным образом с ядерным спином. Приложив напряжение к атому, изменяют энергию, необходимую для "обращения" к ядерному и электронному спинам. Воздействуя радиоволной нужной частоты, меняют спин ядра на противоположный. Таким образом, приложив напряжение к определенному электроду и воздействовав на массив радиоволнами новой частоты, можно управлять отдельным ядром.

Но для контролируемой логической операции НЕ необходимо, чтобы два кубита были перепутаны. Кейн сумел сделать и это. Напряжения между управляемыми атомами в массиве могут включать и выключать взаимодействия между электронами любых атомов, делая возможной двухкубитовую операцию.

Конечно, с теорией все хорошо. Однако трудность заключается в построении такого устройства. Но команда Кейна уже работает над этим.

Команда Роберта Кларка (Robert Clark) из Австралии надеется преодолеть все трудности, с которыми столкнулся Кейн. Особенно сложно построить атомный массив и предотвратить перемещения внутри кремния. Кейн основывает лабораторию, в которой будет изучаться другой сложный аспект его устройства — считывание. После того как закончена одно- или двухкубитовая операция, необходимо считать

информацию с ядерных спинов. Кейн снова полагается на связь между ядерным и электронным спинами. Тщательно измеряя спин электрона, можно в результате получить спин ядра. И хотя измерение спина одного электрона еще никогда не было сделано, Кейн считает, что скоро это будет возможным.

Идея Кейна привлекла большое внимание, из-за того что многие из таких логических элементов могут быть соединены, чтобы образовать квантовый компьютер. И хотя это потребует определенного времени, Кейн считает, что это произойдет в недалеком будущем.

Квантовое явление сверхпроводимости также может пригодиться для построения квантовых компьютеров. В 1999 г. в Дельфтском университете технологий (Голландия) команда ученых сконструировала сверхпроводящую цепь, в которой суперпозиционные, противоположные по направлению токи могут оказаться полезными для хранения и управления кубитами. Цепь состоит из замкнутой цепи с тремя или четырьмя джозефсоновскими соединениями для измерения состояния цепи. Тот факт, что она сделана с помощью обыкновенной литографической техники с обыкновенным электронным лучом, делает ее особенно удобной для массового внедрения. Тем не менее, у сверхпроводящих цепей короткое время декогеренции, и сегодняшние технологии для измерения состояния цепей слишком примитивны для практического применения кубитов.

Более совершенной твердотельной технологией является квантовая точка (quantum dot), особенно полупроводниковая ловушка, которая содержит дискретное число электронов [5]. Она изучается с 90-х годов, потому что удерживаемые в квантовой точке электроны действуют как искусственные атомы со своей собственной периодической таблицей и химией. Затем в 1998 г. Дэвид ДиВинчензо (David DiVincenzo) из IBM и Даниель Лосс (Daniel Loss) из Базельского университета в Швейцарии предложили использовать квантовые точки в качестве основы для квантового компьютера, и с тех пор было выдвинуто множество идей по применению квантовых свойств точек для компьютеризации. Одна из них представляет собой двухкубитовую систему из двух электронов, распределенных между четырьмя квантовыми точками в прямоугольнике. Электроны, стремясь минимизировать свою энергию, занимают противоположные углы прямоугольника, и т.к. такая расстановка имеет две конфигурации, они существуют как суперпозиция, которой можно управлять посредством электродов в углах прямоугольника. Ряд других методов включает в себя записывание и считывание данных из квантовых точек с помощью лазерных импульсов и помещение отдельных ядер в центре каждой точки, адресация к которым может быть произведена с помощью ЯМР-технологий, подобных предложенным Кейном.

Квантовый Интернет. Проблемы в реализации многих из этих идей убедили ученых в том, что если квантовому компьютингу и суждено в ближайшем будущем стать полезным, то это потребует создания сети, которая бы объединила маленькие компьютеры вместе. Но передача квантовой информации из одного места в другое является хитроумным и тонким процессом. Один из вариантов заключается в физической передаче кубитов, но тогда они будут подвержены декогеренции. В 1993 г. Чарльз Беннетт (Charls Bennett) из лаборатории IBM Томаса Дж. Уотсона (Нью-Йорк) пришел к другому решению — телепортации.

Телепортация использует глубокую связь между различными точками пространства, которую устанавливает перепутывание. Беннетт утверждает, что перепутывание может служить в качестве телефонной линии, по которой будет передаваться информация. Другими словами, надо создать пару перепутанных частиц и, удерживая одну из них, вторую посылать получателю. Такой процесс связи позволяет обмениваться квантовой информацией между кубитами.

Беннетт и его коллеги вынуждены были ждать четыре года, чтобы увидеть, как сбываются их предсказания. В 1997 г. в Инсбрукском университете (Австрия) группа физиков, возглавляемая Антоном Цайлингером (Anton Zeilinger), продемонстрировала первый эксперимент по телепортации. Фотоны передавались на расстояние одного метра. Сегодня, спустя более четырех лет, Цайлингер работает над следующим шагом, который заключается в телепортации фотонов на расстояния около километра.

Вскоре после прорыва Цайлингера, Цирак и Цоллер предложили идею о том, чтобы телепортация стала основой квантового Интернета. В марте 2000 г. Сэс Ллойд (Seth Lloyd) и Селим Шариар (Selim Shahriar) в исследовательской лаборатории военно-воздушных сил в Массачусетсе предложили передавать перепутанные фотоны по оптическим волокнам в узлы с охлажденными атомами, которые бы абсорбировали (собирали) фотоны и таким образом сохраняли бы перепутывание. Такое перепутывание можно было бы использовать для исправления ошибок, телепортации и для других целей. Ряд групп в Калифорнии и Лос-Анджелесе работает над этой идеей и надеется, что им удастся запустить сеть на базе трех узлов.

Некоторые ученые ожидают и более грандиозных вещей от перепутывания и считают, что однажды оно станет "током" квантового Интернета. Но для того чтобы явления такого плана стали возможными, необходим значительный прогресс в этой области. Но даже при сегодняшнем уровне развития эти открытия могут сделать самые смелые мечты ученых реальностью. Еще 5 лет назад многие полагали, что квантовые компьютеры появятся не ранее чем через 20 лет, но компьютер на ЯМР уже через год доказал, что они были неправы. Только самые смелые предсказатели осмеливаются строить прогнозы, какой может стать эта область науки через 5 лет [6].

Подтверждением значимости квантовых компьютеров и залогом их будущего развития является тот факт, что в России стал выпускаться первый в мире журнал, посвященный этой области, который называется "Квантовые компьютеры и квантовые вычисления".

Создание квантовых компьютеров предъявляет такие требования к экспериментальной реализации систем с высокой квантовой когерентностью, о выполнении которых пока можно только мечтать. В настоящее время экспериментальные возможности в атомной и квантовой физике, оптике и в других областях науки позволяют осуществить только самые элементарные квантовые вычисления. Но полученные результаты вдохновляют и обнадеживают. Возможно, нереальные сегодня технологии и опыты станут привычными завтра.

Литература

1. Лутковская Н., Лутковский В. История и география квантовых компьютеров. — Радиомир. Ваш компьютер, 2001, №12, С.2.

2. Лутковская Н. Азбука квантовых компьютеров. — Радиомир. Ваш компьютер, 2002, №3, С.2.

3. Килин С.Я. Квантовая информация. — Успехи физических наук, 1999, Т. 169, № 5. С.520.

4. Chuang I. L. — Nature, 1998, № 393, P.143.

5. Лутковская. Н. Транзистор из одного атома? — Радиолюбитель. Ваш компьютер, 2000, №6, С.2.

6. Mullins J. The Topsy Turvy World of Quantum Computing. — IEEE Spectrum, 2001, № 2, P.45.

С миру по нитке, или лекарство от паранойи

Serrgio /HI-TECH,
E-mail:serrgio@gorki.unibel.by
http://www.wasm.ru/

Не так давно я посмотрел художественый фильм под названием "Рыба-меч". Фильм так себе, но американцы, скорее всего, восприняли его "на ура". Помимо всей прочей мути, там был еще и такой эпизод — кулхацкера сажают за комп и просят взломать 128-битный ключ за 60 с. Конечно, это невозможно — самые крутые хакеры ломают его не менее чем за час! ;) Но, как оказалось, любое невозможное становится возможным, если должным образом простимулировать творческую активность. В качестве пряника — девушка, а в качестве кнута — приставленная ко лбу пушка. Без лишних вопросов — пассворд был взломан на 59-й секунде!

Ну, и бог с ним, с этим фильмом — мы-то с вами в сказки уже давно не верим, точно зная, что в реальности все намного сложнее. Вернее, даже не столько сложнее, сколько дольше. Взять хотя бы банальный, якобы случайно забытый пассворд на zip-архив. Хорошо, если он короткий — символов этак шесть еще можно перебрать прямым брутфорсом за разумное время. Или если он содержит осмысленное слово, и число таких слов намного меньше, чем число возможных комбинаций из тех же шести символов, даже если учесть различные варианты написания этих слов (транслит, кириллица на английской раскладке и пр.) — тогда тоже возможно. А вот если мы имеем, например, восьмисимвольный пассворд, составленный по всем правилам, то, как говорится, "пилите, Шура, пилите" — столько не живут.

Так что же это получается? Достаточно использовать длинный "хороший" пассворд с "криптостойким" алгоритмом шифрования, и никакому ЦРУ/ФСБ/КГБ (нужное подчеркнуть, ненужное зачеркнуть) ни в жизнь не прочитать какой-нибудь там "сИкретный план очередной партизанской вылазки"? Запуганные многочисленными публикациями о тотальном контроле за "личной-жизнью-в-интернете", дружно скажем — не верим! Наверняка в мрачных засекреченных подвалах этих служб стоят суперкомпьютеры, которые за несколько минут могут "в лоб" подобрать любой пароль! Ну нам-то, славянам, это глубоко фиолетово, мы, понимаешь ли, привыкли, что если "органы" пришли в гости, то перед ними нужно "упасть-отжаться", ибо попытки "качать права", цитируя соответствующие статьи Конституции, чреваты отбитием почек и пилением зубов напильником. А вот юсы (граждане USA) — это дело другое! От органов они, как от чертей библией, прикрываются Конституцией — прочитают вслух пару волшебных строчек, тех и след простыл. Привыкли, понимаешь ли, наивные, к тому, что их право частной жизни блюдется! Зажрались до того, что такая, казалось бы, мелочь, как "кто-то прочитает мое письмо", возводится в ранг угрозы демократическим принципам как фундаменту государства. Ведь это так страшно и недемократично, когда все, что зашифровано — обязательно смогут расшифровать! Если и не органы, так уж соседский мальчишка-хакер — точно. И сколько бы ни рекламировала какая-нибудь там RSA свой мега-пупер-криптоалгоритм RC5, мы-то, истинные американцы, знаем, что все равно, НАШЕ ПРАВО ЧАСТНОЙ ЖИЗНИ нарушается повсеместно!

Наверное, именно для таких параноиков RSA и решила провести одну интересную акцию. То ли эксперимент, то ли оригинальную рекламную кампанию, то ли просто лотерею ;) — тут уж кому что больше нравится. Суть ее заключается в следующем: "кто ломанет наш пассворд, тому приз 10000$"!

Ну и кого это заинтересует, спрашивается? Сотню-другую хакеров и на несколько порядков больше недохакеров? Ни черта! Любой человек, мало-мальски владеющий информацией, знает, что единственно возможный путь для решения подобной задачи — это только перебор "в лоб" всех возможных вариантов пароля, что, естественно, требует колоссального количества машинного времени. И где его взять обладателю обыкновенного PC? И снова паранойя: "У меня ж всего пятый пень с тактовой частотой 10 тыс. — дохлятина, а вот в органах, там СУПЕРКОМПЬЮТЕРЫ, вот на них это сделать — элементарно!".

И тогда RSA делает вот что — бухает кучу денег в проект ***distributed.net***, который под девизом "The Fastest Computer on Earth" (Самый Быстрый Компьютер в Мире) начал заниматься такой интересной штукой как "использование свободного времени компьютеров во всем мире для осуществления распределенных вычислений". В том числе для взлома кодов RC5 — с целью демонстрации надежности 64-битных ключей (читай — лекарство от паранойи).

В настоящее время частями этого "суперкомпьютера" являются более чем 300 тыс. персональных компьютеров! А если учесть, что определенная часть "зарегестрированных пользователей" подключают на одну "учетную запись" не один компьютер, а несколько (администраторы компьютерных классов, системные администраторы), то на самом деле и того больше. Например, на момент написания этих строк (01.08.2002) по статистике за вчерашний день самый большой кусок работы выполнил "мембер" под номером 275282. Он обработал 1409939 блоков. Для сравнения: ваш покорный слуга, подключивший к суперкомпьютеру два линуксовых сервака и три крутые пользовательские тачки, за это же время обработал всего лишь 1848 блоков. Спрашивается, какие вычислительные ресурсы задействовал этот самый #275282, если ВСЯ СИСТЕМА, все 300 тыс. с хвостом мемберов обработали 10412746 пакетов? Загадка, однако...

Среди причин, по которым вам ОБЯЗАТЕЛЬНО ;) нужно подключиться к этому проекту, следующие:

1. Счастливчик, которому повезет обработать "тот самый блок", совершенно официально получит приз в 2000 $. При том, что расходов на участие — никаких ;).

2. Интересующиеся распределенными вычислениями и планирующие собрать собственный кластер — вот вам уникальная возможность проверить суммарную производительность имеющихся в вашем подчинении компьютеров.

3. Спортивный интерес ,)). Отдельные участники собираются в команды (помимо приза в две штуки, ровно столько же получает еще и команда победителя). След-

ствие этого — общение, совместное распитие пива и т.д. (вписать желаемое).

4. Будьте патриотами своей страны! ("Юноши и девушки! Комсомольцы и комсомолки! Вступайте в сплоченные ряды Молодой Гвардии!"). В настоящее время третье место в командном зачете уверенно держит команда HZTeam. United power of Russia and other xUSSR countres — Объединенная сила России и прочих экс-СССР стран (Хе... "красный кулак коммунистической заразы" — не могу не приколоться!). И не только держит, но и имеет хорошие шансы выйти на второе место, потеснив при этом буржуйскую "AnandTech". Помочь ей — святой долг каждого гражданина, имеющего выход в Инет! Как говаривали советские пропагандисты: "Догоним и перегоним Америку!".

Технологически все выглядит проще простого. Вы устанавливаете на своей машине "клиента", который осуществляет off-line перебор ключей, совершенно не загружая ваш компьютер, так как имеет очень низкий приоритет. В момент вашего выхода в Сеть он обменивается с сервером ключей блоками данных (порядка 10 с). Если, дочитав до этого места, вы еще не почувствовали глубокую внутреннюю потребность присоединиться к этому проекту, то дальше можете не читать, так как пойдет практика.

1. Для начала необходимо получить клиента. Для этого нужно сходить по адресу http://www.distributed.net/download/clients.html и сделать свой выбор. Как видите, существуют версии клиента практически для любой операционной системы. Для Windows смело можете качать это: http://http.distributed.net/pub/dcti/current-client/dnetc-win32-x86-setup.exe. Пре-релизные клиенты, которые якобы немного производительнее обычных, вы можете использовать на свой страх и риск, так как в том случае, если в них будет найдена ошибка, все блоки, которые вы обработали, будут аннулированы.

2. В процессе инсталляции программа задаст вам несколько глупых вопросов наподобие: "Прописываться в автозагрузке или нет?" "Запускаться как программа или как сервис (на клонах NT)?". И в конце концов запустит клиента в конфигурационном режиме, чтобы вы могли внести необходимые установочные данные и изменить параметры системы (**рис.1**).

Выбираем опцию "General Client Options", в ней первую позицию "You email address" (**рис.2**).

Введите ***свой реальный*** адрес электронной почты, впоследствии он будет служить идентификатором обработанных блоков, т.е. к этому пункту необходимо относиться с надлежащей внимательностью ;).

3. Если у вас не постоянное подключение к Интернету, а диалап, то имеет смысл вернуться в основное меню, зайти в "Buffer and Buffer Update Options" / "Keyserver <-> client connectivity Options" / "Dialup-link detection..." и изменить режим работы с 0 (по умолчанию) на 1, чтобы клиент передавал блоки кейсерверу только при наличии соединения, а не надоедал постоянными мольбами о коннекте (**рис.3**).

4. Вновь возвращаемся в основное меню и выходим из режима конфигурации через "Save settings and exit". Сразу же после этого клиент начинает свою работу. Если в данный момент вы — онлайн, то клиент обнаружит это, законнектится с сервером и примет первые блоки для обработки. В случае оффлайна, чтобы не терять времени, начнут обрабатываться случайные ("random") блоки, вплоть до того момента, когда появится возможность забрать настоящие.

Рис. 1

```
distributed.net RC5DES client
RC5DES Client Configuration:
------------------------------------------------------------
 1) General Client Options
 2) Buffer and Buffer Update Options
 3) Performance and Processor Options
 4) Logging Options

 9) Discard settings and exit
 0) Save settings and exit

Note: You have not yet provided a distributed.net ID.
      Please go to the 'General Client Options' and set it.

Choice -->
```

Рис. 2

```
distributed.net RC5DES client
RC5DES Client Configuration: General Client Options
------------------------------------------------------------
 1) Your email address (distributed.net ID) ==>
 2) Complete this many packets, then exit ==> 0 (no limit)
 3) Run for this long, then exit ==> 0:00
 4) Pausefile Path/Name ==>
 5) Disable all screen output? (quiet mode) ==> no
 6) Disable exit file checking? ==> no
 7) Disable the packet completion indicator? ==> no
 8) Project Priority ==> DES,CSC,OGR,RC5

 0) Return to main menu

Choice -->
```

Рис. 3

```
distributed.net RC5DES client
RC5DES Client Configuration: Keyserver<->client connectivity opti
------------------------------------------------------------

Modem detection options:

0) Normal mode: the client will send/receive packets only when it
        empties the in buffer, hits the flush threshold, or
        the user specifically requests a flush/fetch.
1) Dial-up detection mode: This acts like mode 0, with the additi
        that the client will automatically send/receive packets
        dial-up networking connection is established. Modem user
        will probably wish to use this option so that their clie
        never runs out of packets.
2) Dial-up detection ONLY mode: Like the previous mode, this will
        the client to automatically send/receive packets when
        connected. HOWEVER, if the client runs out of packets,
        it will NOT trigger auto-dial, and will instead work
        on random packets until a connection is detected.

Default Setting: 0
Current Setting: 0
New Setting --> 1_
```

Проконтролировать работу клиента (получение и отсылку блоков) можно по логу работы (**рис.4**, синяя стрелка — принято, красная — отправлено).

Рис. 4

```
.....10%....R20%.....30%...
[Oct 10 18:29:16 UTC] Connection detected on 'ppp0/lan0'...
[Oct 10 18:29:18 UTC] Connected to distributed.net 134.93.247.34:2064...
[Oct 10 18:29:18 UTC] The keyserver says: "Feeeeeeeed me, Seymour! (cyp)"
[Oct 10 18:29:20 UTC] Retrieved RC5 packet 3 of 3 (100.00% transferred)
[Oct 10 18:29:20 UTC] Retrieved 3 packets (23 work units) from server.
[Oct 10 18:29:21 UTC] Sent RC5 packet 3 of 3 (100.00% transferred)
[Oct 10 18:29:21 UTC] Sent 3 packets (13 work units) to server.
.....10%....R20%.....30%...
[Oct 10 18:30:01 UTC] Connection lost
```

После отправки первого "готового" блока вы становитесь зарегистрированным участником проекта distributed.net ;)).

5. Еще один маленький нюанс — это настройка приоритетов проектов. По умолчанию подпроект по взлому RC5 имеет самый низкий приоритет у клиента (наверное, для того чтобы он попутно принимал участие и в решении других, менее долгосрочных задач, таких как OGR). Команда HackZone отдает наибольший приоритет именно RC5, поэтому, чтобы не распылять силы команды, настоятельно рекомендуем вам изменить настройки клиента так, чтобы он обрабатывал исключительно блоки RC5.

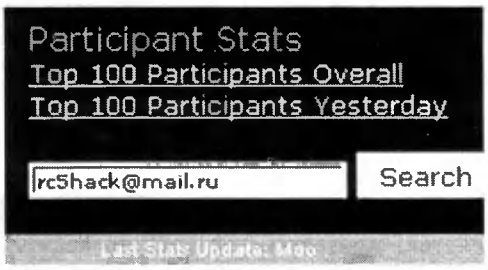
Для этого идем в папку с установленным клиентом, ищем там файл dnetc.ini и подправляем его следующим образом:

```
...
[misc]
project-priority=RC5,CSC=0,DES=0,OGR=0
...
```

Сохраняем и перезапускаем клиента ;).

6. Сутки спустя (сервер статистики обновляет свою статистику только раз в сутки — в 23.45 UTC) идем по адресу http://rc5stats.distributed.net/rc5-64/. Некоторое время любуемся общей статистикой, а потом ищем собственную. Для этого вводим свой мыл в специальное поле и жмем на Search (**рис.5**).

Рис. 5



Видим свою статистику (**рис.6**).

Рис. 6

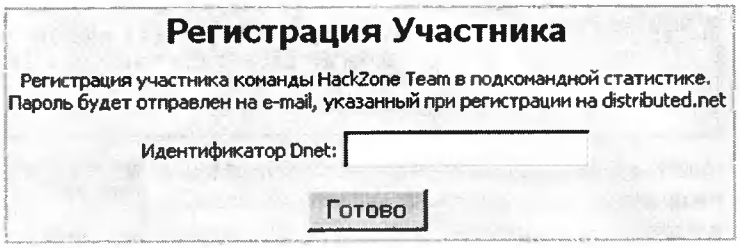
	Rank	Blocks
Overall:	93957 (▲1970)	30,346
Yesterday:	6924 (▲1276)	1,848
Total Blocks to Search:		68,719,476,736
Keyspace Checked:		0.00004416%
Total Keys Tested:		8,145,942,347,776
Time Working:		19days
Overall Rate:		4,962 KKeys/sec

В самом низу находим большую кнопку "Please email me password". Кликаем и ждем письма с паролем! Этот пароль нужен для того, чтобы, кликнув по ссылке "Edit Your Information" и введя запрашиваемые авторизационные данные (мыл и этот самый пароль), вы могли заполнить свой мемберский профиль.

7. Кликаем по ссылке "Top 100 Teams Overall", ищем в списке команду "HZTeam. United power of Russia and other xUSSR countries", кликаем по соответствующей ссылке. Попадаем на страничку с логотипом этой команды. Внизу страницы есть ссылка I want to join this team!, то есть "я хочу присоединиться к этой команде". Кликаем, у нас снова спрашивают логин с пассвордом — вводим. Все! Теперь вы в команде!

8. Осталось только присоединиться к *подкоманде*. Ну, здесь я уповаю на то, что глубокоуважаемый и глубоколюбимый читатель сделает свой выбор в пользу команды HI-TECH, к которой относится и которую продвигает автор всего этого бреда ;). Для того чтобы сделать это, вам необходимо получить еще один пассворд — "хакзоновский" (или, если угодно, "багтраковский"). Идем по адресу http://rc5.bugtraq.ru/sub/, введим свой ID в следующем поле (**рис.7**).

Рис. 7



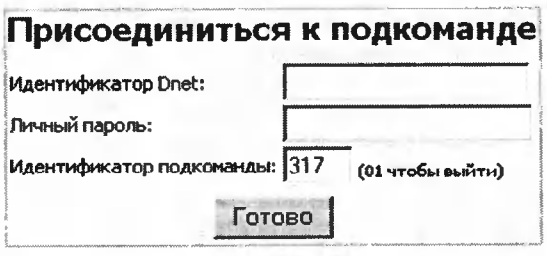
Под ID'ом подразумевается вовсе не ваш e-мыл, который, вроде, тоже идентификатор, а порядковый номер вашей учетной записи на distributed.net. Выдрать его можно, например, из письма с пассвордом. В частности, из вот этой вот ссылки: http://stats.distributed.net/participant/pedit.php?**id=394113**&pass=qwerty.

После этого вам должно придти письмо от bugtraq'а приблизительно следующего содержания:

Your password for user '394113' on www.hzteam.net is '12345676859abcdef'

С этим пассвордом идем на страничку с подкомандной статистикой HI-TECH (http://rc5.bugtraq.ru/sub/subteam.php?id=317), ищем там формочку для регистрации в подкоманде (**рис.8**).

Рис. 8



Вводим свой идентификатор, личный пароль, который только что получили, и жмем на кнопку "Готово".

И... спасибо за соучастие ;) Догоним и перегоним Америку! Windows Must Die!

Ссылки:

1. http://distributed.net/ — главный сайт проекта.
2. http://stats.distributed.net/ — сервер статистики.
3. http://rc5.bugtraq.ru/ — сайт команды HackZone.

3DNEWS Daily Digital Digest

Глоссарий современных 3D-терминов

Д.ЧЕКАНОВ,
E-mail: dmitry@vspu.kirov.ru,
FRANCHY (ЗоринD),
E-mail: dima@bws.rosugol.ru,
С.ВАСИЛЬЕВ,
E-mail: stanislav@3dnews.ru,
А.ГУЛЯЕВ,
E-mail: jeff3d@delphicity.net

(Окончание. Начало в №10/2002)

Game Engine, см. engine, движок. Программный код, на котором работает игра.

Gamma Correction, гамма-коррекция. Возможность управлять красной, зеленой или синей составляющей пиксела или текстуры для определения требуемой яркости свечения.

Glide3D (Glide). Собственный API 3dfx для работы с 3D-графикой. По-моему, программировать под него легче и удобнее, чем под Direct3D. Да и выглядит графика субъективно лучше. Единственная проблема этого API — он является собственностью 3dfx, поэтому его поддерживают только карты Voodoo. Если у вас есть старый эмулятор PS или N64, то он будет работать только под этим API. С точки зрения кодирования, Glide и OpenGL очень друг на друга похожи. Сейчас Glide3D "умер", как и 3dfx.

Gouraud Shading, алгоритм затенения по методу Гуро. Позволяет получить на поверхности объекта плавное затенение — во всех вершинах объекта строятся вектора нормалей; в зависимости от угла между нормалями и направлением на источник света определяется цвет пикселов (как при flat shading), соответствующих вершинам полигонов; цвета пикселов интерполируются (между вершинами) по поверхностям полигонов. Блики выглядят нереалистично (**рис.7**, все рисунки на 2-й странице обложки).

Graphic Aperture Size, размер апертуры. Вы, наверное, никогда не встретитесь с этим термином, если не заглянете в BIOS компьютера. Величина апертуры указывает на размер системной памяти, которая может использоваться AGP-видеокартой. Если у вас много лишней памяти, то вы можете получить небольшой прирост производительности, увеличив величину апертуры.

Graphics Pipeline, графический конвейер. Путь информации в компьютере весьма сложен. Обычно он начинается в центральном процессоре, далее информационный поток проходит через материнскую плату, шину AGP или PCI, заходит на видеоускоритель и затем попадает на экран монитора. Чем быстрее информация проходит этот путь (или конвейер), тем быстрее работает графика.

Integer, целый. Если вы вспомните среднюю школу, целыми числами назывался ряд 0, 1, 2..., а также их отрицательные значения. В компьютерах целые числа определяются точно так же, но они играют другую важную роль. Компьютер обрабатывает целые числа намного быстрее десятичных дробей, хотя целые числа не всегда удобны с точки зрения точности вычислений.

Internal Rendering, внутренний рендеринг. Количество цветов, с которым видеокарта создает изображение. Обычно оно бывает 16-битным, 24-битным или 32-битным. Большее количество цветов обуславливает большую красочность игры, но, в то же время, и некоторую потерю в производительности и количестве кадров в секунду. Большинство современных игр могут осуществлять 32-битный внутренний рендеринг.

I/O Connector, разъем ввода/вывода (I/O означает ввод/вывод). Такие разъемы, или порты, существуют на устройствах, к которым могут подключаться другие устройства. Примером могут служить шины AGP и PCI.

IRQ, Interrupt Request, запрос на прерывание. Как только устройство, например, видеокарта или звуковая карта, пожелает обратиться к центральному процессору, оно посылает прерывание. Прерывания в компьютере нумеруются от 0 до 15. По старым стандартам, два устройства не могли использовать одно и то же прерывание. Сейчас существуют технологические решения, обходящие это правило.

ISA, Industry Standard Architecture, стандартная промышленная архитектура. Одна из стандартных шин расширения. Существуют и более современные шины — AGP или PCI. ISA работает на частоте 8 МГц и предназначена для не очень требовательных к скорости передачи информации устройств — звуковых карт, модемов и сетевых карт. Сегодня ISA уже не используется в современных компьютерах, благодаря широкому продвижению PCI.

LAN — см. Local Area Network

Level of Detail, уровень детализации. Зависит от разрешения текстуры, к примеру, 256x256 и т.д. Низкий уровень детализации означает маленький размер текстур, а не визуальное качество картинки. Зато при низком уровне частота кадров увеличивается. Вот и решайте, что вам больше нужно — потрясная визуализация или бешеная скорость?

Lighting Effects, световые эффекты. Процесс создания эффектов, симулирующих свет в 3D-графике. Реализуется с помощью подсветки текстур и пикселов около виртуального источника света. Освещение является одной из самых красивых возможностей ускорителя, так как оно значительно улучшает восприятие графики.

Local Area Network, локальная сеть. Группа компьютеров, физически соединенных кабелями в сеть и находящихся недалеко друг от друга.

Local Memory, встроенная память. Такая память устанавливается на устройства. Например, на видеокарту встроена оперативная память для хранения данных. Эта память используется ускорителем для хранения текстур, Z-буфера и т.д.

Low-Frequency Response, низкая частота. Низкими считаются частоты звука от 3 Гц до 120 Гц. Если не вдаваться в детали, то такой звук похож на бас, вы чувствуете его не только ушами, но и всем телом.

MIDI, Music Instrument Digital Interface, цифровой интерфейс музыкальных инструментов. Здесь мы подразумеваем музыку в формате MIDI. При ее проигрывании используются

короткие звуковые файлы (возможно, сгенерированные компьютером), у которых меняется частота и темп, а затем из таких кусочков создается музыка. Это очень напоминает использование алфавита для составления слов. Что самое хорошее в MIDI — такая музыка быстро загружается и может использоваться как на Web-страницах, так и в играх. С другой стороны, звук не всегда получается хорошим.

miniGL. Драйвер, специально разработанный для частичной поддержки OpenGL-ускорителями. Позволяет карте работать с OpenGL-играми.

Mip-Mapping, мип-мэппинг (мип-текстурирование). Процесс преобразования изображения или текстуры в меньшие по размеру изображения. MIP означает "многое в одном" (lat. "Multum In Parvum"). Для разных частей обьекта алгоритм использует текстуру с различным разрешением (256x256,128x128,64x64 и т.д.), в зависимости от расстояния между наблюдателем и поверхностью и от угла, под которым находится поверхность (**рис.8**).

Побочным эффектом mip-mapping'а является banding — разрывы между mip-уровнями (текстурами с различным разрешением). К тому же, теряется резкость текстур.

На **рис.9** приведено изображение без (!) mip-mapping'а.

MMX. Разработанный Intel набор из 57 новых инструкций для x86 процессоров, ускоряющий работу с мультимедийными приложениями. На самом деле, этот набор не очень быстр и не очень хорошо подходит для 3D-приложений типа игр.

Multitexture (Multitexturing), мультитекстурирование. Процесс добавления множества текстур к объекту при программировании 3D-игры. Они могут накладываться друг на друга для отображения шероховатостей поверхности, или использоваться друг с другом для создания одной большой 3D-модели, и т.д. При этом цвета текселов этих текстур смешиваются по определенному закону — add (сложение), modulate (умножение), substractive (вычитание) и др (**рис.10**).

Obstruction Effects, звуковой эффект препятствия. Наблюдается при прохождении звука через препятствия типа стен или других твердых предметов. Звуковой сигнал при этом ослабляется. Такой звук кажется глухим, его частота меньше оригинала.

OpenGL, Open Graphics Language. Графический API, созданный и поддерживаемый Silicon Graphics. Мне этот API нравится намного больше Microsoft Direct3D, и я всегда предпочитаю играть под OpenGL.

OpenGL ICD, installable client driver, устанавливаемый клиентский OpenGL-драйвер. Как мне кажется, такой драйвер призван обходить "нагроможденность" API OpenGL. ICD реализует только игровые свойства API, например только те, которые нужны для игры в Quake II.

Overclock, превышение тактовой частоты, "разгон". Обычно под этим термином подразумевают изменение перемычек или параметров BIOS для ускорения "железа" вашего компьютера. Сейчас производители видеокарт уже встраивают возможность разгона прямо в драйверы — вы меняете несколько цифр и увеличиваете тактовую частоту. Раньше же всем этим приходилось заниматься вручную, с помощью отвертки/пинцета и...

PCI, Peripheral Component Interface, шина периферийных компонентов. Является промышленным стандартом шины расширения компьютера. Через шину расширения подключаются различные устройства— видеокарты, сетевые карты и т.д. Стандартной частотой шины является 33 МГц, хотя существуют и более скоростные варианты. Думаю, PCI проживет еще достаточно долго.

Per-Pixel Mip-Mapping, попиксельный мип-мэппинг. Это самая точная версия мип-мэппинга. Для увеличения производительности, мип-мэппинг может применяться не только к пикселу, но и к полигону, и т.д. Но только при попиксельном мип-мэппинге тени получаются такими же детальными, как и отбрасывающие их объекты.

Perspective Correction, исправление перспективы. Заключается в возможности правильно отображать текстуру под любым углом. Если вы посмотрите под прямым углом на квадратную текстуру, стороны выглядят одинаковыми. Но стоит только вам изменить угол, как квадрат оказывается перекошенным. Почему же так происходит? В реальном мире, чем дальше объект, тем он кажется меньше. Поэтому и в нашем примере одна сторона кажется больше, а другая — меньше, что и создает эффект трехмерности.

Phong shading (затенение Фонга) — один из самых качественных типов затенения (**рис.11**). Отличается от gouraud-затенения тем, что вектора нормалей строятся для каждой точки изображения (соответственно, требует много вычислений).

Pipeline, конвейер. В нем хранится очередь операций для микропроцессора. В известной степени конвейер — это то место, где скапливаются все инструкции, чтобы процессор не простаивал. Или, может быть, это область памяти процессора (кеш) для хранения данных.

Pixel, пиксел. Самый маленький объект на экране монитора. Дисплей состоит из пикселов (см. разрешение).

Pixelation, пикселизация. Когда вы слишком близко приближаетесь к текстуре в игре, то можете заметить "квадратики", из которых она состоит. Это и есть пикселизация. Для примера, вы можете поставить на 21"-мониторе разрешение 640x480.

Point sampling — самый простой метод фильтрации текстур (ее отсутствие :-)). Для определения цвета конечного пиксела используется цвет одного (самого близкого) тексела.

Как видно на **рис.12**, результатом point sampling является пикселизация (гранулированность) изображения, хотя это самое точное представление исходной текстуры (bitmap'а).

Polygon, полигон, многоугольник. Если вы вспомните геометрию, то многоугольником называется замкнутая двумерная фигура, образованная рядом прямолинейных отрезков. Многоугольники часто называют полигонами. Полигоны комбинируются с сотнями других для создания цельной модели в 3D-движках.

Raytrace, метод бегущего луча. При этом методе просчитывается виртуальный луч света от источника через все отражающие поверхности до того объекта, на который падает свет. Метод создает очень реалистичные эффекты, а также прозрачные поверхности.

Realtime, в реальном времени. Действие, производимое компьютером с той же самой скоростью, что и в реальной жизни.

Reflective Mapping, отображение отражающих поверхностей. Техника, сходная с методом бегущего луча. Позволяет создавать текстуры, правдоподобно отражающие объекты вокруг.

Refresh Rate, частота обновления. Частота, с которой монитор выводит

статические изображения для симуляции движения. Измеряется в герцах (Гц). Я считаю, монитор с частотой обновления меньше 80 Гц вообще не следует покупать, так как это будет вредить вашим глазам. Кстати, если вы посмотрите на монитор компьютера через телекамеру, то заметите полосу. Она вызвана разными частотами обновления.

Resolution, разрешение. Выражается в виде "количество пикселов по горизонтали × количество пикселов по вертикали", произведение дает число пикселов, отображаемых монитором. Например, 640x480, 800x600 и т.д.

Reverberation, реверберация, отражение. Естественный звуковой эффект, появляющийся в результате отражения звука от многих объектов. Проявляется в виде продолжительного эха. Производиться может звуковой картой. Реверберацией называют и нежелательный дефект звука, когда происходят короткие перерывы звучания. Послушайте продолжительный взрыв в Half-Life, и вы поймете, о чем я говорю.

RISC, Reduced Instruction Set Computing, вычисления с сокращенным набором команд. Архитектура, при которой процессор в большом количестве использует упрощенные инструкции. Так как простые инструкции работают быстрее, это позволяет увеличить скорость выполнения различных задач. Если у вас есть приставка типа Playstation, Nintendo64 или Dreamcast, то они как раз и используют RISC-процессоры.

SIMD, Single Instruction Multiple Data, один поток команд и много потоков данных. Упрощенно, SIMD позволяет процессору выполнять одну и ту же операцию над несколькими потоками данных. Таким образом, процессор освобождается от повторного ввода одинаковых инструкций, что увеличивает производительность. Как Intel MMX и SSE, так и AMD 3DNow! используют SIMD для повышения производительности мультимедийных и игровых приложений.

Scan Line Interleave (SLI), чересстрочная развертка. Мне не очень понятно, как появился такой термин, но 3dfx Voodoo2 активно его использовала. Две карты соединялись друг с другом, одна выводила четные строки, а другая — нечетные. Производительность, соответственно, увеличивалась. Такая технология позволяла не только повышать частоту кадров, но и использовать более высокое разрешение. Раньше считалось шиком иметь две карты Voodoo2 через SLI-соединение.

SGRAM, Synchronous Graphics Random Access Memory, синхронная графическая память с произвольной выборкой. Является основным типом памяти для графических карт. Считается, что SGRAM быстрее своего близкого родственника — SDRAM, благодаря возможности синхронизироваться с центральным процессором для получения оптимальной скорости и производительности.

Software Rendering, программный рендеринг. Весьма ужасный режим рендеринга графики, не задействующий 3D-ускоритель. Обычно медленный и не очень красивый. Является главной причиной изобретения 3D-ускорителей. Если игра не поддерживает ускоритель, то она использует программный рендеринг.

Specular highlight (блики) — имитация прямого отражения источника света (**рис.13**).

При простом наложении текстур могут возникать некоторые дефекты в изображении (глюки :-) — когда камера (наблюдатель) близко приближается к объекту, текселы становятся больше чем пикселы (один тексел заполняет собой несколько пикселов), проявляется эффект пикселизации (блочности) изображения (см. point sampling). Если текстура находится на большом расстоянии от наблюдателя, и текселы становятся меньше пикселов (на один пиксел "претендуют" несколько текселов текстуры — см. mipmapping), то проявляется эффект муара (ряби), так как в одном и том же пикселе все время рисуются разные текселы.

Sprite, спрайт. Двумерное графическое изображение. Примером можно считать прицел в Quake2 или Half-Life.

S-Video. Не будем углубляться в технические детали, S-Video является стандартом, который поддерживается многими 3D-ускорителями. На них вы можете обнаружить разъем S-Video для подключения цифровых камер, видеомагнитофонов, телевизоров и т.д.

Texel, texture element, элемент текстуры, тексел. Обычно текселом называют пиксел применительно к 3D.

Texture. Графическая картинка, "натягиваемая" на полигональные каркасы в 3D. С помощью текстур мы получаем тот прекрасный трехмерный мир, который наблюдаем в играх.

Texture Compression, сжатие текстур. Возможность видеоускорителя уменьшать размер картинки, кодируя повторяющиеся строки и уменьшая цветовую палитру текстуры. Технология может радикально повысить частоту кадров, ухудшая качество изображения, или повысить визуальное качество с помощью поддержки постоянной палитры. В любом случае, сжатие текстур делает нашу жизнь приятнее.

Texture Mapping, текстурное отображение. Процесс "натягивания" текстуры или картинки на 3D-полигональный скелет. Также этот процесс называют текстурированием.

Texture Memory, текстурная память. Часть памяти видеоускорителя, выделенная для хранения используемых для обтягивания объекта текстур.

TFT (Thin Film Transistor), тонкопленочные транзисторы. Используются для создания тонких жидкокристаллических дисплеев.

3D Positional Sound, звук с трехмерным позиционированием. Примерами реализации такого звука являются технологии Sensaura 3D, Aureal3D и DirectSound3D. Они позволяют программировать звук, позиционированный в трехмерном пространстве. Используют сложные математические алгоритмы.

3DNow!-инструкции. Набор AMD из 21 инструкции для увеличения скорости выполнения математических операций с плавающей точкой. Впервые набор был реализован в процессоре K6-2, также он был лицензирован Cyrix и Centaur Technology. Игры, использующие 3DNow!, работают быстрее.

Transparency, прозрачность. Объект, через который мы можем видеть другие объекты, называется прозрачным. Примерами в 3D-графике могут служить стекло, вода, взрывы или лед.

Trilinear Filtering, трилинейная фильтрация. Процесс применения билинейной фильтрации к каждой стороне текстуры. Улучшает четкость изображения и убирает пикселизацию. При этом типе фильтрации фильтруются не только текселы, но и mip-уровни, т.е. сначала вычисляются цвета двух пикселов в двух (соседних) mip-уровнях, а затем эти два значения смешиваются (**рис.14**).

Единственный недостаток трилинейной фильтрации — потеря резкости текстур.

Tweak, твик, разгон. Увеличение тактовой частоты, оптимизация и другие подобные вещи. В общем случае подразумевается процесс изменения некоторых параметров системы для увеличения производительности.

VGA, Video Graphics Accelerator/Array, стандарт графики VGA. В начале 90-х годов цветные мониторы VGA были весьма распространены, затем появился SVGA, с увеличенными количеством цветов и разрешением.

VGA Output, VGA-выход. Предназначен для подключения кабеля от VGA-монитора к компьютеру.

Volumetric Lighting, пространственное освещение. Эффект прохождения света через трехмерную преграду типа тумана, облака пыли, дыма, пара и т.д.

Volumetric Fogging, пространственный туман. Так называется использование "виртуального" тумана, который для увеличения производительности скрывает непрорисованные текстуры на некотором расстоянии. Этим очень "увлекалась" игра Turok 2. Хотя, конечно, лучше видеть туман, чем текстуры, внезапно появляющиеся неизвестно откуда.

VRAM, Video Random Access Memory, видеопамять с произвольной выборкой. Такая память используется в 3D-устройствах. Ее преимущество заключается в возможности одновременного обращения двух устройств. А это как раз и бывает весьма актуально в 3D.

Vsync, Visual/Vertical Synchronization, вертикальная синхронизация. Возможность видеокарты синхронизировать обновление буфера с частотой развертки монитора, устраняя различные артефакты и аномалии. Некоторые производители видеокарт позволяют отключать vsync для повышения частоты кадров, однако это обычно приводит к ухудшению визуального качества изображения.

Wavetable, таблица волнового синтеза. Сначала познакомьтесь с MIDI, так как они очень друг на друга похожи. Волновой синтез является возможностью звуковой карты воспроизводить короткие звуковые отрывки (сэмплы) в нужном порядке для составления мелодий.

Wavetable Synthesis — см. Wavetable и MIDI.

Wavetracing, отслеживание звуковой волны. Так называется возможность звуковой карты воспроизводить реалистичные звуки с помощью отслеживания пути их распространения. Если вы слышите звук на некотором расстоянии от источника, то wavetracing дает возможность определить, откуда он пришел. Также отслеживание звуковой волны позволяет корректно воспроизводить эхо.

Z-Buffer, Z-буфер. Является частью памяти 3D-ускорителя, выделенной под хранение координаты Z для трехмерных точек. Если вы вспомните геометрию, у каждой точки в трехмерном пространстве существует три координаты — X, Y и Z. Смысл использования Z-буфера очень прост — он позволяет видеоускорителю не прорисовывать текстуры, скрытые позади других текстур. Например, если вы заходите в игре за стену, то за ней вы не можете видеть другие объекты, а с помощью Z-буфера карта не будет их отрисовывать лишний раз. Z-буфер позволяет значительно увеличивать производительность. Также при его использовании улучшается точность позиционирования по оси Z.

О значении разрешающей способности

А.ГОРЯЧКИН,
г.Кыштым, Челябинской области,
E-mail: anatoliy_goryach@mail.ru

Очень часто в технических характеристиках того или иного устройства вывода (принтера, плоттера и др.) в качестве одного из параметров указывают разрешающую способность. От этого параметра напрямую зависит качество печати. Измеряется разрешающая способность в dpi (dot per inch). Она показывает, какое количество точек размещается на одном дюйме. Чем больше количество точек, тем, соответственно, выше качество печати. Поэтому при покупке струйного или лазерного принтера стоит обращать внимание на этот параметр.

При указании технических характеристик устройств с высокой разрешающей способностью, таких как фотонаборные аппараты, лазерные фотоплоттеры и т.д., дополнительно указывают еще один немаловажный параметр — дискрету, или шаг записи. Измеряется этот параметр в микронах (мкм). Чем меньше значение дискреты, тем больше количество точек на единицу длины и, соответственно, выше качество получаемого изображения. Между разрешающей способностью и дискретой существует обратно пропорциональная зависимость. Зная величину дискреты X, можно по формуле (1) вычислить значение разрешающей способности, и наоборот, зная значение разрешающей способности Y, можно по формуле (2) подсчитать величину дискреты X:

$$Y = 25{,}4\frac{1000}{X}, \qquad (1)$$

$$X = 25{,}4\frac{1000}{Y}. \qquad (2)$$

В таблице приведены наиболее распространенные значения разрешающей способности и соответствующие им величины дискрет.

Разрешающая способность, dpi	Дискрета, мкм	Количество точек на 1 мм
300	84,7	12
360	70,6	14
508	50	20
600	42,3	23
720	35,3	28
1016	25	40
2032	12.5	80
2540	10	100
3387	7,5	133

Serrgio /HI-TECH,
E-mail:serrgio@gorki.unibel.by
http://www.wasm.ru/

Протоколы Интернет

(Практикум для шаловливых ручек)

Однажды (да простит читатель маленькое лирическое отступление) родители купили мне программируемый калькулятор. Кнопки "равно" я там почему-то не нашел, поэтому взял отвертку и начал медленно и методично разбирать его на части. За этим занятием меня и застала мама — она схватила шланг от пылесоса и начала гонять меня по квартире. В тот самый момент, когда мое нежное ушко уже почти начало отделяться от головы, с работы пришел папа и прикрыл меня волосатой грудью. Мама орала, что "эта вещь стоит денег". Папа орал, что "ну и что, зато правильной дорогой идет, и вообще бить детей непедагогично". А я, виновато потупив очи, с тоской смотрел на телевизор... Потом я смотрел на него с интересом... А потом, совершенно случайно уронив в рукав отвертку, начал помаленьку ерзать попой по дивану в направлении телевизора...

С тех пор прошло уже много лет, но нездоровая страсть "разбирать на части" по-прежнему не дает мне покоя ;). Когда у меня не было Интернета, я "разбирал" программы — с этой частью моих "разборок" вы можете познакомиться в цикле статей "Низкоуровневое программирование для дЗенствующих", который "Ваш компьютер" публикует с №9/2001, и конца этому не предвидится ;). А вот с протоколами Интернета я начал разбираться сравнительно недавно — и это оказалось не менее интересно, чем исследование программ ;).

Я не могу обещать вам систематичность и последовательность изложения — скорее, это будет нечто, похожее на "записки охотника". Но мне хочется надеяться, что практическое использование этих материалов все же доставит вам несколько приятных вечеров ;). Во всяком случае, я очень постараюсь, чтобы это было именно так ;).

На этой лирической ноте берем в руки "отвертку" и "молоток", скачиваем из Инета все существующие в природе RFC'шки (ftp://ftp.rfc-editor.org/in-notes/tar/RFC-all.zip) и лезем в паутину Web — охотиться на пауков и прочих нехороших тварей.

Post Office Protocol 3 (POP3)

Для тех, кто не в курсе — именно через этот протокол ваш *почтовый клиент* забирает почту с *почтового сервера*. Сервер **POP3** прослушивает *110-й* порт (TCP) на предмет запросов на установку соединения, и как только обнаружит попытку соединиться — выдает *клиенту* специальную строку-приглашение, свидетельствующую о готовности сервера начать "конструктивный диалог" по передаче почты.

Первоисточники по данной теме находятся по адресу http://www.rfc-editor.org/ (RFC-1081, RFC-1082, RFC-1225, RFC-1725, RFC-1939). Мы же займемся практикой.

Запускаем в командной строке терминалку со следующими параметрами:

```
telnet <хост> <110>
```

Под хостом подразумевается имя или IP-адрес *почтового сервера*, с которого вы собираетесь коннектиться. Естественно, что этот сервер должен быть не абы каким, а *почтовым* ;). И "телнетиться" к нему нужно не на какой-нибудь порт, а именно на *110-й*.

При успешном соединении сервер ответит нам приблизительно такой строкой:

```
+OK POP3 mail.domain.net v2000.70rh server ready
```

Это означает, что почтовый сервер "на той стороне" действительно присутствует и находится в полной боевой готовности обработать все ваши запросы.

Что же, давайте начнем с ним общение. Чтобы получить доступ к своей почте, нужно себя идентифицировать и, естественно, секретным пассвордом (password, пароль) подтвердить свою "подлинность".

Первое делается при помощи команды:

```
user <логин>
```

Здесь под логином подразумевается слово, которое в адресе электронной почты находится до "собаки".

Сервер ответит:

```
+OK User name accepted, password please
```

Это означает, что команда прошла успешно, и можно вводить следующую.

Вводим пароль:

pass <пароль>

Если на сервере отсутствует юзверь (user) с таким логином-пассвордом, то появится сообщение:

```
-ERR Bad login
```

А если такой пользователь есть (что нас, собственно, и интересует), то появится:

```
+OK Mailbox open, 4 messages
```

То есть мы успешно открыли свой почтовый ящик, в котором в настоящий момент находятся 4 штуки мессаг (message, сообщение).

Теперь, когда мы должным образом авторизовались, можно начинать работу с нашим почтовым ящиком ;).

Для начала проверим, сколько в нашем ящике накопилось мессаг, при помощи команды:

```
stat
```

Получаем ответ:

```
+OK 4 5445
```

что переводится как "в вашем почтовом ящике 4 письма, и "весят" они 5445 байтов".

Вводим команду:

```
list
```

и получаем ответ:

```
+OK Mailbox scan listing follows
1 720
2 2105
3 856
4 1764
```

Нетрудно догадаться, что это нумерованный список мессаг и "вес" каждой в байтах. Ту же команду можно вводить и с параметром, в качестве которого выступает номер мессаги. В этом случае сервер вернет размер только той мессаги, номер которой вы ввели.

Хорошо, список мы просмотрели. Давайте теперь читать нашу почту ;). Для этого существует команда:

```
retr <номер мессаги>
```

в ответ на которую сервер ответит нам чем-то похожим на:

```
+OK 809 octets
Return-Path: <serrgio@gorki.unibel.by>
Received: from master ([192.168.1.8])
        by gorki.unibel.by (8.11.6/8.11.6) with ESMTP id g5I2M4M28676
        for <serrgio@gorki.unibel.by>; Mon, 17 Jun 2002 19:22:06 -0700
Date: Mon, 17 Jun 2002 20:15:58 +0300
From: Serrgio <serrgio@gorki.unibel.by>
X-Mailer: The Bat! (v1.46) Personal
Reply-To: Serrgio <serrgio@gorki.unibel.by>
X-Priority: 3 (Normal)
Message-ID: <12277216736.20020617201558@gorki.unibel.by>
To: serrgio@gorki.unibel.
Subject: privet!
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Status:

Hello serrgio,

 Privet pridurok! To, 4to ti pishesh pis'ma samumu sebe - pervij
 priznack shizofrenii!

_
Best regards,
 Serrgio
mailto:serrgio@gorki.unibel.by

.
```

Вот она — мессага во всей ее красе! Именно так она "в натуре" и выглядит — с непонятными `X-Mailer`, `Reply-To` и еще более страшными `Message-ID: <12277216736.20020617201558@gorki.unibel.by>` и обязательной точкой в конце текста. Что это за словеса такие, я обязательно расскажу в следующий раз, а пока что посмотрим, какие еще команды у нас понимает POP3.

Следующая команда:

`dele <номер мессаги>`

удаляет письмо с почтового сервера. Но не сразу, а только после того как вы должным образом вылогинитесь из своего почтового ящика. То есть если вы просто закроете консольку reset'ом, то мессага не удалится.

Команда **noop** ничего не делает, просто просит у сервера подать признаки жизни, которые он с удовольствием и подает, да еще и прикалывается:

`+OK No-op to you too!`

Команда **rset** сбрасывает текущий сеанс и возвращает его к той точке, когда пользователь только что прошел процедуру аутентификации. Все изменения, сделанные пользователем в системе (например, удаление мессаги), отменяются.

Команда **quit**— догадайтесь сами, что делает.

Есть еще команды **top** (получение краткого описания мессаги) и **uidl** (что-то связанное с уникальными номерами), но они поддерживаются не всеми серверами. Поэтому, черт с ними. Вышеперечисленных и так за глаза хватит, чтобы легко и не напрягаясь продемонстрировать свою "кулхацкерную" крутизну перед вашими ламерствующими коллегами.

Теперь снова вернемся к авторизации. Вас, должно быть, неприятно удивило то, что при вводе пароля командой **pass** пароль отображается на экране в открытом виде, а не какими-нибудь там "звездочками". Вынужден вас огорчить — по сети он также передается "как есть". И подсмотреть его — раз плюнуть! Так как же, спрашивается, защититься от подобного "подглядывания" со стороны нехороших товарищей, считающих себя крутыми хакерами? А очень просто — использовать для аутентификации команду **apop**, которая поддерживается, к сожалению, не всеми POP3-серверами.

Синтаксис этой команды следующий:

apop name digest,

где **name** — имя пользователя, а **digest** — шифрованная по хитрому алгоритму строка пароля.

Должно быть, вы уже обратили внимание на то, что после установления коннекта большинство серверов, помимо всего прочего, отвечают еще и строчкой наподобие **<1372.1025853497@mail.webrider.ru>**. Это так называемая "временная строка" (timestamp), которая генерится почтовым сервером в формате **<process-ID.clock@hostname>**, где process-ID — это идентификатор процесса, clock — состояние таймера на момент установления соединения, ну а hostname — он и в Африке имя хоста.

Параметр digest вычисляется следующим образом:

1. К временной строке конкатенируется (операция связывания — concatenation) пассворд (внимание, "угловатые скобки" — это часть временной строки!). Например, если пассворд — **qwerty**, а временная строка такая, как приведено выше, то получится: **<1372.1025853497@mail.webrider.ru>qwerty**

2. Получившуюся строку хэшируем алгоритмом MD5. Результатом для нашего примера будет "шифрованная строка" 453070495AA789DB0F6A6CF039182FB3.

3. Заменяем в хэше заглавные буквы на строчные. Получится 453070495aa789db0f6a6cf039182fb3.

Все ;). Наш digest готов к употреблению, что мы с удовольствием и проделаем:

apop username 453070495aa789db0f6a6cf039182fb3.

И пускай какой-нибудь кулхацкер попробует догадаться, что эта длинная строка символов и есть наш незамысловатый пароль qwerty!

Подозревая, что отнюдь не все из вас могут вычислить MD5-хэш в уме, мы написали программу (простенький хэш-калькулятор), которая проделает за вас всю черновую работу по коннактенации, хэшированию и замене символов, скачать которую вы можете на сайте журнала (http://radio-mir.com).

(Продолжение следует)

Рисунок О.Попова

ЛИСТАЯ СТРАНИЦЫ

Пользователи достаточно редко озабочены тем, насколько сильно шумит компьютер. Как правило интересуются другими его характеристиками. Однако если компьютер станет основой "домашнего кинотеатра", этот вопрос приобретет актуальность. На высококачественные звуковую карту и колонки тратятся немалые средства, но может оказаться, что эти затраты напрасны, если высококачественный звук сопровождается неприятным акустическим фоном компьютера. **Об источниках шума в компьютере и о способах его уменьшения** рассказывает *Р.Никитин* в статье "Борьба за тишину" **(Hard'n'Soft, №5/2002, С.68...72)**.

Если пользователь работает в офисе, выходящем окнами на оживленную улицу, то уровень шума в этом помещении приближается к отметке 35...45 дБА. Поскольку среднестатистический компьютер выдает порядка тех же 35...45 дБА, его просто не будет слышно за шумом кондиционеров, офисной техники и ходящих по помещению сотрудников. Среднее значение шума в тихой комнате днем приблизительно равно 25...30 дБА, то есть заметно меньше, чем шум от среднестатистического компьютера. В этом случае имеет смысл снизить уровень шума компьютера хотя бы до уровня шума в помещении без компьютера. Уровень шума в квартире ночью оценивается в 15...20 дБА, что накладывает еще более жесткие ограничения на компьютер, который не должен мешать спать домочадцам.

Особым случаем является "домашний кинотеатр". Полноценное восприятие симфонической музыки рядом с компьютером, дающим акустический шум 40...45 дБА, затруднительно. Даже если установить верхнюю границу громкости воспроизводимого колонками звука на уровне болевого порога, то самые тихие звуки, потеря которых для искушенного слуха неприемлема, окажутся замаскированными акустическим шумом компьютера. Вот почему полноценная реализация мультимедийных возможностей современного компьютера зачастую требует снижения уровня его акустического шума.

Акустическое воздействие компьютера на окружающее пространство определяется совокупностью шума вентиляторов охлаждения, шума работы некоторых электронно-механических устройств (жесткий диск, приводы дисководов и CD-ROM), шума клавиатуры и шума периферийных устройств, например, принтера.

Общее количество вентиляторов в современном высокопроизводительном компьютере зачастую достигает четырех. А иногда в корпусе компьютера устанавливаются еще один-два дополнительных прокачивающих вентилятора. Особенно остро проблема шума стоит для систем на базе процессора AMD Athlon, требующего интенсивного охлаждения.

Двумя основными источниками звука в любых вентиляторах являются шум подшипника и шум воздуха, прокачиваемого через ребра охлаждения радиатора. Кроме того, для любого вентилятора издаваемый им шум зависит от типа опор подшипника, диаметра крыльчатки, формы лопастей и скорости вращения ротора. Подшипники для компьютерных вентиляторов изготавливают двух типов — качения (на вентиляторе или его упаковке написано "ball bearing") и скольжения ("sleeve bearing"). Последние — более тихие, так как лишены подвижных шариков, присущих подшипникам качения. Зато подшипники качения считаются более надежными и долговечными. С точки зрения снижения уровня шума от вентилятора, предпочтение следует отдавать именно подшипникам скольжения. Срок службы таких вентиляторов можно продлить, регулярно смазывая опорные втулки жидким машинным маслом.

Производительность вентилятора определяется диаметром крыльчатки и формой ее лопастей. При этом наблюдается такая зависимость: вентиляторы большего диаметра, как правило, имеют большее количество лопастей, однако их угол атаки меньше. Маленькие крыльчатки порой обходятся четырьмя...шестью лопастями, но с большей кривизной и углом атаки. К тому же, геометрия крыльчатки взаимосвязана со скоростью вращения ротора вентилятора. Обычно большие по диаметру вентиляторы имеют меньшую скорость вращения, а маленькие, наоборот, большую, что обеспечивает примерно равную их производительность. С точки зрения снижения шума, предпочтение следует отдавать тихоходным вентиляторам больших размеров. Однако это не всегда возможно ввиду конструктивных ограничений по габаритам.

Конструкция современного блока питания для компьютера не предусматривает возможности вскрытия и замены вентилятора на менее шумный. Но в некоторых моделях блоков питания, например, в Power Man FSP300-60GTB, в блоке питания в комплекте корпуса ASUS FK-320 и др., предусмотрен автоматический регулятор скорости вращения вентилятора, реагирующий на температуру выбрасываемого из системного блока воздуха. Блоки питания с изменяемой скоростью вращения вентилятора сравнительно мало шумят, если работают с нагрузкой, значительно меньшей максимально допустимой. Чтобы еще сильнее уменьшить шум от работы такого блока питания, можно попробовать установить дополнительный вентилятор с низкой скоростью вращения на предусмотренное конструкцией корпуса место. При этом улучшится отвод тепла из корпуса, и поэтому блок питания снизит скорость вращения своего вентилятора. Чтобы эффект от такой меры был более ощутимым, следует заменить перфорированную сетку в стенке корпуса, закрывающую дополнительный вентилятор, на проволочную решетку.

Дисбаланс крыльчатки вентилятора является одной из главных причин издаваемого им шума, особенно для современных процессорных кулеров, где скорость вращения достигает 7000 об/мин. Если в процессе производства вентилятор не был хорошо сбалансирован, и кулер сильно шумит сразу же после покупки, то единственно верным решением проблемы будет его замена. Дисбаланс крыльчатки вентилятора, проявившийся после нескольких месяцев эксплуатации, иногда можно ликвидировать тщательной очисткой лопастей, если причиной радиального биения крыльчатки стала неравномерно налипшая пыль.

Самые современные модели кулеров имеют блок управления частотой вращения вентилятора, например, Thermaltake Volcano 7+ и Elan Vital FCUG9C-6FC. Кулер Thermaltake Volcano 7+ предусматривает возможность выбора пользователем одного из трех режимов работы вентилятора. Для процессоров с относительно небольшим уровнем тепловыделения можно установить пониженную частоту вращения вентилятора и тем самым добиться снижения уровня шума. В кулере Elan Vital FCUG9C-6FC регулировка частоты вращения крыльчатки осуществляется автоматически, что позволяет снижать уровень шума в моменты, когда процессор не выполняет большого объема вычислений — эта технология получила название Whisper (в переводе с английского языка означает "шептун").

Радикальным методом снижения шума вентилятора процессорного кулера является упругое подвешивание его вентилятора над радиатором и создание герметичного канала для воздуха между крыльчаткой вентилятора и пластинами радиатора. Этот метод практически устраняет передачу колебаний на корпус компьютера через радиатор и материнскую плату, однако он сложен в реализации и поэтому интересен в основном лишь компьютерным энтузиастам.

Существуют решения, позволяющие уменьшить число вентиляторов в современном настольном компьютере или даже вовсе обойтись без них. Так, компания Zalman Tech выпускает кулеры с радиатором из меди (в некоторых модификациях — из меди и алюминия, а также позолоченной меди), внешне похожие на цветок. Такой радиатор обдувается малошумящим вентилятором с невысокой скоростью вращения. Однако, как показывает практика, подобный радиатор может быть использован вообще без вентилятора, например, для процессора Pentium III с тактовой частотой не более 700 МГц. Правда, большие размеры радиатора кулеров Zalman Tech не позволяют использовать эти кулеры совместно с некоторыми типами материнских плат и корпусов.

Процессоры VIA C3 могут обходиться

без вентилятора в кулере, достаточно лишь радиатора традиционной прямоугольной формы. Низкая потребляемая мощность этих процессоров позволяет отказаться от вентилятора в блоке питания. Таким образом, теоретически на базе VIA C3 можно создать компьютер без единого вентилятора. Но реально безвентиляторные блоки питания для настольного компьютера пока являются экзотикой, которая нечасто встречается в продаже. Обычный блок питания не может быть превращен в "безвентиляторный" просто отключением вентилятора.

Шум современных жестких дисков в режиме idle (ожидание без активности магнитных головок) невелик — около 30 дБА, что сопоставимо с шумовым фоном относительно тихого помещения. Основной источник шума в этом режиме — подшипниковый узел шпинделя. В режиме активности современные жесткие диски способны издавать шум с уровнем около 35 дБА. Можно ли этим пренебречь и считать, что шум жесткого диска незначителен? Будучи жестко закрепленным в корпусе компьютера, жесткий диск своими низкочастотными колебаниями с небольшой амплитудой (почти неслышимыми в обычных условиях) заставляет колебаться элементы корпуса. Большая поверхность корпуса компьютера становится своего рода усилителем шумов, издаваемых жестким диском.

Одним из методов снижения уровня шума связки "жесткий диск + корпус компьютера" является эластичный монтаж винчестера в корпусе. Иногда с этой целью применяются специальные упругие шайбы в точках крепления поверхности диска к корпусу компьютера, а порой производители просто используют прямоугольные упругие прокладки, которые применяются либо как подставка, либо как упругий подвес для винчестера. Однако если подобные прокладки не предусмотрены конструкцией жесткого диска, то их самостоятельное изготовление и последующее применение может быть небезопасным. Некоторые винчестеры в процессе работы очень сильно греются, и хороший тепловой контакт с корпусом компьютера является непременным условием надежной работы.

Ни трехдюймовый дисковод, ни привод CD-ROM не являются постоянно работающими устройствами. Однако если вы используете мультимедийные возможности современных компьютеров, то будете слушать музыку с CD или смотреть фильмы на DVD-носителях. Для обоих случаев от привода CD-ROM/DVD-ROM требуются невысокие скорости (обычно однократные) вращения диска, чего можно добиться, используя программные замедлители, например, Drivespeed 2000 или CDbremse. При снижении скорости вращения диска уменьшится и уровень акустического шума.

Что касается повышенных скоростей вращения, то они нужны для копирования дисков, инсталляции программ, поиска нужной информации, но никак не для просмотра фильмов. Если же вам часто приходится использовать дисководы на высоких скоростях, и шум их работы вас раздражает, его можно снизить установкой виброизолирующих прокладок.

Как уже отмечалось, большая поверхность корпуса является своеобразным усилителем акустических колебаний, возникающих внутри компьютера. Кроме того, корпус может вносить в эти колебания свои шумы, возникающие из-за вибрации его составных частей. Бороться с этим можно путем прокладывания мест крепления этих частей (чаще всего боковых крышек) поролоном. Еще одной проблемой является механический резонанс, усиливающий колебания частей корпуса. С этим можно бороться, приклеивая к стенкам корпуса дополнительные массы (например, кусочки металла), тем самым изменяя собственную частоту колебаний стенок корпуса. Иногда начинает "гудеть" поверхность стола — с этим можно бороться, положив под ножки системного блока резиновый коврик или пенопластовые пластины.

О результатах тестирования актуальных в наших условиях устройств — сетевых фильтров — рассказывает А.Бутрин в статье "Ловись, помеха, большая и маленькая..." **(ПЛ: компьютеры, №6/2002, С.66...72).**

Сетевой фильтр — это устройство, содержащее варисторный фильтр для подавления импульсных помех и режекторный фильтр для подавления высокочастотных помех. Наличием этих фильтров он отличается от сетевого удлинителя, хотя внешне они похожи. Нередко производители называют сетевым фильтром обыкновенный удлинитель.

Импульсная помеха — это кратковременный (10^{-6}...10^{-9} с) скачок напряжения в сети амплитудой более 4000...6000 В. Импульсные помехи вызываются природными и техногенными источниками. Природный источник импульсных помех — удары молний вблизи кабелей или линий электропередач. Техногенные причины — коммутационные процессы при включении или выключении мощной сетевой нагрузки, выбросы тока при включении/выключении напряжения в сети, аварии на подстанциях. Максимальные величины напряжения коммутационых импульсов даже в бытовых сетях могут достигать 20 кВ.

Для подавления таких помех применяют варисторные фильтры. Варистор — это полупроводниковый резистор, сопротивление которого нелинейно зависит от приложенного напряжения — чем выше напряжение, тем ниже сопротивление варистора. Последний включается параллельно защищаемому объекту. В рабочем режиме ток, текущий через варистор,

Табл. 1

Модель	Режекторный ВЧ-фильтр	Количество варисторов	Защита телефонной линии	Дополнительная индикация	Цена, USD
APC Surge Protector P5T-RS	Есть	3 (2)	Есть	-	21
APC Surge Arrest E20-G	Есть	6	Нет	Земля	32
Defender Surge Protector Advance	Нет	1	Нет	-	5
Defender Surge Protector Phone/Fax/Modem	Есть	3 (2)	Есть	Защита	12
Defender GA01R	Нет	3 (2)	Есть	-	7
On Tech Universal	Нет	1	Нет	-	5
Pilot Pro	Есть	4	Нет	Защита	21
Pilot SPA	Нет	1	Нет	-	26
Power Cube	Нет	1	Нет	-	5
Silvershield Advanced Surge Protector	Есть	3	Нет	Защита, земля	16
Silvershield Advanced Surge Protector (tel)	Есть	3 (2)	Есть	Защита, земля	18
Sven Classic	Нет	1	Нет	-	6
Sven Gold	Есть	3 (2)	Есть	Защита	10
Sven Optima	Нет	1	Нет	-	5
Sven Silver	Есть	3	Нет	Защита	7
Sven Special	Нет	0	Нет	-	6
Vector MAX	Есть	4	Нет	Защита	17
Vector Solo	Есть	3	Нет	Защита	13

В скобках указано число варисторов для защиты телефонной линии

чрезвычайно мал, поэтому варистор в этих условиях представляет собой изолятор. При возникновении импульса высокого напряжения сопротивление резко падает, благодаря чему варистор защищает нагрузку, рассеивая энергию импульса в виде тепла. В этот момент через варистор может протекать ток силой в несколько тысяч ампер. После гашения импульса сопротивление варистора вновь становится очень большим.

Для полноценной защиты устройств в двухпроводной сети с защитным занулением варисторы должны быть включены между фазой и нулем, фазой и защитным занулением, а также нулем и защитным занулением. Такая схема включения называется варисторным треугольником.

Высокочастотная помеха — неопределенный по амплитуде и длительности сигнал в диапазоне 100 Гц...100 МГц, который искажает синусоиду стандартного сетевого напряжения и, таким образом, негативно влияет на работу любого электрооборудования. Источниками ВЧ-помех являются электродвигатели, генераторы, сварочные аппараты и т.п.

Для подавления ВЧ-помех применяют режекторные фильтры, содержащие катушки индуктивности (дроссели) и конденсаторы.

Тестировались фильтры APC Surge Protector P5T-RS, APC Surge Arrest E20-G, Defender Surge Protector Advance, Defender Surge Protector Phone/Fax/Modem, Defender GA01R, On Tech Universal, Pilot Pro, Pilot SPA, Power Cube, Silvershield Advanced Surge Protector, Silvershield Advanced Surge Protector (tel), Sven Classic, Sven Gold, Sven Optima, Sven Silver, Sven Special, Vector MAX, Vector Solo. Технические характеристики, заявленные производителями, приведены в **табл.1**, результаты измерений режекторных характеристик — в **табл.2**, оценки редакции "ПЛ: компьютеры" — **в табл.3**.

Подводя итоги, автор отмечает, что награду "Выбор редакции" получил самый достойный продукт — Silvershield Advanced Surge Protector с защитой телефонной линии. В нем сочетается максимальное количество полезных для пользователя функций — продвинутая индикация, серьезная защита телефонной линии и продуманная схема защиты от импульсных помех. Хорошие результаты показал этот фильтр и в тесте на подавление ВЧ-помех.

Табл. 2

Модель	Ширина полосы режекции, кГц	Глубина режекции, дБ	Центральная частота, кГц
APC Surge Protector P5T-RS	1700	29,28	500
APC Surge Arrest E20-G	50	31,41	40
Defender Surge Protector Advance	-	-	-
Defender Surge Protector Phone/Fax/Modem	2000	33,20	600
Defender GA01R	-	-	-
On Tech Universal	-	-	-
Pilot Pro	80 (600)	33,71 (26,24)	60 (2000)
Pilot SPA	-	-	-
Power Cube	-	-	-
Silvershield Advanced Surge Protector	1890	36,48	100
Silvershield Advanced Surge Protector (tel)	1890	35,14	100
Sven Classic	-	-	-
Sven Gold	1310	31,41	400
Sven Optima	-	-	-
Sven Silver	1210	30,64	400
Sven Special	-	-	-
Vector MAX	1590	37,23	100
Vector Solo	1510	34,54	400

Для фильтра Pilot Pro приведены данные для 2 диапазонов — 10...100 кГц (100...2100 кГц)

Табл. 3

Модель	Конструкция	Функциональность	Ширина полосы режекции	Глубина режекции	Итоговая оценка
APC Surge Protector P5T-RS	3	2	4	2	3
APC Surge Arrest E20-G	5	5	1	3	4
Defender Surge Protector Advance	1	2	1	1	1
Defender Surge Protector Phone/Fax/Modem	3	4	5	4	4
Defender GA01R	3	1	1	1	2
On Tech Universal	2	2	1	1	2
Pilot Pro	5	4	2	2	3
Pilot SPA	3	2	1	1	2
Power Cube	1	2	1	1	1
Silvershield Advanced Surge Protector	4	4	5	5	5
Silvershield Advanced Surge Protector (tel)	4	5	5	4	5
Sven Classic	1	2	1	1	1
Sven Gold	3	4	4	3	4
Sven Optima	1	2	1	1	1
Sven Silver	3	3	3	3	3
Sven Special	-	2	1	1	1
Vector MAX	5	3	4	5	4
Vector Solo	3	3	4	4	4

В номинации "Лучшая покупка" победа присуждена фильтру Defender Surge Protector Phone/Fax/Modem. Несмотря на низкое качество сборки, он прекрасно защищает технику, о чем свидетельствуют результаты измерений. Значок "За техническое совершенство" был отдан APC Surge Arrest E20-G. И не без оснований: рекордное число варисторов, огромные дроссели, идеальная пайка, надежная и продуманная конструкция — вот факторы, благодаря которым это изделие признано лучшим. Что касается устройств Defender Surge Protector Advance, On Tech Universal, Power Cube, Sven Classic, Sven Optima и Sven Special, то сетевыми фильтрами они не являются.

AMD, следом за Intel, готовится выпустить на рынок **64-разрядный процессор**, который она назвала Hammer. Краткие сведения о нем приводятся в заметке "Молотом по наковальне" **(CHIP, №5/2002, С.26)**.

Его особенностью является совместимость с 32-битными приложениями, что делает Hammer (в отличие от Itanium) интересным для всех пользователей. На выставке CeBIT он прекрасно, без всяких "падений", работал как с 32-битной Windows XP, так и с 64-битной версией Linux. Представленный уменьшенный вариант Hammer под названием Clawhammer имеет 754 контакта и, соответственно, подходит к новому Socket 754.

Его дополняет чипсет AMD 8000

(Stepping AO) с поддержкой шины HyperTransport, который установлен на материнскую плату под названием "Solo" производства самой AMD. Порт AGP 3.0 был оснащен видеокартой Radeon 8500.

Официальных сведений относительно тактовой частоты Hammer AMD сообщила немного, но пообещала значительный прирост производительности при работе как с 32-битными, так и с 64-битными приложениями — при одинаковых тактовых частотах Clawhammer должен работать примерно на 25% быстрее, чем Athlon XP. Чтобы достичь обещанного рывка в производительности, процессоры Hammer должны будут уметь обрабатывать значительно большее количество инструкций за один такт. Таким образом, AMD твердо придерживается своей линии на повышение производительности не только за счет наращивания тактовой частоты, но и за счет совершенствования архитектуры самого процессора.

Так, "старший брат" по имени Sledgehammer имеет 940 контактов, которые предназначены для шины HyperTransport, а также для поддержки 128-битной (у Clawhammer — 64-битной) оперативной памяти DDR SDRAM. Процессоры Hammer изготавливаются по SOI-технологии (Silicon on Insulator, кремний на изоляторе) и имеют 0,13-микронную структуру. Их производство будет налажено в Дрездене (Германия) с конца 2002 года.

Процессоры Hammer могут исполнять 32-битные приложения, а если запрашиваются новые регистры, то и 64-битные программы тоже будут работать с ними. Тем самым AMD рекламирует свои новые процессоры как универсальный инструмент для работы как с сегодняшними программами, так и с перспективными приложениями.

Процессоры Hammer отличаются от существующих процессоров наличием расширений x86-64. Тем самым AMD пошла "другим путем", нежели Intel, предлагавшая или 32-битную (у процессоров Pentium), или 64-битную архитектуру (Itanium).

Универсальность является преимуществом новых процессоров — Hammer позволит пользователям расслабиться и спокойно ожидать появления новых 64-битных приложений. По слухам, Intel, озабоченная возможностями новой концепции, срочно взялась за разработку 32/64-битного процессора под названием Yamhill. Ожидается, что он будет работать даже с расширениями AMD x86-64, которые Intel могла бы приобрести в обмен на свои расширения SSE2.

Со времени появления 486-х процессоров, необходимым атрибутом компьютера является кулер процессора с шумным вентилятором. Процессор греется как утюг, растрачивая впустую львиную долю подводимой энергии. **Возможен ли "холодный" процессор?** Этот вопрос обсуждается в заметке "Процессор, остынь" **(CHIP, №5/2002, С.6).**

Специалистам из Intel удалось создать два экспериментальных процессора, достигших гигагерцовой частоты и потребляющих на 70% меньше энергии, чем ныне существующие.

На прошедшей в этом году в Сан-Франциско Международной конференции по интегральным схемам Intel продемонстрировала один из двух "экономичных" процессоров. Его транзисторы при переключении потребляют на 23% меньше тока, чем приборы в обычных процессорах. Еще более впечатляющим выглядит уменьшение так называемых токов утечки (токов, текущих через закрытый транзистор). Потери здесь сокращены на 71%. Все это значит, что данные экономичные процессоры, созданные по 0,13-мкм технологии, способны работать на тактовых частотах 5...10 ГГц при комнатной температуре и без всякого охлаждения. Новые технологии уже постепенно внедряются в производство. В будущем, создав структуры размером 0,065 мкм, Intel планирует полностью перейти на технологию "холодных" транзисторов. Новый PowerMac производства Apple уже сегодня использует кое-что из этой технологии. Не отстают и AMD с IBM, которые заявляют о своих намерениях в ближайшем будущем значительно уменьшить токи утечки.

О новом процессоре AMD сообщается в статье "Плюс перевалил за 2000, частота — еще нет" **(Hard'n'Soft, №5/2002, С.26...28).**

Самые активные участники процессорной гонки — Intel и AMD — не дают друг другу почивать на лаврах, продолжая гонку скоростей и цен. Пользователям, конечно, относительное равновесие очень на руку, поскольку именно в такой ситуации производителям приходится пускать в ход последний аргумент — снижение цены. Поэтому сейчас, при примерном равенстве возможностей Athlon XP и Pentium 4, самые современные и скоростные модели дешевы как никогда.

Тестировался процессор AMD Athlon XP 2100+, выполненный на основе процессорного ядра Palomino (0,18 микрон). В отличие от процессоров Intel, чипы AMD маркируются не в соответствии с реальной тактовой частотой, а согласно принятым производителем единицам измерения быстродействия. Так, например, Athlon XP 1700+ работает на частоте всего 1466 МГц, а самый быстрый Athlon XP 2100+ имеет тактовую частоту 1733 МГц. Получается, что процессоры AMD до сих пор не перешагнули через двухгигагерцовый рубеж, хотя маркировка уже вовсю преодолевает промежуточные пункты внутри третьего гигагерца. Но, как бы то ни было, в гонке частот, даже с учетом "опережающих" значений маркировки, AMD пока заметно проигрывает Intel. Последняя уже предлагает свои Pentium 4 с частотой 2,4 ГГц, и эта частота — реальная.

Несмотря на то что мегагерц у Pentium 4 больше, с точки зрения реальной производительности позиции Athlon XP на редкость прочны. Процессоры AMD и Intel сравнивались при помощи PCMark2002, 3DMark2001 SE, ZD Business Winstone 2001 и ZD Content Creation Winstone 2001. В тесте, кроме упомянутого Athlon XP 2100+, участвовали Athlon XP 1700+, Athlon XP 2000+ и Pentium 4 2ГГц. Результаты получились очень близкие, причем двухгигагерцовый Pentium 4 с ядром Northwood по своей производительности сопоставим лишь с Athlon XP 1700+.

Справедливости ради надо заметить, что тестовые утилиты — синтетические, построены на основе коммерческих программ и не оптимизированы для работы с Pentium 4. Да и чипсет i845 (при проведении тестов использовалась материнская плата AOpen AX4B Pro) не поддерживает память DDR333 (только DDR200 и DDR266), в то время как Soltek SL-75DRV5 для процессоров AMD построена на системной логике VIA Apollo KT333 и такой поддержкой обладает.

Разумеется, Pentium 4, работая с более быстрыми памятью (некоторые чипсеты, в частности, VIA Apollo P4X333, ALi ALADDIN-P4 и SiS645, уже предоставляют такую возможность) и системной шиной, наверняка сможет значительно улучшить свои показатели производительности. Будет ли он при этом столь же "демократичным" по цене, как процессоры AMD? Вероятно, нет. Да и говоря о перспективах, надо учитывать появление в ближайшем будущем Athlon XP с ядром Thoroughbred, эти процессоры должны быть еще более производительными. Насколько велик будет прирост скорости и будет ли он, станет ясно уже совсем скоро. Естественно, Intel тоже не собирается стоять на месте, так что противостояние продолжится, и оно обещает быть крайне интересным.

DL vs. DL: две различные концепции удаленного обучения

А.КАН,
E-mail: anton_kan@tut.by

Со времени возникновения всемирной сети Интернет она всегда так или иначе использовалась и в образовательных целях. Вначале она просто представляла собой альтернативный способ связи между университетами, позже в сети начали появляться учебники и справочная литература. Наконец, возникли целые системы дистанционного обучения (Distance Learning, DL), позволяющие не только получить информацию, но и проверить, насколько хорошо она усвоена, сравнить свои результаты с результатами других студентов, пообщаться с преподавателями и многое другое.

Вопреки распространенному мнению, идея образования через Интернет достаточно популярна и в странах бывшего СНГ, в частности, в Беларуси. Сегодня мы попытаемся сравнить две системы, сделанные в Беларуси.

Первая — это Distance Learning Belarus (*http://dl.gsu.unibel.by*), созданная и поддерживаемая силами сотрудников, преподавателей, аспирантов и студентов Гомельского государственного университета им. Ф.Скорины (*http://www.gsu.unibel.by*). Система функционирует с 1999 г., за это время в ней зарегистрировалось около 4000 участников из 40 стран мира. По словам авторов ресурса, это первый в РБ сайт дистанционного обучения. На данный момент все сервисы, предоставляемые системой, совершенно бесплатны и доступны любому пользователю сети Интернет. Система имеет русскоязычный и англоязычный интерфейсы.

Вторая тоже носит название Distance Learning, но создана лишь в этом году двумя школьниками из Минска — Антоном Ярошуком (*anton@artwebgroup.com*) из средней школы №196 г.Минска и Егором Шитиковым (*egor@artwebgroup.com*) из МГВРК, которые вместе занимаются на курсах Минского дворца детей и молодежи. Впервые была представлена на 7-м республиканском конкурсе исследовательских работ школьников, где получила диплом первой степени и отдельную премию Intel. На момент написания этой статьи система еще не была запущена, поэтому она интересна прежде всего собственно самим подходом к проблеме обучения через Интернет. Эта система не предназначена для всех желающих, имеющих доступ в Интернет. Планируется, что каждый ВУЗ сможет получить свой домен, настроить систему под себя, заполнить ее своим информационным содержанием и использовать для обучения своих студентов. Естественно, каждому из ВУЗов такая услуга будет предоставляться за определенную плату. Система состоит из трех центров: студенческого, преподавательского и административного. Концепция системы схожа с российским "Прометеем" (*http://www.prometeus.ru*), который уже довольно успешно используется в БГУИР. Для теста была предоставлена пробная версия студенческого центра без информационного содержания (т.е. собственно того, чему можно было бы научиться). Как сказали разработчики системы, административный и преподавательский центры они пока не могут предоставить для теста, поскольку они еще не закачаны и не установлены на сервере.

Рис. 1

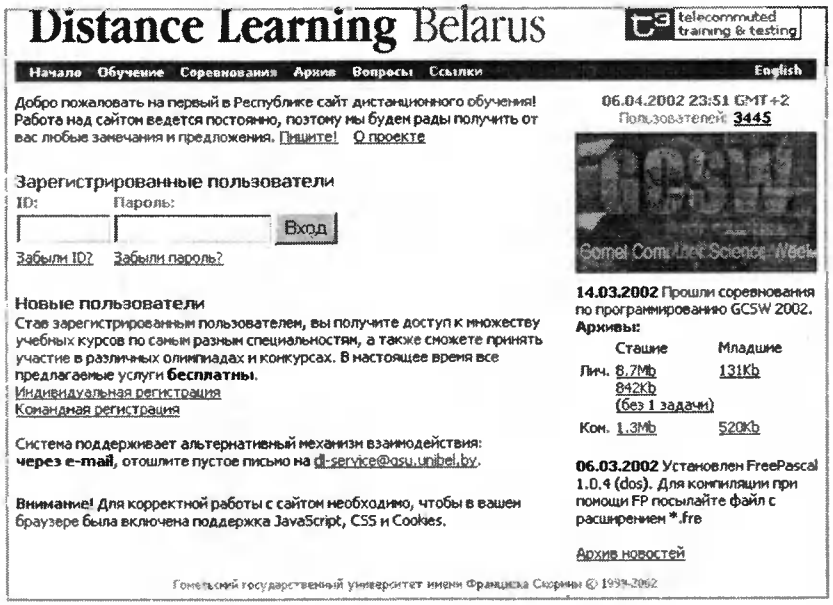
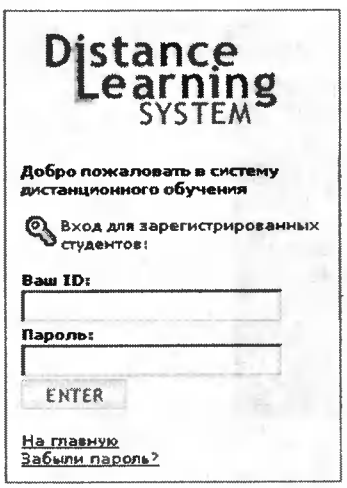


Рис. 2

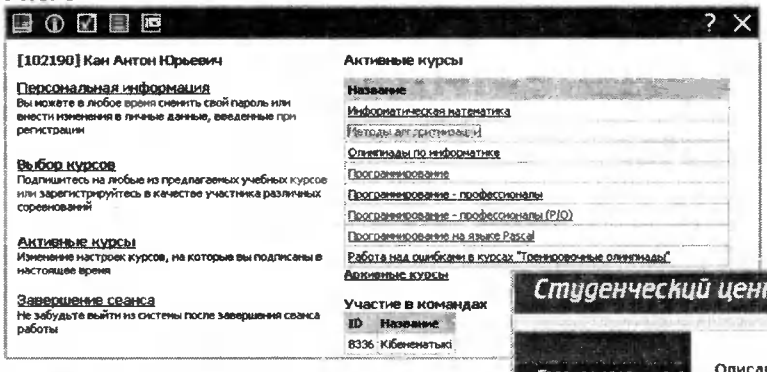


Итак, начнем с начала, т.е. со стартовой страницы каждого сайта. Сразу бросается в глаза некоторая схожесть. Заголовки отличаются только словами "Belarus" у гомельской системы (**рис.1**) и "System" у системы, сделанной минскими школьниками (**рис.2**). Оба сайта просят ввести логин и пароль для входа в систему, а также предлагают помощь для тех, кто забыл свой пароль. Отличие состоит в том, что система Антона и Егора не предлагает регистрацию новых пользователей (поскольку по замыслу доступ к ней будет предоставлен только ограниченному кругу лиц). Кроме того, на странице DL Belarus присутствует информация о системе, новости и описание работы с почтовым роботом (в отличие от своего сегодняшнего "соперника", система поддерживает альтернативный механизм взаимодействия через E-mail). Впрочем, новости сервера также есть и во второй системе, но они вынесены в отдельный раздел, который становится доступен после входа в систему.

Еще одним отличием является то, что зайти в гомельскую систему можно как под логином пользователя, так и под логином команды (команды создаются пользователями для участия в соревнованиях. Предполагается, что пароль команды известен каждому из пользователей). Во второй системе каждый студент обязательно принадлежит к некоторой группе, поэтому она определяется автоматически, в то время как студенты заходят под своими именами по отдельности. Немаловажно и то, что у гомельской системы есть англоязычный интерфейс.

Итак, имена и пароли вспомнены и введены, и мы попадаем на главные страницы сайтов. Для гомельской системы это

Рис. 3



еще только "Рабочий стол" пользователя, где он может просмотреть и изменить свою личную информацию, а главное — выбрать курс и подписаться на новые курсы (**рис.3**). Курсы — основа всей системы. Каждый курс состоит из задач и таблицы результатов, в которой учитываются только пользователи, подписанные на этот курс. В некоторых курсах также содержатся теоретические материалы. Особой разновидностью курсов является олимпиада, которая отличается тем, что задачи доступны ограниченное время, причем в это время недоступны результаты. После выбора курсов открывается второй "Рабочий стол", содержащий такие пункты как "Теория", "Задачи", "Протокол", "Результаты" и "Консультации". Кроме того, в верхней части окна есть кнопки "Помощь" и "Почта DL". В минской системе после ввода логина и пароля сразу открывается доступ ко всем разделам сайта: "Новости курса", "Новости сервера", "Консультации", "Ваша группа", "Учебные материалы", "Тестирование", "NetMeeting", "Форум группы", "Помощь/FAQ" (**рис.4**). NetMeeting является интересной и уникальной особенностью именно этой системы. В разделе показаны все студенты и преподаватели, которые активны в данный момент. Щелкнув по ссылке, можно связаться с ними при помощи одноименной программы от Microsoft. К сожалению, эту возможность протестировать не удалось, поскольку система еще не была запущена, и активных пользователей не оказалось. Сразу бросилась в глаза бедность раздела "Помощь/FAQ". В отличие от аналогичного сервиса на Distance Learning Belarus, в нем оказался всего один вопрос — "Какие системные требования необходимы для работы с сайтом?". Впрочем, это нельзя отнести к недостаткам, поскольку система на момент теста еще не была запущена, и сайт просто не был наполнен текстовой информацией. Разделы "Почта" в обеих системах практически идентичны. Тут можно отправить сообщение любому зарегистрированному пользователю (в минской системе — еще и преподавателю или администратору сайта). Поскольку в Distance Learning System не предусмотрено отдельной процедуры входа для команды (вернее, группы), есть пункт меню "Ваша группа", где можно ознакомиться с составом группы, в которой вы учитесь.

Теперь пора приступить собственно к обучению. Начнем с минской Distance Learning System. В разделе "Учебные материалы" можно ознакомиться с материалами по выбранному курсу, которые отправляются пользователю на E-mail или скачиваются им на жесткий диск в виде архива. В разделе "Консультации" представлена более краткая информация по более конкретным вопросам. Видимо, для общения с преподавателями предназначены разделы "Почта" и "NetMeeting". После получения теоретической базы (надо признаться, что на тестовой версии сайта ей размеры невелики — есть только

Рис. 4

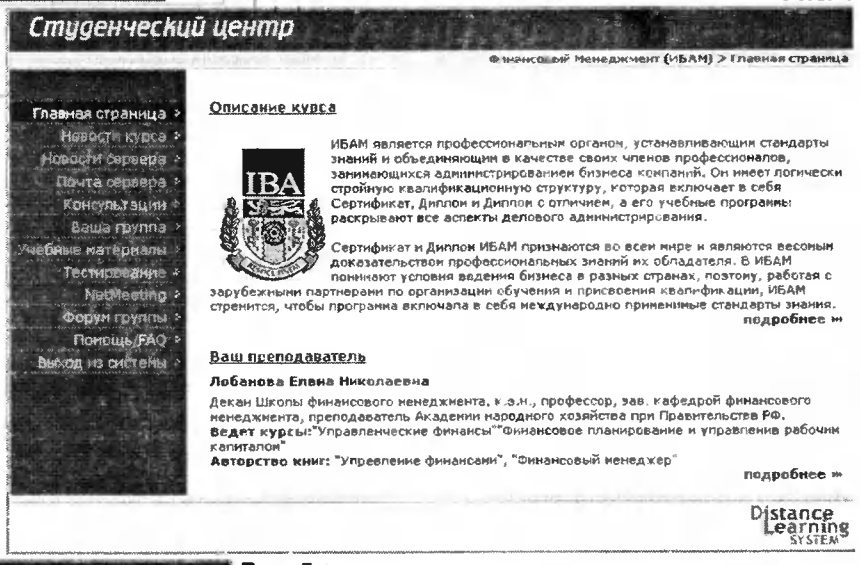


Рис. 5

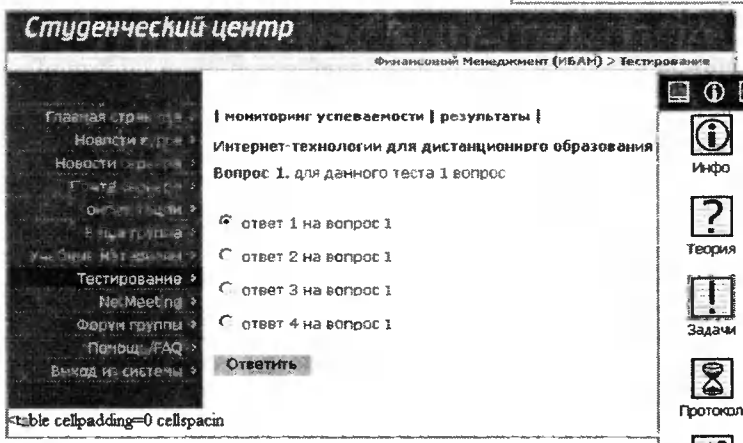
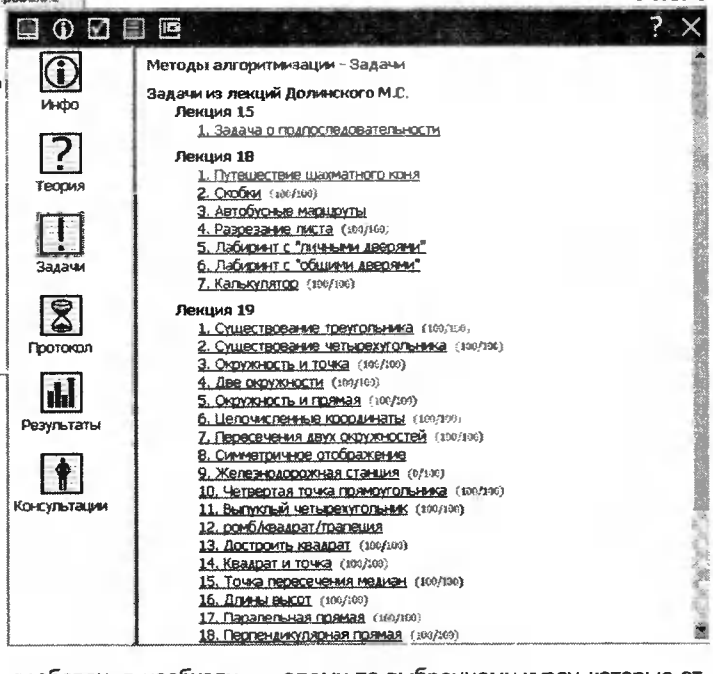


Рис. 6



некоторая информация по финансам и бизнесу, финансовому менеджменту и технологиям дистанционного обучения) можно приступить к тестированию (**рис.5**). К сожалению, в данной версии были представлены только два набора тестов, один из которых был заблокирован, а второй имел вид "Вопрос 1. Вариант 1. Вариант 2. Вариант 3.", что, впрочем, не помешало мне довольно успешно его пройти. Приятно, что разработчики исключили возможность двойного нажатия на кнопку "Ответить" — после ответа она автоматически становится недоступной. После прохождения тестов можно посмотреть свои результаты (какие тесты пройдены, сколько было правильных ответов, сколько набрано баллов) и сравнить их с результатами группы. Кроме того, существует раздел "Мониторинг", где можно проследить свою успеваемость и сравнить ее не только со средней по группе, но и с любым студентом-одногруппником.

Теперь возьмемся за "взрослую" гомельскую Distance Learning Belarus. Для тестирования был выбран курс "Методы алгоритмизации". В разделе "Инфо" можно ознакомиться с информацией о курсе: когда он был создан, когда будет закрыт (в основном, закрываются только олимпиады), сколько участников подписалось на него, краткое описание самого курса. В разделе "Теория" содержатся лекции (как в полном варианте, так и в виде конспекта) М.С.Долинского, доцента ГГУ, которые касаются не только методов решения задач по информатике, но и основ языка Паскаль (**рис.6**). В принципе, даже неподготовленный человек после прочтения лекций может начать решать олимпиадные задачи. Лекции не предлагаются для отправки на E-mail или скачивания, а выложены прямо на сайте (загрузки страницы не приходится ждать слишком долго, поскольку каждая лекция находится на отдельной странице, и существуют краткие варианты), что, на мой взгляд, удобнее. В разделе "Задачи" — сами задачи по информатике. После выбора задачи открывается окно с ее текстом и формой для отправки решения. Решения отправляются в виде программы (система поддерживает C++, Pascal, Assembler, Basic и FreePascal) или файла с ответами (для некоторых задач). Впрочем, есть курсы (например, "Информатическая математика"), на которых нужно просто ввести полученный ответ в форму (что уже больше напоминает Distance Learning System), а также курсы по шахматам, где ходы указываются пользователем с помощью мыши. Результаты тестирования задачи можно просмотреть в разделе "Протокол", причем там будет указано, какие именно тесты не прошли и по какой причине. В прошлом году была введена возможность просмотра содержимого самих тестов. В разделе "Результаты" можно увидеть таблицу результатов для всех участников, подписавшихся на данный курс, или только некоторой их части. В разделе "Консультации" предлагается форма написания письма в службу поддержки (в отличие от раздела "Консультации" минской системы). Совсем недавно появился полезный раздел "Форум", в котором можно обсудить каждую из задач. Его отсутствие до последнего времени немного огорчало, в то время как во второй системе он существовал изначально.

В заключение хотелось бы сказать, что этим тестом я не пытался показать, что одна система хуже или лучше другой, поскольку они принципиально различаются как минимум по двум причинам.

Во-первых, Distance Learning System, сделанная минскими школьниками, предоставляет возможность обучения студентам одного ВУЗа, а гомельская Distance Learning Belarus позволяет обучаться любому желающему, имеющему доступ в Интернет.

Во-вторых, первая позволяет тестировать теоретические знания, в то время как вторая предназначена в основном для решения задач. К сожалению, на момент написания статьи полнофункционально работала только вторая система.

Литература

1.7-й республиканский конкурс исследовательских работ школьников. — Компьютерные Вести, 2002, №12.

Инструментальная web-система автоматизации процессов дистанционного обучения (ИСДО) эксплуатируется в ГГУ с осени 1999 г. На базе данной системы в ГГУ запущен проект "Дистанционное обучение в Беларуси" (Distance Learning Belarus — DLB): http://dl.gsu.unibel.by. В настоящее время в проекте более 3000 зарегистрированных пользователей из 40 стран мира. ИСДО обеспечивает многоязычный интерфейс с пользователем (в настоящее время поддерживаются английский и русский языки) и работает по стандарту "24•7", то есть 24 часа в сутки, семь дней в неделю.

Самые совершенные из аналогов позволяют, используя web-сайт и/или электронную почту, выбрать учебный курс, получить теорию и контрольные задания. Далее, изучив теорию, выполнить и прислать обратно решения. Поддерживается также простейший тестовый контроль вида: серия вопросов, для каждого из которых требуется выбрать правильный ответ из нескольких предложенных или ввести правильный ответ (с проверкой на точное совпадение). Во всех остальных случаях проверкой должны заниматься преподаватели курсов, вручную обеспечивая прием решений, их проверку, отсылку ученику результатов проверки, подведение и анализ статистических итогов.

В дополнение к тому, что обеспечивают лучшие из аналогов, ИСДО позволяет автоматизировать процессы проверки полученных решений, отсылки их обучаемому, накопления результатов и анализа процесса обучения.

Интерфейс подключения программ проверки решений унифицирован, что обеспечивает возможность использования нашей системы дистанционного образования для эффективного обучения практически в любой предметной области. В частности, в текущий момент обеспечивается автоматическая проверка для:

- программ для ПК на языках программирования Паскаль, C/C++, Ассемблер, Perl, Basic;
- решений шахматных задач (обучаемый видит картинку шахматной позиции, перемещением с помощью мыши фигур с исходной позиции на конечную указывает ходы, которые он считает правильными);
- проектов цифровых систем, созданных на языке VHDL и или в системе HLCCAD;
- ассемблерных программ для микроконтроллеров фирм Intel, Atmel, Motorola, Microchip и Texas Instruments, проверяемых в специализированных системах симуляции.

Обеспечена возможность замены проверяющей программы человеком в цепочке автоматизированной обработки. Таким образом, на первых порах принципиально обеспечивается чтение текстов, наблюдение рисунков, прослушивание ответов и просмотр видео, с целью "анализа работы и выставления оценки человеком". Все остальные операции по приему решений, занесению оценок и комментариев в базу данных, отсылке результатов проверки обучаемому выполняются автоматически.

Накопление, визуализация и статистический анализ результатов проверок в базах данных ведется с использованием SQL-технологий.

Долинский М.С., научный руководитель Distance Learning Belarus,
E-maill: dolinsky@gsu.unibel.by,
http://www.gsu.unibel.by/people/dolinsky

Представленная система дистанционного образования (СДО) является законченным программным продуктом, выполненным на высоком профессиональном уровне (с использованием языка программирования PHP4.0.5 и реляционной БД MySQL 3.23).

Основное, что необходимо отметить в представленной авторами СДО, это комплексный подход к построению системы и, как следствие, логическую продуманность и завершенность структуры, что позволило создать программный продукт, обладающий рядом достоинств:

- простота использования (нет необходимости в специальной подготовке);
- широчайшие возможности по общению между студентами и преподавателями (почта, конференции, консультации, чат, видео/аудио);
- возможность контроля учебного процесса на любой стадии со стороны преподавателя;
- гибкая система самотестирования и мониторинга, централизованная база данных, позволяющая анализировать и корректировать процесс обучения.

Сферы использования данной СДО в ВУЗах, колледжах и др. учебных заведениях: дистанционные подготовительные курсы для абитуриентов; дистанционное обучение и консультирование студентов; самостоятельная работа студентов дневного обучения; сетевое тестирование; преподавание за отдельную плату дополнительных дисциплин по выбору студента (помимо официально утвержденной программы обучения); последипломное обучение (дистанционные курсы повышения квалификации); привлечение преподавателей со стороны (из других ВУЗов, городов и стран).

А.Ярошук, один из разработчиков Distance Learning System,
E-mail:anton@artwebgroup.com

М.ДОЛИНСКИЙ,
г.Гомель.

Решение задач на графах

(Окончание. Начало в №№7-10/2002)

5. Поиск в ширину

5.1. Задача "Уличная гонка"

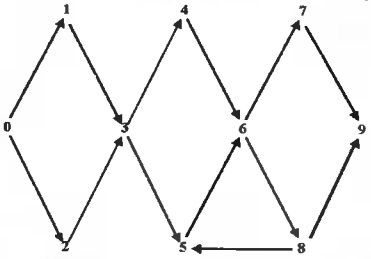
(*IOI'95 — Международная олимпиада по информатике, Голландия*)

План уличной гонки состоит из точек, помеченных числами от 0 до N (где N=9), а также стрелок, соединяющих их. Точка 0 является стартовой; точка N — финишной. Стрелками обозначены улицы с односторонним движением. Участники гонки передвигаются от точки к точке по улицам только в направлении стрелок. В каждой точке участник гонки может выбрать любую из исходящих стрелок.

План улиц является "хорошим", если он обладает следующими свойствами:

- каждая точка плана может быть достигнута со старта;
- финиш может быть достигнут из любой точки плана;
- из финишной точки нет исходящих стрелок.

На рисунке изображен пример плана улиц для гонки.



Для достижения финиша участник не обязательно должен пройти через все точки. Однако некоторые точки невозможно обойти. Назовем их неизбежными. В примере такими точками являются точки 0, 3, 6, 9. Для заданного хорошего плана ваша программа должна определить множество неизбежных точек (за исключением стартовой и финишной), которые должны посетить все участники (подзадача A).

Предположим, что гонка должна проводиться за два последовательных дня. Для этой цели план должен быть разбит на два "хороших" плана — по одному на каждый день. В первый день стартом является точка 0, а финишем служит некая "точка разбиения". Во второй день старт находится в этой точке разбиения, а финиш находится в точке N. Для заданного "хорошего" плана ваша программа должна определить множество всех возможных точек разбиения (подзадача B).

Точка S является точкой разбиения для "хорошего" плана C, если S не является стартовой и финишной точками плана, и если план разбивается ею на два "хороших" плана без общих стрелок и с единственной общей точкой S. Для приведенного примера точкой разбиения является точка 3.

Входные данные

В файле INPUT.TXT описан "хороший" план, содержащий не более 50 точек и не более 100 стрелок. В файле имеется N+1 строка. Первые N строк содержат конечные точки стрелок, исходящих соответственно из точек от 0 до N-1. Каждая из этих строк заканчивается числом -2. В последней строке содержится число -1.

Выходные данные

Ваша программа должна записать две строки в файл OUTPUT.TXT. Первая строка должна содержать количество неизбежных точек в заданном плане, после чего в той же строке должны следовать номера этих точек в любом порядке (подзадача A). Вторая строка должна содержать количество точек в заданном плане, за которым в той же строке должны следовать номера этих точек в любом порядке (подзадача B).

Пример ввода и вывода

INPUT.TXT
```
1 2 -2
3 -2
3 -2
5 4 -2
6 4 -2
6 -2
7 8 -2
9 -2
5 9 -2
-1
```

OUTPUT.TXT
```
2 3 6
1 3
```

Для решения подзадачи A нужно найти все неизбежные точки. Это делается поиском в ширину — если после взятия вершины из очереди очередь становится пустой, то эта вершина — *неизбежная*.

Для решения подзадачи B нужно проверить каждую *неизбежную* точку, является ли она точкой разбиения. Для этого удаляем эту точку из графа, и поиском в ширину строим два множества — достижимых точек от начала и недостижимых точек от начала.

Если существует хоть одно ребро из недостижимой точки в достижимую — то текущая *неизбежная* точка не является возможной точкой разбиения (по определению точки разбиения).

Рассмотрим алгоритм решения задачи более подробно, основываясь на исходном тексте программы, решающей поставленную задачу:

Тело главной программы выглядит следующим образом:

```
InputData;                                  {Ввод исходных данных}
BFS;          {Поиск в ширину - находим "неизбежные" точки}
OutSubA;                          {Вывод ответа для подзадачи A}
for k:=1 to SubA do                {Для каждой неизбежной точки}
  begin
    DelArcs(Duty[k]);                  {Удаляем неизбежную точку}
    BFS;                                          {Поиск в ширину -}
                  {Строим множества достижимых Color[i]=1}
                          {и недостижимых Color[i]=0 вершин}
    RetArcs(Duty[k]); {Возвращаем неизбежную точку в граф}
    for i:=0 to N-1 do                            {Для всех веришин}
      begin
          j:=1;      {j - номер дуги, исходящей из вершины}
        while (a[i,j]>=0) do          {пока дуга существует}
          begin
            if (color[i]=WHITE} and
                                {Если исходная вершина достижима,}
                (color[a[i,j]]<>WHITE)
                                  {а "конец дуги" - не достижима}
                then begin
                    Duty[k]:=0; {Это - НЕ точка разбиения}
                    break    {Перейти к проверке следующей}
                     end;                        {неизбежной точки}
            inc(j);                          {Взять следующую дугу}
          end;
      end;
  end;
OutSubB;                          {Вывод ответа для подзадачи B}
```

Процедура ввода исходных данных:

```
Procedure Inputdata;
  begin
   i:=0; j:=1; read(a[0,1]);          {Начинаем ввод с вершины 0}
   while (a[i,j]<>-1) do  {Пока не закончена обработка вершин}
      begin
        while(a[i,j]<>-2) do {Пока не закончена обработка дуг}
          begin
            inc(j);                              {Увеличить номер дуги}
            read(a[i,j])                                 {Прочитать дугу}
          end;
        if j >2 then Sort;       {Если дуг больше чем одна, то}
                                 {отсортировать их по возрастанию}
                                          {номера конечной вершины}
        readln; inc(i); {Перейти к считыванию следующей строки}
        j:=1; read(a[i,j])  {Прочитать первый элемент строки}
      end;                              {Вернуться к началу цикла ПОКА}
   N:=i;                      {Занести в N номер последней вершины}
  end;
```

Поиск в ширину осуществляется с помощью процедуры BFS (Breadth-First Search):

```
procedure BFS;
var   i,j : longint;
begin
  for i:=0 to N do color[i]:=WHITE;       {Все вершины свободны}
  color[0]:=GRAY;                 {Начальная вершина - обработана}
```

```
  SubA:=0;                    {Количество неизбежных вершин = 0}
  BegQ:=1;                                        {Начало очереди}
  EndQ:=0;                                        {Очередь пуста}
  Put(0);              {Поместить в очередь начальную вершину}
  while (BegQ<=EndQ) do                  {Пока очередь не пуста}
     begin
       Get(i);                          {Взять вершину i из очереди}
       if BegQ>EndQ                  {Если очередь осталась пустой,}
         then
           begin
             inc(SubA);              {инкрементировать количество}
                                                {неизбежных вершин}
             Color[i]:=ColSubA     {Пометить неизбежную вершину}
           end;
    j:=1;                                    {Номер дуги из вершины i}
    while (a[i,j]>=0) do                    {Пока дуги не кончились}
      begin
       if color[a[i,j]]=WHITE     {если вершина a[i,j] свободна}
         then
           begin
             Put(a[i,j]);                  {поставить ее в очередь}
             color[a[i,j]]:=GRAY;       {пометить вершину a[i,j]}
           end;                                  {как использованную}
       inc(j);                                 {Взять следующую дугу}
      end;
     end;
end;
```

Вывод ответа для подзадачи A и накопление неизбежных точек в массив Duty осуществляется так:

```
procedure OutSubA;
begin
   SubA:=subA-2;                {Начальная и конечная точки не}
   write(subA);                  {считаются неизбежными точками}
   j:=0;                        {J - количество неизбежных точек}
   for i:=1 to N-1 do                              {Для всех вершин}
      if (color[i]=ColSubA)                {Если она - неизбежная}
         then
             begin
                write(' ',i);                              {Выводим ее}
                inc(j); {Инкрементируем к-во неизбежных вершин}
               Duty[j]:=i; {Запоминаем номер неизбежной вершины}
             end;
   writeln;
end;
```

Далее приводится полный текст решения задачи

```
program io96d2t4;
Const
    White      =  1;
    Gray       =  2;
    ColSubA =  4;
var
  A             : array [0..101,0..101] of integer;
  Color,Q,Duty  : array [0..100] of integer;
  N,i,j,subA,SubB,BegQ,EndQ,k : longint;
  Procedure Put(x:longint);
     Begin inc(EndQ); Q[EndQ]:=x; end;

  Procedure Get(var x:longint);
      Begin x:=Q[BegQ]; inc(BegQ); end;

  Procedure Sort;
     var
         m,min,rmin,k : longint;
     begin
         for m:=1 to j-2 do
             begin
                min:=a[i,m]; rmin:=m;
                for k:=m+1 to j-1 do
                    if a[i,k]<min
                      then begin min:=a[i,k]; rmin:=k; end;
                   a[i,rmin]:=a[i,m]; a[i,m]:=min;
             end;
     end;

  Procedure Inputdata;
      begin
          i:=0; j:=1; read(a[0,1]);
          while (a[i,j]<>-1) do
              begin
                 while(a[i,j]<>-2) do
                     begin inc(j); read(a[i,j]) end;
                 if j >2 then Sort;
                 readln;
                 inc(i); j:=1; read(a[i,j]);
              end;
         N:=i;
    end;

  Procedure OutSubA;
      begin
      SubA:=subA-2; write(subA); j:=0; Duty[0]:=0;
      for i:=1 to N-1 do
         if (color[i]=ColSubA)
           then begin write(' ',i); inc(j); Duty[j]:=i; end;
      writeln;
    end;

  Procedure BFS;
     var i,j : longint;
     begin
         for i:=0 to N do color[i]:=WHITE;
         for i:=0 to N do Q[i]:=0;
         color[0]:=GRAY; SubA:=0; BegQ:=1; EndQ:=0;
         Put(0);
         while (BegQ<=EndQ) do
              begin
                  Get(i);
                  if BegQ>EndQ
                    then begin inc(SubA); Color[i]:=ColSubA end;
                  j:=1;
                 while (a[i,j]>=0) do
                     begin
                         if color[a[i,j]]=WHITE
                           then begin
                                       Put(a[i,j]);
                                       color[a[i,j]]:=GRAY;
                                end;
                         inc(j);
                     end;
              end;
     end;

  Procedure OutSubB;
      begin
          SubB:=0;
          for i:=1 to SubA do
             if Duty[i]<>0 then inc(SubB);
          write(subB);
          for i:=1 to SubA do
             if Duty[i]<>0 then write(' ',Duty[i]);
          writeln;
      end;

  Procedure DelArcs(k:longint);
      begin
          for j:=N Downto 1 do a[k,j+1]:=a[k,j];
          a[k,1]:=-2;
          for i:=0 to N-1 do
              begin
                  j:=1;
                 while (a[i,j]>=0) do
                      begin
                         if a[i,j]=k then a[i,j]:=maxint;
                         inc(j);
                      end;
              end;
     end;

  Procedure RetArcs(k:longint);
      begin
          for i:=0 to N-1 do
              begin
                  j:=1;
                 while (a[i,j]>=0) do
                      begin
                          if a[i,j]=maxint then a[i,j]:=k;
                          inc(j);
                      end;
              end;
          for j:=1 to N do a[k,j]:=a[k,j+1]
      end;

begin
    assign(input,'input.txt'); reset(input);
    assign(output,'output.txt'); rewrite(output);
```

```
    InputData;
    BFS;          {Поиск в ширину - находим "неизбежные" точки}
    OutSubA;

    for k:=1 to SubA do
      begin
        DelArcs(Duty[k]); BFS;  RetArcs(Duty[k]);
        for i:=0 to N-1 do
          begin
            j:=1;
            while (a[i,j]>=0) do
                begin
                  if (color[i]=WHITE) and (color[a[i,j]]<>WHITE)
                     then begin Duty[k]:=0; break end;
                  inc(j);
                end;
          end;
      end;
    OutSubB;
    close(input); close(output);
end.
```

6. О размерностях использованных в задачах массивов

Внимательный читатель уже обратил внимание, что в некоторых случаях массивы объявлены меньшей размерности, чем реально требуется по условиям задачи. Это сделано намеренно, из следующих соображений.

1. На международных олимпиадах по информатике для школьников с июля 2001 г. используются 32-битные компиляторы (Free Pascal и GNU C), для которых разрешаемые размерности объявляемых массивов существенно больше, чем те, что требуются по условиям задачи.

2. Основная цель данного материала — продемонстрировать предлагаемые базовые алгоритмы решения задач на графах, и автору не хотелось усложнять материал, "затемняя" тем самым суть базовых алгоритмов.

3. Возможно, методам эффективного использования оперативной памяти будет посвящена отдельная статья.

7. Обзор представленной теоретической информации

Цель данного раздела — кратко напомнить приведенные в статье теоретические сведения, необходимые для решения задач на графах. Читатель может использовать этот перечень для самоконтроля усвоения предложенного материала. Если вы понимаете, что скрывается под тем или иным понятием, и умеете выполнять названную работу — значит, цели, поставленные автором (и вами), достигнуты.

Итак, перечень вопросов для самоконтроля.

1. Сведение задач к графам.

2. Метод Флойда, применяется для:

- поиска кратчайших расстояний между всеми парами вершин графа (одновременно можно построить и сами кратчайшие пути от каждой вершины до каждой);
- построения матрицы достижимости графа;
- построения множества истоков и множества стоков (исток — вершина, в которую нет входящих дуг; сток — вершина, из которой нет исходящих дуг).

3. Метод Дейкстры, применяется для:

- поиска кратчайших расстояний от одной вершины до всех остальных вершин;
- построения оптимальных маршрутов от одной вершины до всех остальных вершин;

4. Поиск в глубину, применяется для:

- решения произвольных задач на графах, требующих соответствующего порядка обхода ("в глубину") графа. Раскрашивая пройденные вершины и/или дуги, можно управлять порядком и способами обхода (например, однократное или многократное использование дуг и вершин);
- построения множества истоков и стоков (как истоков на транспонированном графе);
- построения сильносвязанных компонент (ССК) графа. ССК — это множество вершин, каждая из которых достижима из всех вершин ССК.

5. Поиск в ширину, применяется для решения произвольных задач на графах, требующих соответствующего порядка обхода ("в ширину") графа. Раскрашивая пройденные вершины и/или дуги, можно управлять порядком и способами обхода (например, однократное или многократное использование дуг и вершин).

Заключение

Данная статья продолжает серию публикаций [1…10], ориентированных на самообучение программированию и подготовку к областным и республиканским олимпиадам по информатике. Очевидно, что для закрепления данного материала необходимо проверить полученные знания на компьютере как самостоятельным решением поставленных в данном материале задач, так и применением полученных знаний к другим задачам. Полезно также ознакомиться с соответствующими материалами учебных пособий, например, [11…13].

Для проверки полученных знаний автор рекомендует использовать сайт Гомельского государственного университета им.Ф.Скорины: http://dl.gsu.unibel.by. Важно подчеркнуть, что пользоваться ресурсами этого сайта для проверки собственных решений предлагаемых там задач можно как в режиме Internet-online, так и с помощью E-mail посредством почтового робота, отослав для начала пустое письмо по адресу dl-service@gsu.unibel.by

Вы можете также присылать нам свои задачи, содержащие условия, тесты, авторские решения, их описания, а также программы, проверяющие ответы в случае возможной неоднозначности правильных ответов. Под словом "свои" понимаются как составленные, так и решенные вами задачи, условия которых получены из любых источников (желательно в таком случае сообщать источник). Эти задачи (в соответствии с вашими пожеланиями) могут быть установлены на тестирование в нашу систему в учебные курсы, либо использованы для проведения олимпиад. Адрес для отсылки архивов с требуемой информацией: dl-support@gsu.unibel.by

Литература

1. М.Долинский. Памятка участнику олимпиады по информатике среди школьников. — Радиолюбитель. Ваш компьютер, 2000, №1, С.20.

2. М.Долинский. Начинаем программировать самостоятельно. — Радиолюбитель. Ваш компьютер, 2000, №№1…6.

3. М.Долинский. Решение задач на тему "Геометрия на плоскости". — Радиолюбитель. Ваш компьютер, 2000, №№8, 9.

4. М.Долинский. Разработка программ с использованием определяемых программистом типов данных . — Радиолюбитель. Ваш компьютер, 2001, №№1, 3.

5. М.Долинский. Решение задач с помощью очереди и стека. — Радиолюбитель. Ваш компьютер, 2001, №4…6.

6. М.Долинский. Обзор возможностей языка программирования Паскаль. — Радиомир. Ваш компьютер, 2001, №7, С.23.

7. М.Долинский. Алгоритмы генерации комбинаторных объектов. — Радиомир. Ваш компьютер, 2001, №10, 11.

8. М.Долинский. Некоторые факты из теории чисел. — Радиомир. Ваш компьютер, 2001, №9, С.20.

9. М.Долинский. Использование рекурсивных функций и процедур. — Радиомир. Ваш компьютер, 2002, №№1, 2.

10. М.Долинский . Решение задач с помощью рекуррентных соотношений. — Радиомир. Ваш компьютер, 2002, №№3…6.

11. Котов В.М., Волков И.А., Харитонович А.И. Методы алгоритмизации. Учебное пособие для 8-го класса общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 1996.

12. Котов В.М., Волков И.А., Лапо А.И. Методы алгоритмизации. Учебное пособие для 9-го класса общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 1997.

13. Котов В.М., Мельников О.И. Методы алгоритмизации. Учебное пособие для 10-11-го классов общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 2000.

Трактать о Dynamic Link Libraries

А.ИВАНЧИКОВ, И.ИВАНЧИКОВА,
г.Минск.

(Продолжение. Начало в №№9-10/2002)

Объектно-ориентированное программирование и DLL — первое решение

Динамические библиотеки с классами как таковыми не работают. Все, для чего DLLs предназначены — это экспорт имен переменных и функций. Поэтому более или менее эффективно организовать работу клиента с типами, определенными в DLL, основанную на явной компоновке, не удастся. Однако при использовании двух других методов подключения DLL эта задача будет успешно решена.

Для демонстрации возможностей объектно-ориентированной работы с DLL с неявной загрузкой, создадим проект с именем Algorithm, который будет представлять собой пустую DLL. DLL предоставляет один класс с именем CAlgorithm, обеспечивающий два метода QuickSort и BinarySearch (быстрой сортировки и двоичного поиска). После создания проекта в него следует добавить два файла — исходного кода Algorithm.cpp и заголовочный Algorithm.h. Содержимое файлов представлено ниже.

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  Algorithm.cpp
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
#include "Algorithm.h"
// Смена значений *pcElem1 и *pcElem2
//-------------------------------------
void CAlgorithm::swapElem(
    char* pcElem1,
    char* pcElem2,
    unsigned unWidth )
{
...
}

// Разделение одного массива на два
//-------------------------------------
char* CAlgorithm::SplitArray(
    char* pcFirstElem,
    char* pcLastElem,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
...
}

// Управление процессом сортировки
//-------------------------------------
void CAlgorithm::RecSort(
    char* pcFirstElem,
    char* pcLastElem,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
...
}

// Начало процесса сортировки
//-------------------------------------
void CAlgorithm::QuickSort(
    void* pvArray,
    unsigned unCount,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
...
}

// Двоичный поиск
//-------------------------------------
void* CAlgorithm::BinarySearch(
    const void* pKey,
    const void* pBase,
    unsigned unCount,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
    const char* pBegin = (const char*)pBase,
              * pLast = (const char*)pBase +
                        (unCount - 1) * unWidth,
              * pMedium = (const char*)pBase +
                        (unCount >> 1) * unWidth;

    while ( compF(pMedium, pKey) )
    {
        if ( compF(pMedium, pKey) < 0 )
            pBegin = pMedium + unWidth;
        else
            pLast = pMedium - unWidth;

        if ( compF(pBegin, pLast) > 0 )
            return 0;

        pMedium = (const char*)pBegin +
         (((pLast - pBegin) / unWidth + 1) >> 1) *
         unWidth;
    }

    return (void*)pMedium;
}

//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  Algorithm.h
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//

//-------------------------------------------
#ifdef ALGORITHM_EXPORTS
  // описатель экспортируемой функции
  #define ALGORITHM_API __declspec(dllexport)
#else
  // описатель импортируемой функции
  #define ALGORITHM_API __declspec(dllimport)
  // необходимость компоновки с библиотекой
  #ifdef ALGORITHM_LIB
    #pragma comment(lib, "Algorithm.lib")
  #endif
#endif

//-------------------------------------------
class ALGORITHM_API CAlgorithm
{
private:
    void swapElem(
        char* pcElem1,
        char* pcElem2,
        unsigned unWidth );
    char* SplitArray(
        char* pcFirstElem,
        char* pcLastElem,
        unsigned unWidth,
```

```
        int (*compF)(   const void *pvElem1,
                        const void *pvElem2) );
    void RecSort(
        char* pcFirstElem,
        char* pcLastElem,
        unsigned unWidth,
        int (*compF)(   const void *pvElem1,
                        const void *pvElem2) );

public:
    void QuickSort(
        void* pvArray,
        unsigned unCount,
        unsigned unWidth,
        int (*compF)(   const void *pvElem1,
                        const void *pvElem2) );

    void* BinarySearch(
        const void* pKey,
        const void* pBase,
        unsigned unCount,
        unsigned unWidth,
        int (*compF)(   const void *pvElem1,
                        const void *pvElem2) );
};
```

Теперь создадим новый проект — пустое консольное приложение с именем TestAlgorithm. Добавим в него всего один файл — TestAlgorithm.cpp — и выполним соответствующие настройки для проекта, использующего DLL с неявной компоновкой. Дополнительно необходимо в окне *Preprocessor definitions:* диалога *Project Settings*, через запятую к имеющимся уже макроименам, добавить имя ALGORITHM_LIB. Данное имя, при компиляции с заголовочным файлом Algorithm.h, приведет к включению обязательной компоновки с библиотекой Algorithm.lib.

Файл TestAlgorithm.cpp имеет вид:

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  TestAlgorithm.cpp
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
#include <stdio.h>
#include <stdlib.h>
#include "../Algorithm/Algorithm.h"
// размерность массива
#define SIZE_TETS_ARRAY 10

// Функция сравнения для QuickSort/BinarySearch
//-----------------------------------------------
int _cmp(
    const void* ppvElem1,
    const void* ppvElem2 )
{
    return *(int*)ppvElem1 - *(int*)ppvElem2;
}

//-----------------------------------------------
void main(void)
{
    // демонстрация работы Algorithm
    int nArray[SIZE_TETS_ARRAY];

    for ( int i = 0; i < SIZE_TETS_ARRAY; i++ )
        printf("%7d", nArray[i] = rand());

    int nKey = nArray[SIZE_TETS_ARRAY - 1];

    printf("\n");

    CAlgorithm algo;
    algo.QuickSort(nArray, SIZE_TETS_ARRAY,
              sizeof(int), _cmp);

    for ( i = 0; i < SIZE_TETS_ARRAY; i++ )
        printf("%7d", nArray[i]);

    printf("\n");

    void* pvKey = algo.BinarySearch(&nKey,
                  nArray, SIZE_TETS_ARRAY,
                  sizeof(int), _cmp);

    if ( pvKey )
       printf(«Array[%d] = %d\n»,
         (const int*)pvKey - nArray,
         *(const int*)pvKey);
}
```

Запуск TestAlgorithm.exe свидетельствует, что экспорт класса произведен успешно. Аналогичным результат будет и при реализации отложенной загрузки DLL.

(Окончание следует)

И.ШИРКО
E-mail: FDC@tut.by

Массивы: сортировка и поиск

Массив в программировании — это специальный тип для резервирования в памяти места под некоторое количество однотипных элементов. Массивы удобно использовать для хранения и обработки однородной информации, например, таблиц. Для подсчета занимаемой массивом памяти нужно количество его элементов умножить на число байтов памяти, отводимых одной переменной. В **таблице** приведено соответствие между типом переменной и объемом памяти, отводимой под нее. Например, задан двумерный массив:

Тип	Объем памяти (в байтах)
Char	1
String	256
Byte	1
Word	2
Integer	2
Longint	4
Real	6
Single	4
Double	8
Extended	8

```
x: array [1..20, 1..5] of integer;
```

Он содержит 20×5=100 элементов. Каждый элемент занимает 2 байта. В итоге получаем, что массив X занимает 100×2=200 байтов.

Рассмотренные далее примеры обработки массивов написаны на Object Pascal в Delphi 5, но пояснения к алгоритмам позволят перевести их на любой другой язык программирования.

Довольно часто к массивам приходится применять следующие операции:

- ввод/вывод;
- поиск минимального/максимального элемента;
- сортировка.

Разберем по порядку эти действия.

Ввод/вывод массива

Получить массив извне можно либо от пользователя, либо из файла. Напишем процедуру, которая выделяет из строки S целые числа и заполняет ими массив N. В строке S числа могут быть отделены друг от друга любым количеством пробелов. Если в строке содержится больше элементов, чем может вместить массив, то переменная Error получает значение "истина". В переменной i хранит-

ся количество заполненных элементов массива.

```
procedure GetArray (s:string; var N:array of integer; var error:boolean;
                                                    var i:integer);
var
  z:Byte;
  st:string;
begin
st:=s; //будем работать с переменной st, чтобы не изменять
                                                    //  заданную строку
  error:=false;
  i:=0;       //i содержит нижний индекс массива
  while pos (' ', st)<>0 do //оставляем в строке st только по одному
    delete (st, pos (' ', st)+1, 1);     //пробелу между числами

  if st[1]=' ' then delete (st, 1, 1); //если первый или последний
                                       //символ в строке – пробел,
  if st[length (st)]=' ' then delete (st, length(st), 1); // то удаляем его
  while (pos (' ', st)<>0) and (i<high (n)) do //пока в строке есть
    begin  //пробел и не заполнен предпоследний элемент массива...
      z:=pos (' ', st); //...определяем положение первого пробела
      N[i]:=strtoint (copy (st, 1, z-1)); //...добавляем число перед
                                  //пробелом в массив под индексом i
      delete (st, 1, z);//...удаляем это число из строки
                                                    //вместе с пробелом
      i:=i+1; //...переходим к следующему элементу массива
    end;
//если в строке не осталось пробелов, то массив может вместить все
//элементы, и можно записать последнее число, что мы и делаем
  if pos (' ', st)=0 then N[i]:=strtoint (st)
  else //иначе в массив не помещаются все числа
    begin
      error:=true;
      N[i]:=strtoint (copy (st, 1, pos (' ', st)-1));
    end;
end;
```

В этой процедуре используется открытый массив, т.е. не указываются нижняя и верхняя границы массива. Это делается для того, чтобы процедура могла обрабатывать массив любого размера.

Позже процедура GetArray будет использована на практике. А теперь — функция, которая делает обратную операцию, т.е. выводит в строку целочисленный массив, который, в свою очередь, можно записать и в файл, и на форму:

```
Function WriteArray (N:array of integer):string;
var
  i:integer;
begin
  for i:=0 to high(N) do result:=result+inttostr (n[i]) + ' ';
  delete (result, length (result), 1);
end;
```

Здесь все просто — один за другим перебираются элементы массива и записываются в результат через пробел.

В процедуре GetArray и функции WriteArray для строк используется тип string, который на самом деле вовсе не тип, а просто зарезервированное слово Паскаля. Оно может указывать на разные строковые типы. Если директива H включена ({$H+}), а по умолчанию она включена, то string указывает на тип AnsiString. А если она выключена ({$H-}), то string соответствует типу ShortString. Я это к тому, что для больших массивов нужно использовать AnsiString — память под переменную такого типа выделяется динамически (по мере поступления символов в строку), и такая переменная может занимать целых два гигабайта (!) памяти. А вот переменная типа ShortString способна вместить только 256 символов — это значит, что в нее нельзя записать даже сотню двузначных чисел, разделенных пробелом.

Поиск минимального/максимального элемента

Само собой разумеется, что речь здесь идет о неотсортированном массиве. Для поиска минимального (максимального) элемента воспользуемся следующим алгоритмом.

Предположим, что первый элемент массива является минимальным (максимальным). Затем он сравнивается с остальными. Если найден элемент меньше (больше) минимального (максимального), то этот элемент сам становится минимальным (максимальным). Это продолжается до тех пор, пока не будут просмотрены все элементы массива.

Для примера рассмотрим программу, которая ищет минимальное из введенных пользователем чисел. Создайте новый проект и поместите на форму следующие компоненты: Label1, Label2:TIabel, Edit1:Tedit и Button1:Tbutton. Расположите и озаглавьте все это так, как показано на **рис.1**.

Рис. 1

Теперь запишите процедуру обработки события OnClick для кнопки Button1, не забыв перед ней записать процедуру GetArray (описана в предыдущем разделе):

```
procedure TForm1.Button6Click(Sender: TObject);
var
  N:array [0..20] of integer;
  i, k, min:integer;
  error:boolean;
begin
  if edit1.text<>'' then
    begin
      getarray (edit1.text, n, error, i);
      min:=0;
      for k:=1 to i do
        if n[k]<n[min] then min:=k;
      Label2.Caption:='Результат:'+#13+'Минимальный элемент: '
   +inttostr(n[min])+#13+'Номер элемента: '+inttostr (min+1);
    if Error then label2.caption:=label2.caption+#13+
'Поиск производился только в первых '+inttostr(i+1)+' элементах.';
  end;
end;
```

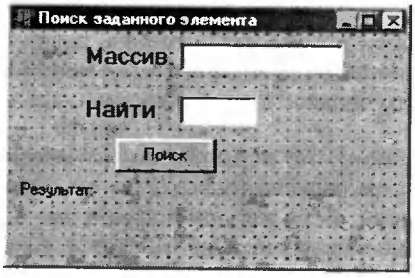
Если юзер ввел чисел больше, чем может вместить массив (в данном случае — более двадцати одного), то ему сообщается, что поиск производился не во всех элементах. Чтобы программа искала максимальный элемент, нужно в инструкции *if n[k]<n[min] then min:=k* заменить знак "<" на ">".

Поиск заданного элемента

Алгоритм простого перебора

Название алгоритма говорит само за себя — перебираем элементы массива один за другим. Если мы нашли искомый элемент, то запоминаем его номер и ищем дальше, пока не просмотрим весь массив. Если задача сводится к тому, чтобы проверить наличие элемента в массиве, то останавливаем просмотр после нахождения первого элемента. Алгоритм простого перебора применяется в неотсортированном массиве.

Рис. 2



Создайте новый проект и расположите на форме следующие компоненты: Label1, Label2, Label3:Tlabel; Edit1, Edit2:Tedit и Button1:Tbutton **(рис. 2)**. Событие OnClick для кнопки:

```
procedure TForm1.Button1Click (Sender: Tobject);
var
  str:string; //строка для хранения номеров
  m:integer; // количество найденных чисел
begin
  if (edit1.Text<>'') and (edit2.text<>'') then
    begin
      str:='';
      m:=0;
      z:=strtoint (edit2.text); // искомое число
      getarray (edit1.text, N, error, i); // заполняем массив
       // просматриваем массив, при нахождении искомого числа
                  //записываем его номер и увеличиваем m на 1
      for k:=0 to i do
        if N[k]=z then
          begin
            str:=str+inttostr(k+1)+' ';
            m:=m+1
          end;
      if m>0 then
        label3.Caption:= 'Результат:' +#13+'Количество найденных'+
                'элементов: '+inttostr (m)+#13+"Номера: '+str
      else
        label3.Caption:='Результат: заданный элемент не найден';
      if error then label3.Caption:=label3.Caption+#13+'Поиск'+
  'производился только в первых '+inttostr (i+1)+' элементах';
end;
end;
```

Бинарный поиск

Часто элементы в массиве упорядочены по возрастанию или по убыванию. Это упрощает поиск в нем заданного числа. Самым популярным и эффективным алгоритмом поиска в таком массиве является метод бинарного поиска (он же — двоичный, он же — дихотомия). Суть этого метода заключается в следующем.

Сначала заданный элемент сравнивается со средним элементом массива. Если они равны, то дело сделано. Если искомое число больше, то оно находится во второй половине массива, иначе — в первой. Анализируем половину, в которой находится заданный элемент. Делаем это до тех пор, пока не просмотрим все элементы, или не найдем нужное число.

Например: даны числа 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Ищем число 8. niz=1; verh=10; — границы поиска.

Вычисляем номер среднего элемента:

[(verh-niz)]/2+niz< = >[(10-1)/2]+1=5.

Искомый элемент больше 5, значит, изменяем нижнюю границу поиска:

niz=5+1=6.

Снова вычисляем середину массива:

[(10-6/2]+6=8 — заданный элемент найден под восьмым номером.

Теперь — программа. Создайте новый проект, форма и компоненты которого — такие же, как и в проекте, демонстрирующем алгоритм простого перебора (рис.2). Запишите процедуру обработки нажатия на кнопку:

```
procedure TForm1.Button8Click (Sender: TObject);
var
  i, z, niz, sered, verh:integer;
  found, error:boolean;
begin
//если юзер ввел массив и число, то..
  if (edit1.Text<>'') and (edit2.text<>'') then begin
    found:=false;
    z:=strtoint (edit2.text); //искомое число записываем в переменную z
    getarray (edit1.text,N,error,i); //заполняем массив N
    niz:=0; //нижняя граница поиска
    verh:=i; //верхняя граница поиска
  repeat
    sered:=trunc((verh-niz)/2)+niz; //средний номер массива
    if n[sered]=z then found:=true //если средний элемент массива
    //равен заданному числу, то found получает значение Истина
    else if z<n[sered] then verh:=sered-1 //если меньше, то
                                  //уменьшаем верхнюю границу поиска
    else niz:=sered+1; //если больше, то увеличиваем нижнюю
  until found or (verh<niz);
                      //выводим результат
    if found then label3.Caption:='Результат: нужное число'+
                        'найдено под номером '+inttostr(sered+1)
    else label3.Caption:='Результат: нужное число не найдено';
    if error then label3.Caption:=label3.Caption+#13+'Поиск'+
 'производился только в первых '+inttostr (i+1)+' элементах';

end;
end;
```

Этот алгоритм заканчивает работу нахождения первого заданного элемента. Если требуется найти все такие числа, то нужно проверять элементы, соседние с первым найденным.

Сортировка массива

Метод прямого выбора

Этот метод заключается в следующем:

- выбирается минимальный элемент массива и помещается на первое место, а первый — на место минимального;
- просматривается массив от второго элемента, находится минимальный и помещается на второе место, а второй — на место минимального;
- и так далее — до предпоследнего элемента.

Разумеется, вместо минимального элемента можно выбирать и максимальный.

Сортировка 10000 элементов у меня происходит за 5 с.

Текст алгоритма очень легко составить самостоятельно, поэтому он приведен не будет. Вместо этого попробуем улучшить метод прямого выбора.

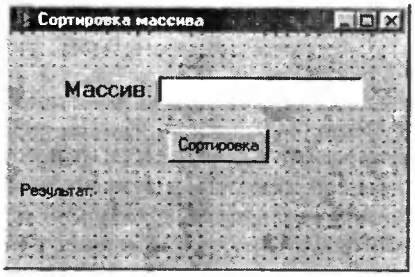
Улучшения напрашиваются сами собой — можно в одном цикле искать не только минимальный элемент, но и максимальный, т.е. сортировать массив с двух сторон. Также на каком-то этапе работы алгоритма массив может быть отсортирован раньше, чем ожидалось, в таком случае сортировку следует остановить. После этих изменений время сортировки массива уменьшается на 30...40% (у меня — 10000 элементов сортируются за 3 с).

Поместите на форму четыре компонента: Label1, Label2:Tlabel, Edit1:Tedit, button1:Tbutton (**рис.3**).

Обработка нажатия на кнопку:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  w, i, z, m, min, max, j, k:Integer;
  error:boolean;
```

Рис. 3




```
  n:array[0..20] of integer;
begin
  Getarray (edit1.text,n,error,w);
  w:=w+1;
  i:=-1;
  m:=(w div 2);
  if odd (w) then m:=m-1;
 repeat
  i:=i+1;
//находим минимальный и максимальный элементы
  min:=i;
  max:=w-i-1;
  z:=w;
  if n[max]<n[min] then
   begin
    min:=max;
    max:=i;
   end;
  for j:=i+1 to w-i-2 do
   if n[j]>n[max] then
    max:=j        else
                   if n[j]<n[min] then
                    min:=j;
//помещаем минимальный и максимальный элементы на края массива
  if min>i then
   begin
    k:=n[min];
    n[min]:=n[i];
    n[i]:=k;
   end;
  if max=i then max:=min;
  if max<w-i-1 then
   begin
    if max=i then z:=i+min-1;
    k:=n[max];
    n[max]:=n[w-i-1];
    n[w-i-1]:=k;
   end;
  j:=i+2;
  z:=0;
//проверяем, отсортирован ли массив
  while (j<=w-i-2)and(n[j-1]<=n[j]) do
   begin
    z:=z+1;
    j:=j+1;
   end;
 until (i=m) or (z=w-i-i-3);
  if (odd(w)) and (i=m) then
   if n[i+1]>n[w-i-2] then
    begin
      k:=n[i+1];
      n[i+1]:=n[w-i-2];
      n[w-i-2]:=k
    end;
//вывод результата
  for z:=0 to w-1 do
    Label2.Caption:=label2.caption+' '+inttostr(n[z]);
end;
```

Сортировка обменом

Каждый элемент, начиная с первого, сравнивается со следующим, и, если он больше следующего, то они меняются местами. Это продолжается до тех пор, пока массив не будет полностью отсортирован.

Форма и компоненты — такие же, как и в прошлом проекте.

Обработка нажатия на кнопку Сортировка:

```
procedure TForm1.Button4Click(Sender: TObject);
var
  w, m, k:Integer;
  error, change:boolean;
  n:array[0..20] of integer;
begin
  Getarray (edit1.text,n,error,w);
     repeat
       change:=false;
       for k:=0 to w-1 do begin
       if n[k]>n[k+1] then
       begin
         m:=n[k];
         n[k]:=n[k+1];
         n[k+1]:=m;
         change:=true;
       end;
     end;
     until not change;
  label2.Caption:='';
  for k:=0 to w do
  Label2.Caption:=Label2.Caption+inttostr(n[k])+' ';
end;
```

Переменная Change следит, поменялись ли между собой хотя бы два элемента. Если нет — значит, массив отсортирован.

10000 элементов у меня сортируются за 12 с.

Сортировка вставками

Это самый быстрый способ сортировки из рассмотренных в этой статье. На моем компьютере 10000 элементов сортируются всего за 2 с.

Просматриваем массив, начиная со второго элемента. Каждому элементу находим подходящее место в массиве. Для этого можно воспользоваться алгоритмом прямого выбора (описан ранее), а можно и более быстрым — методом бинарного поиска (также описан ранее), который использован в приведенном ниже алгоритме.

Компоненты — такие же, как и в двух прошлых программах.

Обработка нажатия на кнопку **Сортировка**:

```
procedure TForm1.Button5Click(Sender: TObject);
var
  min, max, k, i, w;
begin
  for i:=1 to w do
    begin
      min:=i;
      max:=0;
//находим место для очередного элемента
      while (max<min) do
        begin
          k:=(min+max) div 2;
          if n[k]>n[i] then max:=k+1
          else min:=k;
        end;
      k:=min;
      z:=n[i];
  // освобождаем место и вставляем туда элемент
      for m:=i downto k+1 do
        n[m]:=n[m-1];
      n[k]:=z;
   end;
//вывод результата
 for i:=0 to w do
 Label2.Caption:=Label2.Caption+inttostr (n[i])+' ';
        end;
```

Работа с мышкой

О.ВАЛЬПА,
E-mail: sandh@narod.ru

Довольно часто при написании программ в операционной среде MS-DOS необходимо использовать манипулятор мышь, который упрощает общение с интерфейсом программы или делает это общение более удобным. О том, как использовать манипулятор мышь в программах, я и хочу рассказать.

Прежде чем использовать в программе под MS-DOS функции управления мышкой, необходимо загрузить ее драйвер, который, как правило, прилагается к этому устройству на дискете при ее покупке, или использовать с вашей мышкой любой другой драйвер. Обычно он имеет название **mouse.exe** или **mouse.sys**. Драйвер mouse.exe можно загрузить в память компьютера обычным способом из командной строки перед запуском вашей программы или прописав в файл **autoexec.bat** строку с указанием пути, где он находится, и с его названием, например, так — **C:\DRV\mouse.exe**. При использовании драйвера mouse.sys необходимо прописать его в файл **config.sys** в виде строки **DEVICE=C:\DRV\mouse.sys** (естественно, если этот драйвер находится в разделе **C:\DRV**, иначе — замените этот путь на нужный). После перезагрузки компьютера драйвер будет загружен в память, и вы сможете использовать все функции мыши в своих программах.

Список функций драйвера "мыши", вызываемых через прерывание INT33H

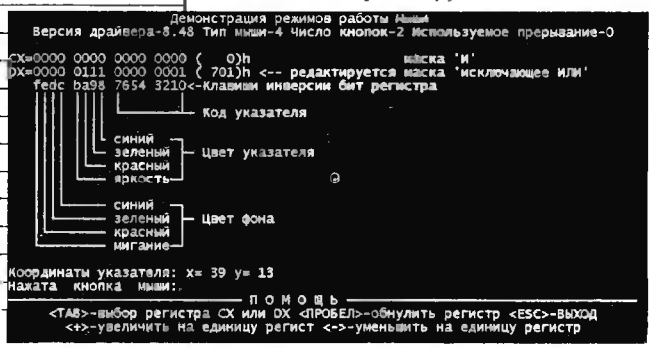
Функция	Описание
00H	Привести мышь в исходное состояние и получить состояние драйвера
01H	Вывести на экран указатель мыши
02H	Погасить указатель мыши
03H	Получить положение указателя мыши и состояние клавиш
04H	Установить положение указателя мыши
05H	Получить информацию о нажатиях клавиш мыши
06H	Получить информацию об отпусканиях клавиш мыши
07H	Установить горизонтальные границы для указателя
08H	Установить вертикальные границы для указателя
09H	Установить форму графического указателя
0AH	Установить тип текстового указателя
0BH	Прочитать состояние счетчиков движения мыши
0CH	Установить определяемый пользователем обработчик событий мыши
0DH	Включить эмуляцию светового пера
0EH	Выключить эмуляцию светового пера
0FH	Установить отношение микки/пиксел (микки — единица измерения числа импульсов от мыши при перемещении ее на 1/200 дюйма, т.е. на 0,127 мм)
10H	Установить область подавления указателя мыши
13H	Установить порог удвоения скорости
14H	Сменить определяемые пользователем обработчики событий мыши
15H	Получить размер буфера хранения состояния мыши
16H	Сохранить состояние драйвера мыши
17H	Восстановить состояние драйвера мыши
18H	Установить дополнительный обработчик событий мыши
19H	Получить адрес дополнительного обработчика событий мыши
1AH	Установить чувствительность мыши
1BH	Получить чувствительность мыши
1CH	Установить частоту прерываний от мыши
1DH	Выбрать видеостраницу для указателя
1EH	Получить видеостраницу для указателя
1FH	Заблокировать драйвер мыши
20H	Разблокировать драйвер мыши
21H	Привести в исходное состояние драйвер мыши
22H	Установить язык сообщений драйвера мыши
23H	Получить номер языка
24H	Получить версию драйвера, тип мыши и номер IRQ

Все функции драйвера мыши вызываются через прерывание **INT33H**. Номер вызываемой функции передается прерыванию через регистр **AX** процессора. Далее будет показано, как это делается, поверьте, в этом нет ничего сложного. Функций драйвера мыши довольно много, но на практике нет необходимости применять их все. Обычно достаточно использовать три-четыре функции. В таблице приводится краткое описание всех функций драйвера мыши Microsoft версии 8.48, вызываемых через прерывание INT33H, которые известны мне на сегодняшний день. Подробное описание всех этих функций можно найти, например, в [1].

Для демонстрации некоторых функций драйвера мыши мною была написана программа на языке C++, которая позволяет изучить пример использования функций мышки в программах и, кроме того, наглядно показывает виды текстовых указателей мыши и способ их получения.

После запуска программы на экране появиться картинка, приведенная на рисунке.

В центре экрана будет отображаться указатель мышки в виде "рожицы". Изменяя содержимое регистров DX и CX с помощью клавиш 0, 1, 2, ..., 9, a, b, c, d, e, f на клавиатуре, вы можете увидеть, как будет меняться указатель мышки. Эти регистры содержат данные для функции 0AH, которая позволяет установить тип указателя. Внизу экрана приведены строки помощи. Выше этих строк отображаются координаты указателя и нажатые кнопки мыши. При перемещении мышки или нажатии любой кнопки мышки, эти строки будут меняться. Таким образом легко определить позицию любого символа на экране. Для получения данных для этих строк мною была использована функция 03H. Ниже заголовка программы выводится информационная строка с отображением версии, типа мышки и других ее характеристик, которые позволяет получить функция 24H.



Для детального изучения способов использования функций мыши ниже приводится полный текст рассмотренной программы.

Текст программы "DMOUSE"

```
//=================================================
// Программа демонстрации работы манипулятора МЫШЬ
//
// Файл:dmouse.cpp
// Дата:05.01.2002
// Автор: Вальпа Олег
```

```
//=========================================================
// Координаты
#define   XS      0  // области сообщений
#define   YS      0
#define   XD      0  // области данных
#define   YD     19
#define   XH      0  // области справки
#define   YH     21

// Подключаемые библиотеки
#include <bios.h>
#include <stdio.h>
#include <stdlib.h>

// Константы
#include <const.def>

// Описатели функций
union REGS regs;
void clrkursor (void);
void setkursor (void);
void gxy(int, int);

int main(void)
{
int x, y, buttons, rdx=0x0701, rcx=0x0, rx, tb=-1, klav=TAB, end=0;
register char verh, verl, type, irq;
register int i, j;

system ("cls");                    // Очистим экран

//Сброс драйвера мыши и запрос состояния
regs.x.ax = 0;
int86(0x33, &regs, &regs);

//Если мышь не инсталлирована - выход с сообщением
buttons = regs.x.bx;
if(regs.x.ax == 0)
 {
 puts("\nМышь не инсталлирована\n");
 puts("\nинсталлируйте ее и вновь запустите программу\n");
 return(1);
 }
regs.x.ax =0x24;
int86(0x33, &regs, &regs);
verh = regs.h.bh;
verl = regs.h.bl;
type = regs.h.ch;
irq  = regs.h.cl;

// Вывод заголовка программы и строк помощи
gxy(XS+20,YS);
printf("Демонстрация режимов работы МЫШИ");
gxy(XS+3,YS+1);
printf("Версия драйвера-%d.%d Тип мыши-%d ",verh,verl,type);
printf("Число кнопок-%d Используемое прерывание-%d",buttons,irq);
gxy(XH,YH);
puts("----------------- П О М О Щ Ь ----------------------");
puts("     <TAB>-выбор регистра CX или DX <ПРОБЕЛ>-обнулить
регистр <ESC>-ВЫХОД     ");
puts("       <+>-увеличить на единицу регистр <->-уменьшить на
единицу регистр       ");
gxy(0,YS+3);
puts("CX=xxxx xxxx xxxx xxxx      маска 'И'                ");
puts("DX=xxxx xxxx xxxx xxxx     маска 'исключающее ИЛИ'   ");
puts("   fedc ba98 7654 3210<-Клавиши инверсии битов регистра  ");
puts("   ¦¦¦¦ ¦¦¦¦ ¦       ¦                            ");
puts("   ¦¦¦¦ ¦¦¦¦ L--------+- Код указателя  ");
puts("   ¦¦¦¦ ¦¦¦¦                                      ");
puts("   ¦¦¦¦ ¦¦¦L синий  -¬                             ");
puts("   ¦¦¦¦ ¦¦L- зеленый +- Цвет указателя ");
puts("   ¦¦¦¦ ¦L-- красный  ¦                            ");
puts("   ¦¦¦¦ L--- яркость--                             ");
puts("   ¦¦¦¦                                            ");
puts("   ¦¦¦L----- синий  -¬                             ");
puts("   ¦¦L------ зеленый +- Цвет фона       ");
puts("   ¦L------- красный ¦                            ");
puts("   L-------- мигание--                            ");

//Отобразить курсор на экране
regs.x.ax = 1;
int86(0x33,&regs,&regs);

//Установка типа и цвета курсора мыши
regs.x.ax = 0xA;
regs.x.bx = 0;
regs.x.cx = rcx;
regs.x.dx = rdx;
int86(0x33,&regs,&regs);

clrkursor ();   //Скрыть курсор
do
 {
 regs.x.ax=3; //Получить положение и состояние мыши
 int86(0x33,&regs,&regs);
 buttons = regs.x.bx;
 x=regs.x.cx;
 y=regs.x.dx;
 gxy(XD,YD);
 printf("Координаты указателя: x=%3d y=%3d",x/8,y/8);
 gxy(XD,YD+1);
 printf("Нажата  кнопка  мыши: ");
 if((buttons) == 0) puts("             ");
 if((buttons) == 1) puts("левая");
 if((buttons) == 2) puts("      правая");
 if((buttons) == 3) puts("левая правая");
 if(klav!=0)
  (
  gxy(31,3);
  if(tb == 1)  puts("<- редактируется ");
  else         puts("                  ");

  gxy(31,4);
  if(tb == -1) puts("<- редактируется ");
  else         puts("                  ");

  }
 if( bioskey(1) != 0 )     // Если нажата клавиша
    klav=bioskey(0);       // Читаем код клавиши
 else klav=0;              // Иначе - пустой код клавиши
 if(klav==TAB) tb = tb * -1;
 if(tb==1) rx=rcx; else rx=rdx;
 switch (klav)
   {
   case KF:    rx=rx ^ 0x8000; break;
   case KE:    rx=rx ^ 0x4000; break;
   case KD:    rx=rx ^ 0x2000; break;
   case KC:    rx=rx ^ 0x1000; break;
   case KB:    rx=rx ^ 0x0800; break;
   case KA:    rx=rx ^ 0x0400; break;
   case K9:    rx=rx ^ 0x0200; break;
   case K8:    rx=rx ^ 0x0100; break;
   case K7:    rx=rx ^ 0x0080; break;
   case K6:    rx=rx ^ 0x0040; break;
   case K5:    rx=rx ^ 0x0020; break;
   case K4:    rx=rx ^ 0x0010; break;
   case K3:    rx=rx ^ 0x0008; break;
   case K2:    rx=rx ^ 0x0004; break;
   case K1:    rx=rx ^ 0x0002; break;
   case K0:    rx=rx ^ 0x0001; break;
   case SPACE: rcx=0; rdx=0; rx=0; break;
   case ESC:   end=1; break;     // Выход из программы
   case KDEC:  rx--;  break;
   case KINC:  rx++;  break;
   default:    break;
   }
   if(tb==1) rcx=rx; else rdx=rx;

gxy(0,YS+3);
printf("CX=");
for(i=0,j=1; i<16; i++,j++) {
   printf("%1X",(rcx>>(15-i)) & 1);
   if(j==4) (j=0, printf(" ");}
   }
printf("(%4X)h",rcx);

gxy(0,YS+4);
printf("DX=");
for(i=0,j=1; i<16; i++,j++) {
   printf("%1X",(rdx>>(15-i)) & 1);
   if(j==4) (j=0, printf(" ");}
   }
printf("(%4X)h",rdx);

   //Установка типа и цвета курсора мыши
```

```
    regs.x.ax = 0xA;
    regs.x.bx = 0;
    regs.x.cx = rcx;
    regs.x.dx = rdx;
    int86(0x33,&regs,&regs);
  ) while(end != 1);

regs.x.ax=2;     // Погасить указатель мыши
int86(0x33,&regs,&regs);

setkursor ();    // Восстановить курсор
system ("cls");
return(0);
}

// Функция позиционирования вывода
void gxy(int x,int y)
{
    regs.h.dl=x;
    regs.h.dh=y;
    regs.h.bh=0;
    regs.h.ah=2;
    int86(0x10,&regs,&regs);
}

// Функция скрытия курсора
void clrkursor (void)
{
    union REGS regs;
    regs.x.ax   = 0x0100;
    regs.x.cx   = 0x0f00;
    int86(0x10, &regs, &regs);
}

// Функция восстановления курсора
void setkursor (void)
{
    union REGS regs;
    regs.x.ax   = 0x0100;
    regs.x.cx   = 0x0708;
    int86(0x10, &regs, &regs);
}
```

В программе используется файл описания констант const.def.

```
//=============================================================
// Файл описания констант (скан-коды клавиатуры и пр.)
//
// Файл:const.def
// Дата:05.01.2002
// Автор: Вальпа Олег
//=============================================================
// Скан-коды клавиатуры
// Клавиши управления
#define   ESC     0x011b // Esc
#define   HOME    0x4700 // Home
#define   END     0x4f00 // End
#define   PUP     0x4900 // Page Up
#define   PDOWN   0x5100 // Page Down

#define   UP      0x4800 // Стрелка вверх
#define   DOWN    0x5000 //         вниз
#define   LEFT    0x4B00 //         влево
#define   RIGT    0x4D00 //         вправо
#define   KINC    0x4E2b //         +
#define   KDEC    0x4A2d //         -

#define   TAB     0x0f09 // Tab
#define   SPACE   0x3920 // Space

// Цифровые и буквенные клавиши
#define   K1      0x0231 // 1
#define   K2      0x0332 // 2
#define   K3      0x0433 // 3
#define   K4      0x0534 // 4
#define   K5      0x0635 // 5
#define   K6      0x0736 // 6
#define   K7      0x0837 // 7
#define   K8      0x0938 // 8
#define   K9      0x0a39 // 9
#define   K0      0x0b30 // 0
#define   KA      0x1e61 // A
#define   KB      0x3062 // B
#define   KC      0x2e63 // C
#define   KD      0x2064 // D
#define   KE      0x1265 // E
#define   KF      0x2166 // F

// Функциональные клавиши
#define   KF1     0x3b00 // F1
#define   KF2     0x3c00 // F2
#define   KF3     0x3d00 // F3
#define   KF4     0x3e00 // F4
#define   KF5     0x3f00 // F5
#define   KF6     0x4000 // F6
#define   KF7     0x4100 // F7
#define   KF8     0x4200 // F8
#define   KF9     0x4300 // F9
#define   KF10    0x4400 // F10
```

Этот файл при трансляции должен находиться в той же директории, что и файл программы dmouse.cpp.

Программа dmouse снабжена достаточным количеством комментариев. Как видно из текста программы, при вызове функций мыши используется команда int86(0x33, &regs, &regs); перед которой в регистр AX заносится номер вызываемой функции командой regs.x.ax = номер функции;. Если функция использует дополнительные данные, они заносятся в соответствующие регистры такой же командой. Например, в следующем фрагменте программы:

```
//Установка типа и цвета курсора мыши   regs.x.ax = 0xA;
regs.x.bx = 0;   regs.x.cx = rcx;   regs.x.dx = rdx;
int86(0x33,&regs,&regs);
```

вызывается функция **0AH**, в регистр BX записывается тип указателя (0 — программный, 1 — аппаратный), а в регистры CX и DX заносятся данные о типе и цвете указателя мышки.

В заключение приведу описание остальных используемых мною в программе функций драйвера мыши.

Функция 00H. Привести мышь в исходное состояние и получить состояние драйвера

Вход: AX=0x0

Выход: AX=состояние (0 — драйвер не инсталлирован, 1 — драйвер инсталлирован)
BX=число кнопок

Функция 01H. Вывести на экран указатель мыши

Вход: AX=0x0001

Выход: нет

Функция 02H. Погасить указатель мыши

Вход: AX=0x0002

Выход: нет

Функция 03H. Получить положение и состояние клавиш мыши

Вход: AX=0x0003.

Выход: BX=состояние клавиш (бит 0=1 — нажата левая кнопка, бит 1=1 — нажата правая кнопка, бит 2=1 — нажата средняя кнопка, биты 3...15=0);
CX=координата X (по горизонтали);
DX=координата Y (по вертикали).

Функция 24H. Получить информацию о мыши

Вход: AX=0x0024.

Выход: BH=основной номер версии драйвера мыши;
BL=дополнительный номер версии;
CH=тип мыши (1 — магистральная, 2 — последовательного порта, 3 — мышь InPort, 4 — мышь PS/2, мышь HP);
CL=номер используемого прерывания IRQ (0 — PS/2, 2...5, 7— номер IRQ).

Литература

1. Р.Данкан. Профессиональная работа в MS-DOS. Пер. с англ. Ю.И.Малахова, К.Г.Финогенова п/р К.Г.Финогенова. — М.: Мир, 1993.

А.НЕБОРСКИЙ,
г.Минск.

Web-страницы на Delphi

Давайте немного подумаем — на чем в основном пишут HTML-страницы? Правильно, на таких языках как HTML, JavaScript, Perl, Cgi, Php... можно еще много примеров приводить. А что делать, если ты знаешь Pascal (Delphi), немного HTML, и уже накачал себе кучу JavaScript'ов? Но ведь ты знаешь такой язык как Delphi, и если думаешь, что между Delphi и Pascal нет никакой связи, то очень заблуждаешься. Не мешкая беги к ближайшему киоску и покупай диск с Delphi 5, затем беги домой и устанавливай. Дальше сам разберешься, а мы возвращаемся к нашим скриптам.

Раньше web-дизайнеры считали, что если "засунуть" на страницу много банеров, счетчиков и графики, то она будет "круче всей планеты". Сейчас же в моде информационный дизайн и всяческие "удобности" для пользователя.

Похоже, что времена меняются, и Delphi — среда создания обычных настольных приложений — может пригодиться и для написания прикладных web-программ. Действительно, что может быть лучше — уже знакомая среда разработки, вдоль и поперек изученный язык, да и достаточно широкий круг специалистов по программированию в Delphi, это ли не плюсы создания web-приложений на Delphi?. Есть конечно, и минусы — созданные программы вряд ли смогут удовлетворить тех, кто считает, что лучший web-сервер — сервер не от Microsoft. Но что поделаешь, версия Delphi под Unix пока отложена до лучших времен.

Что ж, если ты, уважаемый читатель, еще не перелистнул журнал на следующую страницу, то начнем. Все, что нам нужно из программного обеспечения Delphi версии 5 или 6 (не пробовал в версиях ниже) — PWS (Personal Web Server) для испытания приложений на своем компьютере без подключения к Интернет.

Открываем File/New/Other в появившемся окне на вкладке Console Application. Можете не ждать — форма не появится. В окне с программой у вас должно появиться нечто похожее на следующее:

```
program Project 2;
{$APPTYPE CONSOLE}
uses
SysUtils;
begin
{TODO-oUser-c Console Main: Insert code here}
end.
```

Стирайте строку между begin и end, чтобы не смущала, здесь мы будем писать наш текст.

Впишем туда:

```
writeln ('CONTENT-TYPE: TEXT/HTML');
writeln;
writeln ('<html>');
writeln ('<head>');
writeln ('<meta HTTP-EQUIV = "Content-Type" Content = "text-
html; charset = windows-1251">');
writeln ('<title> Delphi the best facility for making web-
publications!</title>');
writeln ('</head>');
writeln ('Hello, world!');
writeln ('</body>');
writeln ('</html>');
```

Вот мы и написали самый простой "каркас" страницы. Теперь перейдем к написанию более сложного приложения. Если вам надо передать какие-нибудь параметры приложению, то используйте стандартные процедуры и функции Delphi. Нас интересуют две функции — ParamCount и ParamStr. Для справки: ParamCount возвращает количество переданных параметров, ParamStr(номер) возвращает строку с параметром по номеру (нулевой параметр — путь к приложению и имя приложения). Например:

```
Program CgiParam;
{$APPTYPE CONSOLE}
Uses SysUtils;
begin
writeln ('CONTENT-TYPE: TEXT/HTML');
writeln;
writeln ('<HTML><HEAD>');
writeln ('<TITLE> Cgidate </TITLE>');
writeln ('</HEAD><BODY>');
if ParamCount >0 then
begin
writeln ('<h4>', ParamStr(1), '</h4>');
end
else begin
writeln ('Параметр отсутствует.');
end;
writeln ('</BODY></HTML>');
end.
```

Это приложение выводит введенный вами параметр. Введите, к примеру, http://1270.0.1/cgi-bin/CgiParam.exe/Proba, и появится слово Proba.

Итак, данные от пользователя web-приложению можно передать через переменные окружения. Вот список наиболее часто употребляемых:

- GATEWAY_INTERFACE — поддерживаемая версия CGI;
- REQUEST_METHOD — метод запроса. Может быть как GET, так и POST;

HTTP_REFERER — адрес страницы (url), активирующей текущее приложение на web-сервере;

PATH_INFO — переданный приложению путь, расположенный между именем приложения и строкой запроса;

QUERY_STRING — строка запроса. Если метод — GET, то она добавляется к url;

REMOTE_HOST — имя хоста удаленного пользователя;

REMOTE_USER — имя удаленного пользователя;

REMOTE_IDENT — IP-адрес удаленного пользователя;

HTTP_USER_AGENT — имя и версия браузера удаленного пользователя.

Например:

```
program CgiVars;
{$APPTYPE CONSOLE}
uses
Windows;
const
VarList: array [1..17] of string [30] =
('SERVER_NAME', 'SERVER_PROTOCOL', 'SERVER_PORT',
'SERVER_SOFTWARE', 'GATEWAY_INTERFACE', 'REQUEST_METHOD',
'PATH_TRANSLATED', 'HTTP_REFERER', 'SCRIPT_NAME', 'PATH_INFO',
'QUERY_STRING', 'HTTP_ACCEPT', 'REMOTE_HOST', 'REMOTE_USER',
'REMOTE_ADDR', REMOTE_IDENT, 'HTTP_USER_AGENT');
var
I: Integer;
ReqVar: string;
VarValue: array [0..200] of Char;
begin
writeln ('Content type: text/html');
writeln;
writeln ('<HTML><HEAD>');
writeln ('<TITLE> CGI Variables </TITLE>');
writeln ('</HEAD><BODY>');
writeln ('<H1> CGI Variables </H1>');
writeln ('<HR><PRE>');
```

```
for I: = Low(VarList) to High(VarList) do
begin
ReqVar: = VarList [I];
if (GetEnvironmentVariable (PChar(ReqVar), VarValue, 200) >0)
then
else
VarValue: = '';
writeln (VarList [I] + ' = ' + VarValue);
end;
writeln ('</PRE></BODY></HTML>');
end.
```

Если вы будете писать англоязычные сайты, то можете приступать, а если вам в текстовом поле нужны русские буквы, например, "АБВ", то в параметры будет передано %C0 %C1 %C2. Ну и как это читать? Маленькая поправка — расхождение с таблицей ASCII составляет 64 символа. Для тех, кто понял, готовая функция:

```
function code2str(s:string): string;
var stemp: string;
      n,dstr: integer;
function hex2integer(str:string): integer;
constHEX: array ['A'..'F'] of integer = (10,11,12,13,14,15);
var Int, i: integer;
begin
Int := 0;
for i := 1 to Length(str) do
if inttostr[i] <'A' then Int := Int*16+ord(str[i])-48
else Int := Int*16+HEX[inttostr[i]];
hex2integer := int;
end;
begin
dstr := 0;
for n := 1 to dstr do begin
if s = '' then Break;
if s[1] = '%' then begin
stemp := copy(s,2, pos ('%',s)+1);
stemp := Chr(hex2integer(stemp)-64);
delete(s,1,1);
delete(s,1,2);
end else begin
stemp := s[1];
delete(s,1,1);
end;
result := stemp;
end;
end;
```

Желаю вам удачи в своих начинаниях. Только не забывайте, что вы еще учитесь, и всегда найдутся люди, которые окажутся умнее вас. Еще в Delphi есть средства для создания более сложных web-приложений. О них я расскажу в следующих статьях.

Фолко&Cooper,
г.Вологда.

Графика и анимация в программировании

Мир вам, юзеры. Мы с другом экспериментировали с тригонометрией и графикой в Паскале и наткнулись на совершенно неожиданный результат — получили очень красивые фигуры, задаваемые тригонометрическими формулами. У нас возникла благородная мысль поделиться своим случайным открытием со всей программистской братией.

Представляем на ваш суд небольшую программку, которая по тригонометрическим формулам строит всякие замысловатые загогулины. Смысл работы проги прост. Угол f изменяется в диапазоне 0...2*pi. Он является параметром тригонометрических функций, которые задаются в параметрической форме (параметрическое представление кривой на плоскости с координатами x и y — это две функции — x=x(t), y=y(t), определенные на одном и том же числовом множестве). Затем происходит прорисовка этой точки.

Исходный вариант программы представляет собой нечто вроде хранителя экрана. Рабочая область разбивается на четыре части, в каждой из которых рисуется своя фигура. Картинка на экране получается симметричной относительно центра экрана. Программа выходит в ОС по нажатию любой клавиши. В общем, пробуйте и судите нас :).

```
uses
graph, crt;
var
      gd, gm:integer;
      x, y, x1, y1:integer;
      f:real; {угол поворота}
      a, b:integer;
      Color, Color_i:byte; {установка палитры}
      Bol:boolean;
const
      r=100; {масштаб одного рисунка}
      n=20; {n - количеству различных рисунков}
procedure Paint; {рисование}
begin
{-------------------------------------------------}
      if f>2*pi then begin
      a:=random(n);
      b:=random(n);
      f:=0;
      end;
      if a=b then inc(a); {исключение рисования окружности}
      while ((f<2*pi) and (color_i>1)) do begin
            if bol then begin
                  color_i:=color_i-1;
                  SetRGBPalette(Color, Color_I, Color_I,
                    Color_I);
            end;
            x:=round(r*cos(a*f)*cos(f));
            y:=round(r*cos(b*f)*sin(f));
            x1:=round(r*cos(b*f)*cos(f));
            y1:=round(r*cos(a*f)*sin(f));
            putpixel(160+x, 120-y, Color);
            putpixel(160+x1, 360+y1, Color);
            putpixel(480+x, 360-y, Color);
            putpixel(480+x1, 120+y1, Color);
            f:=f+0.005;
            If KeyPressed then break;
            Delay(50);
      end;
{-------------------------------------------------------}
end;
begin
      {Инициализация графического режима, переменных и палитры}
      gd:=Detect; InitGraph (gd, gm, 'c:\tp\bin\');
      SetFillStyle (0, 0);
      Randomize;
f:=0;
Color:=5;
bol:=false;
SetRGBPalette (Color, Color_i, Color_i, Color_i);
{Рисуем до тех пор, пока не нажата клавиша}
repeat
      Paint;
      If not KeyPressed then bar (0, 0, GetMaxX, GetMaxY);
until KeyPressed;
bol:=true;
{Продолжаем рисовать, но уже со временем рисунок гаснет}
repeat
      Paint;
      {Выход осуществляется, когда рисунок полностью погаснет}
Until color_i<=1;
CloseGraph;
end.
```

"Секретные" порты в "Sega Mega Drive-II"

С.РЮМИК,
г.Чернигов.

Если вам кажется, что все хорошо, значит, вы чего-то не заметили.
Следствие из законов Мерфи.

Казалось бы, что нового можно обнаружить в архитектуре 16-битной игровой приставки "Sega Mega Drive-II" (MD2), если ее изучение ведется начиная с 1988 г. Тем не менее, тайны и слабоизученные места остаются. Взять, к примеру, способы переключения игр в картриджах-"многоигровках". Известны два способа — аппаратный и программный.

Аппаратный заключается в смене игр по сигналу начального сброса /RES, выведенному на контакт XB27 разъема CARTRIDGE. Меню выбора игр на экране телевизора не высвечивается, и об их количестве можно узнать только из надписей на обложке футляра или из практического опыта. В адресном пространстве ПЗУ картриджа прошивки игр располагаются последовательно, друг за другом, причем каждая программа запускается и работает в начальной области памяти.

Внутри картриджа обычно находится микросхема серии 74LS74 (аналог К555ТМ2) с D-триггерами, которые меняют свое состояние после очередного нажатия кнопки сброса RESET. На **рис.1** приведен фрагмент типовой электрической схемы картриджа с четырьмя играми. Выходы триггеров DD2.1, DD2.2 подключаются к старшим разрядам шины адреса A18, A19 микросхемы ПЗУ DD1. Все внешние сигналы выводятся на разъем CARTRIDGE в соответствии с нумерацией, приведенной в [1].

На **рис.2** и **3** показаны встречающиеся на практике схемы замещения микросхемы DD2 (рис.1), что следует учитывать при ремонте MD2. Обычно на печатных платах картриджей делается универсальная разводка проводников с маркировкой недостающих микросхем соответственно "4013" (DD2*) или "4520" (DD2**).

Количество задействованных D-триггеров в схемах на рис.1 и 2 определяет число игр: один триггер — 2 игры, два триггера — 3 или 4 игры. В схеме на рис.3 используется двоичный счетчик, и поскольку микросхема DD2** содержит два счетчика, общее число игр может составлять от 2 до 256.

При программном способе переключения игр, на экране телевизора предварительно отображается меню, в котором можно выбрать одну из нескольких игр, нажимая на кнопки джойстика. В этом случае программа прошивки ПЗУ картриджа компилируется так, чтобы игры запускались с ненулевого адреса и работали не выходя за пределы определенного участка памяти. Разумеется, разные игры должны занимать разные, непересекающиеся между собой участки ПЗУ. Не все программы допускают подобное видоизменение, все зависит от методов адресации, которые применил разработчик игры. Для компиляции используются специальные программы, поддерживающие систему команд процессора MC68000 и приспособленные к архитектуре MD2.

Рис. 1

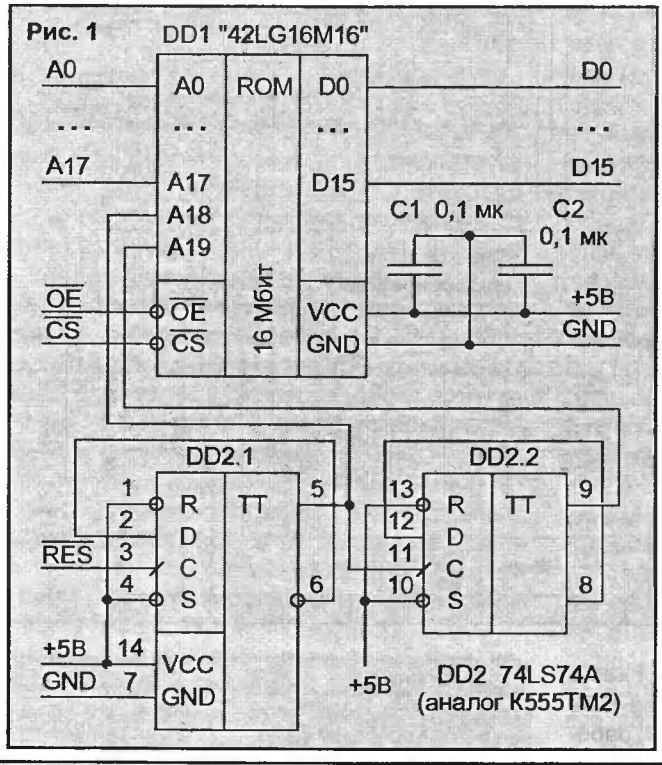


Рис. 2

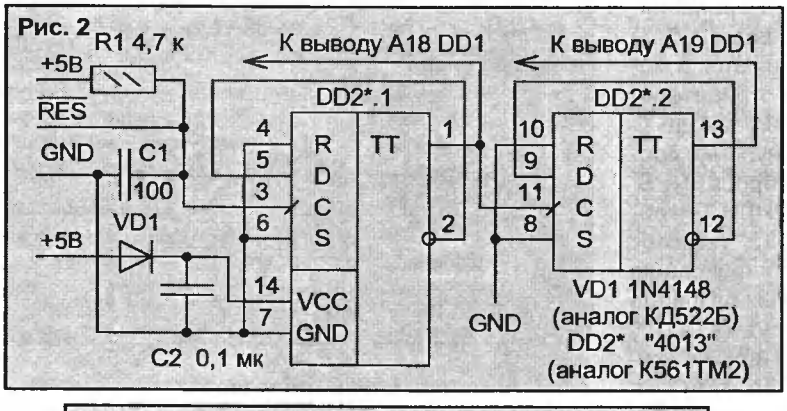
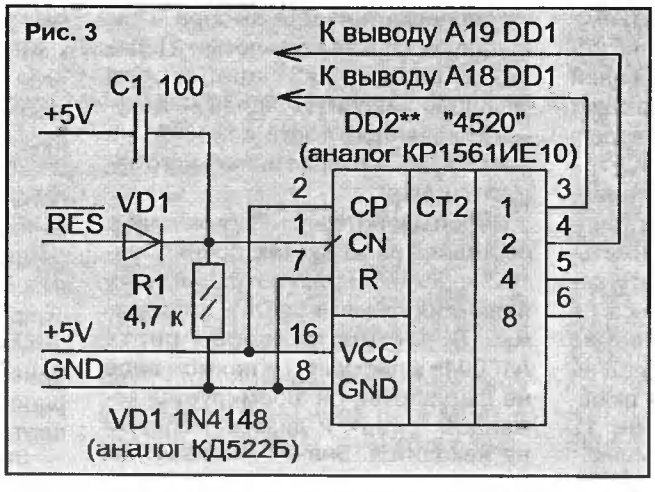
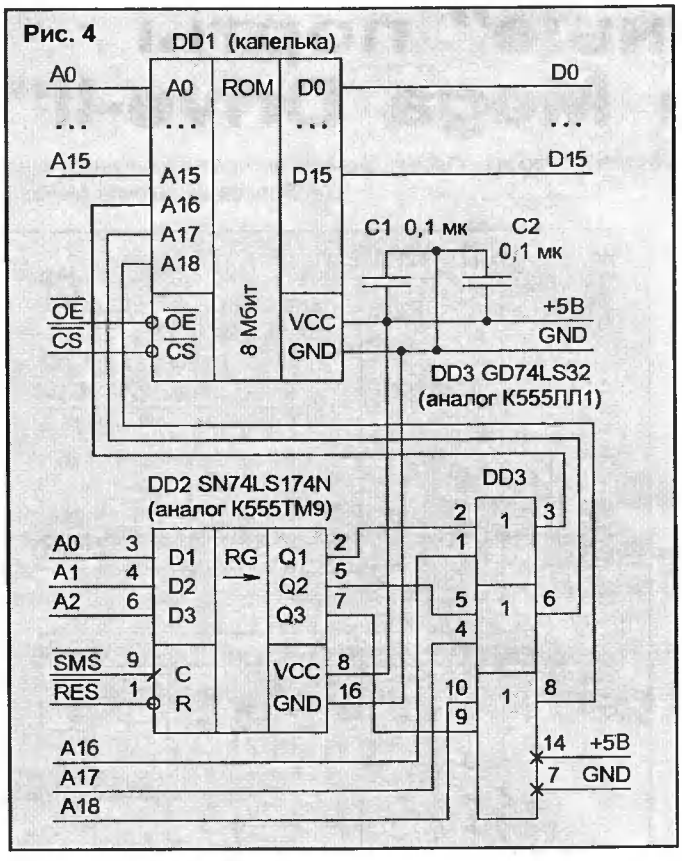


Рис. 3



"Считыватель" SEGA-картриджей, электрическая схема которого приведена в [1], успешно определяет "многоигровки" в автоматическом режиме. Проблем при чтении не возникало до тех пор, пока очередь не дошла до китайского картриджа "Super 7in1", который "считыватель" ошибочно определил как одноигровку. На самом деле это типичная "многоигровка", в которой имеется меню из 7 игр, выбираемых джойстиком. Но, в отличие от картриджей с программным способом переключения игр, в данном картридже коды прошивок не были скомпилированы! Все игры оказались рассчитанными на запуск и работу в начальном участке памяти. Тогда, что же заставляет их переключаться?

Рис. 4



Ключ к разгадке секрета находится внутри картриджа. На его электрической схеме (**рис.4**) видно, что микросхема DD2 SN74LS174N (аналог К555ТМ9) работает в качестве регистра хранения. На ее выходах 1, 2, 3 (выводы 2, 5, 7) логические уровни изменяются в момент спада импульсов отрицательной полярности на синхровходе C (вывод 9). Начальное обнуление производится по сигналу сброса /RES. Логические элементы 2ИЛИ микросхемы DD3 GD74LS32 (аналог К555ЛЛ1) осуществляют фиксацию выбора старших адресных разрядов A16, A17, A18 ПЗУ картриджа DD1.

Переключение с одной игры на другую производится сигналом /SMS, выведенным на контакт XB31 разъема CARTRIDGE. Этот сигнал обычно используется для доступа к внутренней памяти RAM в картриджах, имеющих микросхему ОЗУ. Обращение ведется программно через порт A130F1h.

Напрашивается предположение, что в прошивке кодов ПЗУ картриджа "Super 7in1" также должно иметься обращение к какому-то порту, отвечающему за выдачу импульса по цепи /SMS. Дизассемблирование программы универсальным отладчиком IDA-3.7 позволило найти любопытный участок кодов (**листинг 1**), расположенный в загрузчике меню.

Листинг 1

```
...
loc_54960: cmpi.b  #1,d0            ; Проверка выбора игры номер 2
           bne.w   loc_54982        ; Переход на следующую проверку
           move.l  #$300400A1,d0    ; В регистре d0 код 300400A1h
           swap    d0               ; В регистре d0 код 00A13004h
           movea.l d0,a3            ; В регистре a3 код 00A13004h
           move.l  #$2160000,d0     ; В регистре d0 код 02160000h
           swap    d0               ; В регистре d0 код 00000216h
           movea.l d0,a4            ; В регистре a4 код 00000216h
           jmp     $FF2000          ; Переход в ОЗУ приставки MD2
loc_54982: ...
-------------------------------------------------------------------
00FF2000h: move.b  d0,(a3)          ; Занесение кода 16h в A13004h
           move.b  d0,(a3)          ; Занесение кода 16h в A13004h
           movea.l (start).w,sp     ; Занесение адреса старта в стек
           jmp     (a4)             ; Запуск игры с адреса 00000216h
```

В строке "loc_54960" производится проверка наличия выбора игры (в данном случае игры номер 2). Далее в регистр "a3" процессора MC68000 заносится адрес недокументированного порта A13004h, а в регистр "a4" — адрес начального запуска игры.

Небольшой нюанс. Программист, составивший загрузчик, принял "антихакерские" меры, затрудняющие понимание логики работы программы. В частности, адрес порта A13004h в листинге 1 в прямом виде не фигурирует, он формируется командой "swap", и дизассемблером не находится. Значит, разработчик специально "засекретил" обращение к порту A13004h, а также свои дальнейшие действия от постороннего глаза.

Заканчивается блок кодов командой "jmp" и передачей управления по адресу 0FF2000h, который находится во внутреннем ОЗУ игровой приставки MD2. Ранее в эту область ОЗУ из ПЗУ картриджа были скопированы команды, которые теперь подлежат выполнению. По команде "move.b d0, (a3)" происходит обращение к порту A13004h. Команда "move" повторяется два раза подряд, и это не ошибка. Очевидно, что обращение к порту A13004h приводит к генерации импульса по цепи /SMS, а двойная команда "move" должна увеличивать его длительность для надежного защелкивания информации в микросхеме DD2 SN74LS174N.

Аналогичные процессы происходят и при выборе других игр, меняются лишь адреса начального запуска и адреса "секретных" портов. В **таблице** приведено соответствие логических уровней, разрядов и адресов игр в картридже "Super 7in1".

Еще одно предположение — запись в любой из портов A13000...A1300Eh сопровождается автоматическим выставлением определенной комбинации сигналов на младших разрядах шины адреса A0, A1, A2: для порта A13000h — 000, для A13002h — 001, для A13004h — 010, для

Адрес порта	Адрес запуска игры	Название игры	Область ПЗУ картриджа	Адресные линии		
				A2	A1	A0
A13000h	0200h	Tecmo World Cup-92	000000...03FFFFh	0	0	0
A13004h	0216h	Flicky	040000...05FFFFh	0	1	0
A13006h	0200h	Art Alive	060000...07FFFFh	0	1	1
A13008h	021Eh	Columns	080000...09FFFFh	1	0	0
A1300Ah	0200h	Block Out	0A0000...0BFFFFh	1	0	1
A1300Ch	06FEh	Shive II	0C0000...0DFFFFh	1	1	0
A1300Eh	0206h	Fatal Labyrinth	0E0000...0FFFFFh	1	1	1

A13006h — 011, для A13008h — 100, для A1300Ah — 101, для A1300Ch — 110, для A1300Eh — 111. Теперь становится понятным механизм смены игр, который можно условно назвать программно-аппаратным. Во-первых, в программу вводится команда обращения к порту A130xxh, приводящая к генерации импульса /SMS. Во-вторых, в картридж вводится микросхема регистра-защелки, который переключает старшие адреса ПЗУ по сигналу /SMS.

Чтобы "считыватель" SEGA-картриджей мог выявлять подобные ситуации, в его программу (см. листинг в [1]) следует внести изменения, выделенные в **листинге 2** жирным шрифтом. В электрической схеме "считывателя" (см. рис.1 в [1]) никаких изменений проводить не надо.

Листинг 2

```
...
printf ("составляет %d Мбит.",b);
/* Поиск дополнительных игр, выбираемых через меню */
for (z=f=0, c=128+MAX*2; z < b-1; c += MAX*2, z++)
{ k2[z] = kc (c,MAX/256);       /* Считывание кодов */
  if (srom[0]==0x53 && srom[1]==0x45 &&
    srom[0x10]==0x28 && srom[0x11]==0x43)
  { if (f == 0)                 /* Начальное сообщение */
    { printf ("\nДополнительно в картридже ");
      printf ("обнаружены игры,\nкоторые ");
      puts ("переключаются через меню:");
    }
    printf (" < ");     /* Левая открывающая скобка */
    for (d=0x50; d<0x80; d++)          /* 150...17Fh */
    { if (srom[d] > 32)             /* Поиск пробелов */
      { printf ("%c", srom[d]);     /* Печать симв. */
        p = 0;
      }                 /* Удаление лишних пробелов */
      else if (++p == 1) printf (" ");   /* Пробел */
    }
    puts (">");         /* Правая закрывающая скобка */
    f++;      /* Признак наличия дополнительных игр */
  }
}
/* Запись содержимого ПЗУ картриджа */
puts("\nПроизвести запись на жесткий диск <Y/N>?");
  ...
```

Упоминание о портах A130xx не встречается в литературе, более того, их не удалось обнаружить ни в одном из описаний архитектуры MD2, размещенных в Интернете. Наиболее близкими по смыслу оказались порты A130F3...A130FFh, которые отсутствовали в ранних моделях "Mega Drive". Через эти порты могут подключаться дополнительные банки памяти в восемь областей адресного пространства MD2, расширяя максимальную емкость картриджей с 32 до 128 Мбит. Ввиду высокой цены масочных ПЗУ с большим объемом памяти, данная возможность была реализована только в одном 40-мегабитном фирменном картридже с версией игры Street Fighter ("Capcom").

Порты переключения банков памяти занимают нечетные адреса A130F3, A130F5, A130F7, A130F9, A130FB, A130FD, A130FFh и позволяют оперировать областями фиксированного размера по 4 Мбит в каждой. В нашем же случае порты занимают четные адреса A13000...A1300Eh, а банки могут иметь варьируемый размер, зависящий только от объема программы. Можно предположить, что речь идет о недокументированных портах MD2, которые могли использоваться фирмой "SEGA" в отладочных комплексах для разработчиков игр. В любом случае следует отдать должное изобретательности китайских программистов, которые сумели воспользоваться оригинальным способом переключения игр через "секретные" порты MD2.

О том, что эти порты действительно не изучены, говорит тот факт, что ни один из эмуляторов MD2 на IBM PC [1] не может запустить игры из прошивки картриджа "Super 7in1". Попробуйте убедиться в этом сами, скачав файл кодов картриджа "Super 7in1" объемом 565 Кб, размещенный по адресу: http://emu-russia.km.ru/files/gen/7in1.zip.

Кстати, на сайте http://www.pe2000.net имеется прошивка аналогичного сборника игр "15in1", который также не работает на эмуляторах. В программе его загрузчика удалось обнаружить обращение к "секретным" портам, но в более широкой области A13000...A1301Fh. Объяснение тривиально — возросло количество переключаемых игр, а также удвоился объем ПЗУ картриджа. Коды загрузчика отличаются от приведенных в листинге 2, но логика работы осталась прежней — пересылка команд во внутреннее ОЗУ по адресу 0FF2000h, двойная запись данных в порт A130xxh и переход на начало программы. Налицо еще одно подтверждение существования программно-аппаратного способа переключения игр.

Литература

1. Рюмик С. Как получить прошивки ПЗУ SEGA-картриджей. — Радиомир. Ваш компьютер, 2002, №2, С.30.

От редакции: *исполняемый файл модернизированной программы "считывателя" SEGA-картриджей ("segarom3.exe", версия 3.0) размещен на сайте журнала (http://radio-mir.com).*

А.ГОРЯЧКИН,
г.Кыштым, Челябинской области.
E-mail: anatoliy_goryach@mail.ru

Мобильный домик для "винта"

Перед каждым пользователем, который по роду своей деятельности связан с работой на компьютере в офисе и дома, рано или поздно возникает вопрос о переносе больших объемов информации. Например, как в случае необходимости завершить в домашних условиях начатую в офисе срочную работу над большим проектом. Положение усложняется при отсутствии локальной сети. Передача же данных по сети Интернет удобна лишь при небольших объемах и на большие расстояния. Возможность архивации данных на дискеты оптимизма не прибавляет, особенно если объем данных составляет десятки, а то и сотни мегабайт. Применение CD-R/RW, во-первых, не позволит быстро скопировать нужную информацию, а во-вторых, связано с немалыми финансовыми затратами на приобретение устройства, позволяющего производить запись на CD-диски, и оправдано только при долговременном сохранении информации или при частом "клонировании" компакт-дисков. Твердотельные диски USB Flash Drive — вещь, конечно, удобная, но эти устройства пока еще относительно дороги (60 $), и поэтому не получили широкого распространения. Однако выход из этого затруднительного положения есть.

У многих пользователей после апгрейда компьютера остаются невостребованными жесткие диски объемом до 1 Гб. Применять их в современных компьютерах в качестве стационарного основного или дополнительного винчестера нецелесообразно из-за их малого объема. Тем более, что два стационарных жестких диска создают дополнительную нагрузку на блок питания и в процессе работы повышают температуру внутри корпуса системного блока. Однако эти винчестеры можно с успехом использовать для временного хранения и переноса данных с одного компьютера на другой.

Перед использованием жесткого диска в качестве съемного, его нужно предварительно подготовить. Для этого с помощью программ **fdisk.exe** и **format.com** необходимо убрать с винчестера все имеющиеся логические диски. В противном случае возможно возникновение проблем с именами дисков. Совсем не лишним будет и форматирование винчестера под файловую систему FAT 32. Не забудьте также установить на жестком диске джамперы в нужное положение. Съемный винчестер лучше всего подключать как "Secondary Slave", он устанавливается, как правило, как диск D. Для увеличения скорости обращения к жестким дискам включите для них режим DMA, открыв "Свойства системы — Устройства — Дисковые накопители" и затем свойства каждого диска. В таблице приведена наиболее оптимальная (по мнению автора статьи) конфигурация устройств с интерфейсом IDE.

Конфигурация устройств IDE	
Primary IDE 1	Master – жесткий диск
	Slave – CD-R/RW
Secondary IDE 2	Master – CD-ROM
	Slave – "mobile rack"

При использовании съемных жестких дисков возникают неудобства, связанные с частым снятием кожуха корпуса системного блока, установкой и подключением переносного винчестера к системе. Потом все приходится проделывать в обратном порядке. Если же вообще не закрывать системный блок кожухом, то внутрь будет попадать пыль, которая является главным "врагом" компьютера.

Для преодоления этих неудобств, разработаны и сейчас имеются в широкой продаже специальные устройства под названием "mobile rack". Это устройство, перешедшее из разряда диковинок в состав необходимых аксессуаров, представляет собой мобильное шасси для жесткого диска. В этой статье рассказывается о производимом в Китае mobile rack Gembird. Gembird — зарегистрированная торговая марка GMB Tech (Голландия). Стоимость этого устройства невелика — около 6 USD, а пользу от его использования трудно переоценить. Применение мобильного шасси позволит устанавливать и снимать жесткий диск, не открывая системный блок. Устанавливается оно в свободный пятидюймовый отсек (как правило, самый верхний) корпуса системного блока. На фрагменте из рекламного буклета показан внешний вид мобильного шасси для жесткого диска.

Mobile rack состоит из массивного пластикового корпуса — фрейма (frame) — с дополнительными ребрами жесткости, которые придают ему прочность и устойчивость. Внутри корпуса устройства имеется выдвижной модуль (картридж), в котором при помощи прилагающихся к

устройству винтов закрепляется жесткий диск. Для извлечения картриджа из корпуса на передней панели имеется удобная ручка, которая позволяет снимать и вставлять его без усилий. Выдвижной модуль используют для транспортировки картриджа с закрепленным в нем жестким диском. При транспортировке необходимо соблюдать осторожность. Помните, что жестким дискам противопоказаны удары и тряска. После снятия картриджа автоматически закрывающаяся шторка предохраняет системный блок от попадания пыли. Кроме жестких

дисков, в картридж можно устанавливать ZIP, MO и другие устройства с интерфейсом IDE. Для установки в картридж ZIP или MO необходимо снять решетчатую заглушку, находящуюся на передней панели. Переходный разъем, соединяющий мобильное шасси с компьютером, имеет позолоченные контакты и, по заявлению фирмы-изготовителя, пожизненную гарантию.

Кроме ручки переноса и решетки, на передней панели мобильного шасси находятся два светодиода, которые индицируют состояние устройства в картридже. Левый светодиодный индикатор свидетельствует о наличии питания, а правый загорается при обращении системы к жесткому диску. Дополнительный вентилятор, установленный на задней стороне мобильного шасси, отводит тепло от винчестера, оберегая его от перегрева. Механический замок обеспечивает безопасность сохранения данных. В комплект с мобильным шасси входят крепежные винты и два ключа от механического замка. На верхней пластиковой крышке выдвижного модуля имеется этикетка для нанесения каких-либо служебных надписей, например; "My Report for Year 2002".

При покупке mobile rack, во избежание недоразумений, необходимо выяснить у продавца, возможно ли подключение выдвижного модуля от данного мобильного шасси к уже ранее приобретенному, но другой фирмы-производителя. В противном случае лучше приобретать одинаковые мобильные шасси с последующей их установкой в компьютеры, между которыми будет осуществляться обмен информацией.

Установку мобильного шасси производят в следующей последовательности:

1. Отключите питание компьютера. Снимите кожух корпуса системного блока и заглушку, закрывающую свободный пятидюймовый отсек. Вставьте корпус мобильного шасси в отсек и закрепите его при помощи прилагающихся к устройству винтов. Подсоедините к устройству шлейф IDE и разъем питания.

2. Возьмите выдвижной модуль и снимите с него верхнюю пластиковую крышку. Подсоедините винчестер к короткому шлейфу IDE, идущему от переходного разъема, а также разъем питания. После этого закрепите винчестер с помощью винтов в картридже. Закройте картридж пластиковой крышкой. Вставьте картридж в мобильное шасси. Закройте корпус системного блока кожухом.

3. Включите питание компьютера. Войдите в программу Setup BIOS и произведите настройку конфигурации с учетом установленного дополнительного жесткого диска.

Кроме вышеописанного мобильного шасси, в продаже имеется mobile rack для винчестеров UDMA 66/100, но его стоимость несколько выше (10 USD). В нем установлены два вентилятора и имеется функция "Hot Swap".

Совет дня №1. Чтобы каждый раз при старте не входить в программу Setup BIOS после совершения манипуляций с подключением или отключением винчестера, в таблице определения параметров съемного жесткого диска включите опцию "Auto". Современные программы BIOS "умеют" автоматически определять конфигурацию компьютера.

Совет дня №2. А можно что-нибудь сделать, чтобы буквы дисков в Windows 9x не "съезжали" при установке дополнительного съемного винчестера? Можно. Делается это, когда съемный жесткий диск установлен, через "Панель управления – Система – Устройства – Устройства чтения компакт-дисков". Далее поочередно выбираются все имеющиеся устройства и их "Свойства". Во вкладке "Настройка" внизу есть закладка "Зарезервированные имена дисков". Нужно поставить в обоих окошках одинаковое значение имени диска (текущее) и перезагрузиться. Теперь, если съемный диск будет снят, система зарезервирует за всеми дисками те имена, какие были за ними при установленном съемном диске.

И в заключение. Использование мобильного шасси создает дополнительные удобства при работе с компьютером, в частности, при использовании дополнительного съемного жесткого диска. Устройство mobile rack несложно в эксплуатации, и может быть рекомендовано всем пользователям без исключения.

ОБМЕН ОПЫТОМ

Б.ШИЛЬНИКОВ,
Читинская обл., п.Дарасун.

Способ изготовления кабеля для подключения модема

Хочу поделиться способом изготовления кабеля для подключения модема (телефона). Потребуются кусок телефонного кабеля "витая пара" длиной 1...1,5 м, такой же длины ПХВ-трубка диаметром 4...5 мм и два телефонных коннектора.

Стандартный телефонный кабель имеет четыре "витые пары". Разрежем оболочку кабеля и вынем их. Для подключения модема достаточно двух проводов, т.е. одной "витой пары". Оденем на провода ПХВ-трубку так, чтобы концы проводов выходили из трубки на 1 см. На концах трубки подмотаем на провода полтора-два слоя ИЗОЛЕНТЫ для более надежного крепления проводов в коннекторе. Концы проводов распрямим и обрежем ровно, так чтобы они выступали из трубки на 4...5 мм. Зачищать изоляцию не нужно. Вставим подготовленные концы в коннектор так, чтобы они подключились к двум центральным контактам. Два крайних контакта не используются, а всего коннектор имеет место для 6 проводов. Инструментом для опрессовки аккуратно обожмем кабель в коннекторе. При небольшом навыке можно использовать и инструмент для опрессовки коннекторов RJ-45 для "витых пар". Положим коннектор защелкой вниз между губками пассатижей, и подходящим лезвием отвертки надавим на фиксатор так, чтобы зажать конец трубки в коннекторе. Аналогично опрессовываем другой конец кабеля. Кабель готов. Таким образом можно изготовить 4 модемных кабеля из одного куска телефонного кабеля "витая пара".

Использование избыточной информации для защиты файлов от повреждений

И.РОЩИН,
г.Москва,
Fido: 2:5020/689.53
ZXNet: 500:95/462.53
E-mail: asder_ffc@softhome.net
WWW: http://www.ivr.da.ru

Введение

Для долговременного хранения информации на "ZX-Spectrum" традиционно используются дискеты. Это не очень-то надежный носитель — наверняка вам не раз приходилось сталкиваться с ситуацией, когда при обращении к файлу появляется сообщение о дисковой ошибке, и прочитать файл не удается. Так что защита файлов от повреждений была бы очень кстати...

Теория

Все предельно просто. Пусть у нас имеется N L-битных чисел. Построим дополнительное число, представляющее собой результат поразрядной логической операции "Исключающее ИЛИ" над исходными числами: если общее число единиц в каком-то разряде четно, то соответствующий разряд результата равен нулю; если нечетно — единице.

Ниже приведен пример для случая N=4, L=10:

```
0110010110
0100111010
1000100111
0110100101
----------
1100101110
```

Это дополнительное число и будет избыточной информацией. Теперь, если любое из исходных чисел окажется утраченным, его можно будет восстановить по оставшимся числам и дополнительному числу. Для этого достаточно выполнить над ними ту же операцию "Исключающее ИЛИ", результат которой и окажется равным утраченному числу.

Пусть для рассмотренного примера утрачено, скажем, третье число. Вот как оно восстанавливается:

```
0110010110 ← 1-е число
0100111010 ← 2-е число
0110100101 ← 4-е число
1100101110 ← дополнительное число
----------
1000100111 ← восстановленное 3-е число
```

Можете сами убедиться, что данный способ сработает и при утрате другого числа — первого, второго или четвертого.

Защита файлов

Файл хранится на диске разбитым на небольшие блоки-сектора. В TR-DOS размер сектора — 256 байтов. При записи каждого сектора дисковый контроллер вычисляет и записывает на диск его контрольную сумму, а при чтении — вычисляет контрольную сумму прочитанных данных и сравнивает с ранее записанной. При несовпадении делается вывод, что сектор прочитан с ошибкой. Иногда после нескольких попыток такой сектор все же удается правильно прочитать, но не всегда. Бывает еще, что не удается прочитать заголовок сектора — тогда хранившаяся в этом секторе информация также оказывается недоступной.

Потеря данных в поврежденном файле обычно не бывает полной, часто "не читается" только один или несколько секторов, а другие остаются неповрежденными.

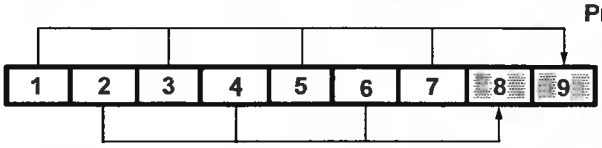
Пусть файл занимает n секторов. Каждый сектор можно считать 256-байтовым числом. Выполним над всеми n числами поразрядную логическую операцию "Исключающее ИЛИ", получив в результате также 256-байтовое число, и допишем его к файлу. Зная это число, можно, как описывалось выше, восстановить любое из утраченных чисел, если известны оставшиеся.

Иначе говоря, добавив к файлу один сектор избыточной информации, мы сможем восстановить любой поврежденный сектор этого файла, если информация в оставшихся секторах уцелела.

Если добавить к файлу два сектора, один из которых (с четным номером) содержит избыточную информацию для четных секторов, а другой (с нечетным номером) — для нечетных, то можно будет восстановить уже два поврежденных сектора файла, если один из них четный, а другой — нечетный (заметим, что этому условию удовлетворяют любые два смежных сектора).

На **рис.1** приведен пример для случая, когда длина исходного файла — 7 секторов. В секторе 8 содержится избыточная информация для секторов 2, 4 и 6, а в секторе 9 — избыточная информация для секторов 1, 3, 5 и 7.

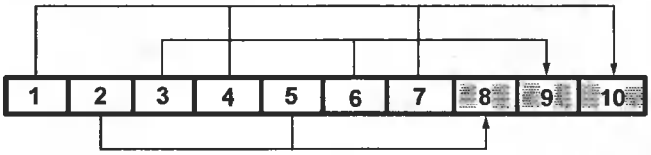
Рис. 1



Если в этом файле окажутся повреждены, например, сектора 2 и 3, то сектор 2 мы сможем восстановить по содержимому секторов 4, 6 и 8, а сектор 3 — по содержимому секторов 1, 5, 7 и 9. Если окажутся повреждены сектора 7 и 8, то надо восстановить только сектор 7 — это можно сделать по содержимому секторов 1, 3, 5 и 9. Заметим, что если бы сектора с дополнительной информацией располагались в другом порядке, восстановить сектор 7 было бы невозможно.

Аналогично, добавив к файлу (в соответствующем порядке) три сектора, содержащие избыточную информацию для секторов, номера которых делятся на три без остатка, с остатком 1 и с остатком 2 (**рис.2**), можно гарантировать возможность восстановления любых трех смежных секторов файла, и так далее.

Рис. 2



Можно разбить файл на участки (например, по 16 секторов) и вычислять избыточную информацию для каждого участка отдельно. Тогда можно будет восстановить файл, даже если повреждения будут в каждом участке.

Выводы

Раньше, чтобы обеспечить сохранность файлов, нам приходилось использовать резервное копирование или архивирование. Что нового дает использование избыточной информации?

Можно хранить вместо копий файлов только избыточную информацию для этих файлов. При этом потребуется значительно меньше дискового пространства, что немаловажно — во-первых, дискеты не бесплатны, а во-вторых, они постепен-

но становятся дефицитом, и если трехдюймовые дискеты еще можно встретить в продаже, то пятидюймовые — увы...

Недостаток этого способа в том, что можно исправить лишь ограниченное число поврежденных секторов файла. Впрочем, как уже упоминалось выше, в файле обычно бывают повреждены лишь один или несколько секторов.

Если надежность этого способа не кажется вам достаточной, и вы предпочитаете использовать резервное копирование или архивирование, то задумайтесь вот о чем — при повреждении файла мало удовольствия узнать, что копия тоже "не читается", не правда ли? Так что использование избыточной информации для защиты копий файлов или архива тоже имеет смысл.

Особенно это относится к архивам. Если в обычном файле будет поврежден сектор, то будет утрачена только информация из этого сектора, а если то же произойдет с архивом, то скорее всего будет утрачен весь конец заархивированного файла, на который пришлось повреждение. Дело в том, что обычно в процессе распаковки используется содержимое уже распакованной части файла.

К сожалению, возможность создания архивов с избыточной информацией для защиты от повреждений не реализована ни в одном из известных мне архиваторов на "ZX-Spectrum". А вот на PC такая возможность есть — например, в архиваторе RAR (опция "Add recovery record").

О программах

Да, а есть ли на "ZX-Spectrum" хоть одна программа для защиты файлов от повреждений с помощью избыточной информации? Увы... Мне, по крайней мере, они не попадались, и я даже не слышал об их существовании. Так что, если хотите защитить свои файлы, придется вам такую программу написать самостоятельно. Дело это не очень сложное, тем более, что некоторые полезные процедуры вы найдете в конце этой статьи.

В простейшем случае такая программа должна уметь создавать для указанных пользователем файлов (например, отмеченных в каталоге) файлы с избыточной информацией (возможно, расположенные на другом диске). Имена создаваемых файлов можно сделать такими же, как у исходных файлов, а расширение — "rec" (от "recovery record"). Обратная операция — восстановление указанных пользователем файлов с помощью файлов с избыточной информацией (возможно, расположенных на другом диске). Если полностью восстановить файл не удается, надо по крайней мере восстановить то, что можно. Необходимо предусмотреть запись восстановленных файлов как поверх исходных, так и на другой диск (на случай, если исходный диск имеет физические повреждения).

Желательно, чтобы программа предлагала выбор из нескольких вариантов формирования избыточной информации (один сектор для каждого файла, или несколько секторов, или один сектор на каждые 16 секторов файла, или как-нибудь еще) и приводила сведения о том, какие повреждения в каждом случае можно будет исправить.

Можете попробовать написать архиватор, в котором была бы возможность защиты создаваемых архивов от повреждений. В этом случае удобно хранить избыточную информацию в конце архива.

Можно использовать защиту файлов от повреждений и в какой-нибудь другой программе. Например, в текстовом редакторе можно при записи текста на диск автоматически записывать и rec-файл с избыточной информацией, если пользователем выбрана соответствующая опция.

Между прочим, с помощью избыточной информации можно защищать от повреждений не только файлы на диске, но и пересылаемую по некоторому каналу связи информацию, если она разбивается на пакеты, и часть пакетов может быть повреждена или потеряна (например, при пересылке разбитого на секции файла в FIDO или в ZXNet). Может быть выгодно передать несколько больший объем информации, но зато получатель не будет "переспрашивать" при каждой ошибке. Соответственно, еще одна область применения предложенной методики — коммуникационные программы.

Полезные процедуры

1. *Получение 256 байтов избыточной информации для группы 256-байтных участков, расположенных в памяти один за другим.*

Вход:

- B — количество участков в группе;
- HL — адрес начала первого участка (должен быть кратным 256);
- DE — адрес, по которому будет размещена избыточная информация (должен быть кратным 256).

```
MK_256REC   PUSH HL
            PUSH BC
            XOR  A

MK_256_1    XOR  (HL)
            INC  H
            DJNZ MK_256_1

            LD   (DE),A
            POP  BC
            POP  HL
            INC  L
            INC  E
            JR   NZ,MK_256REC

            RET
```

2. *Получение 512 байтов избыточной информации для группы 256-байтных участков, расположенных в памяти один за другим. Первые 256 байтов избыточной информации соответствуют участкам 1, 3, 5...; вторые 256 байтов — участкам 2, 4, 6...*

Вход:

- B — количество участков в группе (не менее двух);
- HL — адрес начала первого участка (должен быть кратным 256);
- DE — адрес, по которому будет размещена избыточная информация (должен быть кратным 256).

```
MK_512REC   PUSH BC
            INC  B
            CALL MK_512_1
            INC  D
            INC  H
            POP  BC

MK_512_1    SRL  B
MK_512_2    PUSH HL
            PUSH BC
            XOR  A

MK_512_3    XOR  (HL)
            INC  H
            INC  H
            DJNZ MK_512_3

            LD   (DE),A
            POP  BC
            POP  HL
            INC  L
            INC  E
            JR   NZ,MK_512_2

            RET
```

3. *Восстановление одного 256-байтного участка из группы таких участков, расположенных в памяти один за другим, по 256 байтам избыточной информации.*

Вход:

- B — количество участков в группе;
- C — номер восстанавливаемого участка, уменьшенный на 1 (т.е. для первого участка — 0).
- HL — адрес начала первого участка (должен быть кратным 256);
- DE — адрес, по которому находится избыточная информация (должен быть кратным 256).

```
US_256REC  LD   A,H
           ADD  A,C
           LD   XH,A

;XH - старший байт адреса восстанавливаемого участка.

US_256_1   PUSH BC
           PUSH HL

           LD   A,L
           LD   XL,A
           LD   A,(DE)

US_256_2   LD   C,A
           LD   A,H
           CP   XH
           LD   A,C
           JR   Z,US_256_3
           XOR  (HL)
US_256_3   INC  H
           DJNZ US_256_2

           LD   (IX),A

           POP  HL
           POP  BC
           INC  L
           INC  E
           JR   NZ,US_256_1

           RET
```

4. *Восстановление одного 256-байтного участка из группы таких участков, расположенных в памяти один за другим, по 512 байтам избыточной информации. Первые 256 байтов избыточной информации должны соответствовать участкам 1, 3, 5...; вторые 256 байтов — участкам 2, 4, 6...*

Вход:

- B — количество участков в группе;
- C — номер восстанавливаемого участка, уменьшенный на 1 (т.е. для первого участка — 0).
- HL — адрес начала первого участка (должен быть кратным 256);
- DE — адрес, по которому находится избыточная информация (должен быть кратным 256).

Если надо восстановить два участка из группы (тогда один из участков должен быть четным, а другой нечетным), то перед вторым вызовом процедуры все входные параметры должны быть установлены заново, так как процедура не сохраняет первоначальные значения соответствующих регистров.

```
US_512REC  LD   A,H
           ADD  A,C
           LD   XH,A

;XH - старший байт адреса восстанавливаемого участка.

           INC  B
           BIT  0,C
           JR   Z,US_512_1
           INC  H
           INC  D
           DEC  B
US_512_1   SRL  B

US_512_2   PUSH BC
           PUSH HL

           LD   A,L
           LD   XL,A
           LD   A,(DE)

US_512_3   LD   C,A
           LD   A,H
           CP   XH
           LD   A,C
           JR   Z,US_512_4
           XOR  (HL)
US_512_4   INC  H
           INC  H
           DJNZ US_512_3

           LD   (IX),A

           POP  HL
           POP  BC
           INC  L
           INC  E
           JR   NZ,US_512_2

           RET
```

5. *Формирование в памяти избыточной информации для файла. Создается 256 байтов избыточной информации на каждые 16 полных или неполных секторов файла. Эта процедура использует описанную выше процедуру MK_256REC.*

Вход:

-B — длина файла в секторах;

- DE — трек-секторный адрес начала файла.

```
BUF_16_SEC EQU  #8000  ;Буфер для загрузки 16 секторов файла.
BUF_RESULT EQU  #9000  ;Буфер для избыточной информации.

;Длина и того, и другого буфера - #1000 байтов; адреса буферов
;могут быть другими, но обязательно кратными 256.

MAKE_INFO  LD   HL,BUF_RESULT
           LD   (ADR_256),HL

MAKE_I_1   LD   C,16
           LD   A,B
           SUB  C
           JR   NC,MAKE_I_2
           LD   C,B
MAKE_I_2   PUSH BC

           PUSH DE
           LD   B,C

;Сейчас в регистре B - количество читаемых секторов (16 или
;меньше), в DE - трек-секторный адрес, откуда читать.

           PUSH BC
           LD   HL,BUF_16_SEC
           PUSH HL
           LD   C,5   ;Номер функции "чтение секторов".
           CALL #3D13
           POP  HL
           POP  BC

;Сейчас HL=BUF_16_SEC. Формируем для прочитанного участка файла
;256 байтов избыточной информации:

           LD   DE,0
ADR_256    EQU  $-2
           CALL MK_256REC
           INC  D
           LD   (ADR_256),DE

;Увеличиваем трек-секторный адрес:

           POP  DE
           INC  D

;Повторяем, пока не будет обработан весь файл:

           POP  BC
           LD   A,B
           SUB  C
           LD   B,A
           JR   NZ,MAKE_I_1

           RET
```

Литература

1. М.А.Карцев. Арифметика цифровых машин. — М.: Наука, 1969.

Быстрая печать спрайта
(Не дай себе засохнуть!)

А.СИЗЕНКО,
Украина, г.Луганск,
E-mail: pat_2000@mail.ru

Для быстрой печати спрайтов, за неимением команды LD (HL),NN (она могла выполняться за 16 тактов), используем команду LD (HL),N (10 тактов), что, конечно, менее удобно. Для печати каждого (!) спрайта используется собственная процедура, занимающая 112 байтов. Для простоты расчета начала процедуры "растягиваем" ее до 128 байтов.

Наиболее целесообразно такие "наборы подпрограмм" поместить в какую-нибудь страницу памяти. Таким образом удается разместить 128 спрайтов. Если же необходимо "иметь под рукой" большее количество спрайтов, используем две страницы памяти. При этом придется предварительно переключать нужную страницу, что будет немного подтормаживать работу всей системы. Например, для страниц 1 и 3 это будет выглядеть следующим образом:

```
        LD      BC,#7FFD
        LD      E,1
        BIT     7,A
        JR      NC,M1
        RES     7,A
        LD      E,3
M1      OUT     (BC),E
```

Вначале рассмотрим "быстрое" умножение на 16. Делаем это "старым" способом:

```
        LD      L,A     ;4 ; помешаем число в регистровую пару
        LD      H,0     ;7 ;
        ADD     HL,HL   ;11; умножаем на 2
        ADD     HL,HL   ;11; *4
        ADD     HL,HL   ;11; *8
        ADD     HL,HL   ;11; *16
        ADD     HL,HL   ;11; *32
```

Итого — 66 тактов.

Что происходит при умножении на 32? Старшие четыре бита "переезжают" в регистр H, а младшие становятся на их место. Отсюда:

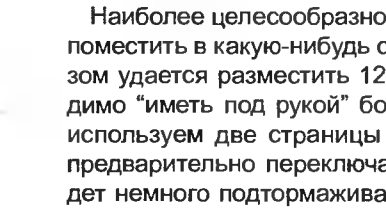
```
        RRC A   ;4 ; убиваем сразу двух зайцев
        RRC A   ;4 ; формируя одновременно старший и младший
        RRC A   ;4 ; полубайты
        RRC A   ;4
        LD H,A  ;4 ; временно сохраним
        AND #F0;7  ; "тушим" младшие биты
        LD L,A  ;4 ; младший байт готов
        LD A,H  ;4 ; восстановили
        AND #0F;7  ; "тушим" старшие биты
        LD H,A  ;4 ; старший байт готов
```

Итого — 46 тактов, или выигрыш в скорости — 30,36%.

Далее привожу только расчеты, сведенные в таблицу.

Входные параметры для процедуры:
- HL — адрес первого знакоместа;
- DE — адрес третьего знакоместа;
- BC — старшие адреса атрибутов первого и третьего знакомест (младшие будем брать с видео).

Чтобы избежать лишних команд INC/DEC L, знакоместа будем выводить на экран "змеевидно":



и т.д.

Итак, текст процедуры:

```
        LD      A,L     ;запоминаем младший байт для вывода атрибутов
                        ;первая строка
        LD      (HL),N  ;первый байт
        INC     L       ;следующее знакоместо
        LD      (HL),N  ;второй байт
        INC     H       ;и опускаемся ниже
                        ;вторая строка
        LD      (HL),N  ;
        DEC     L
        LD      (HL),N
        DEC     H
        --------------;и т.д.
        LD      (HL),N  ;до 16 строки
        DEC     L       ;
        LD      (HL),N  ;

        LD      L,E     ;теперь адрес третьего знакоместа грузим в HL
        LD      H,D     ;

        LD      (HL),N  ;все аналогично - 17 строка
        INC     L       ;
        LD      (HL),N
        DEC     H
        LD      (HL),N
        DEC     L
        LD      (HL),N
        DEC H
        ------------- ;и т.д.
        LD      (HL),N  ;до 32 строки
        DEC     L       ;
        LD      (HL),N  ;

        LD      L,A     ;далее адрес атрибутов
        LD      H,B     ;
```

Умножение на	Количество тактов старым способом	Количество тактов предлагаемым способом	Выигрыш в скорости, %	Комментарии
8	44	42	4,55	Выполняем три команды RRC A. Байты маски соответственно будут #F8 и #07
16	55	46	16,36	
32	66	42	36,36	"Крутим" A в другую сторону (RLC A)
64	77	38	50,65	
128	88	34	61,36	
256				По понятным причинам не рассматривается

Процедура печати одного спрайта 2 x 2 знакоместа

Схематично такой спрайт можно представить так:

1	2
3	4

```
        LD      (HL),N ;первое знакоместо
        INC     L      ;
        LD      (HL),N ;второе
        LD      L,E    ;
        LD      H,C    ;
```

```
        LD      (HL),N ;третье
        INC     L      ;
        LD      (HL),N ;четвертое
        RET            ;возвратились
```

Проиллюстрируем применение данной процедуры.

Здесь применим несколько модифицированный способ "быстрого умножения" на 128:

```
        XOR     A              ; обнуляем A и очищаем флаг переноса
        LD      L,A            ; L=0
        RRA                    ; формируем старший байт
        RRL                    ; младший бит "переносим" в L
                               ;
        ADD     A,#C0          ; добавляем к старшему начальный
                               ; адрес страницы
        LD      H,A            ; старший байт готов
                               ;
        LD      (BODY+1),HL    ; грузим в тело программы адрес
                               ; обращения к п/п
                               ; или LD ($+13),HL
        LD      HL,#4021       ; начальные данные
        LD      DE,#4041       ;
        LD      BC,#5858       ;
BODY    CALL    #0000          ; вывод спрайта
```

М.ФАХРИЕВ,
г.Ижевск.

Welcome to life

Азартные игры — адреналин в крови, бешеные деньги, судьбы, поставленные на кон.

И вот, когда вы в двух шагах
От сказочных богатств,
Он говорит вам:
Хэ-хэ, бог подаст...
Песня из мультфильма "Остров сокровищ"

Азарт игры выплескивается из казино на улицы и, что самое страшное — захватывает детей и проникает в школы. Все это порождает преступность. Чтобы увлечь детей и дать возможность играть в домашних условиях с куском маминого пирога в руках, мною написана небольшая игровая программа на Бейсике для компьютера "ZX-Spectrum", имитирующая игру "в кости". После запуска программы командой RUN выберите режим игры — игра с компьютером или игра с партнером. Вбрасывание кубиков осуществляется нажатием клавиши "R" на клавиатуре.

Листинг программы

```
10  REM "DA MOSCOW RAP IS BULLSHIT LTD"
20  CLS
30  PRINT AT 1,1; "WELCOME TO LIFE"
40  PAUSE 20
50  PRINT AT 2,1; "YOU WANNA PLAY WITH SYNDICATE
                   OR YOU PLAYING  ON THE STREET
                   (1 OR 2)"
60  INPUT A
70  IF A=1 THEN GOSUB 100
80  IF A=2 THEN GOSUB 230
90  GOTO 60
100 CLS
110 PRINT AT 3,1 "LET'S GET DEATH..."
120 INPUT A$
130 IF A$="R" THEN GOTO 150
140 GOTO 120
150 LET Y=INT(12*RND)
160 LET M=INT(12*RND)
170 IF Y<M THEN GOSUB 400
180 PRINT AT 6,2 "SOMETIMES I LOOSE..."
190 PRINT AT 7,2; M
200 INPUT "CONTINUE? (Y/N)"; A$
210 IF A$="Y" GOTO 100
220 NEW
230 CLS; PRINT AT 11,10 "IS ON!"
240 PRINT AT 5,1 "PLAYER 1"
250 INPUT A$
260 IF A$="R" THEN GOTO 280
270 GOTO 250
280 LET G=INT(12*RND)
290 PRINT AT 6,1; G
300 PRINT AT 6,10 "PLAYER 2"
310 INPUT A$
320 IF A$="R" THEN GOTO 350
340 GOTO 310
350 LET J=INT(12*RND)
360 PRINT AT 7,10; J
370 INPUT "CONTINUE? (Y/N)"; A$
380 IF A$="Y" THEN GOTO 230
390 NEW
400 PRINT AT 6,2; "YOU PLAYED YOURSELF"
410 PRINT AT 7,2; Y
420 GOTO 200
```

КОМПЬЮТЕРНЫЕ ХРОНИКИ

И.РОЩИН,
г.Москва,
Fido: 2:5020/689.53
ZXNet: 500:95/462.53
E-mail: asder_ffc@softhome.net
WWW: http://www.ivr.da.ru

BestView 2.13

Итак, вышла очередная версия BestView — программы для просмотра и запуска файлов на "ZX-Spectrum". Какие же улучшения появились в новой версии?

Добавлена возможность удаления файлов (правда, пока только по одному). Очень удобно — посмотрел содержимое файла и, если он не нужен, тут же удалил.

Появилась необходимая в некоторых случаях возможность просмотра любого файла в виде текста.

Теперь, когда в ClipBoard'е не хватает места для размещения всего отмеченного текста, BestView сигнализирует об этом изменением цвета бордюра.

В записываемых файлах обнуляется неиспользуемое пространство в конце последнего сектора (о важности этого я уже упоминал в [1]).

Есть и другие, не столь значительные усовершенствования.

Также исправлены замеченные ошибки. При этом размер программы не только не вырос, но, напротив, уменьшился на три сектора!

Программа доступна на крупных спектрумовских сайтах, а также я могу выслать ее по E-mail. При желании ваш адрес может быть добавлен в список рассылки, и вы будете получать новые версии BestView сразу после их выхода.

Литература

1. И.Рощин. Обрубаем файлам "хвост". — Радиомир. Ваш компьютер, 2002, №4, С.37.

Солдат Удачи II: Двойная Спираль

ELROND,
E-mail: elrond@land.ru

Наконец-то вышло долгожданное продолжение "самой кровавой игры", и вышло оно, как мне кажется, на славу.

Итак, **Soldier of Fortune II: Double Helix**. Чем же эта игра отличается от своего знаменитого предшественника?

Прежде всего, конечно, графикой. Движок SoF-2 основан на блестяще переработанном Quake III: Arena engine. Думаю, это название говорит само за себя. Пейзажи стали более реалистичными, враги — почти как настоящие.

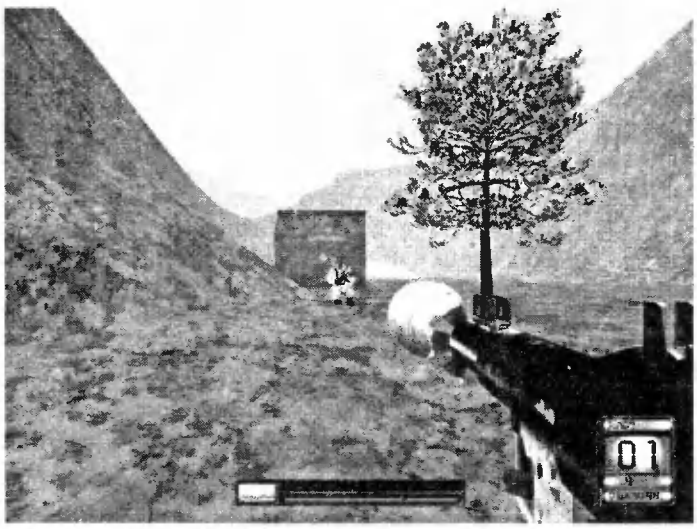
Улучшена также система повреждений GHOUL — теперь она называется GHOUL II. Но это, к счастью, не единственное изменение. Если раньше тела противников были поделены на тринадцать зон, то теперь их 36, что значительно повышает реалистичность процесса боя.

Но все же в этой "бочке меда" не без "ложки дегтя". Например, возникают вопросы, когда врага удается убить с помощью точного выстрела в... пятку. Времена ахиллесовой пяты прошли, да и патроны в обоймах вряд ли отравлены. Другой вопрос — поразительная точность при стрельбе. Практически из любого оружия можно с первого раза попасть в голову врага, силуэт которого еле-еле виднеется за завесой дождя, кучность стрельбы почти идеальна даже из автоматического оружия.

Однако это компенсируется повышением уровня AI. Противники не нарываются на пули, а пережидают в укрытии, и время от времени ведут оттуда прицельный огонь.

Отдельная тема — траектория полета тел убитых врагов. В ближнем бою со многими противниками это практически незаметно. Но если издалека выстрелить в солдата, стоящего на краю какой-нибудь вышки, то можно понаблюдать, как его обмякшее тело будет красиво планировать с высоты.

Оружия в игре немало, оно разнообразно, и его выбор порой является главным фактором прохождения



очередной миссии. Кроме того, ко многим видам оружия можно добавлять определенные части — глушитель, лазерный прицел и т. д. Особенно хорош штык-нож к автомату Калашникова. Даже самые простые пистолеты имеют альтернативный огонь — под названием "рукояткой по голове". Также введены такие понятия как "опасное" и "безопасное" оружие. Опасным оружием нужно пользоваться осторожно — оно обладает высокой убойной силой, и при неумелом обращении может ранить своего хозяина.

Оружие следует применять в зависимости от противостоящего врага. Например, если дробовик налево и направо "выносит" незащищенных противников, то против врага в бронежилете его КПД сильно падает.

Большее внимание в игре стало уделяться бесшумности проведения операций (вот где часто может пригодиться штык-нож АК). Кроме приседания, добавлен способ передвижения ползком по-пластунски. Передвигаясь таким образом, можно спрятаться в траве и незаметно проползти к месту назначения. Ну а если вас все-таки заметили, здесь понадобится что-нибудь из "опасного" оружия.

В игре появилась возможность взламывать некоторые запертые двери. На это, конечно, требуется время. Хотя это и ближе к реальности, но все же иногда действует на нервы, особенно когда приходится еле живым убегать от толпы жаждущих смерти несчастного Джона Маллинза террористов/бандитов/просто плохих парней.

В SoF-2 можно играть как на прохождение (набор миссий, сюжет

которых продолжает первую часть игры), так и в одиночную или многопользовательскую игру. Традиционно большое количество настроек позволяет настроить игру, исходя из лучшей производительности или лучшего качества изображения.

И, конечно, чтобы увидеть все прелести игры, нужно играть на самом высоком уровне сложности. Именно тогда и станет ясно, что Солдат Удачи II — это не простая стрелялка-мясорубка, а игра с интеллектуальным подходом, где главную роль порой играет не мощное вооружение, а правильно выбранная позиция или время для атаки, умение действовать молниеносно и незаметно.

Вывод — в целом, я думаю, сиквел удался; точнее, получился образцовый сиквел. Таким и должно быть продолжение — с улучшенной графикой, большим количеством оружия и с разнообразными мелочами, которые не отягощают игрока, а еще больше приближают игру к реальности и поднимают ее на более высокий уровень.

Минимальные системные требования

Процессор Pentium® II 450 МГц или Athlon®
ОС Windows® 98/ME/2000/XP
128 Мб ОЗУ
1,3 Гб на жестком диске (плюс 140 Мб после установки)
Видеокарта AGP 16 Мб
DirectX 8.1

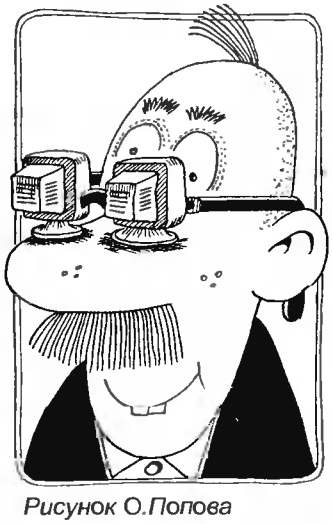


Рисунок О.Попова

Сага о поедании пуда соли с Henchman'ом

К.КЛИЩЕНКО,
г.Минск.

И мир смотрел на него
миллиардом своих воплощений.
"Неизвестный эпиграфер"

Был когда-то такой продукт — вроде игра, а вроде — и алфавит для Эллочки-людоедки. Назывался он AD&D Unlimited Adventures и позволял создавать свои собственные игры Gold Box-серии. Честно говоря, мне не удалось застать его в момент выпуска, чтобы насладиться созиданием на той технической базе (1992 г., мне одиннадцать лет, ну какой из меня созидун?). Так вот, Neverwinter Nights это, грубо говоря, "вышеупомянутый продукт"-2 (правда, в отличие от первой игры, в NN можно озадачить не только своего соседа, но и группу этих самых соседей в локальной сети. Можно также отвлечь пользователя Интернета от чата и порнографии на online-игру).

Вопрос о необходимости написания этой статьи — это почти что главный вопрос философии (соотношение материального и духовного). Ведь никто не станет обвинять программистов, создавших Soundforge, в том, что прилагаемые сэмплы не слишком напоминают произведения Баха или Хансена (это глубоко личное, и не трогать грязным скепсисом). Или примеры корректной установки шрифта, напоминающие о необходимости постоянной поставки французских булок в неограниченных количествах (особенно мягких). При всем уважении к маленьким батонам имени Марсельезы, я не могу сравнивать их с Шекспиром. Так и эта игра, на первый взгляд, является лишь демонстрацией возможностей редактора. В таком контексте описание "Ночей" будет техническим, а писать "Neverwinter Nights для чайников" я не собираюсь.

Заняться рассуждениями о ценности вложенного повествования меня заставили три причины. Во-первых, большинство пользователей даже не возьмутся за создание своих произведений (стремление стать новыми Канегемами или Брэдли есть не у всех, да и труда нужно вложить даже в небольшой модуль немало, а это не всем под силу). Потому им нужна не только потенциально великолепная игра, но и сиюминутное наслаждение. Во-вторых, в игру вложена не только сила разума, но и душа, а марание души и есть долг критики. Ну а в-третьих, что, я зря получал удовольствие в различных порциях в течение полутора недель?

Не будучи любителем пространных, в треть статьи, словесных излияний по поводу фабул, а тем паче заставок, я не стану вас утомлять этими самыми фабулами. Ясно только главное — в городе Неверуинтер (транскрипция ужаснейшая) эпидемия, и надо найти лекарство. А дальше история вьет сама себя — красивая, душевно-пафосная, но до слез реальная. Реальность ее, конечно, для многих эфемерна, но не меньше и тех, кто воспримет ее с должным чувством, и судьбы ее героев пересекутся с вашей. Нет, я вас не заставляю понять Арибет, сочувствовать Голгофе, решать проблемы жителей детской игрушки..., но попробуйте, может, вам понравится сказание. Тогда не стоит читать дальше... Конечно, прекрасна история, но у нее есть и техническая часть, которую стоит рассмотреть беспристрастно (по крайней мере, попытаться).

К сожалению, у авторов не получилось сделать историю равномерно плывущей (или падающей — кому как нравиться). Очень интересные и цепляющие за душу третья и четвертая части не оставляют

и тени воспоминаний о чересчур Might&Magic-овских (в худших его проявлениях, то есть девятых) первой и второй. Построение чересчур стандартно — "а не принесешь ли ты мне четыре забубовины", "а потом еще три афэлочины". Ладно бы только это, ведь и третья часть, на первый взгляд, похожа на них, но у начальных есть еще и реализация на уровне "пойду вырежу все вокруг" (а точнее, в строго выделенном направлении). Справимся, ведь правда, и не такое выносили, тем более, что сюжет скармливается в приемлемых порциях. Постоянно знаешь, что и зачем делаешь (хотя любителям квестов такого подхода не понять). Но в принципе, такой скудный рацион поначалу оправдывается принципиальным взглядом большинства людей на развитие как таковое. Ведь никто сразу же (ну почти никто) не доверит "слабэнкому, но гордому" герою общение с такими тонкими структурами как время и параллельные пространства.

Статья выдается рвано, периодический наплыв чувств и вдохновения сменяется нечетким будущим не начатых абзацев. Но впрочем, обратим свои взоры куда-нибудь в иное...

Именно NN помогла мне почувствовать расхваливаемую D&D эпику. Bioware удалось практически невозможное — сделать игру, которая стоит в нужном количестве шагов от позора РПГ, называемого Diablo, и дарящую непередаваемое чувство героизма. Враги не лезут толпами, но и не являются редкостью, доступной лишь пытливому приключенцу. Моя пара полуорков-варваров свободно превращается в универсальную нарезку Бернера, действия которой захватывают дух почище Блэйда или Конана. Именно в силу этой самой эпики на игру и сыплются обвинения в смещении баланса. В любой другой игре такое обвинение могло бы быть рассмотрено под микроскопом и после этого с позором (Arcanum) или восторгом (Severance) вынесено наружу волнами общественного сознания. В NN даже глупо об этом начинать вести речь. Не нравится? Ты можешь отредактировать сценарий, и даже маленький гоблин пооткусывает коленные чашечки высокоуровневой группе, прежде чем та позовет на помощь. К тому же, усилиями этого самого "смещенного" баланса меня привязало к игре с такой страшной силой. Единственным же по-настоящему дисбалансирующим элементом в игре является stone of recall — специальный манчкинский предмет. Он позволяет переместить вас практически из любого места в тепло храма, где вас "на халяву" вылечат. Таким образом, можно чуть ли не каждый хит подлечивать. Но это ведь нечестно. Хотя в порыве "еще и еще игры" немагический персонаж может достаточно долго прошляться без отдыха. Как видите, игра предоставляет неограниченные возможности в подлости по отношению к ней. Бой за мага (по-видимому, гораздо более интересный, ввиду многообразия спелятника) просто не попался мне под руку, а поэтому, извините, очень уж хотелось высказаться по горячим следам еще дымящейся тушки Мораг. К тому же, даже при игре за варвара было доступно удивительное многообразие тактик (особенно с учетом факта мультионлайновости игры).

Приятно заметить, что диалоги привлекают содержательностью и богатством вариантов развития. Лгать, убеждать и подмечать дано не каждому, нужно и харизмой выйти, интеллект развить (ну хотя бы получить о нем представление). Хотя стремление угодить каждому и здесь дает о себе знать. Можно не обращать слишком много внимания на диалоги и пронестись сквозь игру с кровавой пеной у рта. При этом NPC не перестают неявно подчеркивать вашу незначимость, хотя под конец вы становитесь даже не пупом отдельно взятого мира, а этим самым местом еще в нескольких мирах.

И главную веху, отделяющую single от непосредственной ориентации игры, составляет Henchman. Это не привычный NPC, не ждите историй Ариэль или клятв Минска, о летающих черепах и освобожденных рабах тоже забудьте. Henchman — это помощник, на первый взгляд, не сильно отличающийся от Diablo-вского (не чересчур ли часто я поминаю великую аркаду, может, это голос разума пробивается сквозь пелену эмоций?). Но вы заблуждаетесь, на самом деле его короткая, но выразительная история не оставит вас в стороне от его стремления или корыстного предложения помочь вам. Подчас он действует полезнее вас, и вы выживаете лишь его страданиями. Требует же он, в лучшем случае, аки младенец, бутылочку, да изредка словцом перекинется с вами. Но не забывайте — вы с ним сиамские близнецы, и если сильно захотите играть без него, то NN потеряет изрядную долю привлекательности, особенно в шаге от финала...

P.S. Браться за создание модуля нужно лишь хорошенько подготовившись. Так как при всем его кажущемся удобстве, он требует тщательной подготовки, особенно если вам не хочется делать AD&D based-аркаду. Язык скриптов напоминает геймеру об ужасах программирования.

Простите, совсем позабыл, но ничего, оставлю постскриптум в середине. О технической части спою вам песню. Графика, безусловно, хороша (на мощных компьютерах), особенно при работе с моделями (мой знакомый в течение нескольких минут восхищался драконами) и освещением (четыре рядом идущих по подземелью людей с факелами создают буйство теней, подвластное лишь реальности). Убедительно смотрятся четыре торчащих из щита горящих стрелы. Блокирование "меч в меч" сопровождается убедительными искрами и лязгом. Да здравствуют музыка и звуки, реализация которых поражает даже на фоне графики.

P.S.(P.S.) На самом деле статью можно было писать совершенно подругому — подробно описывая реализацию звука и графики, вносить в нее и собственно игровой элемент.

P.P.S. Верю, что удалось заинтриговать, к тому же, я твердо рассчитываю заниматься созданием модулей не в одиночку.

P.P.P.S. Не верьте мне, меня купили.

P.P.P.P.S. Хотя лучше верьте. Купили-то меня дорого, за хорошую игру.

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

*Для публикации бесплатных объявлений **некоммерческого характера** о покупке и продаже компьютерных комплектующих и программ, их текст можно присылать в письме по адресам:*

119454, г.Москва, а/я 37 или
220095, г.Минск-95, а/я 199,

*передавать по тел. в Минске **(017) 221-01-10**, либо через E-mail:*

rl@radiopage.by
rm@radio-mir.com
*WWW: **http://radio-mir.com***

Ученик 11-го класса с благодарностью **примет в дар** компьютер и комплектующие.
397600, Воронежская обл., г.Калач, ул.Гагарина, 13, Лебединский Алексей.
Тел. 27-3-36.

Вышлю схему изготовления кабеля для подключения любых сотовых телефонов к компьютеру (микросхема Max232 заменена на 2 транзистора КТ315).
E-mail: n_mar@tut.by

Продаю видеокарту S3 Savage3D (8Mb, AGP 2x, TV-out — композит и S-video), поддержка оверлея, поддержка 3D, диск с последними драйверами.
Тел. в Минске 258-36-33.

Куплю б/у CD-ROM.
E-mail: taras000@atlasua.net

Бесплатно **вышлем** каталоги радиосхем, радиодеталей, книг, радиоконструкторов, видео- и аудиокассет, дискет с подборкой полезных программ и каталог на CD по радиолюбительству. К заявке на каталог необходимо прикладывать оплаченный конверт с обратным адресом.
422981, Татарстан, г.Чистополь, а/я 248, Романов А.А. (B.R.S.)

Куплю контроллер жесткого диска для ПК "Искра 4816.03", программное обеспечение для ПК "Искра-1030М".
453500, Башкортостан, г.Белорецк, а/я 21, Сафонов С.Н.
Твл. (34792) 4-45-67.

Ищу схему и описание EPSON-совместимого принтера EC-7245 (применялся в КУВТ "КОРВЕТ").
E-mail: zgo@mail.chita.ru

Куплю: игровую приставку "Panasonic 3DO" модели FZ-10; видео-CD-адаптер (68 pin x 1); игровые компакт-диски системы 3DO; джойстики; пистолеты; мышь, можно от других приставок (GoldStar 3DO, Sonyo 3DO и т.д.) с логотипом 3DO; карту Creative Labs 3DO Blaster и другие устройства для "Panasonic 3DO".
681000, Хабаровский край, г.Комсомольск-на-Амуре, ул.Краснофлотская, 26 — 42, Решетник К.П.

Куплю для ПК "Scorpion ZX-256 turbo" плату GMX, контроллер SMUS, контроллер IBM-клавиатуры и Kempston-mouse с документацией, а также ПО.
681090, Хабаровский край, Комсомольский р-н, ст.Гурское, ул.Владивостокская, 4 — 1, Курбатов Н.В.
Тел. 3-91.

Ищу программу или описание способа установки и изменения элементной базы для программы-симулятора "Electronics Workbench EDA".
191123, г.С.-Петербург, а/я 12, Ильин А.

Приму в дар любой компьютер и комплектующие.
456430, Челябинская обл., Уйский р-н, п.Вишневка, Бичан С.

Школьник с большой благодарностью **примет в дар** любые комплектующие к IBM-совместимым компьютерам и цифровым видеокамерам.
Беларусь, г. Барановичи, ул. Фабричная, 14 — 84.

Приму в дар компьютер или комплектующие.
E-mail: zkr@freemail.ru

Приму в подарок б/у компьютер без монитора.
40034, Украина, г. Сумы, ул.Коротченко, 15/311. В.Стороженко.

Бесплатно высылаю схемы на любые темы и любой сложности, с подробным описанием и рисунком печатной платы. Для получения каталога схем высылайте конверт с обратным адресом.
357081, Ставропольский кр., Андроповский р-н, ст.Воровсколесская, ул.Центральная, 91. Ширяев А.

Куплю контроллер дисковода "Электроника МС8414" для ПК "Электроника МС1502".
E-mail: mishsv@om.msi.ru

Уважаемые читатели!

Подписаться на наши журналы (индексы: **48996** —"Радиомир", **48924** — "Радиомир. КВ и УКВ", **48925** — "Радиомир. Ваш компьютер") в I полугодии 2003 г. можно по каталогу Агентства "Роспечать", **стр. 283**, или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь" (раздел "Издания РФ, распространяемые по прямым договорам").

Те, у кого возникли проблемы с подпиской, могут получить их из редакции. Там же можно заказать имеющиеся в наличии отдельные номера журналов за предыдущие годы.

Год	Можно заказать следующие номера журналов			Расценки на 1 экз. (с учетом пересылки)		
	Радиолюбитель	Радиолюбитель. Ваш компьютер	Радиолюбитель. КВ и УКВ	По России и СНГ (рос. руб.)	По Беларуси (бел. руб.)	По Украине (гривны)
1998	3, 5, 8	1 — 5, 8 — 12	1, 6, 7, 11, 12	25	800	6
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	1 — 3, 5, 6	30	1000	7
2000	2 — 5, 7 — 12	1 — 8, 10 — 12	4 — 7, 9 — 12	35	1200	8
2001	2 — 6	1 — 6	2 — 6	40	1400	8,5
	Радиомир	Радиомир. Ваш компьютер	Радиомир. КВ и УКВ	По России и СНГ (рос. руб)	По Беларуси (бел. руб.)	По Украине (гривны)
2001	7 — 12	7 — 12	7 — 12	40	1400	9
2002	1 — 12	1 — 12	1 — 12	45	1500	10

Наши платежные реквизиты:
- для жителей России и стран СНГ (кроме Беларуси) —
получатель: ООО "Радиомир Пресс", ИНН 7729404741,
р/с 40702810900010000084 в ООО КБ "Агропромкредит", ф-л Центральный, г.Москва,
корр. счет 30101810500000000109, БИК 044525109
Адрес банка: 113054, г.Москва, ул.Зацепа, 43Б;
- для жителей Беларуси —
получатель: УП "РЛД", УНН 190218688
р/с 3012000004882 в ф-ле №524 АСБ "Беларусбанк" в г.Минске код 121.
Адрес банка: 220028, г.Минск, ул.Физкультурная, 31.

На бланке почтового перевода необходимо очень четко написать свой почтовый индекс, полный адрес, фамилию, имя и отчество полностью. В графе "Для письма" необходимо точно перечислить, какие конкретно номера какого из журналов Вы заказываете.

При оплате платежным поручением нужно предварительно выписать счет.

Вся информация по E-mail: rm-sales@radio-mir.com или по тел./факсу в г.Минске (017) 221-01-10

Приобретение отдельных номеров журналов

В РОССИИ:

В **ЗАО "Интерпресс — Экспресс"**: тел. в Москве (095) 208-85-55, 208-67-55, e-mail: inter@aif.ru

В магазинах радиодеталей **"ЧИП и ДИП"**:
- г.Москва, ул.Гиляровского, д.39 (ст. метро "Проспект Мира" — радиальная);
- г.Москва, ул.Ивана Франко, д.40, к.1, стр. 2 (платф.Рабочий поселок, 15 мин. от Белорусского вокзала);
- г.Москва, ул.Беговая, д.2;
- г.Ярославль, ул.Нахимсона, 12, тел. (0852) 27-57-15.

На радиорынках в Москве: Митинском (места R4, S8, K52, E50, G56). Царицынском (место 121).

В **ООО "ТРЭНТЭКС"**: 111024, г.Москва, 4-я Кабельная, 2А (ст. метро "Авиамоторная"), тел. (095) 258-91-94, 258-91-95. E-mail: abook@mail.ru, abook@inbox.ru
Радиорынки: Царицынский (место 13/А), Митинский (ряд 8, контейнер 12, место Z25).

В **"Фонде содействия радиолюбителям-радистам"**: г.Москва, 4-й Войковский проезд, 10, тел. (095) 150-09-72, 150-04-16 (ст. метро "Войковская").

На УКРАИНЕ:

В Киеве в фирме **"Торм"**, тел. (044) 227-72-73.

В БЕЛАРУСИ:

В Минске в магазине **"Книга XXI век"**, пр.Ф.Скорины, д.92, тел. (017) 264-27-97 (ст.метро "Московская").

САГА О ПОЕДАНИИ ПУДА СОЛИ С HENCHMAN'ОМ