ISSN 0869-2343

Журнал Д-ра Доббса

RUSSIAN DR. DOBB'S JOURNAL

№ 3' 1991

Русский Парадокс

Стандарт языка С

АВТОРСКОЕ ПРАВО НА ПРОГРАММЫ

СТРАННЫЙ РЕЖИМ РАБОТЫ АДАПТЕРА VGA

СОВЕТСКИЙ СУПЕРКОМПЬЮТЕР: ВОЗМОЖЕН ЛИ ОН СЕГОДНЯ?

T9000
ТРАНСПЬЮТЕРЫ НОВОГО ПОКОЛЕНИЯ

Clipper 5.0

ЗНАЕТ ЛИ КОМПЬЮТЕР ПОЭЗИЮ?

ПО Орловский завод управляющих вычислительных машин им. К.Н. Руднева

ЛОКАЛЬНАЯ СЕТЬ *"ЛИНИЯ"*

ПО "Орловский завод управляющих вычислительных машин им. К.Н. Руднева" предлагает кольцевую локальную сеть "ЛИНИЯ", основанную на маркерном методе доступа, которая обеспечивает работу ПК СМ 1810.60, НЕЙРОН И9.66, IBM PC XT/AT в среде MS-DOS.

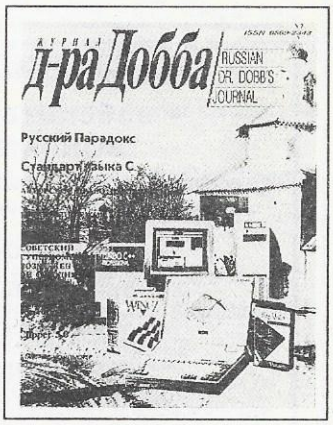
- *Передающая среда - коаксиальный кабель/витая пара*
- *Удаление между станциями - 1 км/200 м*
- *Скорость передачи данных - 1 Мбайт/с*

"ЛИНИЯ" позволяет разделять физические и программные ресурсы любого ПК в сети. Возможна установка защиты программных данных. Цена одного подключения - примерно 2,2 тыс. руб. (зависит от комплектности). Телефоны: 3-84-98, 3-12-10 (г. Орел)

Содержание

Журнал д-ра Добба

Компьютерный журнал для всех



Журнал
зарегистрирован
Министерством печати
и массовой
информации РСФСР
Свидетельство
о регистрации № 791
Учредитель -
трудовой коллектив
издательства "ТРИАДА"
Главный редактор
А.А. Эйдес
Зав. редакцией
О.В. Ксендзацкая
Художественные
редакторы
А.М. Свердлов,
С.Я. Грозовская
Компьютерная верстка:
М.М. Егорова
Производство
и распространение:
Л.Н. Морозова,
Н.М. Крупенченок

ТРИАДА
TRIAD

Рукописи
не рецензируются
и не возвращаются.

Адрес для переписки:
125190, Москва
А-190, а/я 240

МКК
ICC

Государственная ассоциация
предприятий, объединений
и организаций полиграфической
промышленности «АСПОЛ».
Ярославский полиграфкомбинат,
150049, Ярославль, ул. Свободы, 97.
Зак. № 491.

Объем 10,6 уч.-изд.л.
Цена 3 руб. 50 коп.

©"Журнал д-ра Добба", 1991

Профессиональные системы защиты информации для IBM-PC фирмы NOVEX Technology, Ltd.

NOVEX TECHNOLOGY

File_PROTECTION™

- победитель конкурса "BORLAND-Contest'91"

- создание самовосстанавливающихся программ
- защита программ:
  - — от нелегального копирования ;
  - — от изучения логики их работы;
  - — от внесения в них изменений;
  - — от нападения файловых вирусов;

File_PROTECTION™

- защищает программы форматов .com и .exe - на "винчестере" - на дискетах 5"25 и 3"5, размеченных в форматах 320K, 360K, 400K, 720K, 806K, 1.2M, 1.36, 1.44M, 1.6M, используя их как ключевые;
- защищает программы от дизассемблирования и трассировки;
- обнаруживает любые файловые вирусы (по факту заражения) и выдает диагностику;
- производит "лечение" заразившейся программы;
- позволяет устанавливать счетчик запусков программы;

Система прошла тестирование в Тестовой Лаборатории Международного Компьютерного Клуба.

Поставка -1 дистрибутивная дискета 5"25 -Руководство Пользователя.

Цена - 1500 руб.

FP_installator™

- генератор дистрибутивных дискет, с которых защищенное программное обеспечение может быть "выгружено" на жесткий диск сгенерированной утилитой install.exe заданное число раз;
- поддерживает режим "реинсталляции";
- инсталляционные дискеты имеют специальную структуру, не воспроизводящуюся при любых видах копирования (CopyIIPC,Copywrite, Option Board..), а защищенные программы не работоспособны до их "выгрузки" на жесткий диск;
- позволяет размещать на дистрибутивных дискетах файлы любых типов и любых размеров и иметь в одном дистрибутиве любое число дискет;
- обеспечивает подготовку и сохранение "модели" создаваемого дистрибутива;
- обеспечивает установку счетчика инсталляций с изготавливаемого дистрибутива;
- в защищаемые программы "вживляется" вакцина системы File_PROTECTION.

FP_installator позволяет избежать использования обременительного в работе ключевого диска, являющегося атрибутом многих известных систем.

Стоимость системы зависит от количества дистрибутивов, которые с ее помощью можно изготовить: 5–390 руб., 10–715 руб., 15–990 руб., 25–1525руб.,50–2765 руб., 100–5015 руб., и т. д.

Lock_MANAGER™ – системный драйвер (LM.SYS),для защиты конфиденциальной информации от несанкционированного доступа, "нелегального" копирования, случайного (или умышленного) стирания. Осуществляет динамическое кодированиеие защищаемой информации при записи на диск, оставаясь "прозрачным" для системных и прикладных задач.

- *позволяет защищать текстовые файлы, базы данных (например dBASE, Clipper, Paradox, AutoCAD, PCAD и т.д.), данные электронных таблиц, графику и т.д.;*
- *обеспечивает доступ к защищенной информации только ее законному владельцу;*
- *обеспечивает защиту информации от модификации (например, от подделки банковских счетов);*
- *дает возможность получать раскодированную копию файла только его владельцу;*
- *позволяет запретить печать информации или ее передачу по сети.*

Драйвер защищен от копирования и попыток вмешательства в его работу, а защищаемые файлы не изменяются в размере и не содержат никакой служебной информации, дающей ключ к разгадке.

Стоимость драйвера - 1285 руб.

Оптовым покупателям и дилерам предоставляются значительные скидки.

NOVEX Technology, Ltd.
129010, Москва, пр. Мира, 18, оф. 220
Телефоны **(095) 588-57-50** днем,
511-38-11 вечером

Окно в Купертино

Сенсация! Произошел прорыв на российском фронте разработки программ - СП "ПараГраф" прорубило окно в международный рынок! Переведем дыхание после восклицаний и изложим факты.
7 октября одновременно в Москве и Боулдере (США, шт. Колорадо) было объявлено о том, что фирма Apple Computer из Купертино и СП "ПараГраф" из Москвы подписали соглашение о совместной разработке программы распознавания рукописного текста.
Работы по созданию технологии распознавания рукописного текста велись в СП "ПараГраф" с 1989 г., и поначалу многие считали, что понадобятся они, в основном, для рекламы или для развлечения публики на всевозможных выставках и форумах.
Сейчас положение изменилось - после весенней выставки Comdex'91 в моду быстро вошли компьютеры с рукописным или, как его еще называют, "бесклавишным" вводом (pen-based). Резко возрос интерес к алгоритмам распознавания и обработки рукописного текста.
Тут-то и выяснилось, что именно СП "ПараГраф" находится на переднем крае, и руководство фирмы Apple приняло решение, которое еще совсем недавно показалось бы парадоксальным - обратилось за помощью к специалистам из Москвы.
Об эффективности программ, разработанных в СП "ПараГраф", пока можно судить лишь косвенно - эксперты фирмы Apple, наверное, хорошо подумали, прежде чем сделать выбор.
Однако не следует забывать, что в последние годы не всегда решения руководства этой фирмы были оптимальными (что, на мой взгляд, и привело к нынешнему сближению фирмы Apple с ее заклятым конкурентом - фирмой IBM), поэтому более точную информацию наши читатели смогут получить, если СП "ПараГраф" предоставит свои программы для тестирования в "Пробирной палате" "Журнала д-ра Добба".
Как бы там ни было, настал-таки и для СП "ПараГраф" звездный час: название не сходит со страниц печати, а президент - с экранов телевизоров. Отдает должное этому совместному предприятию и "Журнал д-ра Добба". В предыдущем номере мы хвалили его инициативу - проведение пресс-конференции по случаю выпуска нового программного продукта, в данном номере посвятили этому продукту отдельную статью, а его замечательное название ("Русский Парадокс"), которое стало особенно актуальным после августовских событий, украсило обложку.
Мне кажется, что то, о чем я говорил до сих пор, как нельзя лучше сочетается с помещенной на обложке идиллической картинкой русской природы. В заключение - несколько ложек дегтя, не для того, чтобы испортить с таким трудом собранную СП "ПараГраф" бочку меда, а чтобы уровень проводимых им рекламных мероприятий соответствовал уровню разрабатываемых им программ.
О подписании исторического соглашения между Apple и "ПараГрафом" С.А. Пачиков впервые поведал в новой телепередаче "Компьютер-инфо" (кстати, передача мне показалась интересной - информативной и динамичной), упомянув между прочим, что фирма Apple является "третьей фирмой в мире".
В пресс-релизе, сообщающем об этом соглашении, про фирму Apple сказано еще проще: "крупнейшая американская компьютерная компания". Заканчивается пресс-релиз фразой, достойной дословного воспроизведения: "ParaGraph JV является основным в Москве и имеет привелегии *(орфография пресс-релиза - А.Э.)* в продолжении разрабатывания и исследования по линии продуктов распознавания". Не хотелось, чтобы эти, мягко говоря, непроверенные факты ставили под сомнение достоверность и всех остальных обстоятельств, связанных с безусловно важным и, наверное, почетным для российской индустрии программирования соглашением.

А.А. Эйдес
Главный редактор

с о б

Начало традиции

Фирма Intel распространила пресс-релиз, датированный 30 сентября 1991 г., в котором сообщается, что расширено семейство микропроцессоров 80C186 - выпущено три новых 16-разрядных микропроцессора:

80C186EC (функционально заменяет 20 электронных компонентов, повышая надежность и экономя пространство);

80C186EA (обеспечивает 80%-ное сокращение потребляемой мощности по сравнению с процессором 80C186);

80C186XL (производительность на 25% выше, чем у процессора 80C186, а потребляемая мощность в два раза меньше).

Эти процессоры отличают высокий уровень интеграции, низкая потребляемая мощность, низкая стоимость. Эти устройства должны прежде всего заинтересовать разработчиков модемов, контроллеров локальных сетей, дисководов.

В этом году фирма Intel поставит заказчикам около 10 млн. микропроцессоров семейства 186, причем можно заказывать процессоры как с 16-разрядной, так и с 8-разрядной шиной.

В пресс-релизе приводятся и другие технические характеристики новых устройств, о которых, если это заинтересует читателей, мы расскажем более подробно в следующих номерах "Журнала д-ра Добба". А сегодня хотелось бы рассмотреть стратегию деятельности фирмы Intel на советском рынке, которая должна существенно измениться.

Во-первых, тот факт, что о выпуске группы новых процессоров в Москве было объявлено уже 26 сентября - даже чуть раньше, чем в Санта-Кларе, Нью-Йорке, Токио, Париже и Лондоне, говорит о многом. Управляющий делами фирмы в СССР Дмитрий Александрович Ротов, назначенный на этот пост в июле нынешнего года, заявил, что и впредь о важнейших событиях, происходящих в фирме Intel, в России будут узнавать не позже, чем в США, Франции или Японии. Таким образом, фирма Intel выступила инициатором новой и очень полезной для всех специалистов нашей страны традиции. Надеемся, что примеру застрельщицы почина последуют и другие ведущие фирмы и не за горами то время, когда Россия перестанет быть "информационной провинцией".

Во-вторых, фирма Intel намеревается продавать свои изделия на советском рынке без ограничений и по тем же ценам, что и в США. Основным партнером фирмы в России был назван центр "Техно" Министерства авиационной промышленности. Однако двери открыты для всех, и фирма с удовольствием вступит в переговоры с солидными партнерами.

В-третьих, фирма собирается обосноваться в нашей стране "всерьез и надолго", поэтому одну из неотложных задач видит в том, чтобы открыть представительства в Москве и (возможно) в Санкт-Петербурге, причем персонал этих представительств должен быть сформирован из местных специалистов, которые лучше разбираются во внутренней ситуации, а уж стратегическое руководство фирма Intel обеспечит!

Таковы в сокращенном изложении ближайшие цели и задачи фирмы Intel в России, изложенные ее полномочным представителем Д.А. Ротовым.

В заключение несколько слов о самом Дмитрии Александровиче. По его словам, первый миллион долларов он "сделал" в 22 года, в 26 - все потерял, а потом проделывал это (правда, неясно, что именно: "делал" или терял, или и то, и другое) еще несколько раз. Без отрыва от предпринимательства Д.А. Ротов закончил Отделение ядерной физики Гарвардского университета, работал в Принстоне (по-видимому, в "почтовом ящике" американского образца). Последние три года служит в фирме Intel. Предки Дмитрия Александровича в начале века выехали из России. Дед его - из казаков, а один из пращуров даже участвовал в Бородинской битве (на стороне русской армии).

А. Аба

Новые достижения в применении транспьютеров

С 28 по 30 августа в Глазго (Великобритания) проходила третья ежегодная международная конференция "Применения транспьютеров". По числу представленных докладов и участников (более 130 докладов из 25 стран) эта конференция была самой крупной за последние годы.

В прозвучавших на конференции докладах отражены последние достижения в применении транспьютеров в таких областях, как обработка сигналов и изображений, системы связи, робототехника, биотехнология, промышленные системы управления, графические системы, базы данных и др.

Впервые на конференции такого уровня были представлены доклады и из нашей страны.

На выставке, проходившей параллельно с конференцией, ведущие изготовители транспьютерной техники из Великобритании, Германии и США представили свои новейшие разработки. Все ожидали, что фирма INMOS, наконец, покажет "живьем" свой транспьютер T9000 (см. статью об этом транспьютере на с. 34-37), однако, кроме макета транспьютерного модуля нового формата на базе T9000, на стенде фирмы ничего примечательного не было.

"Гвоздем" выставки была презентация фирмой Texas Instruments (США), впервые участвовавшей в подобной конференции, микропроцессора TMS320C40 нового поколения, который, обладая значительно большей, чем у транспьютера T800, вычислительной мощностью (на уровне вычислительной мощности микропроцессора i860 фирмы Intel), в то же время по своей архитектуре близок к нему. Наличие у микропроцессора TMS320C40 шести встроенных параллельных двунаправленных каналов ввода-вывода, подобных транспьютерным линкам, позволяет строить на основе этого микропроцессора параллельные суперкомпьютеры с высокими технико-экономическими показателями.

Как следствие вступления на рынок параллельных вычислительных систем такого гиганта, как фирма Texas Instruments, практически все известные транспьютерные фирмы объявили о выпуске аппаратных средств на базе микропроцессора TMS320C40 и соответствующего программного обеспечения. Например, фирма 3L (Великобритания), известная своими компиляторами языков программирования Си, Фортран, Паскаль для транспьютеров, объявила о разработке для микропроцессора TMS320C40 компилятора, аналогичного по возможностям компилятору 3L C, а фирма Perihelion (Великобритания) - о выпуске соответствующей версии своей операционной системы Helios. Английские фирмы Paratech и Transtech показали на выставке платы для персональных компьютеров на базе этого микропроцессора.

Аналогичная картина наблюдалась и год назад, когда практически все транспьютерные фирмы выпустили транспьютерные модули с микропроцессором i860. Особую позицию

ЫТИЯ

занимает фирма Parsytec (Германия), за несколько лет превратившаяся в одного из крупнейших изготовителей суперкомпьютеров на основе транспьютерной технологии.

Демонстрируя свою приверженность транспьютерам, фирма показала на выставке фрагмент нового суперкомпьютера на базе транспьютера T9000. Если фирма INMOS не подведет с выпуском нового транспьютера, то фирма Parsytec попытается к 1993 году первой в мире создать суперкомпьютер с производительностью 10^{12} операций с плавающей запятой в секунду, опередив тем самым на два года своих американских и японских конкурентов. Чтобы достичь такой производительности, фирме Parsytec придется решить проблему надежного функционирования вычислительной системы, состоящей из 65 тысяч транспьютеров! Поскольку все основные европейские изготовители суперкомпьютеров на базе транспьютерной технологии (английские Meiko и Parsys, французские Telmat и Archipel) согласились участвовать в этом проекте, у Европы появляются хорошие шансы занять лидирующие позиции на рынке суперкомпьютеров.

Следующая конференция по транспьютерным приложениям, впервые состоявшаяся за пределами Великобритании, в Барселоне (Испания), с 21 по 25 сентября, в 1993 году переедет в Германию.

Есть некоторые надежды, что в 1994 году конференция может состояться в Москве. Возможно, в порядке подготовки к этому событию Советская транспьютерная ассоциация совместно с акционерным обществом "Центр параллельных систем и технологий" с 8 по 12 октября провели в Звенигороде первую конференцию "Транспьютерные системы и их применение", в которой помимо наших специалистов участвовали представители пяти региональных транспьютерных центров Великобритании и ряд английских транспьютерных фирм (подробности в следующем номере).

В.А. Лопатин

"Пилотный проект" в образовании

24-26 сентября 1991 г. в Алма-Ате проходил III Международный симпозиум "Новые информационные технологии в образовании", организованный Государственным комитетом СССР по народному образованию, фирмой IBM World Trade (Europe/Middle East/Asia) и Министерством народного образования Казахстана. Основная тема - реализация "Пилотного проекта", осуществляемого в нашей стране совместно с корпорацией IBM, а участники - преимущественно представители учебных заведений и других организаций, участвующих в проекте. Этот проект составляет, в свою очередь, часть обширного проекта "Культура, наука, образование: США - СССР", Меморандум о взаимопонимании по которому был подписан во время визита Президента М.С. Горбачева в США в июне 1990 г.

"Пилотный проект" направлен на совершенствование системы образования в нашей стране на основе внедрения современных методов обучения с использованием ПК и инструментальных программных средств фирмы IBM и, в частности, предусматривает оснащение нескольких тысяч учебных заведений разного уровня компьютерными классами на базе ПК семейства IBM PS/2. Руководит проектом с советской стороны Геннадий Ягодин (Председатель бывшего Госкомитета СССР по народному образованию), с американской - проект курирует Первый вице-президент корпорации IBM Майкл Армстронг (Michael Armstrong).

В "Пилотный проект", рассчитанный на период до 2000 г., входит несколько субпроектов. И если год назад работа велась по трем субпроектам - "Пилотные школы", "Нетрудоспособные дети и инвалиды" и "Школы бизнеса" (см. журналы "Интеркомпьютер" №№ 5,6 за 1990 г. и "Байтик" №№ 2,3 за 1991 г.), то теперь в стадии подготовки или реализации находится уже девять субпроектов, охватывающих различные уровни образования - начиная с детского сада, через среднюю школу, ПТУ, техникум, вуз и кончая (послевузовским) инженерным образованием.

Сейчас завершается первый этап "Пилотного проекта" (1990-91 гг.). Создана инфраструктура, состоящая из 42 республиканских и региональных центров поддержки. Каждый центр оснащен компьютерным классом (12 компьютеров IBM PS/2 моделей 30 и 50) и имеет в своем составе специалистов, получивших сертификацию фирмы IBM. Компьютерными классами на базе IBM PS/2 оборудованы более 1000 средних школ, 37 школ для детей с дефектами зрения и слуха, 24 вуза, несколько ПТУ и техникумов. Проведена переподготовка 2,5 тыс. преподавателей.

Вторая фаза проекта (1992-93 гг.) предусматривает наряду с дальнейшим развитием его инфраструктуры оснащение подобными компьютерными классами еще более 2,5 тыс. учебных заведений. Ведутся переговоры о совместном производстве в нашей стране компьютеров IBM PS/2, предназначенных прежде всего для сферы образования (потребности которой оцениваются в 5 млн. ПК). Общая стоимость проекта до 1995 г. составит 205 млн. дол.

Конечно, внедрение компьютеров в сферу образования в СССР началось не с "Пилотного проекта". Уже в 1985 г. в школах был введен курс информатики (на этом настоял тогда акад. А.П. Ершов), и к настоящему времени компьютерными классами оборудовано около 25 тыс. школ, большое число других учебных заведений. К сожалению, установленная в этих классах отечественная вычислительная техника не отвечала и не отвечает современнымм требованиям к информатизации учебного процесса: ни по надежности, ни по санитарно-гигиеническим нормам (ученик может работать за дисплеем лишь 20 мин в неделю), ни по объему и качеству программного обеспечения, а ее несовместимость с доминирующими в развитых странах мира семействами ПК еще более усугубляет отставание. "Пилотный проект" позволяет реализовать новую стратегию информатизации в сфере образования, обеспечивая ее интеграцию с современными компьютерными технологиями, вхождение в европейские и мировые информационные сети.

Уже не существует страны СССР, в которой начиналась реализация "Пилотного проекта", и Госкомитета СССР по народному образованию, представители которого подписывали договоры с фирмой IBM, но важность и перспективность этого проекта для образования ничуть не уменьшилась, поэтому присутствовавшие на симпозиуме в Алма-Ате представители из 13(!) республик бывшего Союза направили Президентам всех республик совместное Обращение.

Смысл этого Обращения - принять решения, обеспечивающие сохранение "Пилотного проекта" и продолжение совместной координированной работы.

А.Д.Плитман

PC-WRITE LITE
"Диетический" программный продукт

А. В. Петроченков

Когда говорят об отечественном рынке программных продуктов, не всегда сразу можно понять, о чем же по сути дела идет речь. Ведь рынка такого у нас еще не существует. Пока еще никто не страдает от слишком активного предложения товара и уж тем более не видно профессиональных приемов маркетинга по формированию спроса среди перспективных пользователей программных продуктов.

Тем не менее в последнее время признаки формирования рынка программных средств, особенно некоторых их видов, становятся все заметнее. Очевидно, уже можно говорить об отечественном рынке текстовых процессоров, предназначенных для работы с текстами на русском языке.

У многих тысяч пользователей ПК очень популярен редактор текста "ЛЕКСИКОН", причем его последние версии пополнились рядом полезных возможностей, свойственных более мощным текстовым процессорам. В октябре минувшего года в Москве представители фирмы Microsoft и СП "Диалог" объявили о выпуске русифицированной версии интегрированного пакета MS Works 2.0, в котором имеется весьма мощный текстовый процессор универсального применения. В феврале нынешнего года в Мюнхене фирма Microsoft и СП "ПараГраф" объявили о совместном выпуске под торговой маркой "Русское Слово" русифицированных версий профессионального текстового процессора Microsoft Word *5.0* и *5.5* в комплекте с русификатором. На прошедшей в апреле этого года в Москве выставке "КОМТЕК'91" была представлена русифицированная версия мощного профессионального текстового процессора WordPerfect *5.1*, работа над которой завершится в самое ближайшее время - отлаживается работа встроенной программы проверки орфографии. Представители фирмы WordStar, присутствовавшие на выставке "КОМТЕК'91", тоже не скрывали своей заинтересованности в поиске партнера, способного адаптировать пакет WordStar 6.0 для советского рынка.

Таким образом, уже сейчас возникает реальная конкуренция среди текстовых процессоров для ПК, которая наверняка будет возрастать в дальнейшем на благо потребителям. Правильный выбор потенциальным пользователем подходящего текстового процессора может оказаться довольно непростой проблемой.

Александр Васильевич Петроченков - технический писатель, автор статей о различных программных продуктах для персональных компьютеров. Занимается также переводами технической документации, подготовкой руководств, справочников и каталогов, предназначенных для пользователей программных продуктов.
Его адрес: 214000, Смоленск, а/я 44
Телефон: (08100)5-58-05

Когда "маленький" оказывается "большим"

Стремление создавать все более громоздкие и сложные программы становится в последнее время все ощутимее.

Как показывает практика, далеко не всегда оправданно приобретение таких программных монстров, какими являются профессиональные текстовые процессоры. Сложность этих программных продуктов не может не пугать новичка: кажется, будто можно угробить полжизни, прежде чем научишься эффективно управлять такими программами.

Как это ни странно, легкие и маневренные программы могут оказаться более удобными, чем их громоздкие конкуренты, хотя последние и обладают разнообразными возможностями. Увы, эти громко рекламируемые возможности нередко оказываются буквально "неслыханными", ибо далеко не каждому пользователю ПК хоть однажды приходится воспользоваться и половиной из них. Далеко не каждому отечественному писателю, журналисту или аспиранту сможет когда-либо понадобиться все богатство протрясающих воображение возможностей пакетов MS Word и WordPerfect. По поводу этих невероятных возможностей впору бы написать хвалебную оду, но вот только писать такую оду следовало бы с помощью чего-то более изящного и послушного.

Чтобы установить на ПК пакет WordStar, Вам придется поочередно вставить в дисковод дюжину дискет. Не говоря уже о том, что на жестком диске не прибавится свободного пространства, за все это богатство придется платить (и немало). MS Word *5.0* - это тоже добрая дюжина дискет, а "РусскоеСлово" - даже 17 дискет, и хотя заплатить за них можно в рублях, но не так уж и дешево.

Однако на рынке есть разработчики программных продуктов, понимающие, что часто меньшее оказывается бóльшим. Эта парадоксальная формула истинна, когда речь идет о программных продуктах, предназначенных для не самых совершенных моделей ПК (каких в нашей стране большинство) или для машин с ограниченными техническими характеристиками (например, для портативных компьютеров). Эта формула также истинна во всех случаях, когда программой пользуются новички либо те, кто в действительности вовсе не нуждается во всех мыслимых и немыслимых возможностях дизайна, верстки и форматирования текста, но предпочитает простой, понятный и удобный инструмент для каждодневной творческой работы над содержанием текста, а не над его внешним обликом.

Метод shareware не знает запретов

Одной из фирм, сумевших разработать удивительно простой, быстрый и недорогой, но довольно мощный текстовый процессор, оказалась фирма Quicksoft (Сиэттл, шт. Вашингтон).

Впервые разработанный нынешним ее президентом Бобом Уоллесом (Bob Wallace) еще в 1983 году этот текстовый процессор, получивший название PC-Write, распространяется в США, Канаде, Австралии и Новой Зеландии методом shareware.

Благодаря простоте и удобству текстовый процессор PC-Write оказался сегодня одним из самых распространенных пакетов в США. По оценке экспертов, число легальных пользователей этого пакета превышает число пользователей пакетов WordStar, WordPerfect и Microsoft Word вместе взятых! Если учесть, что в США сегодня используются свыше 50 млн. персональных компьютеров, этот успех громаден не только в относительном сравнении, но и по абсолютной величине, хотя она точно не известна никому, так как регистрируются не все пользователи shareware-программ. Не случайно компьютерные журналисты в США нередко в порыве восторга называют Боба Уоллеса гениальным программистом и ставят его в один ряд с Биллом Гейтсом и Питером Нортоном.

Регистрационный взнос за последние версии пакета PC-Write составляет 99 дол., вполне умеренную сумму по американским стандартам. Надо отметить, что, хотя этот пакет распространяется методом shareware только в названных выше странах, он прекрасно известен почти во всем мире. Например, в ФРГ у официальных дистрибьютеров можно приобрести как английскую, так и немецкую версию пакета PC-Write, а многие "компьютерные" издательства считают необходимым выпускать подробные руководства и справочники по работе с ним. Известен и многими любим этот пакет и в нашей стране, хотя официально он никогда в СССР не продавался, и поэтому у нас круг его пользователей, вероятно, гораздо уже, чем в других странах.

PC-Write 3.0 - последняя версия пакета. Она явно оказалась перегруженной различными дополнительными возможностями, вплотную приблизившими ее к "разжиревшим" профессиональным коммерческим продуктам. Боб Уоллес явно не устоял перед соблазном попытаться удовлетворить все возможные пожелания пользователей. И хотя версия PC-Write 3.0 очень хороша в работе, эта попытка Боба Уоллеса, похоже, оказалась не самой плодотворной.

Борьба с ожирением

Очередной всплеск восторженных отзывов в прессе по поводу успехов фирмы Quicksoft произошел в 1990 году, когда было объявлено о выпуске "облегченной" программы PC-Write Lite 1.0, унаследовавшей все лучшие достоинства программы PC-Write. Полная версия программы PC-Write Lite занимает на диске всего 330 Кбайт, хотя для ее работы достаточно лишь одного файла LITE.EXE, размер которого не превышает 150 Кбайт.

Журнал "Computing Now!" писал в марте 1990 года: "Фирма Quicksoft выпустила с конвейера обновленную версию программы PC-Write, словно новую компактную гоночную машину: быструю и мощную. Программа PC-Write Lite, обладая всеми возможностями предшественницы, просто фантастически быстра. Ну почему же, почему другие текстовые процессоры не обладают такой скоростью?"

И действительно, PC-Write Lite даже на ПК с процессором 8088 работает явно быстрее, чем другие известные программы на ПК с процессором 286. Кроме того, скорость работы этой программы особенно бросается в глаза во время молниеносной загрузки и выхода из программы после завершения работы.

"PC-Write Lite - это изящная и богатая возможностями версия классической shareware-программы PC-Write", - писала газета "The Washington Post". А другая газета - "San Francisco Chronical" - утверждала, что программа попроще, чем классическая программа PC-Write, но сделано это умышленно.

"Компактность этой программы делает ее очень подходящей для учащихся, пользователей портативных ПК и для всех начинающих пользователей ПК", - писала газета "The Seattle Times", а журнал "BYTE" подтверждал: "Если Вам нужен простой текстовый процессор, рекомендуем обратить внимание на PC-Write Lite. Программа развивает отличную скорость, но не прожигает дыру в Вашем кармане."

Журнал "Compute!" написал в июне 1990 года: "Если Вы объелись сложными текстовыми процессорами и потеряли аппетит к работе, попробуйте диету PC-Write Lite. Действует потрясающе и без каких-либо неприятных ощущений."

Действительно легко!

Lite переводится с английского, как "легкий". Прошу поверить, так оно и есть, когда речь идет о PC-Write Lite: полистав лишь с полчаса превосходно изданное руководство пользователя, включающее в себя подробный учебник, я установил программу на своем компьютере и сразу же без проблем приступил к срочной работе, словно всю жизнь проработал именно с этим текстовым процессором!

Достаточно скоро я узнал, что текстовый процессор PC-Write Lite хоть и "выступает в наилегчайшем весе", но отнюдь не слабак. Здесь есть практически все, что нужно для эффективной творческой работы. Интерфейс прост и ненавязчив: кроме меню в верхней строке экрана, весь остальной экран в Вашем полнейшем распоряжении, словно чистый лист бумаги!

Программа позаботится, чтобы каждые пять минут файл вашего документа автоматически записывался на диск, так что неожиданные отключения электроэнергии, погубившие уже столько творческих усилий и озарений, Вам почти не угрожают. PC-Write Lite позволяет перед печатью предварительно увидеть схематическое расположение текста на странице. Программа автоматической проверки английской орфографии со словарем из 50 000 слов может чрезвычайно легко пополняться неограниченным числом новых слов из Ваших текстов.

Управлять текстовым процессором можно как из меню, так и с помощью комбинаций клавиш.

Можно смело утверждать, что возможности макрокоманд программы PC-Write Lite точно такие же, как у любого другого текстового процессора. Такое утверждение может показаться просто нахальным, но ничуть не расходится с истиной: за полчаса любой пользователь может изменить схему расположения командных клавиш программы, чтобы работать с ней в привычной и удобной для него манере. Перекодировка и настройка программы осуществляются редактированием отдельного маленького файла ED.DEF, что довольно просто и не вызывает проблем.

Shareware

Метод распространения программных продуктов shareware предполагает возможность свободного копирования пользователями программ друг у друга или распространение таких программ через сеть компьютерных клубов, по электронной почте или через "электронные доски объявлений" (BBS). Пользователь shareware-программы, если программа ему действительно понравилась и оказалось полезной, добровольно регистрируется у изготовителя или автора, перечисляя ему указанную (небольшую) сумму. Поэтому всякая копия shareware-программ является легальной, однако не всякий пользователь является зарегистрированным.

Другая черта легкой приспособляемости программы PC-Write Lite касается весьма широких возможностей по использованию имеющегося в Вашем распоряжении аппаратного обеспечения, так как программа весьма нетребовательна к техническим возможностям компьютера. Для работы достаточно лишь 256 Кбайт оперативной памяти в любой IBM-совместимой машине (однако потребуется 384 Кбайт ОЗУ для применения автоматической проверки орфографии) с операционной системой DOS 2.0 или более поздней версией. Можно обойтись вообще без жесткого диска, но при этом желательно иметь два дисковода для 5,25- или 3,5-дюймовых дискет. Чтобы работать с русскими буквами, нужен адаптер EGA или VGA, но монитор не обязательно должен быть цветным. Кроме того, программа поддерживает работу принтеров чуть ли не всех марок и эпох.

Славянская версия

В конце января 1991 года фирма Quicksoft объявила в Сиэттле о выпуске версии PC-Write Lite 1.02 с поддержкой кириллицы. Это обстоятельство и послужило основной причиной написания статьи. Славянская версия программы основана на кодовой странице 866 CP и поддерживает русифицированную версию операционной системы MS-DOS 4.01, поэтому кроме возможности работать с буквами русского алфавита она позволяет, переключив клавиатуру, использовать также буквы украинского и белорусского алфавитов.

Приспособление программы для работы с кириллицей заключается в возможности свободно переназначать любые клавиши в схеме клавиатуры, придавая им нужные значения в случае включения клавиши CapsLock, которую обычно наиболее удобно использовать для переключения режимов "Лат/Рус". При этом также загружается информация о парных буквах верхнего и нижнего регистров, что необходимо для правильного перемещения курсора по словам, для подсчета слов и проверки орфографии. Чтобы видеть на экране буквы кириллицы, используется набор знаков для экрана, загружаемых в адаптер EGA или VGA. С этой версией программы PC-Write Lite могут применяться разнообразные принтеры, используемые для печати буквами русского алфивита благодаря встроенным ПЗУ с альтернативной кодировкой символов или в графическом режиме.

Короче говоря, PC-Write Lite может использоваться в качестве средства русификации машин импортного производства, чем в нашей стране промышляют многие компьютерные кооперативы и фирмы. Лишь отсутствие в комплекте пакета PC-Write Lite наклеек с русскими буквами для клавиатуры (на мой взгляд, это серьезный недостаток) затрудняет быструю комплексную русификацию компьютера. Впрочем, справедливости ради должен отметить, что в документации я нашел адрес американской фирмы, которая за 22 дол. пришлет всем желающим наклейки с русскими буквами для клавиатуры.

К сожалению, пакет PC-Write Lite не может распространяться в Советском Союзе методом shareware. Цена пакета PC-Write Lite 1.02 составляет 79 дол. За эти деньги Вы приобретете дискеты, на которых записаны программа и утилиты, 196-страничное руководство пользователя на английском языке, право на техническую поддержку зарегистрированного пользователя по телефону (пока только на английском языке, причем оплату международного телефонного разговора с Сиэттлом осуществляет сам пользователь), а также подписку на ежеквартальный печатный бюллетень фирмы Quicksoft, содержащий полезные советы, рекомендации и ответы на вопросы, задаваемые пользователями наиболее часто.

Дополнительную программу для проверки русской орфографии, включающей словарь на 95 000 слов русского языка, фирма Quicksoft предлагает приобрести отдельно за 29 дол. Если Ваш текст состоит из смеси английских и русских слов, полностью сохраняется возможность проверки правописания слов английского языка, имеющаяся в стандартном пакете PC-Write Lite.

Славянскую версию фирма Quicksoft предлагает по сниженной цене всем зарегистрированным пользователям пакетов PC-Write и PC-Write Lite. Кроме того, фирма Quicksoft предлагает ощутимые скидки в случае закупки пакетов партиями и для использования в любых учебных заведениях, а также ожидает предложений от дилеров и дистрибьютеров в Советском Союзе.

Адрес фирмы: Quicksoft Inc.
219 First North Avenue #224
Seattle, WA 98109 U.S.A.

Телефон для заказов и деловых предложений, по которому можно звонить за счет фирмы: (800) 888-8088

Понятно, что программа эта пока явно не предназначена для нашего рынка, а ориентирована на американский или иной зарубежный рынок. Едва ли можно рассчитывать на покорение рынка программным продуктом, в котором все 45 справочных экранов, вся документация и техническая поддержка реализованы на английском языке.

Заключительные замечания

Фирма Quicksoft подчеркивает, что текстовый процессор PC-Write Lite предназначен в первую очередь для пользователей портативных компьютеров, для студентов, писателей, журналистов, бизнесменов, программистов и для всех, кому не требуются мощные возможности форматирования текста и функция mail-merge (т.е. автоматическое слияние текста стандартного письма и адреса). Однако, на мой взгляд, этими ограничениями не исчерпываются недостатки программы. Она позволяет делать сноски и примечания, но в ней отсутствуют такие распространенные и становящиеся стандартными возможности, как автоматическое составление оглавлений и предметных указателей, а также отсутствует возможность печатать текст в несколько колонок на странице. Впрочем, эти недостатки определяются стремлением разработчиков обеспечить максимально возможные простоту и компактность программы.

Мне представляется, что у этого текстового процессора может быть неплохая судьба на нашем рынке программных средств. После проведения необходимых мероприятий по переводу и локализации эта программа вполне смогла бы составить конкуренцию известным коммерческим продуктам. Такая работа не представляет большой трудности и поэтому может быть быстро осуществлена небольшой группой специалистов.

Чрезвычайно удобно использовать PC-Write Lite совместно с программами верстки. Тем, кто работает с пакетом Ventura Publisher фирмы Xerox, хорошо известно, что редактировать сколько-нибудь длинные тексты с помощью этого пакета неудобно. Гораздо удобнее корректировать текст с помощью специального текстового процессора, а затем экспортировать файлы. Благодаря скорости работы и ASCII-формату программа PC-Write Lite подходит для этого идеально.

Итак, чтобы достичь большего результата при наименьших затратах, стоит обратить внимание на легкую и изящную программу PC-Write Lite. Отнюдь не случайно эта программа завоевала почетный приз американского журнала "The Computerian Review" за лучшую программу года, который называется "Three Thumbs Up" ("Три больших пальца вверх"). Я готов присоединить к ним и свой большой палец - за новую русифицированную версию PC-Write Lite.

Правовая охрана программ в СССР: cui prodest (кому выгодно)?

Л. П. Малков

О необходимости введения в СССР правовой охраны программ для ЭВМ долго говорили в кругах, связанных с программированием. И вот свершилось: 31 мая 1991 года Президент СССР М.С. Горбачев подписал "Основы гражданского законодательства Союза ССР и союзных республик", предусматривающие помимо прочего и распространение авторского права на программы для ЭВМ с 1 января 1992 года. Учитывая масштабы отрасли программирования, я бы назвал это событие историческим.

Однако радоваться можно только до чтения самого законодательного акта. При ознакомлении с ним спектр эмоций, по-видимому, должен колебаться от горького разочарования до горячего огорчения по поводу пренебрежения интересами индустрии программирования.

Всякий законодательный акт должен быть кем-то побуждаем, кому-то выгоден. Похоже, что подписанный документ в части, относимой к программированию, явился результатом произвольного переноса имевшихся правовых конструкций на программирование без попытки оценить, а главное, учесть специфику отрасли.

Введение правовой охраны программ затрагивает интересы программистов, предпринимателей в сфере программирования, пользователей программ.

Опыт развитых стран с абсолютной очевидностью показывает, что наиболее важную роль в развитии отрасли программирования играют предприниматели и фирмы. Действительно, отрасль существует только при наличии промышленного производства программ, которое может осуществляться только предприятиями (фирмами).

Программисты заинтересованы в соблюдении интересов фирм, поскольку, если не будет серьезного производства программ, их доходы будут складываться из относительно нерегулярных продаж либо сам программист будет вынужден стать предпринимателем, организуя собственную фирму. Однако в этом случае он тем более заинтересован в законодательстве, учитывающем интересы предпринимателей и фирм.

Со своей стороны, пользователи заинтересованы в высококачественных программных продуктах, их постоянном развитии, доступности, сопровождении и т.д. Очевидно, что все это не может обеспечиваться отдельными авторами, а возможно только в рамках предприятий (фирм).

Зависимость от промышленного производства и ориентация на индустриализацию - основные отличия программирования от подавляющего большинства сфер деятельности, охватываемых авторским правом. Писатель, художник, композитор создает не "промышленное произведение" и, более того, будет обижен, если его произведение считать таким. Хотя, конечно, и в более традиционных сферах деятельности, на которые распространяется авторское право, наблюдаются явная тенденция к индустриализации и высокая степень характерного для промышленности разделения труда, особенно в производстве видеофильмов, клипов и т.п.

Напротив, программирование с самого начала было частью промышленного производства, еще до того как стало развиваться отдельно от создания аппаратных средств. В этом есть свои плюсы и минусы. С одной стороны, программирование - не только творческое занятие, с другой - благодаря индустриализации оно добилось фантастических успехов и темпов развития. В настоящее время для американской экономики программирование является более значимой (в денежном выражении) отраслью, чем производство книг, кассет и фильмов вместе взятых. Оценка годовых продаж программ и сопутствующих услуг во всем мире превышает 100 млрд. дол. Еще раз подчеркну, что эти успехи немыслимы при условии защиты только интересов отдельных авторов - они стали возможны только благодаря правовой поддержке интересов предпринимателей, занятых промышленным производством программ. Состояние Билла Гейтса, основателя фирмы Microsoft, достигло почти 3 млрд. дол. (!), конечно же, потому, что государство защищало его интересы не только и не столько как автора, но как предпринимателя. Да и его сотрудники процветают, поскольку защищены права фирм-изготовителей программ.

К сожалению, это важнейшее положение выпало из поля зрения разработчиков "Основ гражданского законодательства Союза ССР и союзных республик". В соответствии с традиционными представлениями о том, что должно обеспечивать авторское право, новый нормативный документ защищает права разработчика программы, подрывая до основания интересы фирм-изготовителей программ. В конечном итоге этот документ наносит разрушительной силы удар одновременно и

Леонид Петрович Малков -
член совета директоров
СП "ПараГраф".
Старший научный сотрудник
ЦЭМИ АН СССР.
Область профессиональных интересов -
экономические аспекты компьютерной индустрии,
правовая охрана программ.
Автор более 30 статей.

по интересам самих изготовителей, и по интересам пользователей.

Можно сказать, что такое законодательство препятствует разделению труда, профессионализации, индустриализации программирования, сводя все экономические отношения в этой отрасли к производству программ кустарями-одиночками.

Рассмотрим некоторые примеры, демонстрирующие враждебность принятого законодательства по отношению к фирме.

Ключевой вопрос - отношение к так называемым служебным разработкам, т.е. к работе, выполненной в порядке служебного задания, в рабочее время, на фирменной технике.

Приведу соответствующую статью Основ полностью.

Статья 140. Служебные произведения: "Авторское право на произведение, созданное в порядке выполнения служебного задания (служебное произведение), принадлежит его автору.

Право использования служебного произведения способом, обусловленным целью задания и в вытекающих из него пределах, принадлежит лицу, по заданию которого создано произведение (работодателю). Вознаграждение автору за использование таким способом и в таких пределах уплачивается в случаях, установленных законодательством.

По истечении трех лет с момента представления произведения, а при согласии работодателя - и ранее права автора на использование произведения и на получение авторского вознаграждения принадлежат ему в полном объеме.

Право автора произведения использовать служебное произведение способом, не обусловленным целью задания, не ограничивается."

Статья 135 (пункт 1) при этом гласит, что "Автором произведения признается гражданин, творческим трудом которого оно создано". Тем самым наниматель исключается из рассмотрения. Ниже мы увидим, насколько нетипично это положение для законодательства развитых стран.

Пункт 2 той же статьи расшифровывает, что понимается под правами автора: "Автору произведения принадлежит исключительное право на свое произведение, включающее:

- право на авторство,
- право на имя,
- право на неприкосновенность произведения,
- право на опубликование произведения,
- право на использование произведения (право осуществлять или разрешать его воспроизведение любыми способами ...; перевод, переработку произведения; распространение экземпляров воспроизведенного произведения; ...;
- право на вознаграждение за разрешение использовать и за использование произведения".

Таким образом, автор может передать фирме только право на использование, но не остальные права. Получается, что автор программы, созданной в фирме, имеет право требовать указания его имени, а также запрещать (ссылаясь на неприкосновенность) любые изменения в программе.

Насколько это абсурдно, понимает каждый, кто как-то связан с программированием. Конечно, не только нормальный предприниматель, но и просто программист оказывается в западне. Вы не хотите писать какую-то часть программы и поручаете это другому сотруднику, но теперь он по нашему лучше всего защищающему права авторов законодательству становится Вашим соавтором, и неизвестно, чего он захочет потребовать за свою прямую и оплаченную работу впоследствии. Получается, что лучше не прибегать к разделению труда и все делать самому, чтобы потом не оказаться в юридической ловушке.

Предприимчивый предприниматель, естественно, будет раздосадован тем, что вместо помощи он получил саботирующий развитие промышленности закон, но будет думать, как его обойти.

Простейшее и естественное решение - заключить с сотрудником контракт, по которому все права на программы, создаваемые в порядке служебного задания, будут принадлежать фирме, но авторы Основ, предвидя такое решение, позаботились о том, чтобы его пресечь.

Как уже цитировалось: "По истечении трех лет с момента представления произведения, а при согласии работодателя - и ранее права автора на использование произведения и на получение авторского вознаграждения принадлежат ему в полном объеме". Чтобы "эксплуататор"-предприниматель (а все эти нормы действительно когда-то были направлены на то, чтобы защитить беспомощных авторов от эксплуата- ции издателей) не смог обойти и это положение в индивидуальном трудовом договоре, предусмотрена еще одна норма.

В статье 139 (пункт 2) сказано: "Условия заключенного с автором договора, ухудшающие его положение по сравнению с положением, установленным в законодательстве, не имеют юридической силы и заменяются условиями, установленными законодательством". Это означает, что даже если сотрудник подписал индивидуальный договор с фирмой о том, что он согласен, чтобы созданные им в рабочее время, по служебному заданию, на предоставленной фирмой технике программы полностью принадлежали фирме, то и тогда предприниматель не может спать спокойно. В любой момент сотрудник может опротестовать этот договор и добиться того, чтобы через три года с момента создания программы права на его программу или созданную им ЧАСТЬ программы принадлежали ему.

Очень забавно будет посмотреть на то, как программист, написавший программу управления бортовой ЭВМ, будет на законных основаниях оспаривать право своего КБ вносить в его программу изменения, если они, например, связаны с изменением аппаратных средств. Конечно, сотрудники программистских фирм тоже не смогут чувствовать себя спокойно, ибо такое законодательство теоретически обеспечивает возможность судебных споров ВСЕХ СО ВСЕМИ.

Можно предположить, что эти фантастические правовые нормы будут игнорироваться, но зачем вообще и, главное, КОМУ нужно столь подрывное законодательство? Тем более это положение неприемлемо, поскольку именно в программировании имеются уникальные гигантские возможности для советских специалистов оказаться на мировом уровне в наукоемкой области. Почти во всех других отраслях отставание носит критический или катастрофический характер, и потому именно программирование следовало бы холить и лелеять в экономическом отношении, предоставляя максимальные возможности для быстрого экономического развития в этой отрасли.

Самые различные положения Основ, не только связанные со служебными произведениями, подчеркивают игнорирование прав фирмы (юридического лица) и выделение роли собственно автора (физического лица).

Роль фирмы как участника экономических отношений настолько игнорируется, что фактически даже в суд подать можно только от лица автора, а не фирмы (статья 143). Исключительное неудобство этого положения для нормального развития отрасли более чем очевидно.

Обладателем прав настолько непременно должно выступать физическое лицо, что даже теоретически нельзя определить длительность срока правовой охраны, не учитывая личности автора. Стандартный срок составляет 50 лет после смерти

автора, точнее, "авторское право действует в течение всей жизни автора и пятьдесят лет после его смерти, считая с 1 января года, следующего за годом смерти автора" (статья 137, пункт 1).

Может быть, только в таком виде и может осуществляться правовая охрана объектов авторского права? Посмотрим на опыт других стран.

Исключительно интересен опыт США, поскольку именно там программирование достигло наибольшего развития и наивысшей степени индустриализации.

Согласно Закону США об авторском праве (Copyright Act) (статья 201): "Авторское право на произведение, охраняемое в соответствии с настоящим Законом, первоначально принадлежит автору или авторам произведения...

В случае создания произведения по найму наниматель или другое лицо, для которого создавалось произведение, считается автором в смысле настоящего Закона и, если стороны специально не обусловили иное в подписанном ими документе, обладает всеми правомочиями, составляющими авторское право...

Обладание авторским правом может быть полностью или частично передано с помощью любого акта передачи правового титула или в силу действия закона и может перейти по наследству на основании завещания или по закону.

Любое исключительное право, составляющее авторское право ..., может быть передано ... отдельно."

Таким образом, мы видим, что по умолчанию права на служебное произведение принадлежат нанимателю, и любую часть правомочий можно уступать, что открывает широчайшие возможности для рыночных отношений.

Все продаваемое на рынке программное обеспечение снабжается знаком охраны авторского права (copyright), защищающего права соответствующих фирм. В каких-то случаях при этом авторы программ как-то упоминаются (например John Socha в пакете Norton Commander), но в большинстве случаев невозможно найти упоминания об отдельных разработчиках (скажем, пакета MS-Word или Paradox).

Особо отмечается в Законе США об авторском праве срок охраны произведения, созданного по найму: "70 лет с года его первого опубликования либо в течение 100 лет с года его создания, в зависимости от того, какой срок истечет первым" (статья 302).

Еще более важным для нас является опыт Франции, поскольку законодательство по авторскому праву в СССР ближе к французскому, чем к американскому.

Во Франции нормы авторского права были распространены (не без споров) на программное обеспечение Законом, принятым 3 июля 1985 года (для сравнения, в США соответствующее событие произошло в 1980 году). При этом в авторское право на программы было внесено несколько очень важных изменений, которые поучительны для СССР.

Во-первых, статья 45 Закона от 3 июля 1985 года гласит, что "если не предусмотрено иное, программы, созданные одним или несколькими служащими или сотрудниками администрации при выполнении служебных обязанностей, принадлежат нанимателю, который обладает всеми правами, предоставляемыми авторам". Более того, та же статья распространяет авторские права также на юридические лица.

Это положение существенно отличается от предусмотренного статьей 1 Закона Франции об авторском праве, согласно которой права на произведение предоставляются автору, даже если оно создано во время работы по найму.

Данная позиция более чем красноречива если сравнивать ее с обсуждаемыми здесь Основами гражданского законодательства Союза ССР и союзных республик.

Во-вторых, статья 46 того же Закона Франции об авторском праве от 3 июля предусматривает, что, если не предусмотрено иное, автор не может отказать в адаптации программы в пределах предоставленных ему прав. Это опять же изменяет положения Закона Франции об авторском праве, где предусмотрено право автора на неприкосновенность его произведения.

В-третьих, статья 47 того же Закона ограничивает общее положение о добросовестном пользовании копиями произведения в отношении программ использованием только одной запасной (back up) копии.

В-четвертых, статья 48 устанавливает особый срок правовой охраны программ в 25 лет с момента создания, в отличие от 50 лет после смерти автора для остальных произведений, охраняемым авторским правом.

В-пятых, статья 49 устанавливает, что вознаграждение за лицензию, уступающую права на программу, может быть выплачено в виде разового платежа, в то время как Закон об авторском праве предусматривает, что плата за лицензию должна быть пропорциональна количеству проданных или лицензированных копий.

Наконец, есть еще одна точка отсчета для рассмотрения вопроса о служебных произведениях и о правах фирм - советское законодательство о изобретениях. Изобретения часто рассматривают наряду с объектами авторского права как предметы интеллектуальной собственности.

Те же Основы гражданского законодательства в Разделе V, посвященном праву на изобретения, предусматривают возможность учета интересов фирм в рамках служебных изобретений.

Статья 146 гласит: "Патент на изобретение, созданное работником в порядке служебного задания (служебное изобретение), выдается работодателю, если между ним и автором заключен договор об уступке прав на такие будущие изобретения работодателю".

В существующем виде "Основы гражданского законодательства Союза ССР и союзных республик" не только не предоставляют, пусть даже крайне неэффективной охраны программ, но оказываются просто враждебными интересам отрасли. Недостатки, связанные со служебными программами, - только часть общего списка претензий к этому нормативному акту со стороны защитника экономических интересов индустрии программирования.

Формально эти Основы вводятся в действие с 1 января 1992 года, но на их базе должны были бы разрабатываться республиканские нормативные акты. В связи с распадом Советского Союза и приоритетом республиканских законодательств правовое значение рассматривавшегося документа более чем неопределенное. Однако он отражает некоторую логику мышления законодателей. И если не обсуждать эти проблемы в профессиональных кругах, то может получиться, что та же логика будет считаться законодателями приемлемой и окажется заложенной в республиканские законодательные акты. Программистской общественности необходимо, объединившись, добиваться введения на республиканском уровне нормативных актов, учитывающих интересы отрасли программирования.

Я благодарен за многочисленные, продолжительные и не всегда простые споры по вопросам правовой охраны программ А.Н. Козыреву, В.Л. Макарову, И.В. Савельевой, Р.М. Горелик, В.Г. Виталиеву, Л.И. Подшибихину, С.А. Пачикову и др.

Особая благодарность представителю фирмы Ashton-Tate Брюсу Маркварту (Bruce Marquart) за беседы и предоставление материалов.

СУПЕРКОМПЬЮТЕРЫ В МИРЕ И В НАШЕЙ СТРАНЕ

Советский суперкомпьютер - возможен ли он сегодня?

А. И. Масалович

...философские соображения, сколь бы важными они ни представлялись, не должны помешать нам искать пути применения новой техники. Если это удастся, то будущее наше будет лучше, чем можно себе вообразить. Если же нет, то у нас вообще может не быть будущего.

Д.Мичи, директор института им. А.Тьюринга (Глазго)

До суперкомпьютеров ли нам сегодня, в голодной отсталой стране, истерзанной многолетним экономическим экспериментированием? В стране, где увеличением объема производства может похвастаться лишь Гознак. Да и можно ли всерьез говорить о суперкомпьютерах, отстав от мирового уровня развития электроники на 6-10 лет (по другим оценкам - навсегда)? Разумеется, было бы наивно полагать, что отечественная наука и промышленность способны своими силами создать действительно конкурентоспособную быстродействующую ЭВМ. Однако, как ни парадоксально, именно сегодня у нас есть реальный шанс включиться в европейские и мировые разработки в этой области. Основания для столь оптимистического заявления дает анализ современного состояния и тенденций развития индустрии суперкомпьютеров, составляющий предмет данной статьи. Что же касается вопроса: "Зачем это нам сегодня?", то наилучший ответ на него содержится в эпиграфе. Образно говоря, индустрия суперкомпьютеров - это локомотив прогресса, призванный доставить человечество в XXI век. И если мы уже сегодня не позаботимся о билетах, то рискуем просто остаться на перроне.

Для того чтобы определить возможные место и роль советской индустрии суперкомпьютеров относительно мировой, необходимо предварительно ответить на три вопроса: "Что собой представляют современные суперкомпьютеры? Для чего они создаются? К чему стремятся их создатели?"

К суперкомпьютерам (часто употребляется также термин суперЭВМ) принято относить наиболее производительные вычислительные машины, быстродействие которых во много раз превышает возможности так называемых "коммерческих" ЭВМ, представленных в данный момент на рынке.

Быстродействие ЭВМ *(performance)* обычно измеряют в миллионах операций в секунду (MIPS), миллионах операций с плавающей точкой в секунду (MFLOPS), а также в единицах, характеризующих скорость выполнения определенных стандартных тестов. Сегодня применяется около двух десятков таких тестов, наиболее известны тесты Whetstone, Dhrystone и так называемые "Ливерморские циклы" *(Livermore Loops)*. В качестве характеристик применяют также пиковое (т.е. предельно достижимое) *(peak)* и среднее *(sustained)* быстродействие. Такое обилие используемых характеристик вызвано невозможностью однозначно ранжировать ЭВМ по быстродействию. Более быстрые при использовании одних тестов модели могут значительно проигрывать в случае использования других. Например, по оценкам Д. Кука (D. Cook) (Суперкомпьютерный центр при Иллинойсском университете), из одиннадцати типов компьютеров, представленных в Центре, самым высоким пиковым быстродействием характеризуется модель Hitachi S-820, в то время как среднее быстродействие выше у модели Cray Y-MP/832, и многие тесты выполняются на ней быстрее.

Что касается конкретных значений быстродействия суперкомпьютеров, то они, во-первых, колеблются в значительных пределах (сегодня это примерно 250 - 30000 MFLOPS), а во-вторых, в последние годы чрезвычайно быстро растут. Достаточно сказать, что еще пять лет назад Оксфордский словарь по вычислительной технике относил к суперкомпьютерам ЭВМ с производительностью свыше 10 MFLOPS. Сегодня этот рубеж перешагнули многие настольные рабочие станции, а пиковое быстродействие, например, компьютера nCUBE 2 фирмы nCUBE оценивается в 27 GFLOPS ("гигафлопс" - миллиард операций с плавающей точкой в секунду).

Вообще же самыми быстрыми однопроцессорными компьютерами считаются SX-X фирмы NEC (5.5 GFLOPS) и VP-2600 фирмы Fujitsu (4 GFLOPS). На роль абсолютного лидера по быстродействию претендуют модели примерно десятка фирм, среди которых Cray Computers, Cray Research, Thinking Machines, BBN, SSI, а также вышеупомянутые nCUBE, NEC и Fujitsu и даже всем известная фирма Intel. Выделить среди них "самую быструю ЭВМ" не представляется возможным по приведенным выше причинам.

Разработка суперкомпьютеров, несомненно, является весьма дорогостоящим делом. Достаточно сказать, что первые десять экземпляров знаменитого суперкомпьютера Cray-2 в четырехпроцессорном исполнении были проданы по 20 млн. дол. каждый. Кто платит эти миллионы? И для чего вообще создаются все новые и новые модели сверхбыстродействующих машин?

Можно назвать четыре основные причины. Прежде всего, во все времена были и будут задачи, для решения которых мощности существующих ЭВМ недостаточны. Около десяти лет назад нобелевский лауреат Кеннет Уилсон (K.Wilson) составил список наиболее трудных научных проблем, занимающих умы современных ученых. В ряду этих проблем - прогнозирование глобальных климатических изменений, картографирование человеческого генома и другие. Для решения большинства из них нужны новые суперкомпьютеры. Проиллюстрируем это на примере. В одном из исследовательских центров Лос-Аламоса сейчас

Андрей Игоревич Масалович известен читателям журналов "Интеркомпьютер", "Мир ПК" и "Вопросы радиоэлектроники". Области его научных интересов - САПР СБИС, объектно-ориентированное программирование, интеллектуальный интерфейс систем проектирования, архитектура многопроцессорных систем.

проводятся работы по моделированию климатических изменений [В мире науки, март 1991, №3, с.75]. Используемая модель планетарного слоя атмосферы содержит более полумиллиона четырехгранников - это близко к предельным возможностям используемого компьютера фирмы Thinking Machines. Однако можно подсчитать, что длина стороны каждого четырехугольника может доходить до 30 км, при этом вычисления требуемых параметров выполняются только в вершинах четырехугольников. Естественно, ученые мечтают о более высокой точности вычислений и, соответственно, о более быстрой ЭВМ. Кстати, в первых рядах заказчиков новых суперкомпьютеров для прикладных нужд традиционно стоят военные. Их аргументы более чем убедительны: например, компьютер, обслуживающий тренажер летчика-истребителя, должен просчитывать все изменения в модели воздушного боя за 1/50 с (и при этом формировать реалистичную картину на экране и приборной доске!). При обработке радиолокационных данных информация обновляется каждые 0,001 с. Понятно поэтому, что при общем сокращении расходов на оборону военные не скупятся на оплату новейших разработок. В США для этих целей даже создано специальное Агентство перспективных разработок в области обороны Defence Advanced Research Projects Agensy - (DARPA) с многомиллионным бюджетом.

Вторая причина лежит в области политики. Помимо конкретной практической пользы, которую приносят суперкомпьютеры той или иной стране, само их существование дает определенные очки в мировой политической игре, как, скажем, наличие ядерной энергетики или золотого запаса. Сегодня термин "вычислительная мощность государства" используется в одном ряду с другими показателями экономического состояния страны.

Третья причина - философская. Какие-то, не изученные еще до конца мотивы двигают человечеством, заставляя вкладывать огромные деньги в полеты на Луну или создание растрового микроскопа, основанного на использовании туннельного эффекта. Усилия, связанные с разработкой новых Суперкомпьютеров, примыкают к проектам подобного рода, проектам, которые зачастую не дают немедленной отдачи, но тем не менее считаются полезными, поскольку раздвигают границы наших знаний о мире и демонстрируют могущество человека. Кстати, вынесенные в эпиграф слова были сказаны в обоснование одного крупного и дорогостоящего проекта суперкомпьютера для задач искусственного интеллекта.

Наконец, четвертая причина, побуждающая разработчиков браться за новые проекты суперкомпьютеров, - это стремление "проверить на прочность" свежие идеи и технологические новинки. Многие ведущие фирмы, разрабатывающие суперкомпьютеры, были когда-то основаны талантливыми людьми, стремившимися наиболее полно реализовать свои новые идеи (не имея, как правило, конкретных заказчиков). Так было с Сеймуром Крэем (S. Cray), основателем фирмы Cray Research; со Стивом Ченом (S. Chen), ушедшим от Крэя и основавшим фирму Supercomputer Systems Inc. (SSI); с Дэнни Хиллисом (D. Hillis), самым колоритным разработчиком самого быстрого из высокопараллельных компьютеров (фирма Thinking Machines). Кстати, взгляд на суперкомпьютеры, как на полигон новых научно-технических решений, объясняет "мирное сосуществование" моделей, различающихся по быстродействию более чем в сто раз.

Какие цели ставят сегодня перед собой разработчики суперкомпьютеров? Удивительно, но в последнее десятилетие XX века наблюдается редкое единодушие в формулировке перспективных планов (вспомним разнобой мнений пять лет назад, после обнародования японского проекта ЭВМ пятого поколения). Разработчики говорят: "К концу века должен быть создан компьютер (условно обозначаемый 3Т) с быстродействием 10^{12} операций с плавающей точкой в секунду, памятью 10^{12} байт и скоростью передачи данных в 10^{12} байт в секунду". Для измерения быстродействия этой гипотетической машины уже придумана единица 1 TFLOPS ("терафлопс" - триллион операций с плавающей точкой в секунду), а Хиллис уже осваивает первые миллионы долларов, вложенные в проект "терафлопной" машины СМ-3 фирмы Thinking Machines.

"Триллион операций в секунду - это то, что нам надо", - вторят разработчикам специалисты-прикладники. - "Этого достаточно для достоверного решения таких задач, как расчет обтекания фюзеляжа сверхзвукового самолета, моделирование торнадо, а также для большинства других известных задач."

"Это то, к чему стоит стремиться", - добавляют пессимисты, поскольку создание такого компьютера позволит в десять раз превзойти физически достижимые предельные характеристики традиционной вычислительной техники. Достигнув нового рубежа, человечество будет вынуждено совершить переход к принципиально новым вычислительным системам. Возможно, они будут основаны на биомолекулярной технологии, или будут выполнены на оптических переключателях, а может быть, в них будут заложены неведомые еще физические принципы. В любом случае это будет уже другая страница истории науки.

Цель достижения "трех Т" замечательна, но как она соотносится с сегодняшним состоянием электроники в нашей стране? Самые быстрые из советских компьютеров с гипотетической максимальной конфигурацией могут достичь пикового быстродействия 10 GFLOPS. И это предел, причем как в чисто техническом плане, так и в научном. С чем можем мы выйти в мир, где мыслят "терафлопсами" и "гигабайтами" ? Чтобы ответить на этот вопрос, бросим беглый взгляд на советскую индустрию суперкомпьютеров.

А был ли советский суперкомпьютер?

В сентябре 1991 года в Новосибирском Академгородке под руководством профессора Миренкова проходила Международная конференция "Технологии параллельных вычислений". Судя по всему, иностранные гости остались довольны как организацией, так и научным уровнем конференции. Однако в последний день, после всех докладов, прогулки на теплоходе и экскурсии в Вычислительный центр СОАН СССР один профессор из Великобритании заметил: "То, что вы рассказываете о советских компьютерах, очень интересно. Хотелось бы знать, какие из них существуют на самом деле?"

Вопрос, как говорится, не праздный. Действительно, десятилетиями существовали две точки зрения. Согласно одной из них, в СССР уже давно имеются потрясающе быстрые компьютеры, только они страшно засекречены, поскольку работают на оборону и освоение космоса. Согласно другой, все это миф, и единственный секрет всех наших "ящиков" - уровень отставания от Запада. Интересно, что оба высказывания недалеки от истины.

Поясним это на примере специализированного процессора ЕС-2720. Если бы сообщение о его возможностях появилось в прессе в начале 80-х годов (когда процессор прошел госиспытания), оно произвело бы эффект разорвавшейся бомбы. Еще бы, кому могло придти в голову, что СССР обладает процессором, который по быстродействию не уступает знаменитой суперЭВМ STARAN, "любимой игрушке" не одного поколения американских военных. Понятно, что появление такой информации в те годы было совершенно невозможно.

Строки, которые вы читаете, - это, вероятно, второе упоминание о ЕС-2720 в открытой печати. И появилось оно благодаря тому, что мне попалось на глаза первое - на английском языке, в книге, изданной в Сингапуре [Parallel Computing Technologies/Ed. by N.N. Mirenkov. - Singapore: World Scientific, 1991. - P. 317]. Что же тут удивляться слухам и разнобою мнений!

Тем не менее время для увлекательного разговора о специализированной вычислительной технике еще не пришло. Поэтому вернемся к компьютерам общего назначения. (Чтобы закончить с ЕС-2720, скажу, что сейчас он мирно доживает свой век в компьютерном центре "Сибирь" СОАН СССР, приводя в восторг сибирских ученых эффективным решением геофизических и иных задач.)

Когда-то давно, на заре компьютерной эры, уровень советской вычислительной техники нисколько не уступал мировому. СССР обладал мощнейшей по тем временам вычислительной системой БЭСМ-6, быстродействие которой достигало 1 MIPS. Эту "рабочую лошадку" до сих пор вспоминают в институтах и лабораториях Академии наук. Операционная система "Дубна" также не уступала другим ОС того периода. Язык программирования РЕФАЛ, разработанный советскими учеными, уже третий десяток лет применяется во всем мире при решении задач искусственного интеллекта.

Однако, начиная с 60-х годов, постепенно наметилось отставание советской компьютерной индустрии от мирового уровня. Различия в быстродействии отечественных и зарубежных процессоров, качестве периферийных устройств, степени интеграции элементной базы становились все заметнее и привели в конечном итоге к утрате конкурентоспособности советских изделий на мировом рынке.

Что послужило причиной отставания? Можно перечислить целый ряд объективных и субъективных факторов, однако наиболее существенными из них представляются два: низкий уровень технологии изготовления основных компонентов компьютеров и ориентация на воспроизведение прототипов. Трудно сказать, какой из этих факторов

серьезнее. Одно можно утверждать определенно: принятый четверть века назад курс на копирование зарубежных образцов (наиболее яркий пример - серия ЕС ЭВМ, воспроизводящая модели ряда IBM 360/370) был по сути своей ошибочным, поскольку предполагал неизбежное отставание от прототипа на несколько лет.

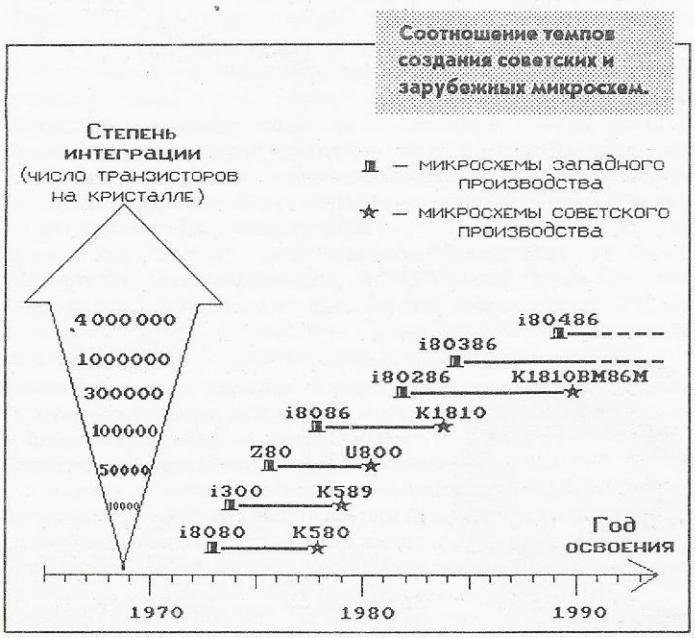
Насколько велико отставание сегодня? Чтобы не заниматься гаданием, посмотрим на рисунок, из которого видно, что советский аналог каждой из указанных западных микросхем появлялся на свет на несколько лет позже своего прототипа. В 70-е годы разрыв составлял около 5 лет, в 80-е - вырос до 7-8 лет. Микропроцессоры сегодняшнего дня, такие, как Intel 860 или Motorola 68040, пока в СССР не реализованы.

Если бы производительность компьютеров определялась только быстродействием процессорных элементов, Советский Союз давно уже сошел бы с дистанции в состязании суперкомпьютерных держав. К счастью, это не так. Для повышения производительности существует много путей. В 70-е годы основные усилия тратились на миниатюризацию и создание более быстродействующих элементов, в 80-е - на упаковку и охлаждение, в начале 90-х перспективными считаются распараллеливание обработки данных и интеллектуализация интерфейса с пользователем. И в течение всех этих лет шли эксперименты с архитектурой многопроцесссорных суперЭВМ. Например, еще в 1971 году фирма Burroughs выпустила трехпроцессорную ЭВМ B6700 со стековой архитектурой и высоким уровнем машинного языка. А в 1978 году увидела свет советская стековая ЭВМ "Эльбрус-1" (которую шутники почему-то прозвали "Эль-Бэрроуз"). В состав комплекса "Эльбрус-1" могли входить до 10 процессоров производительностью 1-1,5 MIPS каждый. В следующей модели, "Эльбрус-2", при том же числе процессоров производительность каждого из них была увеличена до 10 MIPS.

Другую архитектуру имеют так называемые матричные процессоры, содержащие большое число сравнительно простых процессорных элементов. По известной классификации Флинна (Flynn) они относятся к классу систем типа SIMD (см. словарь на с.15). Из советских матричных процессоров наиболее известна система ПС-2000, принятая Государственной комиссией в конце 1980 года. Максимальное число процессорных элементов в системе составляет 64, при этом считается, что может быть достигнута производительность 200 MIPS, хотя при решении реальных задач среднее быстродействие системы составляет примерно 10 MIPS. Как и многие SIMD-системы, ПС-2000 является специализированной ЭВМ, ориентированной на решение задач определенных классов.

Развитием этого направления стала система более широкого применения - ПС-3000, имеющая смешанную архитектуру. В ПС-3000 представлены как универсальные процессоры, использующие общую память в режиме SIMD, так и специализированные векторные процессоры.

В начале 80-х годов в Европе обрела популярность концепция транспьютеров как процессоров, способных непосредственно обмениваться информацией друг с другом и обслуживать несколько независимых процессов. Естественно, в Советском Союзе, как и в других странах, предпринимались попытки разработать свой оригинальный транспьютер. Наиболее удачной из них была работа над процессором КРОНОС в рамках проекта "МАРС-Т".



Разработка мини-суперкомпьютера "Марс-Т" была начата в середине 80-х годов. В его создании принимали участие специалисты из Сибирского отделения Академии Наук СССР (Новосибирск), Института кибернетики АН Эстонской ССР (Таллинн) и Научно-исследовательского института управляющих вычислительных машин (Северодонецк). Разработка архитектуры нового компьютера базировалась на идее минимизации количества связей путем группирования функциональных модулей в иерархическую систему. Базовым вычислительным модулем компьютера "МАРС-Т" являлся специально разработанный транспьютероподобный процессор "КРОНОС". Число процессоров в системе могло варьироваться от одного до нескольких десятков.

Несмотря на то, что компьютер "МАРС-Т" был полностью оригинальным, на его разработку оказали большое влияние идеи, заложенные в транспьютерах серии T400 английской фирмы INMOS и в рабочей станции Lilith, разработанной Н. Виртом в Швейцарии. Как и в компьютере Lilith, в компьютере "МАРС-Т" набор команд был ориентирован на эффективную реализацию языка Модула-2. Однако по сравнению с компьютером Lilith "МАРС-Т" обладал рядом серьезных преимуществ: увеличенной длиной слова, упрощенной выборкой команд, возможностью организации асинхронных связей между процессорами. В наборе команд процессора "КРОНОС", как и в T400, были средства управления процессами и организации связи между ними по каналам. Программа представлялась как набор принципиально независимых процессов, управляемых от данных, событий или условий.

К сожалению, отличие процессора T424 от процессора "КРОНОС" было не в пользу последнего. Уровень технологии не позволил реализовать "КРОНОС" на одном кристалле. Процессор занимал целую плату. Одним из следствий этого явилось пятикратное по сравнению с обеспечиваемым процессором T424 увеличение времени цикла.

Самобытный компьютер "МАРС-Т" и его "сердце" - процессор "КРОНОС", постигла судьба многих талантливых разработок. Компьютер, реализованный на восьми процессорах и успешно прошедший испытания, в серийное производство так и не попал.

Из существующих на сегодняшний день советских суперкомпьютеров можно назвать две модели: "Эльбрус-3" и "Электроника ССБИС". Создатели многопроцессорного вычислительного комплекса "Эльбрус-3" утверждают, что максимальный 16-процессорный вариант их детища способен показать пиковое быстродействие до 10 GFLOPS (!). Авторы суперкомпьютера "Электроника ССБИС" (который кто-то из иностранцев назвал "Красным Крэем") немного осторожнее в оценках. Производительность каждого из двух векторно-конвейерных процессоров суперкомпьютера они оценивают в 250 MFLOPS.

Вообще-то компьютеры с такими характеристиками могли бы неплохо выглядеть на мировом рынке. Однако этого не наблюдается, так как названные модели проигрывают по сравнению с западными по эксплуатационным и массогабаритным характеристикам, надежности, сетевым и графическим возможностям и ряду других параметров. Главный же недостаток этих моделей (в глобальном смысле) - "выработка ресурса" заложенных в них идей. Нет шансов на то, что дальнейшее развитие архитектуры этих компьютеров приведет к существенному увеличению быстродействия. И это проблема не только для советских суперкомпьютеров, это головная боль всех разработчиков компьютеров с векторно-конвейерной архитектурой. Несмотря на то, что сегодня модели этого типа задают тон на рынке суперкомпьютеров, направление главного удара в борьбе за мегафлопсы находится в другой области.

Куда двигаться дальше?

Мы бросили беглый взгляд на состояние дел в мире суперкомпьютеров. Пришел момент вернуться к обещанным в начале статьи перспективам участия СССР в передовых научно-технических разработках. Для этого предлагается сыграть в игру, когда-то предложенную "Литературной газетой": "Если бы директором был я". Представим себе, что нам в руки попала дирижерская палочка, позволяющая управлять "оркестром" участников современных суперкомпьютерных программ. В первых рядах этого "оркестра" - заказчики во главе с военными, за ними следует компьютерная индустрия, затем - отраслевая и академическая

наука, к которой примыкают университеты и вузы. Разберем с каждым участником его партию.

Прежде всего, всем заказчикам, и в первую очередь военным, необходимо провести короткие научно-исследовательские разработки с целью определения оптимальной по критерию производительность/стоимость конфигурации вычислительных средств для решения их задач. Такие НИР (стоимостью не более 200 тыс. каждая) позволят локализовать и объединить области, где применение суперкомпьютеров действительно необходимо, и тем самым сэкономить миллионы рублей. Опыт показывает, что большинство из тех, кто выступает за создание новых суперсистем, в действительности может обойтись существующими мини-суперЭВМ, расширенными одной-двумя специализированными платами.

Во-вторых, предлагается на время вывести из игры в суперкомпьютеры нашу электронную промышленность. "Если Вам по весне захотелось обзавестись возлюбленной, не стоит брать амебу и ждать, пока она эволюционизирует", - говорил когда-то доктор У.Маккалох. Для создания современных суперкомпьютеров необходимы БИС со степенью интеграции свыше 800 тыс. транзисторов на кристалле. Наша промышленность потратила последние пять лет на то, чтобы освоить в восемь раз меньшую плотность упаковки. Необходимо время для коренного переоснащения отрасли.

Тем временем отраслевая наука может создать современный и вполне конкурентоспособный мини-суперкомпьютер (Да-да!). Представим себе процессорное поле, состоящее из нескольких (от 1 до 16) транспьютеров, например, типа T805. Пусть каждый из транспьютеров имеет собственную локальную память емкостью до 16 Мбайт и делит ее с быстродействующим математическим сопроцессором (например, типа i860). Управляется процессорное поле локальной сетью 32-разрядных рабочих станций (типа Sun SPARC, RS/6000 и т.д.). Большинство используемой аппаратуры стандартно, свои только системы питания, охлаждения и коммутации. (По такому пути идут сейчас многие мировые фирмы, в том числе широко известная Meiko). Уже разработано несколько советских проектов подобного типа, в их числе "Квант", "Интеркуб", "Парвек" и другие. Такой проект может быть реализован за 1,5-2 года и потребует примерно 3 млн. долларов и 20 млн. рублей, что по европейским меркам немного и вполне по силам любому нашему ведомству. Большая часть усилий при этом будет потрачена на адаптацию стандартного и разработку нового программного обеспечения.

Возникает резонный вопрос: "А как же быть с правилами Комитета по контролю над экспортом в страны Восточной Европы - пресловутого COCOM?" Ведь закупка современных микропроцессоров и любой мало-мальски мощной рабочей станции попадает под запреты этой хмурой организации. Будем реалистами. Запреты COCOM в их сегодняшнем виде не лишают нас компьютеров. Они скорее лишают прибылей их создателей. Сотня-другая рабочих станций (чего в принципе будет достаточно для оснащения суперкомпьютерами описанного выше типа всех заинтересованных академий, ведомств и штабов) способна "просочиться" через любые барьеры.

Специалисты по теоретическим аспектам параллельных вычислений имеют сегодня реальный шанс поучаствовать в самых передовых мировых программах разработки суперкомпьютеров. Это связано не только с тем, что достаточно полной теории автоматического распараллеливания вычислений не существует, а потребность в ней огромна. Сегодня мы становимся свидетелями рождения целого "созвездия" новых научных дисциплин, связанных с нейрокомпьютерами, оптоэлектроникой, компьютерами, управляемыми потоком данных *(dataflow machine)*. Сосредоточив усилия на этих направлениях, можно реально приблизить рождение суперкомпьютеров новых типов.

Наконец, для приобщения вузов и университетов к работам над суперкомпьютерами необходимо создавать широкодоступные суперкомпьютерные центры, подобно тому, как это было сделано в США в начале 80-х годов. Когда Национальный научный фонд США принял решение субсидировать пять университетских центров для этих целей, что ознаменовало новую эпоху в разработке и освоении суперкомпьютеров. Тысячи научных работников и аспирантов получили доступ к невиданным доселе вычислительным мощностям. Из числа этих аспирантов вышли Чен, Хиллис, Дэвенпорт (Chen, Hillis, Davenport) - все те, кто задают тон в разработке новейших машин. Если мы хотим, чтобы наших аспирантов ждала такая же судьба, необходимо уже сегодня предоставить им доступ к "Эльбрусу-3", "Электронике ССБИС" - и что там у нас есть еще? И как знать, может быть, к 2000-му году исчезнет с повестки дня вопрос: "На сколько же лет мы отстали?".

СЛОВАРЬ

benchmarking - оценка производительности вычислительной системы с помощью эталонной тестовой задачи.

concurrency *(параллелизм)* - параллельное выполнение нескольких процессов. Термин применяется при описании общих принципов организации одновременного выполнения процессов в вычислительной системе.

dataflow machine (*компьютер, управляемый потоком данных*). Сегодня является предметом интенсивных исследований, поскольку основан на новых принципах управления, отличных от классической фон-Неймановской концепции последовательного потока управляющих команд.

Flynn classification *(классификация Флинна)* - один из наиболее распространенных способов классификации вычислительных систем. Согласно классификации Флинна, вычислительные системы делятся на четыре категории :

- **SISD (Single Instruction, Single Data)** - система с одним потоком команд и одним потоком данных ;
- **SIMD (Single Instruction, Multiple Data)** - система с одним потоком команд и несколькими потоками данных ;
- **MISD (Multiple Instruction, Single Data)** - система с несколькими потоками команд и одним потоком данных ;
- **MIMD (Multiple Instruction, Multiple Data)** - система с несколькими потоками команд и несколькими потоками данных.

Практически все сегодняшние суперкомпьютеры относятся к категориям SIMD и MIMD, причем в каждой из категорий представлен широкий диапазон архитектур.

MFLOPS (Million Floating-Point Operations Per Second) (*миллион команд с плавающей точкой в секунду*) - одна из основных единиц измерения производительности вычислительных систем.

MIPS (Million Instructions Per Second) (*миллион команд в секунду*) - одна из единиц измерения производительности вычислительных систем.

multicomputer (*многомашинная система*) - этим термином обозначают слабосвязанные многопроцессорные системы с распределенными локальными запоминающими устройствами.

neural network (*нейронная сеть*) - архитектура вычислительной системы, в той или степени моделирующая процесс обработки информации нейронами мозга.

VLSI (Very Large-Scale Integration) (*сверхвысокий уровень интеграции*) - технология изготовления интегральных микросхем, позволяющая объединять в одном кристалле более 1 млн. транзисторов. Технология VLSI является основой изготовления элементной базы практически всех современных суперкомпьютеров.

Роман с компьютером

Н. Е. Покровский

Говорят, крестьянин любит свою лошадь какой-то особой любовью. Охотник благоговеет перед породистой собакой и нарезным карабином. Водитель души не чает в своем автомобиле.

В силу разных причин я начисто лишен всех этих благородных чувств.

Зато у меня есть компьютер, и он, мой товарищ и друг, совмещает в себе все самые дорогие и ценные качества: он везет меня (по тернистой дороге научной жизни), кормит (тощими гонорарами за публикации), охраняет (от агрессивных профессиональных машинисток) и утешает (в силу возможностей своих игровых программ). Притом делает все это весело, элегантно, подчеркивая мое высокое технологическое превосходство над всем остальным населением. Разве этого мало? Честно говоря, я и предполагать всего этого не мог.

Так что, если хотите, называйте это романом. Между мной и моим "INTEL CWS-386".

Год 1989-й, положительно, был для меня удачен, хотя бы в общих чертах. Я, скромный преподаватель истории социологии московского вуза, победил в американском национальном конкурсе ученых-гуманитариев, что дало мне право на годичную командировку в США и самостоятельную научную работу в одном из привилегированных академических центров, расположенных в Северной Каролине.

Такова предыстория нашего рассказа, а вот и его начало.

Роковое решение

По приезде в расположение центра, мало-мальски сориентировавшись в ситуации, я с удивлением обнаружил, что помимо общей зарплаты мне положена сумма, называемая "грантом" и предназначенная исключительно для приобретения оборудования, отвечающего целям моего научного поиска. Если откровенно, то все мое "оборудование" ограничивалось ручкой и чистой бумагой. Но я стал думать, как бы оприходовать эти с неба свалившиеся деньги. И тогда меня осенило: а не купить ли замечательный новый компьютер. Чем не оборудование?

Некоторый опыт общения с компьютерами семейства IBM PC у меня уже был в Москве, так что сомнений не возникло: с порога приобрести современный персональный русифицированный компьютер фирмы IBM, нужный мне для работы над текстами рукописи. Наверное, и не стоило бы об этом упоминать, если бы не некоторые любопытные "последствия" этой моей инициативы.

Опыт общения с компьютером XT у меня действительно уже был, да опыт неоднозначный. Помнится, я уже пытался русифицировать плату адаптера Hercules старого выпуска, но так и не смог этого сделать до конца. Только перед самым отъездом в Америку приятель (болгарин) привез из Софии новый экранный знакогенератор. Но и тут не все получилось, как надо. Какая-то странная пошла перекодировка. Но уже было не до этого.

Итак, созрело решение обзавестись "настоящей" машиной.

Хотя я в Америке был не в первый раз, но тем не менее наивно полагал, как до сих пор полагает и большинство моих сограждан, что ежели у тебя есть необходимые средства, то там, в Новом Свете все можно купить.

После двух недель, которые я провел в телефонных переговорах, мне, наконец, удалось оторваться от торговой сети фирмы IBM в провинциальной Северной Каролине и достичь высшего уровня - Агентства всемирной торговли фирмы IBM в Нью-Йорке. Почему я так высоко залетел? Да потому, что здесь, на Юге, эксперты фирмы никак не могли в толк взять, чего я от них добиваюсь. Кое-кто из них с удивлением узнавал от меня, что в России славянский алфавит, а не английский.

Не наша проблема

Как бы то ни было, но милый женский голосок в Агентстве всемирной торговли фирмы IBM четко отрапортовал:

- Ваш вопрос нам известен. Но помочь ничем не можем. Это не наша проблема.

- Что не ваша проблема? - не сразу сообразил я. - Иметь наготове самое обыкновенное, если не сказать примитивное

Никита Евгеньевич Покровский - историк американской философии и социологии, журналист, автор книг "Генри Торо" (М: Мысль, 1983), "Ранняя американская философия" (М: Высшая школа, 1989), "Henry Thoreau" (М: Прогресс, 1988), нескольких научных сборников, в том числе сборника "Лабиринты одиночества" (М: Прогресс, 1989) и многих статей. Преподает на социологическом факультете Московского университета им. М.В. Ломоносова. Ведущий научный сотрудник Высшей школы международного бизнеса в Академии народного хозяйства СССР. Консультант фирм "Пепсико", "Старки" и др. Лауреат Премии Ленинского комсомола в области науки.

оборудование для возможного использования в стране с почти 300-миллионным населением, чей язык является одним из шести официальных языков ООН?

Но подобные пафосные сантименты, исполненные благородной обиды за родное отечество, не очень тронули мою собеседницу.

- Объясняю Вам, сэр, это не наша проблема. В России у нас не открыт рынок. Надо ждать, - поставила она точку в нашем разговоре.

Не скрою, я расстроился.

Нелепость какая-то: страна под самое горлышко забита-перезабита компьютерами, но русскоязычного среди них найти нельзя. А в Москве, отнюдь не испытывающей перенасыщенности этой полезной техникой, толковый программист или техник-электронщик русифицирует современный компьютер - либо программными средствами, либо "прошивкой" микрочипа.

Я к Вам пишу, чего же боле...

Расстройство свое и свои соображения я в самых изысканных выражениях и оборотах делового стиля положил на бумагу и отослал ее президенту фирмы IBM.

Как принято говорить, ответ не заставил себя долго ждать. На третий день запыхавшаяся секретарша принесла мне только что привезенный из ближайшего аэропорта пакет срочной почты ("Из самой IBM", - произнесла она с таким же почтением, как если бы на конверте стоял гриф Белого дома или Капитолийского холма).

Не без любопытства я вскрыл конверт.

Отнюдь не президент, а некто из глубин среднего уровня вежливо отвел все мои недоуменные вопросы и в конце пообещал, что фирма вскоре разрешит все мои проблемы.

Далее последовали события удивительные и непредсказуемые, по крайней мере для меня.

Люди фирмы IBM

Плотный чередой ко мне в центр стали наведываться делегации из географически близко расположенных подразделений фирмы.

"Композиция" всех этих делегаций была до удивительности одинаковой. Они состояли, как правило, из двух джентльменов.

Один из них всегда был полный, упитанный, разговорчивый, веселый и добродушный. Костюм на нем сидел прекрасно, несмотря на слегка располневший животик, чемоданчик-дипломат матово поблескивал настоящей кожей. Туфли, которые он постоянно демонстрировал, высоко закидывая одну ногу на другую, являли совершенство сапожного искусства.

Второй человек всегда был прямой противоположностью своему спутнику. Он был худ, молчалив, иногда с бородкой, часто в свитере-водолазке. В отличие от "веселого", "грустный" внимательно слушал историю моих бедствий и задавал только технические вопросы, ответы на которые тщательно заносил в блокнот. В это время "веселый", не умолкая, говорил и говорил - видимо, у него по плану была такая установка: не молчать ни при каких обстоятельствах.

Все эти делегации (а их было три) роднили и еще некоторые черты. "Айбиэмовские" ребята буквально опивались дармовым черным кофе, которое подавалось в моем центре, а если поспевали к ланчу, т.е. к обеду по-нашенски, то аппетитно закусывали в моем обществе, также наслаждаясь безвозмездным гостеприимством Гуманитарного центра, который к тому же обнаружил при этом и черты гуманности.

В конце встреч они обещали вскоре решить мои проблемы, просили ни в коем случае не покидать "семью IBM" и не

НАШИ ПРОГРАММНЫЕ ПРОДУКТЫ - КЛЮЧ К УСПЕШНОМУ РЕШЕНИЮ ВАШИХ ПРОБЛЕМ!

Центр "ИНТЕРФЕЙС"
предлагает пользователям ПК, совместимых с IBM PC:

FORGRAF - НОВАЯ БИБЛИОТЕКА ГРАФИЧЕСКИХ ПРОГРАММ НА ФОРТРАНЕ77
Компактный набор графических функций; возможность использования курсоров и окон; работа с графиками, осями координат, гистограммами, текстом; перемещение и копирование сегментов изображений; ввод символов; интерактивная и научная графика. EGA/VGA-мониторы. *Цена 650 руб.*

GRAPH - ПАКЕТ ПРОГРАММ РАСШИРЕННОЙ ГРАФИКИ НА ЯЗЫКЕ СИ
Поставляется в исходных текстах. Позволяет отображать на экране оси координат, графики, гистограммы. Поддерживает работу со спрайтами, движущимися окнами, математической системой координат. Обладает другими уникальными возможностями. Допускает использование дополнительного буфера экрана, а также работу с "мышью". CGA- и EGA-мониторы. Turbo C и Microsoft C. *Цена 395 руб.*

ISP (Interactive Signal Processing) - ДИАЛОГОВАЯ СИСТЕМА МОДЕЛИРОВАНИЯ И ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ
Идеальное инструментальное средство для автоматизации научных исследований и анализа экспериментальных данных. Характеризуется простотой использования графических средств, развитым диалогом, наличием команд статистического анализа, спектральных преобразований (БПФ и др.), вычисления свертки и корреляционных функций, фильтрации, восстановления сигналов и др. Поставка в исходных текстах на языке Си. *Цена 750 руб., при поставке в исходных текстах - 2500 руб.*

По Вашим запросам будут высланы условия поставки, описание, демо-дискета. Для частных лиц скидка до 50 %.

Вопросы и заявки направляйте по адресу:
142432, Черноголовка, ННЦ АН СССР, а/я 33, "Интерфейс", Б.Н. Гайфуллину

перебегать к конкурентам из фирмы Apple. После этого следовало традиционное американское: "Будем поддерживать связь. Скоро получите от нас известия."

Никогда более я ничего не слышал от этих милых людей.

Да, и еще одна деталь. Каждая последующая делегация ничего не знала о посещении предыдущей.

Мои коллеги по Гуманитарному центру и, что более существенно, его администрация, видя подобное половодье "айбиэмовцев", решили, что у меня с ними какой-то "деловой роман", а коммерческая деятельность любого рода в стенах центра не допускалась.

Наконец последовал один любопытный звонок.

Очередной "айбиэмовец" из головного "НИИ" фирмы во Флориде представился как главный разработчик русскоязычной клавиатуры и экранных драйверов для русского алфавита. Грандиозно! Сердце мое наполнилось радостью путешественника, перед которым после перехода через безводную пустыню раскрывается панорама прекрасного города.

Но город этот, увы, оказался миражом.

Почему в Грузии грузинский алфавит?

Собеседник мой из Флориды отрекомендовался также и как дизайнер клавиатуры для других славянских языков. Упомянул он и прибалтийские языки, и грузинский. Над всем этим он, по его же собственным словам, с согласия советской Академии наук работал чуть ли не полтора года. Вскользь он поинтересовался у меня, правда ли, что в Прибалтике используют "не русские буквы". Когда же я сообщил ему, что в Грузии также почему-то имеют свой алфавит, а не русский, разработчик из Флориды воспылал ко мне огромным уважением и забросал массой лингвистических вопросов. У меня на душе потеплело. Моя скромная личность наконец обратила на себя внимание серьезного человека. Кому это не приятно?! Даже в Америке.

Как-то, после очередного продолжительного разговора по телефону разработчик из Флориды попросил меня помочь ему по всему кругу вопросов, связанных с его проектом. Для этого он собирался прилететь ко мне в Северную Каролину.

Обращался он ко мне теперь уже запросто и даже с некоторой требовательностью в голосе, словно я чем-то был ему обязан. "К моему приезду, - наставлял он меня, - подготовьте компьютер. На нем мы будем отрабатывать клавиатуру." - "Но где же я его возьму, ведь у меня его все еще нет? В этом же и состояла проблема." - "Ах, да, - "вспомнил" он. - Ну, не знаю. Подумайте сами. Неужели в вашем центре нет наших компьютеров?!"

Чаша терпения русского человека переполнилась. Пружина сжималась, сжималась, а потом и распрямилась.

"Сэр, я внимательно собеседовал с вами в течение ряда дней, - как мне казалось, спокойно промолвил я. - Но все же мне не ясен главный вопрос: "Кто кому должен помогать? И еще одно. По своей второй профессии я журналист. И потому собираюсь слово в слово описать историю моих хождений по мукам и опубликовать все это в Советском Союзе, а, возможно, и в США".

Последнее было мелко с моей стороны, я признаю это. Кроме того, мой выпад относительно журналистского расследования обнаруживал "русский синдром" административной мести.

В трубке молчали.

Никогда более фирма IBM ни прямо, ни косвенно мною не интересовалась и никак себя не обнаруживала.

Одесса начинается в Северной Каролине

Своего я все же добился. Это я насчет компьютера, "говорящего" по-русски.

Мой коллега по Гуманитарному центру, американский китаист, также столкнулся с проблемой переделки "машины", но на китайский иероглифический язык.

Короче, стопроцентный американский специалист по Китаю с видом стопроцентного советского заговорщика представил мне скромного неразговорчивого парня, произносившего не более одного слова в минуту. Парень этот работал инженером в местном отделении все той же фирмы IBM. А в "свободное от основной работы время" он собирал из новеньких серийных блоков своей фирмы, якобы присланных братом из Нэшвилла (ох уж этот Нэшвилл - американская Одесса!), "левую" продукцию. Сборка происходила у него в гараже, где я и получил свой компьютер самой последней модификации. (Я не отказал себе в удовольствии проверить все комплектующие по новейшему каталогу и обнаружил, что "молчаливый" на совесть впаял в мою машину самые "горячие" достижения компьютерного гения.)

Машина работала превосходно.

Она давала такие картинки на своем мониторе VGA NEC MiltiSync 3D, что даже видавшие виды американцы не могли отвести взгляда от всяких слайд-шоу, выплывающих на экран. Машина "болтала" по-русски, печатала, только что не пела, впрочем и пела тоже все, начиная от фуг Баха до рождественских песенок "Джингл белз".

Но не подумайте, ради Бога, что меня "купили" за бисер. Все эти слайд-шоу, песенки и прочая ерунда были лишь внешней оболочкой грандиозного внутреннего содержания: 33 МГц, два дисковода, 180-мегабайтный "винчестер" и CD ROM. (Даже CD ROM. Трудно поверить в это!) Что же касается периферии, то это тема для иной, не менее возвышенной поэмы.

Теперь я был счастлив. Счастлив вполне и наверняка.

И моя американская администрация тоже повеселела. Наконец она уверовала в то, что я все же социолог и у меня нет никаких коммерческих планов в отношении фирмы IBM.

Paradox говорит по-русски

В. Э. Фигурнов

В семействе русифицированных программных продуктов появилось новое солидное пополнение - система управления базами данных (СУБД) Paradox 3.5 фирмы Borland International. Русификация выполнена СП "ПараГраф" и группой программистов "КОКОМ" для обеих разновидностей версии 3.5 - 3.50 и 3.51, но пока не содержит перевода документации.

"Ну и что? - скажут многие читатели. - Мы используем для баз данных Dbase (Foxbase, Clipper, "Ребус", "КАРАТ" и т.п.), и зачем нам какой-то Paradox?". Однако все же стоит подумать, почему на Западе доля продаж СУБД Paradox, составлявшая в 1989 г. 9% от всех продаваемых СУБД, возросла до 27% в 1990 г. и до 35% к середине 1991 г.? Можно не сомневаться, что после приобретения фирмой Borland International ее основного конкурента в области создания СУБД - фирмы Ashton-Tate (разработчика DBase) объем продаж СУБД Paradox возрастет еще больше. Стоит разобраться в причинах столь большого успеха СУБД Paradox на западном рынке (ведь нельзя же все объяснить только хорошей маркетинговой политикой фирмы Borland).

Что такое Paradox

СУБД Paradox была первоначально разработана фирмой Ansa Software. Затем последняя была приобретена фирмой Borland International, которая и осуществляет дальнейшее развитие этой СУБД. Версия 3.5 СУБД Paradox была выпущена в 1990 г.

СУБД Paradox с самого начала была задумана, как наглядная и легкая в использовании система, рассчитанная на все категории пользователей - от новичков до опытных разработчиков информационных систем. Начинающие пользователи могут без какого-либо программирования выполнять достаточно сложные задачи по обработке данных, а опытные программисты с помощью встроенного языка программирования - создавать высокоэффективные информационные системы (приложения). Разумеется, все это не так-то просто реализовать, и даже такому гиганту в области программных систем, как фирма Borland International, понадобилось несколько лет, чтобы достичь необходимого сочетания удобства, мощности и производительности. Уже в версии Paradox 3.0 содержались практически все имеющиеся в этой СУБД в настоящее время возможности, но только в версии 3.5 они были реализованы при обеспечении необходимого быстродействия.

Перечислю наиболее характерные свойства СУБД Paradox.

Реляционная модель данных. К несомненным достоинствам СУБД Paradox следует отнести использование реляционной модели данных, в которой все данные представляются только в виде двумерных таблиц, а связи между данными задаются только в виде совпадающих ключевых полей в таблицах. Эти соглашения позволяют обеспечить простоту, единообразие и наглядность представления данных и средств обработки данных.

Средства манипулирования данными. Позволяют работать на уровне как отдельных записей, так и таблиц и групп таблиц. С помощью одного запроса к базе данных можно по достаточно сложным правилам поиска найти информацию в нескольких таблицах, изменить информацию в найденных записях таблицы, а также удалить эти записи либо создать на основе найденной информации новые записи в других таблицах. Задание запросов весьма наглядно - оно основано на использовании запроса по образцу (Query by Example). Определенным недостатком является ограниченность выражений (формул), которые можно использовать в запросах: в них могут входить только константы, имена полей и арифметические операции.

Простота и удобство интерфейса. Обеспечиваются такими свойствами СУБД Paradox, как:

- полная интегрированность диалоговой части СУБД Paradox, средств программирования и отладки программ;
- простота и интуитивность меню и запросов, наличие подробного встроенного справочника.

Средства представления данных. Обеспечивают возможности вывода данных на экран (экранные формы и графики) и на печать (отчеты). Экранные формы могут использоваться для просмотра и редактирования данных, а также для запросов информации. Удобство средств представления данных обеспечивается:

- наличием диалоговых средств для создания экранных форм и выводимых на печать отчетов;
- возможностью создания форм, включающих в себя вычисляемые или защищенные поля, данные из нескольких таблиц или нескольких записей одной таблицы, а также автоматически выполняемые проверки правильности вводимой информации;
- возможностью использования примерно 50 функций встроенного языка программирования PAL для создания вычисляемых полей в отчетах и экранных формах;
- наличием встроенных средств деловой графики, поддерживающих различные типы графиков и диаграмм.

Неприхотливость и производительность. Эти свойства СУБД Paradox особенно впечатляют в последней версии 3.5, в которой была установлена новая система управления памятью VROOM. Она позволила обеспечить:

- высокую скорость выполнения запросов даже при минимальной конфигурации компьютера;
- крайне скромные требования к аппаратуре - достаточно иметь компьютер IBM PC XT с 512 Кбайт оперативной памяти;
- эффективное использование расширенной и дополнительной (EMS) памяти.

СУБД Paradox использует математический сопроцессор, если он имеется в компьютере. На компьютерах с процессором 286, 386 или 486 и достаточным объемом (не менее 512 Кбайт) расширенной памяти СУБД Paradox

Виктор Эвальдович Фигурнов известен как автор книги "IBM PC для пользователя", трехтомного комплекта документации и программ "Работа пользователя с IBM PC", различных статей. Является директором НПО "Информатика и компьютеры" (103473, Москва, а/я 81).

может работать в "защищенном режиме" процессора, что весьма ускоряет выполнение запросов. Для построения графиков пригодны большинство видеоадаптеров (CGA, EGA, VGA, Hercules, 8514/A) и почти все виды принтеров и плоттеров.

Возможности программирования. Paradox является достаточно мощной системой и без использования программирования. Но для опытных разработчиков в СУБД Paradox имеется богатый и мощный встроенный язык программирования PAL. Этот язык весьма хорошо интегрирован в диалоговую подсистему СУБД Paradox:

- из PAL-программы непосредственно доступны все возможности диалоговой подсистемы СУБД Paradox;
- с помощью одного нажатия клавиш можно записать любую последовательность выполняемых действий в виде PAL-программы;
- редактор и отладчик PAL-программ, встроенные в СУБД Paradox, вызываются из ее общего меню.

PAL - достаточно ясный и логически стройный паскалеподобный язык, содержащий все необходимые средства структурного программирования: условные операторы, циклы, процедуры, средства локализации имен и т.д. Однако PAL в освоении не сложнее, чем Basic, - в нем нет обязательных описаний и строгой типизации переменных, жестких требований к структуре программы и т.д. Удобство программирования на PAL обеспечивают и около 300 встроенных процедур и функций (в основном ориентированных на задачи обработки данных), а также поставляемые вместе с СУБД Paradox библиотеки и генератор программ.

Средства работы в сети. Автоматически обеспечивают сохранность и целостность данных при одновременной работе нескольких пользователей с базой данных. Кроме того, в PAL-программах для большей их эффективности можно явно предусматривать меры по обеспечению целостности данных. Paradox поддерживает локальные сети Novell, 3COM, AT&T Starlan, Banyan VINES, IBM PC LAN. Для того, чтобы при функционировании в сети несколько пользователей могли одновременно работать с СУБД Paradox, следует приобрести пакет Paradox MultiPack, который обеспечивает подключение пяти дополнительных пользователей и стоит приблизительно столько же, сколько стоит сама СУБД Paradox.

Итак, Paradox - достаточно эффектная, удобная и производительная СУБД, не уступающая ни по одной из существенных характеристик своим основным конкурентам DataEase, R:Base, FoxPro и, тем более, DBase III и DBase IV. СУБД Paradox может быть успешно применена для построения информационных систем, для задач бухгалтерского учета, автоматизированных систем управления производством, хранения и обработки больших массивов данных, в том числе требующих одновременной обработки данных несколькими пользователями в локальной сети. Впрочем, Paradox отнюдь не является идеальным программным продуктом - как и все в этом мире. Некоторые недостатки СУБД Paradox будут обсуждены ниже.

СП "ПараГраф"

Основано в 1989 г. Специализация - разработка и распространение программного обеспечения и программно-аппаратных систем. СП "ПараГраф" является разработчиком системы русификации компьютеров BetaPlus, русифицированной версии текстового редактора Microsoft Word 5.0 "РусскоеСлово", текстового редактора "ЛЕКСИКОН" и интегрированной системы "Мастер", полиграфических шрифтов для лазерных принтеров типа Laserjet и Postscript.

Сопутствующие программные продукты

Для расширения своих возможностей пользователи СУБД Paradox и разработчики PAL-программ могут приобрести за дополнительную плату различные сопутствующие программные продукты. Рассмотрим некоторые из них.

Поддержка SQL-серверов. С помощью пакета Paradox SQL Link пользователь может получить доступ к удаленным SQL-серверам (базам данных, доступ к которым осуществляется с помощью запросов на языке SQL). После установки пакета SQL Link его возможности становятся доступными из меню СУБД Paradox. При этом очень удобно то, что пользователь не обязан изучать язык SQL - СУБД Paradox сама преобразует обычные запросы по образцу к удаленным базам данных в SQL-запросы. Результаты запросов к удаленной базе данных, как и результаты обычных запросов, выдаются в таблице Answer. Пользователи, знающие язык SQL, могут составлять собственные SQL-запросы в программах на языке PAL. Пакет Paradox SQL Link поддерживает работу программ OS/2 EE Database Manager 1.2, Oracle Server 6.0, Microsoft SQL Server 1.0.

Paradox Runtime и компиляция PAL-программ. Хотя разработанные на языке PAL программы принадлежат тому, кто их разработал, последний не может передать их другим лицам, переписав им СУБД Paradox, - это было бы незаконным копированием программ. Чтобы не заставлять пользователей PAL-программ покупать Paradox, разработчик программ может приобрести одну копию относительно дешевого продукта Paradox Runtime, который позволяет выполнять любые PAL-программы, и поставлять копию Paradox Runtime вместе со своими PAL-программами. Файлы Paradox Runtime занимают около 2 Мбайт.

Разумеется, было бы лучше, если бы в состав СУБД Paradox входил компилятор, преобразующий PAL-программы в исполняемые файлы, но пока что фирма Borland такого компилятора не разработала. Впрочем, фирма TSR Systems предоставляет компилятор PalCom, преобразующий PAL-программы в программы на языке Си. Этот подход позволяет подключать существующие библиотеки Си-программ, производить оптимизацию программ и вставлять их в существующие программные системы.

Paradox Engine. Для программистов, желающих использовать средства СУБД Paradox в своих программах, фирма Borland предлагает Paradox Engine - библиотеку подпрограмм, доступных из Паскаля и Си и позволяющих работать с СУБД Paradox.

Русификация СУБД Paradox

Теперь обсудим созданную СП "ПараГраф" и группой "КОКОМ" технологию русификации СУБД Paradox - в ней имеются весьма любопытные моменты.

Технология русификации позволяет создавать русифицированные версии без исходных текстов программ и без вмешательства в исполняемые файлы программы. Оригинальный программный продукт остается на диске без изменений, но перед его использованием должна быть запущена небольшая резидентная программа RUSS.EXE, которая на основе информации, содержащейся в нескольких дополнительных файлах на диске, модифицирует образ оригинальной программы в оперативной памяти так, что эта программа начинает "говорить по-русски".

Достоинства технологии. Несомненным достоинством этой технологии русификации являются:

- соблюдение авторских прав разработчиков оригинального продукта;
- возможность достижения высокого качества русификации: без каких-либо сокращений обозначений и с добавлением дополнительных возможностей (например, переноса как русских, так и английских слов или использования как русских, так и английских названий пунктов меню).

Недостатки технологии. Недостатки данного подхода - увеличение времени загрузки программ (весьма заметное на компьютерах типа XT) и некоторое уменьшение объема доступной программе оперативной памяти. Кроме того, требуемая для русификации программа RUSS.EXE работает только в том случае, если на компьютере установлена система базовой русификации BetaPlus того же СП "ПараГраф", а та сама требует достаточно большого объема оперативной памяти. Общий объем оперативной памяти, доступной русифицируемой программе, уменьшается приблизительно на 45 Кбайт, а доступной остальным программам - на 30 Кбайт. Однако для СУБД Paradox такое уменьшение объема доступной оперативной памяти вполне терпимо - ведь Paradox 3.5 может работать и на компьютерах с 512 Кбайт памяти.

Зачем нужна специальная русификация? Если на компьютере можно вводить русские буквы с клавиатуры и они изображаются на экране в текстовом режиме и выводятся на печать (так называемая базовая русификация компьютера), то на таком компьютере оригинальная СУБД Paradox будет воспринимать и выводить на экран русские буквы. Однако при этом:

- меню, сообщения и справочник СУБД Paradox останутся на английском языке;
- СУБД Paradox будет неправильно производить поиск для полей, содержащих русские буквы (поскольку поиск в СУБД Paradox не учитывает различий между прописными и строчными буквами);
- в запросах на поиск русские слова надо будет указывать в кавычках;
- в выводимых на экран графиках нельзя будет использовать русские буквы;
- сортировка полей, содержащих русские буквы, также будет неправильной (не алфавитной);
- названия месяцев придется указывать по-английски и т.д.

КОКОМ"

Это независимая группа программистов в составе: П.Зелинский, А.Ханов, В.Молонов, П.Сазонов и А.Сазонов. Основные направления деятельности - русификация программных продуктов и создание программных продуктов для сферы образования. Группа "КОКОМ" осуществляла русификацию пакетов Norton Commander, Paradox и Quattro Pro.

Поэтому для СУБД Paradox требуется специальная русификация, устраняющая указанные недостатки при работе с русскими буквами. Эта русификация и осуществляется программой P3R, разработанной группой "КОКОМ" и СП "ПараГраф". Данная русификация официально одобрена разработчиком СУБД Paradox фирмой Borland International.

Варианты русификации. Пользователю предоставляются два варианта русификации - базовый и полный. В обоих случаях адаптируется обработка русских букв (сортировка, перевод из прописных в строчные и т.д.) и дат, добавляются графические шрифты для вывода русских надписей на рисунках. В базовом варианте меню, сообщения и справочник СУБД Paradox остаются на английском языке, а в полном они представлены на русском языке. В обоих вариантах имена встроенных процедур и функций в PAL-программах остаются английскими. Для переключения с базового на полный вариант русификации и обратно достаточно запустить соответствующий командный файл.

Базовая русификация может больше устроить программистов, привыкших к английскому интерфейсу СУБД Paradox. В остальных случаях, по-видимому, целесообразно использовать полную русификацию.

Следует заметить, что даже в "полном" варианте русификации некоторые функции СУБД Paradox остались нерусифицированными. Например, с помощью функции Format значения логических переменных можно вывести как "Yes" или "No", но нельзя - как "Да" или "Нет".

Качество перевода в русификаторе является весьма высоким. Практически нигде нет программистского жаргона, неоправданных сокращений и т.д., хотя перевод некоторых терминов меню можно расценить как спорный.

Совместимость. Приятно отметить, что данная русификация обеспечивает хорошую совместимость с оригинальной версией СУБД Paradox: практически все программы (приложения) для стандартной СУБД Paradox могут без изменений работать с русификатором. Например, выбор пунктов меню может осуществляться с помощью первых букв как английского, так и русского названий, например пункт "Формы" (Form) выбирается при нажатии букв "Ф", "ф", "F" и "f". Аналогично в PAL-программе на этот пункт меню можно ссылаться, как на {Формы} и как на {Form}. Однако имеются причины и несовместимости в будущем оригинальной и русифицированной версий СУБД Paradox, в частности:

- в оригинальной версии СУБД Paradox названия месяцев в датах воспринимаются и выводятся только по-английски (например, 1-Jan-91), а в русифицированной - только по-русски (например, 1-Янв-91);
- сортировка и упорядоченность полей с русскими буквами в оригинальной и русифицированной версиях СУБД Paradox различны. Для изменения порядка сортировки в существующих базах данных вместе с русификатором поставляется специальная программа P3RCONV.

Использование русских имен. Русификатор СУБД Paradox позволяет использовать русские имена полей в таблицах и переменных в PAL-программах. Однако имена самих таблиц могут быть только английскими, так как они используются одновременно в качестве имен файлов.

Защита от копирования. Сама СУБД Paradox поставляется фирмой Borland без защиты от копирования, но русификатор СП "ПараГраф" защищен от копирования. Каждый комплект русификатора может быть установлен на компьютер пять раз, но, согласно договору о поставке, повторные установки на компьютер можно осуществлять только в том случае, если установленная копия русификатора испорчена. Впрочем, размещенную на жестком диске копию русификатора можно сохранить на дискете и, если копия будет испорчена, восстановить ее на том же компьютере с этой дискеты.

Пожелания

Как сообщается в западной прессе, фирма Borland International сейчас готовит новую версию СУБД Paradox - 4.0. СП "ПараГраф", несомненно, займется ее русификацией. По-видимому, в этих версиях будут учтены многие пожелания, которые можно высказать по поводу их улучшения.

Возможные улучшения СУБД Paradox можно классифицировать по нескольким направлениям (интересно, насколько они совпадут с действительностью).

Изменение системы меню стало бы наиболее очевидным и давно назревшим усовершенствованием. Горизонтальные меню СУБД Paradox смотрятся весьма архаично, и в версии Paradox 4.0 они, наверняка, будут заменены на ниспадающие меню.

Графический интерфейс - значительно более проблематичное, но весьма перспективное возможное усовершенствование. Фирма Borland уже внедрила графический интерфейс в своем табличном процессоре Quattro Pro 3.0, и, по-видимому, ей по силам сделать это и для СУБД Paradox. Несомненно, СУБД с красивыми и разнообразными шрифтами в экранных формах и печатаемых отчетах выглядит весьма эффектно, однако удобство ее использования будет зависеть от того, насколько естественно будут осуществляться выбор шрифтов и позиционирование надписей в формах и отчетах.

Компилятор PAL-программ фирмы Borland был бы весьма полезным инструментом для программистов, разрабатывающих информационные системы на языке PAL. Компилятор обеспечил бы ускорение выполнения PAL-программ (что очень важно для сложных программ) и избавил бы пользователей этих программ от необходимости хранить на диске СУБД Paradox или Paradox Runtime (занимающие соответственно 3,5 и 2 Мбайт).

Устранение ограничений, присущих версии Paradox 3.5, также было бы весьма полезным усовершенствованием. В самом деле, почему в отчетах можно использовать произвольные выражения от элементов текущей записи таблицы и "присоединенных" к ней таблиц, в формах - только от элементов текущей записи, а в запросах - вообще только арифметические выражения от элементов текущей записи? Почему в программе можно узнать установленный режим обработки пустых значений, но нельзя его изменить? Почему в форме можно вывести сведения о сумме всех чисел в колонке таблицы, но нельзя вывести сведения о сумме чисел, удовлетворяющих определенному условию? Подобные ограничения СУБД Paradox весьма досадны, и их обход часто требует большого объема программирования.

Средства для описания логического представления базы данных - весьма важное возможное направление развития СУБД Paradox. Некоторые примитивные средства такого рода (формы, управление видом таблиц) уже присутствуют в СУБД Paradox, но они, конечно, недостаточны и не позволяют автоматически обеспечить целостность баз данных, содержащих несколько связанных таблиц. Наилучшим решением было бы введение языка описания логического представления базы данных, как это сделано, например, в СУБД DB/2, но как паллиатив приемлемо повышение удобства и гибкости имеющихся средств, например форм, состоящих из множества записей или таблиц.

Пожелания по поводу русификатора, разумеется, тоже имеются. Вот некоторые из них:

- предложить в составе русификатора перевод документации по СУБД Paradox;
- обеспечить автономность русификатора от программы Beta;
- снизить требования русификатора к (обычной) оперативной памяти, по крайней мере, на компьютерах, имеющих расширенную или дополнительную память;
- обеспечить возможность отключения и деинсталляции русификатора;
- устранить оставшиеся английские слова в интерфейсе пользователя, например заменить значения функции Format "On" и "Off" на "Да" и "Нет";
- обеспечить возможность восприятия дат в виде, принятом в англоязычной версии (типа 1-Jan-91).

Впрочем, несмотря на все перечисленные пожелания, следует сказать, что СУБД Paradox и ее русификатор и в существующем виде являются первоклассными программными продуктами и, безусловно, стоят тех денег, которые за них нужно заплатить.

Что, где, почем?

СУБД Paradox и ее русификатор, а также сопутствующие программные продукты фирмы Borland можно приобрести в СП "ПараГраф" по адресу: 103051, Москва, Петровский бульвар, д. 23, тел. (095) 200-25-66, факс (095) 928-27-68. Цены (без 5%-ного налога по состоянию на 1 октября 1991 г.) на программные продукты вместе с соответствующими русификаторами и отдельно на русификаторы приведены ниже.

ПРОГРАММНЫЙ ПРОДУКТ	ЦЕНА ПАКЕТА, РУБ.	ЦЕНА РУСИФИКАТОРА, РУБ.
Paradox 3.5	9900	2000
Paradox Runtime 3.5	1400	500
Paradox MiltiPack 3.5	11900	4000
Paradox Engine 2.0	5500	2700

Эти пакеты можно приобрести и у других распространителей продукции фирмы Borland (как правило, по тем же ценам).

БЛОКИ ПРОГРАММНОГО КОДА

Рекомендации по написанию и использованию блоков программного кода и функций, в которых эти блоки используются

Г. Лиф

Блоки программного кода *(code blocks)* представляют собой программы, откомпилированные для среды Clipper 5.0. Они могут быть откомпилированы как составная часть Clipper-программы или во время выполнения программы с помощью оператора "&" (конечно, я знаю, что в этом нет особого смысла, но так часто делают).

Представим блок программного кода в самом общем виде:

```
{ | [<список аргументов>] | <список выражений> }
```

Блоки программного кода выглядят аналогично описаниям массивов в языке Clipper версии 5.0. И те, и другие начинаются с открывающей фигурной скобки ("{") и заканчиваются закрывающей фигурной скобкой ("}"). Однако блоки программного кода отличаются наличием двух символов "конвейера" ("|") сразу после открывающей фигурной скобки. По желанию Вы можете вставить между этими двумя символами <список аргументов>, который при вычислениях будет передан в блок программного кода. Объекты в <списке аргументов> должны отделяться друг от друга запятыми ("a,b,c..." и т.д.).

Содержащийся в блоке программного кода <список выражений> - это список допустимых выражений языка Clipper, разделенных запятыми. Как Вы увидите в дальнейшем, с их помощью можно делать все, что угодно.

Пробелы между символами "|" и фигурными скобками являются необязательными, хотя для улучшения "читабельности" я рекомендую их использовать.

Как написать блок программного кода

Блоки программного кода могут быть написаны таким образом, что они будут откомпилированы либо вместе с остальной частью программы, либо при ее выполнении. В приведенном ниже примере блоки программного кода создаются во время компиляции:

```
local myblock := { | | fname }
```

С помощью оператора "&" Вы можете написать символьную строку, которая во время выполнения будет откомпилирована в блок программного кода. Это тот же самый оператор, который используется для записи макросов, однако в данном случае он не является макрозаменой.

© DBMS, April 1991.
All rights reserved. M&T Publishing, Redwood City, CA, USA.

Предположим, что Вы захотели ввести некоторый объект TBrowse(), чтобы просматривать базу данных. Вам потребуется установить номер столбца для каждого поля базы данных. При установке номеров столбцов для объекта TBrowse() Вы должны указать блок программного кода, в котором хранится содержимое данного столбца. Если Вы заранее знали, что в базе данных есть поля FNAME, LNAME и SSN, то написание блоков программного кода таким образом, чтобы они могли быть откомпилированы во время сеанса компиляции, не составит для Вас труда:

```
local x, browse := TBrowseNew(3, 19, 15, 60), column
use test
column := TBColumnNew("FNAME",{ | | fname })
browse:AddColumn( column )
column := TBColumnNew("LNAME",{ | | lname })
browse:AddColumn( column )
column := TBColumnNew("SSN",{ | | ssn })
browse:AddColumn( column )
```

Предположим, однако, что Вы хотите сделать эту подпрограмму независимой от конкретной базы данных. Тогда Вы не сможете напрямую использовать имена полей, поскольку до выполнения программы структура базы данных не будет известна. В следующем примере функция FIELD() языка Clipper возвращает имя поля по его порядковому номеру в структуре базы данных:

```
local x, browse := TBrowseNew(3, 19, 15, 60), column
use test
for x := 1 to fcount()
column := TBColumnNew(field(x) ,&("{ | | "+;
  field(x)+"}"))
browse:AddColumn( column )
next
```

Функция FIELD(2) возвращает имя второго поля в базе данных (в моем примере это "LNAME").

Если Вы не хотите сами браться за написание блоков программного кода, то можете позволить препроцессору выполнить эту работу за Вас. Например, если Вы запишете:

```
index on fname to customer
```

препроцессор создаст следующий блок программного кода:

```
__dbCreatIndex( "temp", "fname",
    {| | fname},if(.F.,.T.,NIL))
```

Вычисление блоков программного кода

Единственной операцией, которую Вы можете выполнить над блоком программного кода, является его вычисление. Это вычисление можно представить как вызов функции, возвращающей какое-то значение. Блоки программного кода вычисляются с помощью функций EVAL(), AEVAL() или DBEVAL(); они могут также вычисляться "внутренне" при передаче их в качестве параметров в функции, в которых такие параметры могут использоваться. После вычисления блок программного кода возвращает значение крайнего правого выражения, расположенного внутри блока. Например, если Вы вычисляете блок программного кода

```
local myblock :={ ¦ ¦ mvar }
```

то функция EVAL() возвращает значение MVAR:

```
local myblock :={ ¦ ¦ mvar }, mvar := 500, x
x := eval(myblock)
? x                          // на печать: 500
```

Поскольку блоки программного кода могут содержать любое допустимое выражение языка Clipper, Вы можете использовать в них значительно более изощренные конструкции. Например:

```
local myblock :={ ¦ ¦ qout(var1), qqout(var2), 500 }
local var1 :="Mister ", var2 :="Grump"
x := eval(myblock)       // на печать: "Mister Grump"
? x                      // на печать: 500
```

Каким образом переменная x получает значение 500? Функции вычисления возвращают значение последнего (самого правого) выражения в блоке. Поскольку таким выражением в блоке Myblock было 500, переменная x принимает это значение.

Далее я расскажу об использовании блоков программного кода с параметрами. В приведенной программе показаны некоторые простые блоки программного кода, в которых параметры не используются.

Блоки программного кода и параметры

Как и в случае использования функций, при передаче параметров у Вас появляется возможность более "продуктивного" использования блоков программного кода. Написание списка параметров для блока программного кода аналогично написанию списка параметров для функции, однако гораздо сложнее разобраться в его использовании в "линейном мире" блоков программного кода. Простой блок программного кода

```
local myblock := { ¦ a, b, c ¦ max(a, max(b, c)) }
```

может быть записан как функция:

```
function mmax(a, b, c)
return max(a, max(b, c))
```

Функция MMax() возвращает значение наибольшего из переданных ей трех параметров. К тому же самому результату приведет вычисление блока программного кода MyBlock. Однако Вы должны сначала преодолеть другое препятствие: передачу параметров в блок программного кода.

Функция EVAL() присваивает необязательные параметры, расположенные после имени блока программного кода. Каждый необязательный параметр представляет собой параметр, который должен быть передан блоку программного кода. Например, если Вы напишете:

```
eval(myblock, 20)
```

ПРОГРАММА

```
local myblock := { ¦ ¦ qout(mvar) }, mvar := "testing"
eval(myblock)     // output: "testing"

myblock := {¦ ¦ 5000 }
x := eval(myblock)
? x               // output: 5000

myblock := { ¦ ¦ x++ }
for y := 1 to 100
  eval(myblock) // crashes because X not defined
next
? x

myblock := { ¦ ¦ x++ }, x := 1  // much nicer thanks
for y := 1 to 100
   eval(myblock)
next
? x               // output: 101

myblock := { ¦ ¦ BlueFunc() }
eval(myblock)    // calls BlueFunc() which displays a message
return nil

static function bluefunc
? "here we are in a BlueFunc() - will we ever escape?"
inkey(5)
return nil
```

то в MyBlock будет передан параметр 20. Давайте еще раз рассмотрим функцию MMax() и блок программного кода, чтобы Вы могли разобраться в механизме передачи параметров с помощью функции EVAL():

```
local myblock :={ ¦ a, b, c ¦ max(a, max(b, c)) }
? mmax(20, 100, 30)              // на печать: 100
? eval(myblock, 20, 100, 30)     // на печать: 100
```

Вот как выглядит блок программного кода, которому можно передать до трех параметров (эти параметры он выводит на экран):

```
local myblock :={ ¦ a, b, c ¦ qout(a, b, c) }
eval(myblock, 1, 2, 3)           // на печать: 1 2 3
x := eval(myblock, 1, 2)         // на печать: 1 2 NIL
? x                              // на печать: NIL
```

Вы ведь уже знаете, почему вторая команда функции EVAL() выводит на печать значения 1, 2 и NIL? Это происходит потому, что все объявленные параметры, которым не присвоены значения, получают начальное значение NIL. (По поводу значения NIL см. мою статью в декабрьском выпуске журнала "The Aquarium" за 1990 г.) Поскольку MyBlock ожидает получения трех параметров (A, B, C), а передано только два, то C принимает начальное значение NIL. Вопрос "на засыпку": "Знаете ли Вы, почему x принимает значение NIL?" Нет, это не имеет никакого отношения к тому, что было передано слишком мало параметров, а связано с тем, что блок

Грег Лиф (Greg Lief) является одним из разработчиков системы Clipper и преподавателем по этой системе, а также одним из авторов книги по системе Clipper версии 5.0, подготовленной для издательства MacMillan Publishing. Данная статья первоначально появилась в посвященном системе Clipper журнале "The Aquarium", выпускаемом на дискетах фирмой Grumpfish (тел: 503-588-1815; факс: 503-588-1980).

программного кода возвращает значение выражения QOut(a, b, c). Функция же QOut() всегда возвращает значение NIL.

Важно отметить, что всем определенным нами в блоке аргументам автоматически присваивается тип LOCAL. Такие аргументы будут недоступны в блоках программного кода, вложенных в данный. Это обстоятельство заслуживает отдельного примера:

```
local firstblock := { | | qout(x) }
local myblock := { | x | x++, eval(firstblock) }
eval(myblock, 3)
```

При попытке вычислить блок FirstBlock() эта программа не сработает. Может показаться, что аргумент x из блока MyBlock() должен быть доступен внутри блока FirstBlock(), однако x является переменной типа LOCAL для блока MyBlock(), и поэтому она недоступна для блока FirstBlock().

В оригинальном варианте этой статьи, написанном для журнала "The Aquarium", подробно обсуждалось использование блоков программного кода с функциями. Для данной публикации подобное обсуждение было бы слишком длинным, однако Вы можете запросить этот материал по каналам сети TelePath BBS.

Передача переменных типа LOCAL

Мы знаем, что переменная типа LOCAL действует только в той процедуре или функции, в которой она была объявлена. Однако на самом деле существует способ передать такую переменную в другую функцию. Для этого требуется использование блока программного кода:

```
function main
local bblock := { | | x }, x := 500
test1(bblock)
return nil

function test1(b)
? eval(b)// на печать: 500
return nil
```

При компиляции блока BBlock() в составе функции MAIN() он будет содержать ссылку на переменную x, которая является локальной для функции MAIN(). Однако при передаче блока BBlock как параметра в функцию TEST1() и последующем его вычислении в этой функции значение переменной x будет присвоено.

Я не призываю к неумеренному применению подобного метода. Это не совсем то, что имели в виду создатели языка для переменных типа LOCAL, не правда ли? Однако однажды это сэкономило мне день работы. Я хотел получить значение переменной с помощью команды GET в режиме, когда пользователь выбирает нужное значение из списка и вводит его нажатием "горячей" клавиши. Сформулирована задача очень просто, за исключением того, что переменная, о которой идет речь, имела тип LOCAL, что ограничивало возможность ее использования рамками той функции, в которой я ее получал. Решить проблему мне помогло использование блока программного кода. Это решение, а также множество других примеров использования обсуждаемых методов Вы найдете в программах по теме данной статьи, передаваемых по каналам сети TelePath (415-364-8315).

Надеюсь, Вы почувствовали преимущества создания блоков программного кода. Как и для многого другого в системе Clipper 5.0, единственным ограничением при работе с блоками программного кода является лишь Ваше воображение.

Библиотека DBL для системы управления базами данных Clipper.

Библиотека DBL разработана специально для высокотехнологичного программирования интерфейса с пользователем.

Данные в табличной форме наиболее наглядны и привычны для пользователя. Всевозможные формы таблиц, пересчет данных, рекурсивные вызовы, сетевые возможности, стандартные средства корректировки данных (get, delete, recall, append blank, insert blank before), поиск "по образцу", одновременная работа с несколькими таблицами на экране, печать таблиц обеспечиваются подсистемами DBA, FCL, SRH, DPA. При этом сохраняется полный контроль над программой.

Вы можете определить и дополнительные действия, выполняемые при добавлении новой записи, после корректировки клетки таблицы, при перемещении к новой колонке или строке, при попытке выйти за пределы таблицы, при нажатии определеных клавиш, при попытке корректировки клетки - всего свыше 14 событий. Возможности явного указания формул упрощают программирование пересчитывающихся таблиц (аналог электронных таблиц). Имеется множество полезных мелочей: выделение отрицательных чисел другим цветом, подсветка текущей строки, задание типа рамки, наличие тени, указание направления позиционирования по таблице и т.д.

Компактность программирования запросов, сообщений, горизонтальных и вертикальных меню обеспечивается подсистемами GSA, MNU, LMU. Указание цвета, типа рамки, вариантов выравнивания и центрирования заменяют расчет координат экрана.

Подсистемы могут работать и поставляться независимо, но наиболее эффективна их совместная работа. Низкая стоимость библиотеки позволит окупить ее мгновенно.

Использование подсистем требует минимума операторов в программе. Особенно ценно, что при модификации программ существенно упрощается внесение изменений.

Тщательно подобранные варианты умолчаний и простота освоения позволяют разрабатывать профессиональные приложения и программистам, только начинающим изучение Clipper'a.

Демонстрационная дискета с примерами использования подсистем и дополнительной информацией может быть выслана наложенным платежом (52 руб).

Адрес: **446025, г. Сызрань, Самарская обл., КИП "Практика".**
Телефон: **59-22-20.**

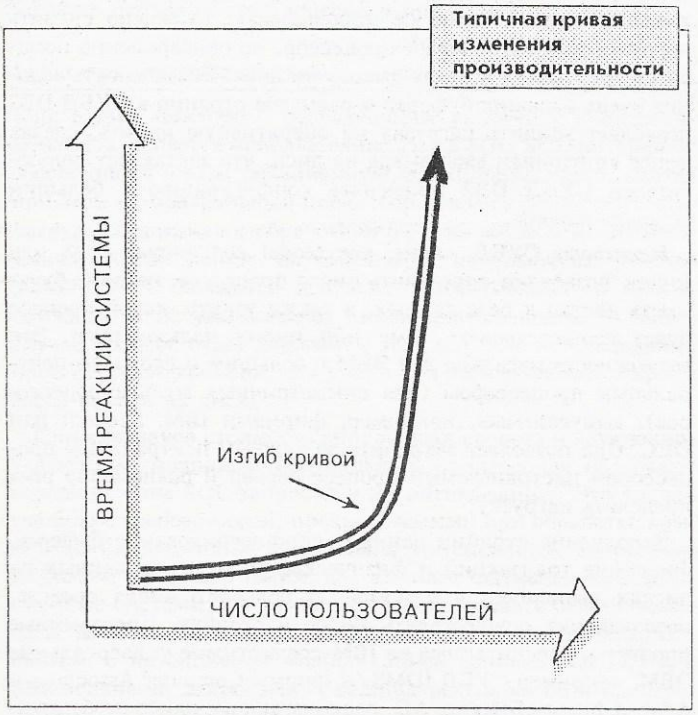
Преодоление "узких мест"

Преодоление трудностей, связанных с повышением производительности баз данных.

М. Батлер, Р. Блор

Обычно при снижении производительности обработки данных пытаются либо получше настроить базу данных, либо усовершенствовать аппаратное обеспечение. Для этого могут потребоваться нежелательные усилия или расходы, но обычно с подобными ситуациями можно справиться. Однако иногда возникают такие проблемы, которые так просто не решить.

Как правило, во всех базах данных есть "узкие места", вызывающие существенное снижение производительности при возрастании числа транзакций. Изгиб кривой, изображенной на рисунке, соответствует возникновению "узкого места", обусловливающего появление очереди ожидания того или иного ресурса. С этого момента добавление в систему каждого



нового пользователя вызывает непропорционально большую задержку во времени реакции (отклика) системы для всех ее пользователей. Если число пользователей будет и дальше расти, то система вообще окажется не в состоянии продолжать работу в режиме реального времени.

Очереди обычно появляются тогда, когда около 80% ресурсов системы уже используются и для управления очередностью обслуживания пользователей требуется принятие дополнительных мер. При этом возникает дополнительная нагрузка на ресурсы, что достаточно быстро приводит к серьезному ухудшению реакции системы. Большинство опытных специалистов по обработке данных сталкивались с подобными ситуациями. Когда кривая изменения времени реакции системы доходит до критического изгиба, наступает время настройки системы, что, возможно, означает настройку базы данных. Такая настройка не повлияет на форму кривой изменения производительности, однако может растянуть эту кривую в горизонтальном направлении таким образом, что система окажется в состоянии принять еще несколько пользователей или обеспечить меньшее время реакции для уже существующих пользователей.

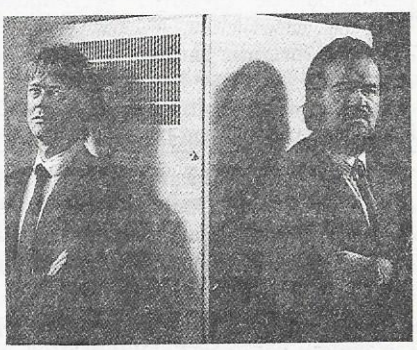
После того как возможности настройки будут исчерпаны, все остальные меры могут быть связаны только с усовершенствованием аппаратного обеспечения.

Существуют четыре основные области, где могут возникнуть "узкие места": центральный процессор, память, ввод-вывод на диск и конкурентный доступ к данным. Когда производительность системы начинает ухудшаться, на основании статистических данных, полученных при исследовании производительности системы, обычно можно определить, какая из этих областей вызвала снижение производительности. Способ решения проблемы зависит от того, что является "узким местом".

"Узкие места", связанные с центральным процессором

До тех пор пока архитектура аппаратного обеспечения позволяет последовательно наращивать мощность центрального процессора, используемого для создания базы данных, решение проблемы "узких мест", связанных с центральным про-

Мартин Батлер (Martin Butler) и Робин Блор (Robin Bloor) являются директорами фирмы ButlerBloor, консультирующей по вопросам работы с базами данных. С ними можно связаться по телефону 602-254-7866 (908-373-311 в Великобритании).



© DBMS, April 1991.
All rights reserved. M&T Publishing, Redwood City, CA, USA.

цессором, обеспечивается добавлением новых процессоров или улучшением качества используемых. Использование дополнительных вычислительных мощностей может оказаться сложным для проектирования и привести к изменениям в архитектуре системы.

Возникновение "узких мест", связанных с центральным процессором, при использовании архитектуры "клиент-сервер" менее вероятно, поскольку при этом центральный процессор специально предназначен для обеспечения доступа к данным.

Рабочие характеристики базы данных зависят от того, насколько хорошо СУБД управляет доступными ей ресурсами и конкурентным доступом.

Если же узкие места все-таки возникают, то единственным способом их преодоления (после того как возможности настройки базы данных исчерпаны) является наращивание вычислительной мощности центрального процессора.

"Узкие места", связанные с памятью

С "узкими местами", связанными с памятью, бороться проще всего. Стоимость 1 Мбайта памяти продолжает снижаться, и покупка дополнительной памяти почти всегда обходится дешевле, чем изменение конфигурации системы.

В большинстве случаев преодоление "узких мест", связанных с памятью, обусловливает замену других ресурсов системы: центрального процессора, диска или же и того, и другого.

Применительно к базам данных память используется в основном для буферизации и объединения данных. Не следует изменять конфигурацию буферов с целью увеличения их объема, поскольку, если он будет слишком большим, это скорее будет мешать, чем помогать.

"Узкие места", связанные с вводом-выводом на диск

Диск обычно наиболее сильно ограничивает быстродействие, поскольку это устройство самое "медленное" из рассматриваемых. Одной из наиболее сложных задач, на решение которой направлены усилия разработчиков баз данных, является задача снижения степени влияния этого ограничения.

На минимизацию времени доступа к диску при различных рабочих нагрузках в системе в совокупности влияют буферизация, применение различных структур данных, размещение таблиц и многое другое.

При возникновении "узких мест", связанных с диском, весьма существенное увеличение производительности может дать настройка базы данных, если в СУБД имеются разнообразные средства регулировки производительности. Простое добавление дисков может и не улучшить дела, если организация данных на диске оказывается недостаточно гибкой.

Конкурентный доступ к данным

Конкурентность, пожалуй, худшее из "узких мест", с которым приходится иметь дело и которое чаще всего возникает в системах с интенсивной модификацией данных, когда данные часто блокируются для обеспечения исключительного доступа к ним.

Обычно в базе данных имеется несколько "горячих точек", где записи часто модифицируются, и конкуренция в получении доступа к такой информации ведет к ухудшению производительности.

Если проблемы, связанные с конкурентным доступом, однажды возникли, они очень быстро становятся хроническими. Хуже того, кроме планирования доступа пользователей к системе (что обычно является весьма непопулярной мерой) или изменения конфигурации системы, в этих ситуациях мало что можно сделать.

Изменение конфигурации, связанное с проблемами конкурентного доступа, обычно производится таким образом, чтобы целые структурные части базы данных удовлетворяли требованиям обеспечения необходимого уровня производительности.

Средства изменения производительности

Рабочие характеристики базы данных полностью зависят от того, насколько хорошо СУБД управляет как доступными ей ресурсами, так и конкурентным доступом.

Что касается аппаратного обеспечения, то большинство СУБД позволяют администратору базы данных сбалансировать работу центрального процессора, памяти и диска. Например, в СУБД Adabas фирмы Software AG и Supra фирмы Cincom используется сжатие данных. Это сокращает время ввода-вывода на диск, однако требует дополнительной работы центрального процессора по распаковке данных при их считывании. Если сжатие не использовать, то можно снизить нагрузку на центральный процессор, но одновременно повысятся требования к вводу-выводу на диск. Возможность задания очень больших буферов и размеров страниц в СУБД DB2 позволяет усилить нагрузку на оперативную память, сделав менее критичным ввод-вывод на диск, что заставляет пользователей СУБД DB2 применять конфигурацию с большим объемом памяти.

Некоторые СУБД, такие, как Model 204 фирмы CCA или Ingres, позволяют определить число процессов, которые будут иметь доступ к базе данных, а также узнать, какой процесс будет соответствовать тому или иному пользователю. Эта возможность идеальна для ЭВМ с большим и сложным центральным процессором (для симметричных мультипроцессоров), выпускаемых, например, фирмами IBM, Sequent или DEC. Она позволяет закрепить за каждым центральным процессором настраиваемый процесс-сервер и равномерно распределить нагрузку.

Заполнение страниц данных, конфигурирование буферов, фиксация транзакций и физическое размещение данных на дисках выполняются с помощью большого числа средств, позволяющих осуществлять такую настройку. Программные продукты, рассчитанные на IBM-совместимые универсальные ЭВМ, например СУБД IDMS/R фирмы Computer Associates и Adabas фирмы Software AG, располагающие наиболее богатым выбором средств настройки, по-видимому, являются наиболее надежными при использовании в больших базах данных, где наиболее важны соображения производительности.

Однако весьма незначительное число средств настройки имеет отношение к конкурентному доступу. Наиболее важным из них является блокировка на уровне записи. Как это ни удивительно, некоторые из известных СУБД, например DB2, Ingres и Sybase, не имеют этого средства. В них блокировка производится на уровне страниц, что приводит к блокировке множества записей, хотя заблокирована должна быть только одна из них. Учитывая высокую вероятность возникновения

проблем с конкурентным доступом, можно считать использование вышеупомянутых СУБД в системах с интенсивной модификацией данных весьма проблематичным.

Другим средством, которое помогает в какой-то степени решать проблемы конкурентного доступа, является получение "снимков" данных, иногда называемое "грязным" считыванием. Это средство обеспечивает доступ к считыванию (в рамках обработки запросов) записей, заблокированных для изменения. Подобное средство должно использоваться очень осторожно, поскольку целостность данных не гарантируется, однако оно может уменьшить вероятность конкурентного доступа, так как при считывании данных блокировка не производится. Некоторые СУБД, например Oracle, обеспечивают доступ с получением "снимков" данных, однако целостность данных при этом по-прежнему гарантируется. В СУБД Oracle это достигается благодаря сохранению в памяти различных версий записи, что позволяет перераспределить "нагрузку" между памятью и конкурентным доступом в пользу последнего.

Производительность - не единственная характеристика, которая принимается во внимание при создании программного обеспечения для работы с базами данных. Большинство СУБД, предназначенных для использования на ПК, написаны на языке С и, с точки зрения скорости работы, не могут конкурировать с такими СУБД, как Model 204 на ПК фирмы IBM или System 1032 на машинах семейства VAX, поскольку эти СУБД написаны на родном для этих сред ассемблере. Для обеспечения мобильности программных продуктов прибегают к другим ухищрениям: СУБД Oracle, например, использует свои собственные типы данных, что приводит к более высокой загрузке центрального процессора при обработке данных.

Типы прикладных систем

Еще одним фактором, позволяющим увеличить производительность, является использование языка SQL. SQL-интерфейс обеспечивает СУБД реляционную функциональность, однако при этом на центральный процессор ложится дополнительная нагрузка,связанная с обработкой SQL-запросов и их оптимизацией. К сожалению, оптимизаторы, используемые некоторыми СУБД, являются довольно примитивными и иногда снижают, а не увеличивают производительность. В любом случае процесс будет дорогим, с точки зрения использования процессорного времени, даже если при этом экономится время ввода-вывода на диск.

Снижающаяся стоимость аппаратных средств и неуклонное увеличение мощности центральных процессоров облегчают осуществление SQL-запросов и их оптимизацию. Это удачно совпало с требованиями, предъявляемыми при обработке коммерческих данных к системам с интенсивным запросным режимом, например таким, как административные информационные системы, где лучше всего использовать реляционные базы данных. В общем, существует два типа прикладных систем: с интенсивной модификацией записей (OLTR) и с интенсивными запросами. Средства регулирования производительности СУБД обычно ориентированы на один из этих типов.

Сочетание таких средств, как блокирование на уровне записи, доступ по рандомизированному ключу *(hash-key)* поиска и возможность конфигурирования буферов малого размера обычно присущи системам с интенсивной модификацией записей, тогда как возможность конфигурирования очень больших буферов и использование B-деревьев или инвертированных списков ориентированы на системы с интенсивными запросами. Например, СУБД DB2, очевидно, предназначена для прикладных систем с интенсивными запросами, тогда как СУБД Rdb фирмы DEC и Supra фирмы Cincom являются универсальными и могут использоваться в системах обоих типов.

В некоторых исследовательских организациях совершают ошибку, полагая, что, если вычислительная система обеспечивает высокую скорость изменения данных, а также предназначена для обслуживания большого потока запросов, то в ней можно использовать одну физическую базу данных. Обычно только в относительно небольших системах можно одновременно удовлетворить противоположным требованиям прикладных систем двух вышеназванных типов.

Намного более практичным является создание двух физических баз данных, каждая из которых настраивается по-своему и даже может иметь свои структуры данных.

Поскольку для удовлетворения большинства запросов часто бывает достаточно иметь не самые последние данные, база данных, ориентированая на обслуживание запросов, может обновляться не синхронно с внесением изменений, а позже.

Пакет ACMS фирмы DEC предлагает изощренный механизм для реализации такого подхода, который может быть использован в СУБД Rdb этой фирмы. Изменения, поступающие в первичную базу данных, могут запоминаться в порядке очередности их поступления, а затем автоматически, в соответствии с определенным регламентом вноситься в базу данных, предназначенную для выполнения запросов. К сожалению, необходимость подобных функциональных возможностей СУБД часто недооценивается, и тонкие механизмы настройки, предлагаемые некоторыми поставщиками программных средств для баз данных, часто не получают должного признания.

Заключение

Проблемы, связанные с повышением производительности баз данных, имеют более общий характер, чем это многим представляется. Неискушенные пользователи баз данных часто ошибочно полагают, что программные средства для работы с ними и так столь совершенны, что о производительности заботиться не нужно. Это, возможно, и справедливо для малых прикладных систем, ориентированных на сравнительно небольшое число пользователей. Однако в случае больших систем выбор программных средств становится критичным. При больших объемах данных и высоких интенсивностях транзакций всегда требуется провести дополнительную работу, чтобы обеспечить адекватные рабочие характеристики СУБД. При этом следует помнить о возможности появления "узких мест", влияющих на производительность базы данных, и принять соответствующие меры.

факты

☐ Фирма WordStar International выпустила набор программ Laptop Collection, предназначенный для повышения производительности портативных *(laptop)* компьютеров. В этот набор входят следующие программы: WordStar Laptop Edition (для редактирования текстов), LapLink Special (для передачи файлов между портативными и настольными компьютерами) фирмы Edition Traveling Software, OnTime (для делопроизводства) фирмы Campbell Services. Для работы Laptop Collection требуется 640 Кбайт оперативной памяти, два дисковода для дисков емкостью 720 Кбайт или один такой дисковод и жесткий диск.

Стандарт языка С
Отчет о текущем состоянии

В каком направлении продолжается развитие языка?

Р. Яшке

Достигнута важная веха в индустрии программирования на языке С: появился первый стандарт. Многие разработчики компиляторов интенсивно работают над тем, чтобы их компиляторы соответствовали этому стандарту, а некоторые уже добились этого. Можно надеяться, что скоро мы получим систему для формальной проверки соответствия стандарту языка. Индустрия программирования на языке С достигла зрелости. В связи с этим поговаривают, что она начинает гибнуть или по меньшей мере загнивать. Действительно, коль скоро все элементы языка могут быть подведены под формальное определение, то уже не остается возможностей для той свободы, благодаря которой язык С вознесся до таких высот. Да и компьютерная пресса в известной мере потеряла интерес к этому (могу поклясться!) общепризнанному языку и вместо него пестует нового любимца С++.

Если Вы хотите идти в ногу с модой, Бог с Вами! Если же мне потребуется выполнить реальную разработку, то я постараюсь обойтись без С++. Со стандартизацией приходит стабильность. И хотя изменения способствуют самому нашему существованию, в еще большей мере оно зависит от стабильности. В большинстве случаев мы можем обойтись без футуристических выкрутасов. Теперь, когда ситуация с нашим любимым языком программирования прояснилась, мы можем приступить к той работе, для выполнения которой нас, собственно, и нанимали: обеспечивать получение полезных результатов, опираясь на реальных людей и ограниченный бюджет. Если нам требуется создать мобильный программный код, то теперь мы имеем гораздо больше шансов на успех, чем когда бы то ни было.

Именно сейчас полезно оглянуться на проделанный путь, представить, как формировался стандарт языка С, и подумать, в каких направлениях можно и нужно продолжить развитие этого языка.

ANSI-стандарт языка С

Первый Стандарт языка С (известный как Стандарт X3.159-1989) принят Американским национальным институтом стандартов (ANSI - American National Standard Institute) в декабре 1989 г. Этот Стандарт разработан Комитетом X3J11 под наблюдением Секретариата X3, которым, в свою очередь, руководила Ассоциация изготовителей компьютерной и организационной техники (CBEMA - Computer and Business Equipment Manufacturer's Association) со штаб-квартирой в Вашингтоне (федеральный округ Колумбия). Строго говоря, X3J11 не является комитетом ANSI, однако обычно его называют Комитетом ANSI С.

© Dr. Dobb's Journal, August 1991.
All rights reserved. M&T Publishing, Redwood City, CA, USA.

Истоки формирования Комитета X3J11 относятся к 1983 г., когда фирма Motorola наняла Джима Броуди (Jim Brodie) для разработки компилятора языка С. Попытавшись найти полное описание языка, препроцессора и библиотеки, он пришел к выводу, что такового не существует. К тому времени имелись лишь книга Б.Кернигана и Д.Ритчи "Язык программирования Си" (часто просто называемая K&R), различные статьи, опубликованные как внутри, так и вне фирмы Bell Labs корпорации AT&T, а также множество компиляторов, большинство из которых либо вели происхождение от мобильного компилятора языка С pcc (portable C compiler), распространявшегося в составе операционной системы Unix, либо эмулировали его. Если разработчик компилятора хотел выяснить, как должен себя вести компилятор языка С в той или иной ситуации, ответ нередко звучал следующим образом: "Посмотрите, что в этом случае сделал бы компилятор pcc".

Установив, что полного описания языка не существует, Броуди доложил об этом своему боссу. Поскольку последний когда-то занимался стандартизацией языков программирования, он посоветовал Броуди самому заняться разработкой стандарта, коль скоро такового не существует. Броуди внял совету, решив, что для завершения описания потребуется свести не такое уж большое число концов с концами и что это не займет много времени. (На самом деле для этого потребовалось семь лет.)

Так Броуди стал ответственным секретарем Комитета ANSI. Его первая большая работа состояла в написании проекта, в котором он сформулировал цели и задачи подобного комитета, а также провел оценку необходимых ресурсов и указал их возможные источники. Этот проект был передан на рассмотрение в Комиссию SPARC (ANSI любит аббревиатуры). После одобрения проекта был официально сформирован новый комитет, получивший название X3J11 (X3 - подгруппа ANSI, к которой относятся компьютерные стандарты; J - категория в подгруппе X3, к которой относятся языки программирования). К тому времени уже работало десять других комитетов: например, комитеты X3J3 (стандартизация Фортрана) и X3J16 (стандартизация С++.)

Первое официальное заседание Комитета состоялось под председательством его прародителя Броуди. На этом заседании были выдвинуты кандидаты в руководители Комитета и проведены выборы. Первыми руководителями Комитета стали: Броуди - председатель, Том Плам (Tom Plum) - заместитель председателя, П.Дж. Плоджер (P.J. Plauger) - секретарь, и Энди Джонсон (Andy Johnson) - ответственный за словарь языка.

Ларри Розлер (Larry Rosler) вызвался быть редактором проекта стандарта. (Хотя редактор и не входит в число руководителей Комитета, обычно он играет весьма важную

роль в его работе.) Все эти люди продолжают работать в Комитете по мере своих возможностей, за исключением Розлера, которого заменил Дейвид Проссер (David Prosser). Впоследствии в состав руководства Комитета вошли редактор по подготовке обоснований Ранди Хадсон (Randy Hudson) и ответственный за международные связи (сначала Стив Херси (Steve Hersee), затем Плоджер, а теперь я).

Как приобщиться к "Высочайшему двору" разработчиков языка С

У рядовых пользователей обычно складывается впечатление, будто стандарты спускаются им сверху некими божественными авторитетами, которые берут на себя бремя установить, что лучше всего подходит простым смертным. На самом деле в Комитет X3J11 входят вполне обычные люди, многим из которых приходится самим программировать на языке С. Коль скоро они регулярно занимаются программированием, как же им удалось оказаться представленными "Высочайшему двору" разработчиков языка С? Это произошло либо по их собственной инициативе, либо по инициативе их работодателей.

В принципе, любой человек может посещать заседания комитетов ANSI по стандартизации и принимать неформальное участие в их работе, даже если он не является зарегистрированным членом одного из этих комитетов. Если он хочет влиять на их работу, участвуя в голосовании, то к нему предъявляются следующие требования:

Если Вы располагаете временем и средствами, то имеете возможность оказать существенное влияние на стандарт ANSI.

- Он должен быть исправно платящим взносы зарегистрированным членом ассоциации СВЕМА (для этого достаточно заполнить анкету и не забывать каждый год платить 250 дол.). Фирма может зарегистрировать и оплачивать членство любого числа своих служащих, но лишь один из них становится основным делегатом в комитете по стандартизации, а остальные рассматриваются как альтернативные. Только один представитель каждой фирмы может голосовать по конкретному вопросу. Член комитета не обязан ни быть гражданином США, ни даже постоянно проживать в США. (В работе Комитета X3J11 участвует большое число иностранных граждан, не проживающих постоянно в США.)
- Чтобы извлекать пользу из обсуждений, проводимых на заседаниях Комитета, ему надо их посещать. Почти все эти годы совещания созывались четыре раза в год, обычно где-нибудь в США, и продолжались четыре с половиной дня каждое.
- Чтобы воспользоваться правом на голосование, он должен участвовать, по крайней мере, в двух из каждых трех совещаний и быть внесенным в списки присутствовавших в течение определенного числа дней на каждом совещании, в котором он принял участие. Каждая отдельная фирма получает только один голос. Например, независимый консультант Фред Смит имеет один голос, равно как и фирмы IBM, AT&T и DEC. Естественно, его голос может сыграть решающую роль.

Короче говоря, если у Вас есть лишние 250 дол. и Вы располагаете временем и средствами, чтобы посещать заседания, а в промежутках между ними готовить и читать статьи, то Вы имеете возможность оказать существенное влияние на стандарт ANSI. Впрочем, определенное влияние можно оказать, и не посещая заседаний, а получив статус наблюдателя и общаясь с Комитетом по обычной или электронной почте.

В Комитете X3J11 преобладают разработчики средств трансляции языка С. (Отсюда вовсе не следует, что созданный ими стандарт в чем-либо неадекватен или необъективен. Напротив, я считаю, что он отличается весьма высоким качеством.) Конечно, каждая фирма-изготовитель неявным образом представляет пользователей выпускаемых ею инструментальных средств, но то, что изготовителю представляется лучшим для соблюдения собственных интересов, может и не отражать пожеланий или нужд пользователей. По мере дальнейшей эволюции языка С будет интересно посмотреть, изменится ли состав Комитета X3J11. Я подозреваю, что это произойдет, но процесс изменения состава будет медленным, и лишь немногие конечные пользователи ощущают необходимость принимать участие в разработке стандарта на этой стадии или могут позволить себе тратить на это время и деньги.

Важно понимать, что членство в Комитете X3J11 - сугубо добровольное, и его членам приходится самим нести все расходы, связанные с участием в его работе. Ни ANSI, ни Секретариат X3 не предусматривают никакой финансовой поддержки. Для проведения каждого совещания приходится отыскивать милосердного устроителя. (До недавнего времени устроителям приходилось копировать и рассылать до и после каждого совещания около 200 комплектов различных материалов, содержащих по нескольку сотен страниц.) Доходы ANSI образуются лишь из взносов членов Комитета и от продажи документов с описанием стандартов.

Цели Комитета

Преследуемые Комитетом X3J11 цели лучше всего изложены в документе "Обоснования" (Rationale), который был выпущен в дополнение к Стандарту языка С.

Конечной целью работы Комитета была разработка четкого, непротиворечивого и недвусмысленного Стандарта языка С, в котором было бы обобщено существующее общераспространенное определение языка С и который бы способствовал переносимости программ пользователей из одного операционного окружения языка С в другое.

Устав Комитета X3J11 четко ориентирует его членов на систематизацию существующих практических результатов. Комитет часто ориентировался на прецедент, если последний был четким и однозначным. Задаваемые Стандартом определения большинства элементов языка в точности совпадают с

Рекс Яшке (Rex Jaeschke) является редактором журнала "The Journal of C Language Translation", выходящего четыре раза в год и рассчитанного на разработчиков средств компиляции языка С. Он также входит в состав Комитета X3J11 ANSI, представляет США в Группе по стандартизации языка С ISO и является ответственным секретарем Группы, которая занимается расширениями языка С, относящимися к численным методам.
Его адрес: 2051 Swans Neck Way, Reston, VA 22091.
Tel.: 703-860-0091 (USA);
с ним можно также связаться через сеть Internet по коду rex@aussie.com.

приведенными в Приложении А книги K&R и реализованными почти во всех компиляторах языка С.

Книга K&R - не единственный источник, из которого черпались сведения о существующей практике. В течение ряда лет много сил было положено на то, чтобы улучшить язык С, устранив имевшиеся в нем недостатки. Комитет формализовал оправдавшие себя расширения языка, которые стали составной частью различных диалектов языка С.

Однако для различных диалектов языка С многие проблемы решались различными, иногда диаметрально противоположными путями. Эти расхождения обусловлены несколькими причинами. Во-первых, книга K&R, которая служила спецификацией языка С почти для всех его компиляторов, не всегда обеспечивает достаточную точность (тем самым допуская различные интерпретации), и в ней не отражены некоторые моменты (например, полная спецификация библиотеки), которые важны для обеспечения мобильности программного кода. Во-вторых, за многие годы, в течение которых язык продолжал развиваться, для устранения ограничений и недостатков в его диалекты добавлялись различные несогласующиеся между собой расширения.

В задачи Комитета входили рассмотрение таких областей несогласованности реализаций и разработка свода четких непротиворечивых правил, согласованных с остальной частью языка. С этой целью Комитет рассматривал расширения, сделанные в различных диалектах языка С, составлял спецификацию полного набора требуемых библиотечных функций и разрабатывал полный, корректный синтаксис языка С.

Работа Комитета состояла в основном в достижении компромиссов. Комитет пытался повысить мобильность языка С, оставляя определения некоторых его свойств машинно-зависимыми. Он пытался реализовывать в языке новые полезные идеи, не нарушая его основной структуры. Он пытался разработать четкий и непротиворечивый язык, не нарушая действия существующих программ. Все задачи были важными, и в попытках достижения жизнеспособного компромисса каждое решение взвешивалось в свете порою противоречивых требований.

При составлении спецификации Стандарта языка С Комитет придерживался нескольких основополагающих принципов, наиболее важные из которых перечислены ниже:

- ☐ Важен данный программный код, а не данная реализация языка.
- ☐ Программный код на языке С может быть мобильным.
- ☐ Программный код на языке С может не быть мобильным.
- ☐ Необходимо избегать употребления конструкций языка не по их прямому назначению.
- ☐ Стандарт представляет собой договор между разработчиком компилятора и программистом.
- ☐ Необходимо придерживаться духа языка С. У последнего есть много аспектов, но наиболее существенным является общественное мнение об исходных принципах, положенных в основу языка. Некоторые аспекты, входящие в понятие духа языка С, могут быть выражены такими фразами:

- Доверяй программисту.
- Не пытайся удержать программиста от того, что должно быть им сделано.
- Сохраняй язык компактным и простым.
- Предусматривай только одну возможность выполнения операции.

Одной из задач Комитета было не лишать компиляторы возможности генерировать компактный эффективный объектный код. В некоторых случаях Комитет улучшал эффективность генерируемого кода (например, операции над числами с плавающей точкой могут выполняться не с двойной, а с одинарной точностью, если оба операнда имеют одинарную точность).

Вехи

Поскольку многое в деятельности комитетов по стандартизации обычно скрыто от широкой публики, Комитету X3J11 в процессе работы было важно время от времени узнавать "общественное мнение". С этой целью был объявлен период получения неофициальных рецензий, в течение которого рабочие черновики Стандарта приобретались по каналам ассоциации CBEMA. После этого Комитет потратил немало времени на анализ многих критических замечаний и конструктивных предложений.

Следующим шагом стал период получения официальных рецензий. (Начиная с этого момента для внесения изменений в проект стандарта требовалось большинство в две трети голосов членов Комитета X3J11, в отличие от простого большинства на предыдущих этапах.) На этот раз черновик Стандарта подвергался рецензированию в течение четырех месяцев. На все комментарии необходимо было давать ответ в соответствии с детальным сводом правил. В каждой ситуации, когда Комитет X3J11 принимал решение отклонить предложение, необходимо было дать обоснование, а неудовлетворенная ответом публика имела право на апелляцию. После рассмотрения всех обращений к Комитету был вновь объявлен период получения рецензий, но на этот раз всего на два месяца. После трех периодов рецензирования черновик Стандарта приводился в завершенное состояние. Этот процесс продолжался до тех пор, пока Комитет X3J11 не добился получения Стандарта, приемлемого для большинства (но не обязательно для всех) членов Комитета, обладающих правом голоса, и ANSI.

К концу 1988 г., после утряски всех вопросов Стандарт был завершен. Однако спустя некоторое время в ANSI обнаружилось письмо, которое было непреднамеренно пропущено. Несмотря на то, что последний проект Стандарта уже включал в себя около половины содержавшихся в письме предложений, автор письма хотел получить свой кусок пирога. После обсуждения этих предложений (во многих из них требовалась поддержка режима реального времени) Комитет X3J11 решил не заниматься реализацией оставшихся предложений. Неудовлетворенный этим вердиктом автор письма подал апелляцию, в результате которой было однозначно решено не вносить дополнительных технических изменений в проект Стандарта. К концу 1989 г., после годичной задержки ANSI-стандарт языка С был окончательно принят. Казалось бы, эпопея пришла к финалу. На самом деле завершился лишь один оборот этой бесконечной карусели. Комитет X3J11 должен не только разбираться с запросами публики, но и думать о пересмотре этого Стандарта.

Этап истолкования

По принятым в ANSI правилам, комитеты по стандартизации должны руководить истолкованием выработанных ими стандартов. В настоящее время, равно как и в обозримом будущем это является основным занятием Комитета X3J11. Поскольку истолкование стандарта требует гораздо меньше времени и сил, чем его начальная разработка, то Комитет X3J11 перешел на работу по новому расписанию, согласно которому заседания созываются на два дня дважды в год. Однако по мере роста объема работы продолжительность заседаний была увеличена до двух с половиной дней.

Ниже приводятся примеры запросов на истолкование, рассмотренных Комитетом на этом этапе:

1. Если заданы макроопределения #define lp (и #define fm(a) a, то как будет расширен оператор fm lp "abc")? (Ответ Комитета X3J11: fm("abc").)

2. Что будет выведено на печать в результате вызова printf("#.4o",345)? Будет ли это 0531 или 00531? (Ответ Комитета X3J11: 0531.)

3. Возвращают ли функции значения структур путем их копирования? (В конце концов Комитет X3J11 решил, что результат исполнения функции должен быть таким, как если бы копирование значений имело место.)

Развитие Стандарта языка С будет происходить на международном уровне.

С большинством запросов удавалось расправиться довольно легко, но были и такие, для разрешения которых требовалось детальное прочтение многих разделов Стандарта. В этих случаях формальный ответ мог быть довольно объемистым и содержать многочисленные цитаты из Стандарта. Некоторые запросы оказались довольно поучительными для членов Комитета X3J11, считавших, что в Стандарте сказано нечто совсем другое, или не соглашавшихся с окончательным выводом.

Данные Комитетом X3J11 истолкования не имеют силы стандарта и спустя некоторое время могут быть опровергнуты самим Комитетом или ANSI. Ожидается, однако, что почти все истолкования будут учтены в будущих версиях Стандарта. Комитет X3J11 собирается публиковать технический бюллетень, содержащий все запросы, рассмотренные к моменту выпуска этого бюллетеня.

ISO-стандарт языка С

В 1986 г. Международная организация по стандартизации (ISO - International Standard Organization) создала Комитет по стандартизации языка С, получивший название SC22/WG14. В это время работа над проектом ANSI-стандарта находилась на стадии завершения. Однако из-за запросов американских поставщиков средств компиляции, желавших продавать их в неанглоязычные страны, а также запросов покупателей и поставщиков из этих стран, значительные усилия были затрачены на развитие проекта ANSI-стандарта в двух различных направлениях: первое было ориентировано на улучшение возможностей использования однобайтовых западно-европейских символов, в начертание которых входят диакритические знаки (например, немецкое ä и испанское ñ), а второе обеспечивало возможность работы с многобайтовыми символами азиатских алфавитов.

Хотя из-за этого ANSI-стандарт был завершен на год позже, наличие указанных выше возможностей позволило принять его без технических изменений в качестве ISO-стандарта. Таким образом, по содержанию ISO-стандарт и ANSI-стандарт оказались эквивалентными. (ISO-стандарт получил название ISO/IEC 9899:1990 (E).)

В то время как Комитет X3J11 занимается истолкованием стандарта, Комитет WG14 ведет новые работы на нескольких фронтах и занимается составлением нормативного приложения:

- ☐ Делегация Великобритании работает над пояснительным документом, освещающим многочисленные "темные углы" Стандарта. Результатом работы будут разъяснения, а не существенные изменения Стандарта.
- ☐ Японская делегация определяет многочисленные изменения в бибилотеках, обеспечивающие лучшие возможности работы с многобайтовыми символами. Они затронут спецификации заголовков, функций, определений типа и макроопределений.
- ☐ Датская делегация работает над более удобочитаемой альтернативой для триграфа, предложенного Стандартом. (Триграф представляет собой трехсимвольную последовательность вида ??x, предназначенную для записи девяти символов (среди которых {, }, | и \), отсутствующих в большинстве систем, где используется набор символов ISO-646.) Это направление вызвало наибольшее количество дискуссий.

С момента публикации нормативное приложение будет иметь силу стандарта.

В настоящее время заседания Комитета WG14 созываются два раза в год и продолжаются два или три дня. В ноябре прошлого года заседание проводилось в Копенгагене, в мае нынешнего оно состоялось в Токио, а в декабре должно пройти в Милане. Обычно в заседаниях участвуют делегации в составе от одного до четырех человек от каждой из четырехпяти стран. Формальное голосование не проводится, и решения принимаются при условии единогласия. Конечно, на уровне ISO работа движется медленнее, поскольку делегаты должны получать информацию от национальных институтов стандартизации, и временной разрыв между действиями Комитетов X3J11 и WG14 может повлиять на возможность США быстро реагировать на запросы, поступающие в ISO. К счастью, значительная часть дебатов происходит по электронной почте.

Работа над расширениями языка

Подспудно предпринимаются многочисленные попытки расширения языка С, его окружения или определения его библиотек. Некоторые из расширений инициируются группами специалистов в рамках работы институтов стандартизации, в то время как другие представляют собой просто-напросто дополнения к коммерчески доступным компиляторам. Ниже приводятся сведения о нескольких достаточно заметных группах, ведующих подобную деятельность.

Группа по численным расширениям языка С (NCEG - Numerical C Extensions Group). Она была организована мною в марте 1989 г. как тематическая рабочая группа. Поскольку было ясно, что в ANSI-стандарте языка С не предполагается рассматривать вопросы, связанные с программированием численных методов, я решил попробовать скоординировать эту работу. Правда, в мои намерения не входило заменить Фортран на язык С с численными расширениями.

На учредительном заседании группы NCEG в качестве наиболее существенных были выделены следующие направления:

- возможность использования синонимов;
- векторизация и синтаксис массивов;
- комплексная арифметика;
- массивы переменной размерности;
- поддержка IEEE-стандарта представления чисел с плавающей точкой;
- исключения и функция errno;
- организация параллельной обработки;
- агрегированное присваивание начальных значений.

Чуть позже направление параллельной обработки было исключено, поскольку оно оказалось составной частью проекта, предложенного другим комитетом ANSI (X3H5). Над остальными направлениями работа продолжалась, а недавно было добавлено новое направление, связанное с расширением типов данных, позволяющим обрабатывать большие целые числа. Рассматриваются также и другие расширения синтаксиса присваивания начальных значений (инициализации).

С середины 1989 г. группа NCEG собирается регулярно. Обычно ее заседания проходят в течение двух дней до или после заседания Комитета X3J11. В марте 1991 г. Комиссия SPARC окончательно приняла проект, предложенный группой NCEG. С тех пор она стала рабочей группой под крышей ANSI и получила название X3J11.1.

Миссия группы NCEG состоит в составлении технического отчета, а не в разработке расширения Стандарта. Широкой публике проекты отдельных частей этого отчета будут представлены скорее всего в начале 1992 г. Первоочередная задача группы NCEG состоит в определении подходов к проблемам, связанных с численными методами. Кроме того, группа должна способствовать принятию этих подходов разработчиками средств компиляции. Тем самым может быть заложена хорошая основа для будущего пересмотра Стандарта языка С.

Другие группы. В настоящее время Комитет X3H5 занят определением модели параллельной обработки в языках Фортран и С. Первый проект ее привязки к языку С появился в декабре 1990 г.; в настоящее время он пересматривается с учетом полученных многочисленных рецензий.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ ПО ЯЗЫКУ С

Чтобы получить дополнительную информацию по вопросам, затронутым в настоящей статье, можно обратиться к указанным ниже лицам или организациям.

ANSI-стандарт языка С имеет официальное название ANSI X3.159-1989. Плата за один экземпляр Стандарта - 65 дол. Учтите, что он доступен только в печатном виде. Чтобы получить экземпляр Стандарта, обратитесь по адресу:

American National Standards Institute
Sales Department
11 West 42nd Street
New York, NY 10036
Tel: 212-642-4900; Fax: 212-398-0023
Sales Fax: 212-302-1286

Ответственный секретарь Комитета X3J11 (ANSI C)
Jim Brodie
Motorola Inc.
Tempe, AZ 85284
Tel.: 602-897-4390
Internet: brodie@ssdt-tempe.sps.mot.com

Если Вы хотите послать официальный запрос на истолкование Стандарта языка С, обратитесь по адресу:
Manager of Standard's Processing
X3 Secretariat, CBEMA
311 First Street N.W., Suite 500
Washington, DC 20001-2178
Tel.: 202-737-8888; Fax: 202-638-4922

Ответственный секретарь Подгруппы X3J11.1 (NCEG)
Rex Jaeschke
Journal of C Language Translation
2051 Swans Neck Way
Reston, VA 22091
Tel.: 703-860-0091
Internet: rex@aussie.com

Председатель Комитета X3J16 (ANSI C++)
Dmitry Lenkov
Hewlett-Packard Company
19447 Pruneridge Avenue, MS 47LE
Cupertino, CA 95014
Tel.: 408-447-5279
Internet: dmitry%hpda@hplabs.hp.com

ISO-стандарт языка С имеет официальное название ISO/IEC 9899:1990 (E). Его экземпляр можно получить в США через ANSI, а в других странах - через официально уполномоченные национальные институты стандартизации.

Рабочая группа WG14 (представитель США)
Rex Jaeschke (его адрес и телефоны см. выше)

Заместитель председателя Комитета X3H5
(параллельная обработка данных)
Walter G. Rudd
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3902
Tel.: 503-737-5553; Fax: 503-737-3014
Internet: rudd@cs.orst.edu

По вопросам верификации в США обращайтесь по адресу:
National Institute of Science and Technology
Software Standards Validation Group
Gaithersburg, MD 20899
Tel.: 301-975-3247

Одобренный NIST комплект тестов на соответствие Стандарту языка С:
Perennial
4699 Old Ironsides Drive, Suite 210
Santa Clara, CA 95054
Tel.: 408-727-2255; Fax: 408-748-2909
Internet: uunet!peren!acvs

BSI/European Validation Service
British Standards Institution Quality Assurance (BSI QA)
PO Box 375, Milton Keynes, MK14 6LL
United Kingdom
Tel.: +44-0908-220908; Fax: +44-0908-226071
Internet: neil@bsiqa.uucp

Одобренный BSI комплект тестов на соответствие Стандарту языка С:
Plum Hall Inc.
1 Spruce Avenue
Cardiff, NJ 08232
Tel.: 609-977-3770; Fax: 609-653-1903
Internet: plum@plumhall.com

Различные комитеты ISO работают или уже завершили работу над расширениями языка C для таких областей, как двумерная и трехмерная графика в стандарте GKS.

Фирма AT&T уже продолжительное время финансирует проект Concurrent C по параллельной обработке данных, а фирма Thinking Machines уже завершила работу над языком C*. Существуют также язык C++ и язык Objective C, разработанный фирмой Stepstone. Компилятор GNU C фирмы Free Software Foundation уже играет существенную роль в применении языка C для программирования численных методов. Еще один любопытный проект разрабатывается первым автором компилятора Turbo C. Создаваемый им язык получил предварительное название OPAL и имеет глубокие корни в языке C. Находятся в стадии разработки многие другие проекты, в том числе C-cured - версия языка C, которая, по замыслу авторов, должна обеспечивать более рациональный способ определения типов данных.

Совместимость языков C и C++

Когда в Комитет X3J11 поступило предложение заняться разработкой стандарта для языка C++, Комитет его отклонил. Это было вызвано не только тем, что подобная задача не входила в устав Комитета и его члены уже потратили более шести лет на разработку Стандарта языка C, - в настоящее время Комитет слишком загружен работой по его истолкованию. Аргументов для отказа было предостаточно, например утверждалось, что C++ - совершенно иной язык, несмотря на его глубокие корни в языке C. В конце концов для работы над Стандартом языка C++ был сформирован новый комитет (X3J16), а спустя год или чуть больше была создана Рабочая группа ISO C++.

В свое время немало разговоров ходило вокруг известного заявления Комитета X3J16 о том, что язык C++ "должен быть как можно ближе к Стандарту ANSI языка C, но не более того". В любом случае в Комитете X3J16 преобладает желание по возможности обеспечить языку C++ совместимость вверх с языком C и разойтись с последним только в тех местах, где это абсолютно необходимо. Вопросами подобной совместимости занимается Подгруппа по совместимости с языком C, входящая в состав Комитета X3J16. Ниже указаны некоторые темы, над которыми работает эта Подгруппа:

- ☐ Смешанные программы, написанные как на C, так и на C++.
- ☐ Корректная обработка компиляторами языка C++ кода, написанного на стандартном языке C.
- ☐ Код на языке C++, в котором есть вызовы библиотечных процедур стандартного языка C, и наоборот.

Верификация компиляторов

Любой продавец средств компиляции может заявить о том, что его компилятор соответствует Стандарту языка C, - многие уже так и поступили. Однако к моменту написания настоящей статьи (май 1991 г.) в США еще не была налажена служба верификации средств компиляции для проверки их соответствия Стандарту. Однако Национальный институт науки и техники (NIST - National Institute of Science and Technology, прежде National Bureau of Standards - Национальным бюро стандартов) уже взял за основу комплект тестов, предложенный фирмой Perennial. Этот комплект несколько месяцев назад был представлен на техническое рецензирование и в настоящее время пересматривается в соответствии с требованиями NIST. Необходимо отметить, что NIST будет осуществлять проверку на соответствие FIPS-стандарту языка C (FIPS - Federal Information Processing Standard). В настоящее время этот стандарт представляет собой ANSI-стандарт языка C с несколькими дополнительными требованиями.

Британским институтом стандартов (BSI - British Standards Institute) и несколькими другими европейскими институтами стандартизации взят на вооружение комплект тестов, предложенный другим продавцом средств компиляции, фирмой Plum Hall. BSI уже проводит верификацию средств компиляции, и некоторые компиляторы уже получили сертификаты о соответствии Стандарту. Этот институт выполняет проверку на соответствие Стандарту ANSI/ISO языка C.

Следует отметить, что для верифицируемой реализации средств компиляции должно быть указано, каким именно образом она ведет себя в тех ситуациях, которые зависят от реализации. Однако документация не может быть автоматически проверена с помощью комплекта тестов, поэтому пока неясно, каким именно образом это затруднение будет устраняться.

Будущее Стандарта языка C

Как уже говорилось, по содержанию Стандарты ISO и ANSI в настоящее время эквивалентны, поэтому Комитет X3J11 и Группа WG14 поощряют всех, кто вместо ANSI- или ISO-стандарта говорит просто о Стандарте языка C (Standard C). На этот счет уже проведены переговоры с большим числом всемирных и многонациональных предприятий. С появлением электронной почты, действующей в режиме реального времени, мир "стал тесен". Национальные границы легко пересекаются и во многих отношениях перестали быть препятствием, поэтому стало выгоднее работать сообща над международными стандартами, чем изолированно над национальными.

Все это означает, что развитие Стандарта языка C будет происходить на международном уровне. Национальные группы, например Комитет X3J11, будут играть достаточно важные роли и, возможно, будут выполнять разработку по согласованию с Группой WG14. Однако центр тяжести переместится на разработку международных стандартов. В связи с этим Комитет X3J11 в настоящее время завершает письменный опрос своих членов по вопросу об аннулировании Стандарта X3.159-1989 и о замене его на Стандарт ISO/IEC 9899:1990 (E). Таким образом, Комитет планирует сделать ANSI-стандарт языка C тождественным ISO-стандарту не только по содержанию, но и по форме. Это подготовит почву для того, чтобы Комитет X3J11 мог официально отражать в своих документах работу Группы WG14: по мере пересмотра ISO-стандарта языка C ANSI-стандарт может пересматриваться таким же образом.

Технический отчет, который выпустит группа NCEG, вероятнее всего, окажет определенное влияние на будущие направления развития языка C, поскольку многие члены Комитета X3J11 активно работают и в Подгруппе X3J11.1.

Что касается языка C++, то я не считаю его логическим преемником языка C. Для определенных классов программ язык C++ может и в самом деле оказаться более подходящим. Однако язык C все еще имеет собственную достаточно значительную нишу, и поскольку ведется интенсивная работа над обеспечением возможности использования многобайтовых и международных наборов символов в языке C, я считаю, что область приложений языка C расширится.

T9000
Транспьютерная революция продолжается!

М. А. Маркин, В. А. Лопатин

Оригинальная архитектура транспьютера позволяет эффективно использовать его для реализации одной из моделей параллельных вычислений - модели взаимодействующих последовательных процессов (ВПП). В этой модели параллельная программа рассматривается как совокупность обычных последовательных программ, выполняемых параллельно на нескольких процессорах, обменивающихся сообщениями по синхронным информационным каналам [Pountain D. Virtual Channels: The Next Generation of Transputers//BYTE. - 1990. - N 4. - P. E&W3-E&W12].

К сожалению, аппаратные и программные средства современных транспьютеров позволяют реализовать модель ВПП только частично. Число процессов и информационных каналов, которые может поддерживать программа, выполняемая на одном транспьютере и имитирующая параллелизм благодаря использованию режима разделения времени, зависит от емкости имеющейся памяти, т.е. в большинстве случаев это число практически не ограничено.

Решив распределить процессы между несколькими транспьютерами для достижения реального параллелизма, Вы обнаружите ряд ограничений, связанных с тем, что каждый транспьютер имеет только четыре физических канала (линка). Это обстоятельство существенно ограничивает возможности написания параллельных программ и приводит также к потере некоторых преимуществ модели ВПП.

Разработчики современных параллельных суперкомпьютеров учли эти недостатки архитектуры транспьютеров, выбрав в качестве основной топологии современных мультипроцессорных систем топологию N-мерного гиперкуба, потому что в этом случае удается минимизировать время обмена сообщениями между процессорами. Но даже для четырехмерного гиперкуба требуется шесть каналов связи на один процессор. Для гиперкубов большей размерности требуется еще большее число каналов на один процессор.

До настоящего времени реализация топологии N-мерного гиперкуба в транспьютерных системах требовала использования в каждой вершине нескольких транспьютеров только для того, чтобы получить необходимое при такой топологии число каналов, связывающих эти вершины.

Главный козырь фирмы INMOS

Аппаратные средства транспьютера нового поколения - T9000, о планах выпуска которого в IV квартале 1991 г. фирма INMOS (Великобритания) официально объявила на международной конференции "Transputing'91", проходившей в апреле в Саннивейле (США), будут усовершенствованы для достижения более высокой скорости вычислений и поддержки более сложных операционных систем, чем это обеспечивают существующие транспьютеры. Следует особо подчеркнуть, что транспьютер T9000 имеет средства, позволяющие решить проблему недостаточности числа каналов связи в существующих транспьютерах. С этой целью фирма INMOS могла бы, конечно, добавить еще несколько линков в новом транспьютере, но их общее число все же должно оставаться достаточно небольшим из-за ограничений в технологии изготовления СБИС. Вместо этого фирма приняла радикальное решение: добавить в транспьютере T9000 аппаратные средства мультиплексирования, чтобы его линки могли одновременно использоваться сразу несколькими процессами. Фирмой INMOS разработана также СБИС маршрутизации C104, которая должна обеспечивать эффективное информационное взаимодействие между удаленными транспьютерами благодаря использованию протоколов, аналогичных применяемым в системах связи с коммутацией пакетов.

Каналы межтранспьютерной связи при использовании транспьютера T9000 становятся виртуальными; это позволяет реализовать столько программных каналов, сколько требуется Вашей программе. Виртуальные каналы могут устанавливаться даже между теми транспьютерами, которые не имеют прямых связей через физические каналы; при этом требуемая эффективность передачи сообщений в указанный адрес обеспечивается с помощью СБИС мар-

Виктор Алексеевич Лопатин - член Правления Советской транспьютерной ассоциации, член Совета директоров А/О "Учебный центр параллельных систем и технологий". Область интересов - вычислительные системы с массовым параллелизмом, распределенные операционные системы. Контактный телефон 499-15-00.

Михаил Александрович Маркин - член Правления А/О "Учебный центр параллельных систем и технологий". Область интересов - вычислительные системы с массовым параллелизмом, гетерогенные вычислительные системы, локальные вычислительные сети. Контактный телефон 499-15-00.

шрутизации. Для существующих транспьютеров возможности организации взаимодействия процессов через линки ограничены наличием лишь четырех линков. Так как для каждого программного канала в транспьютере назначается физический канал, то при необходимости доставки сообщения в удаленный транспьютер приходится делать несколько таких назначений. При этом программа становится зависимой от топологии транспьютерной системы.

При использовании транспьютера T9000 для обмена сообщениями между двумя любыми процессами можно использовать программный виртуальный канал, а выбор конкретного физического маршрута прохождения сообщений обеспечивают аппаратные средства транспьютера и СБИС маршрутизации. При этом разрабатываемая программа становится независимой от топологии сети транспьютеров и может переноситься на другие многотранспьютерные системы практически без изменений. Транспьютер T9000 имеет четыре физических канала, но он также содержит коммуникационный контроллер, обеспечивающий установление множества виртуальных каналов благодаря разбиению сообщений на пакеты и передаче пакетов от нескольких сообщений по одному физическому каналу.

Линки транспьютера T9000 по быстродействию (100 Мбит/с в дуплексном режиме) в пять раз превосходят линки современных транспьютеров. Кроме того, система с коммутацией пакетов обеспечивает более эффективное использование физических каналов, так что скорость передачи сообщений увеличивается несмотря на использование режима разделения физического канала.

Реализация обмена сообщениями

В транспьютере T9000 сообщение произвольной длины разбивается на последовательность пакетов длиной 32 байта каждый совершенно иначе, чем во всех других транспьютерах, которые передают сообщения по каналу байт за байтом. Для смешивания пакетов, принадлежащих разным сообщениям (т.е. разным виртуальным каналам), необходимо, чтобы каждый пакет содержал заголовок, предназначенный для идентификации виртуального канала, по которому этот пакет передается. Последний символ в каждом пакете специальный - Конец Пакета (End-Of-Packet), за исключением последнего пакета передаваемого сообщения, где это символ Конец Сообщения (End-Of-Message).

Пакет имеет следующий формат:

<Заголовок> <0...32 байта данных> <EOP/EOM>

Для обеспечения синхронизации связи процесс, принимающий сообщения (приемник), должен подтверждать получение каждого пакета; посылаемый приемником пакет подтверждения - это обычный пакет, не содержащий данных. Каждый виртуальный канал является дуплексным, т.е. содержит два подканала, по одному из которых передаются пакеты данных, а по другому принимаются пакеты подтверждений. Передающий процесс не может продолжить свою работу до тех пор, пока не получит подтверждения приема последнего переданного пакета данных. Это подтверждение посылается сразу после получения первого байта пакета, что позволяет обеспечить непрерывную передачу сообщений, если приемник готов к приему.

Запросы на передачу ставятся в очередь на каждом физическом канале, так что центральный процессор не должен находиться в состоянии ожидания во время передачи пакета. В случае, если принимающий процесс не готов к приему пакета, транспьютер T9000 обеспечивает буферизацию одного пакета для каждого из виртуальных каналов. По той же причине современные транспьютеры осуществляют буферизацию одного байта принимаемого сообщения во внутреннем регистре. В транспьютере T9000 буферизация осуществляется в памяти, а не в регистре, так что при наличии достаточного объема памяти в одном транспьютере можно реализовать любое число виртуальных каналов.

СБИС маршрутизации C104

Для эффективного использования виртуальных каналов необходимо использовать СБИС маршрутизации C104. Эта СБИС представляет собой электронный коммутатор с коммутацией пакетов, реализованный на одном кристалле. По своим функциям СБИС C104 близка к учрежденческим АТС: она содержит 32 физических транспьютерных канала и может осуществлять маршрутизацию сообщения от любого из 32 транспьютеров T9000 к любому другому. Следует подчеркнуть, что не обязательно подключать все 32 транспьютера к СБИС C104; можно, например, вместо этого подключить восемь транспьютеров, используя все их линки. Такое подключение не влияет на число виртуальных каналов, которые могут быть скоммутированы, так как каждый физический канал может поддерживать установление любого числа виртуальных каналов. Однако при таком соединении пропускная способность увеличивается в четыре раза. В ряде случаев можно присоединить некоторые из каналов СБИС C104 к другим аналогичным СБИС, для того чтобы образовать более крупную и более сложную коммутируемую многотранспьютерную систему. СБИС C104 содержит полнодоступный коммутатор 32x32, обеспечивающий соединение между любыми двумя каналами, а также сравнительно простую логическую схему для определения пункта назначения каждого принятого сообщения. Разработанная специалистами фирмы INMOS схема маршрутизации позволяет обходиться без использования в СБИС C104 процессора или значительного объема оперативной памяти. Для большинства программистов механизм маршрутизации будет полностью прозрачным, при этом многотранспьютерная система будет представляться программисту таким образом, что можно будет посылать сообщения от любого транспьютера любому другому даже при отсутствии между ними прямых соединений.

Развитие языка Оккам 2

Применение пакетной коммутации позволяет снять ряд ограничений в языке Оккам 2, используемом для программирования транспьютеров. Текущая версия языка Оккам 2 представляет собой статический язык программирования, в котором все используемые программой ресурсы должны быть определены на стадии компиляции; поэтому Оккам-программы во время выполнения не могут порождать новых параллельных процессов на удаленных транспьютерах. Более того, топология программ, распределенных по нескольким транспьютерам, существенно ограничена тем, что каждому процессу доступно только четыре канала. При написании программы программист должен в явном виде устанавливать соответствие каждого программного канала физическому каналу транспьютера и не может менять эти связи без повторной компиляции.

Появление транспьютера T9000 и СБИС маршрутизации C104 позволит реализовать полную версию языка Оккам - Полный Оккам (называемую также Оккам 3). Эта версия будет выглядеть точно так же, как и текущая версия языка, но Вам, например, больше не нужно будет использовать оператор PLACE для установления соответствия между программными и физическими каналами. Можно будет объявлять в программе любое необходимое число каналов, а обеспечение соответствующих соединений возьмет на себя система маршрутизации сообщений, реализованная с помощью СБИС C104. Отпадет необходимость в предварительном размещении процессов по транспьютерам, так как компилятор будет их размещать самостоятельно. Такие программы могут

быть перенесены на любую многотранспьютерную систему с достаточным числом транспьютеров.

Следующим этапом может быть разработка Динамического Полного Оккама. В этой версии языка Оккам ограничения, связанные с размещением процессов на стадии компиляции, будут полностью устранены, появится возможность реализации рекурсивных процедур, динамического назначения процессоров и т.п. Архитектура транспьютера T9000 содержит ряд усовершенствований, делающих разработку такого языка возможной.

Хотя механизм взаимодействия с помощью виртуальных каналов, конечно же, является наиболее интересным свойством нового транспьютера, в транспьютере T9000 имеется ряд других усовершенствований, облегчающих жизнь разработчикам операционных систем, а также улучшающих поддержку таких языков программирования, как Си и Ада. Например, важным усовершенствованием является добавление новых команд, обеспечивающих возможность взаимодействия многих программных каналов с одним физическим каналом. Этим усовершенствованием целесообразно пользоваться в том случае, когда нескольким процессам необходимо предоставить возможность разделения общего программного кода, причем более эффективно, чем это возможно при использовании конструкции ALT в языке Оккам 2.

Особенности архитектуры нового транспьютера

По данным фирмы INMOS, пиковая производительность транспьютера T9000 при внутренней тактовой частоте 50 МГц будет достигать 200 млн. операций/с при выполнении обычных операций и 25 млн. операций/с при выполнении операций над числами с плавающей точкой (25 MFLOPS). Увеличения производительности на порядок по сравнению с производительностью существующих транспьютеров удалось достичь не только благодаря совершенствованию технологии изготовления транспьютера, но и благодаря использованию специальных системных решений, обеспечивающих одновременное выполнение нескольких команд за один цикл работы процессора. Для выполнения команд используется пятиступенчатый конвейер: первая ступень может выбирать одновременно две локальные переменные, вторая - заново вычислять производные адреса переменных, что необходимо для работы с индексированными массивами или нелокальными переменными, третья ступень позволяет выбирать одновременно две нелокальные переменные или два элемента массива, четвертой ступенью является арифметико-логическое устройство (АЛУ) или устройство обработки чисел с плавающей точкой (FPU - floating-point unit), последняя ступень конвейера отвечает за запись результата операции в память или выполнение команд условного перехода. Блок-схема такого конвейера приведена на рисунке.

Основы быстродействия

В транспьютере имеется специальное устройство, группирующее поток входных команд, чтобы обеспечить наибольшую загрузку конвейера. В одну группу объединяются команды, обрабатываемые различными блоками конвейера, что позволяет в установившемся режиме обеспечивать выполнение процессором за один такт всей группы команд (загружать в каждом цикле процессора новую группу команд). При создании управляющего (группирующего) устройства и выборе блоков конвейера фирма INMOS проанализировала структуры кода, вырабатываемого компиляторами языков высокого уровня, что позволило построить процессор таким образом, чтобы он эффективно выполнял наиболее типичные сочетания команд.

В описании транспьютера T9000 [The T9000 Transputer. Products overview. Manual. - INMOS. - 1991] приводится следующий пример, поясняющий процедуру группировки команд и работу конвейера.

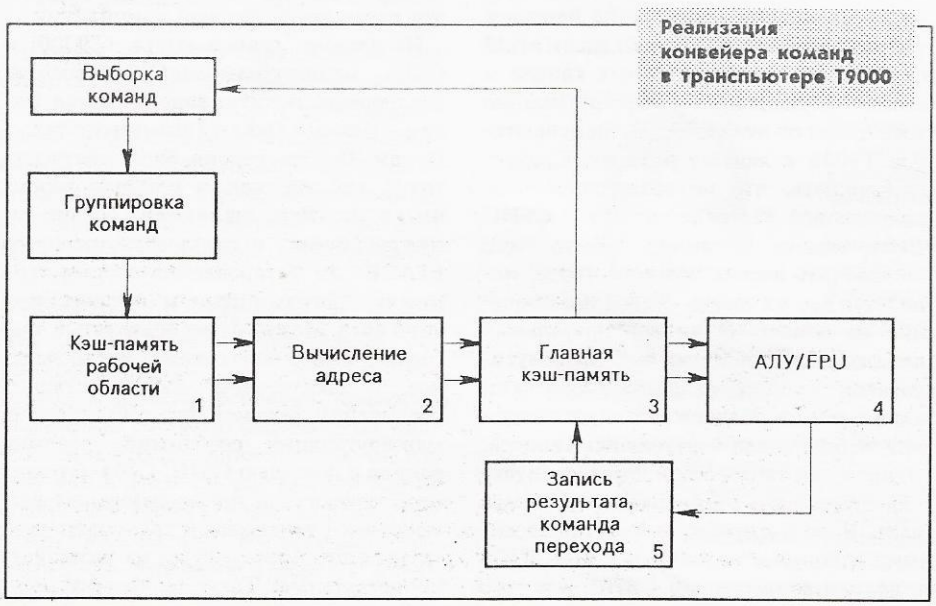
Пусть необходимо вычислить выражение a[i+1] = b[j+15]+c[k+7].

1-я группа	ldl	j	[1]	
	ldl	b	[2]	
	wsub		[2]	
	ldnl	15	[2]	[3]
2-я группа	ldl	k	[1]	
	ldl	c	[1]	
	wsub		[2]	
	ldnl	7	[2]	[3]
	add		[4]	
3-я группа	ldl	i	[1]	
	ldl	a	[1]	
	wsub		[2]	
	stnl	1	[2]	[5]

Справа в квадратных скобках приведены номера блоков конвейера, которые используются при выполнении каждой из команд. С учетом того, что блоки 1, 2 и 3 могут одновременно обрабатывать две команды, произведена группировка команд. Получилось три группы команд. Таким образом, использование конвейера позволяет транспьютеру T9000 вычислить данное выражение за 3 такта. Для сравнения: вычисление аналогичного выражения на транспьютере T805 занимает 25 тактов.

Кроме того, применение в транспьютере T9000 дополнительных аппаратно реализованных устройств позволило сократить число тактов процессора, необходимых для выполнения арифметических и логических операций. В таблице приведены времена выполнения (в тактах процессора) некоторых логических и арифметических операций транспьютерами T805 и T9000:

Тип операции	Транспьютер	
	T805	T9000
multiply	38	2-5
fractional multiply	35-40	3-6
divide	39	5-12
remainder	37	6-13
long add, long substract	2	1
long sum, long diff	3	1
long multiply	33	3-6
long divide	35	15
reverse bit in word	36	1



Особенности управления

Если для инициализации существующих транспьютеров и управления ими предусматривалось использование трех сигналов (Reset, Analyse, Error), то в новом транспьютере для этого предусмотрены два специальных канала, аналогичных линкам. Они обеспечивают двунаправленный обмен информацией и позволяют связать все транспьютеры отдельной сетью управления. Все элементы (узлы) сети могут быть соединены последовательно, а для сетей с большим числом транспьютеров или более сложной кофигурацией управляющие линки могут коммутироваться с помощью СБИС C104. Наличие таких линков, для которых определен специальный протокол управляющих команд, существенно расширяет возможности управления транспьютерами T9000. Управляющий процесс может посылать команды по линкам в виде специальных управляющих пакетов и принимать ответы от удаленных транспьютеров, расположенных в управляемой сети. Принимаемое сообщение может содержать информацию, переданную в ответ на содержавшийся в команде запрос, или просто подтверждать прием и выполнение команды.

Имеющийся набор команд позволяет управлять режимами загрузки транспьютера и запуска прикладной программы, вызывать останов прикладной программы, выполнять инициализацию транспьютера, считывать или модифицировать содержимое отдельных ячеек памяти и конфигурационных регистров транспьютера. Последние управляют режимами работы коммуникационного процессора виртуальных каналов, кэш-памятью, программируемым интерфейсом памяти и линками транспьютера.

Память

В новом транспьютере предусмотрена внутренняя быстродействующая память емкостью 16 Кбайт (в транспьютерах предыдущего поколения - 4 Кбайт). При инициализации транспьютера можно задать один из трех режимов ее использования: вся память отводится под кэш-память для быстрого доступа к наиболее часто используемым данным и фрагментам программы; вся память используется как быстродействующее внутреннее ОЗУ, расположенное в адресном пространстве транспьютера; половина памяти используется как кэш-память, а вторая половина как внутреннее ОЗУ. Кроме указанного блока кэш-памяти в транспьютере для быстрого доступа к локальным переменным предусмотрена кэш-память для верхних 32 слов рабочей области транспьютера *(workspace)*. Этот блок памяти аналогичен блоку регистров общего назначения в других микропроцессорах.

В транспьютере T9000 реализован механизм управления доступной программе пользователя памятью, позволяющий контролировать ее поведение при возникновении ошибочных ситуаций. Для этого процесс пользователя запускается с помощью специального управляющего процесса.

При возникновении в процессе пользователя ошибочных ситуаций, таких, как обращение к памяти по несуществующему адресу или выполнение привилегированной команды, происходит прерывание выполнения процесса пользователя и управление передается управляющему процессу для корректной обработки данной ситуации.

С точки зрения процесса пользователя, вся область физического адресного пространства делится на четыре логические области. Для каждой из этих областей может быть задан размер страницы памяти от 256 до 2^{30} байт и могут быть определены программные сегменты и установлены права процесса пользователя по записи. Эти области могут быть использованы для хранения программ, данных, стека и организации пула свободной памяти при реализации Unix-подобных операционных систем.

Среди других особенностей нового транспьютера отметим наличие специального блока управления внешней памятью. В отличие от существующих транспьютеров, у которых разрядность внешней магистрали данных совпадает с внутренней разрядностью, в новом транспьютере интерфейс управления памятью позволяет настраиваться на внешнюю память с различной длиной слова: от 8 до 64 разрядов. Все адресное пространство внешней памяти может быть разбито на четыре области, и в каждой из них интерфейс может быть настроен на используемую память независимо от соседних областей. Интерфейс позволяет работать с 8-, 16-, 32- и 64-разрядной статической памятью и 32- и 64-разрядной динамической памятью. С учетом того, что основная область применения существующих транспьютеров - встраиваемые системы, можно эффективно строить компактные управляющие вычислительные системы с минимальным числом дополнительного оборудования, например использовать помимо транспьютера только одну микросхему 8-разрядного ПЗУ для хранения управляющих программ. Интерфейс управления внешней памятью позволяет без дополнительных логических элементов подключать к транспьютеру T9000 динамическое ОЗУ емкостью до 8 Мбайт.

Ключ к будущему успеху

Благодаря совместимости по системе команд транспьютера T9000 с существующим поколением транспьютеров значительная часть имеющегося на рынке прикладного и инструментального программного обеспечения может быть использована в вычислительных системах на базе этого транспьютера.

О новых программных средствах, разработанных для транспьютера T9000, будет подробно рассказано в следующих номерах "Журнала д-ра Добба". В первую очередь мы расскажем об операционной системе CHORUS, которую фирма INMOS выбрала в качестве основной операционной системы для своего нового поколения транспьютеров.

факты

☐ Периодически возникающие в среде советских программистов слухи о "смерти" языка Кобол сильно преувеличены. Во всем остальном мире он все еще достаточно широко распространен. Недавно индийская фирма Mafatlar Consultancy Services выпустила пакет Clarient, который анализирует исходный текст на Коболе и генерирует отчеты о перекрестных ссылках, "мертвом" коде *(dead-code)* и операторах GO TO. Разработаны версии пакета Clarient для операционных систем Unix, Xenix и DOS.

☐ Выпущенная американской фирмой Kingston Technology плата SX/Now позволяет в 2,5 раза повысить производительность компьютера на базе микропроцессора 80286. Плата снабжена микропроцессором 80386SX и гнездом для подключения сопроцессора, она совместима с компьютерами производства фирм IBM, Hewlett-Packard и др. Цена платы 645 дол. (16 МГц) или 695 дол. (20 МГц).

Знает ли компьютер поэзию?

Л.Н. Санжаров, А.В. Финьков

Представьте, что Вам предлагают прочесть два стихотворения и просят назвать их авторов:

Какая тающая нежность![*]
Какая млеющая боль!
Что за чеканная небрежность!
Что за воздушная фиоль!

Сыновеет ночей синева,[**]
Веет во все любимое.
И кто-то томительно звал,
Про горести вечера думая.

Мы можем поручиться, что не всякий, даже хорошо знающий поэзию литератор сумеет назвать авторов приведенных строк.

А теперь эти стихотворения транскрибируем, т.е. запишем знаками, отражающими их звучание в живой речи:

КАКА'ЙЬ ТА'ЙУШ'Ш'ЬЙЬ Н'ЕЖНЪС'Т'
КАКА'ЙЬ МЛ'ЕЙУШ'Ш'ЬЙЬ БО'Л'
ШТЬЗЪЧИКА'ННЪЙЬ Н'ИБР'ЕЖНЪС'Т'
ШТЬЗЪВАЗДУ'ШНЪЙЬ Ф'ИОЛ'

СЫНАВ'ЕЙЬТ НАЧЕ'Й С'ИН'ИВА'
В'ЕЙЬТ ВАФС'О Л'УБ'И'МЪЙЬ
ИКТО-ТЪ ТАМ'И'Т'ЬЛ'НЪ ЗВА'Л
ПРАГОР'ЬС'Т'И В'ЕЧЬРЪ ДУ'МЪЙЬ

(Для тех, кто не знаком с транскрибированием, поясним, что апостроф (') после согласных обозначает мягкость, а после гласных - ударность: знак Ъ передает краткий гласный после твердых согласных, знак Ь - краткий гласный после мягких согласных, знак Й обозначает йотирование, а сочетание знаков ЙЬ - безударные гласные.)

Транскрибированные тексты вводим в компьютер. Через некоторое время на экране появляется сообщение о том, что автор первого текста - Игорь Северянин, а второго - Велемир Хлебников. Компьютер не ошибся.

Такой результат - следствие наших долгих размышлений, многочисленных экспериментов, проб и ошибок. Исходной была мысль о том, что композиторы смогли создать тысячи непохожих одно на другое музыкальных произведений, располагая всего лишь семью нотами. А ведь после чтения стихов еще долго помнишь ритмику, мелодику - словом, музыку стиха, которая создается звуками речи: вокальными, высокими, низкими, бемольными, диезными и т.д. Пользуясь этими звуками, поэты создали множество разнообразных стихотворных произведений. "Наверное, - подумали мы, - изучив звуковую структуру стихов, их мелодику, можно определить "звуковой почерк" поэта и, формализовав этот почерк, затем устанавливать, кто же является автором того или иного стихотворения."

Размышляя об этом, мы пришли к выводу, что при звуковом анализе, по крайней мере, поэтической речи нужно исходить из акустической теории звуков, ибо она, раскрывая физическую природу каждого звука, позволяет понять, как эта природа отражается в звучании стиха.

Далее мы рассуждали так: "Чтобы помочь компьютеру "узнавать" поэтический текст по его звучанию, необходимо собрать данные, которые позволят установить закономерности, присущие "звуковому почерку" того или иного поэта. Для этого мы создали специальную программу. Оператор вводил в компьютер транскрибированный текст, и компьютер по определенному алгоритму устанавливал акустические особенности этого текста - как качественные, так и количественные.

Как правило, стихи одного поэта по звучанию не похожи на стихи другого поэта. Это проявляется и в числе звуков одного и того же разряда, и в частоте появления разных звуков в тексте. Так, если сравнить установленные компьютером аку-

Леонид Николаевич Санжаров - и.о. проф. Кафедры русского языка Измаильского пединститута, автор более 100 работ, свыше 60 из которых посвящены использованию компьютеров и аудиовизуальной техники в преподавании лингвистики.
Андрей Владимирович Финьков - старший преподаватель Кафедры педагогики Измаильского пединститута, физик по образованию.

[*] Цит. по кн.: Игорь Северянин. Соловей (Поэзы). - М.: В/О "Союзтеатр СТД СССР", 1990. - 208 с. (Репринтное издание. - Берлин/Москва: Акц. о-во "Накануне", 1923).

[**] Цит. по кн.: Велемир Хлебников. Творения/Под общей ред. М.Я. Полякова. - М.: Советский писатель, 1986. - 736 с.

стические характеристики приведенных в начале статьи стихотворений, то картина будет следующей:

В. Хлебников "Сыновеет ночей синева..."	И. Северянин "Какая тающая нежность..."
Вокальные звуки	
83 %	83 %
Самые распространенные из них	
А, И, Ъ, Ь, Н, Й	А, Ъ, И, Ь, Й, Н
Высокие звуки	
50 %	56 %
Самые распространенные из них	
И, Ъ, Н, Т, Й, Л	Ъ, И, Й, Н, Ш, Е
Диезные звуки	
49 %	42 %
Самые распространенные из них	
И, Н, Т, Ы, Л, З	И, Н, В, Л, У, Н

Проведение аналогичных сопоставлений позволило нам определить основные количественные показатели акустического "облика" поэтической речи, делающие поэтов разными, придающие каждому их произведению определенное музыкально-акустическое звучание.

Собранный с помощью разработанной нами программы материал послужил основой для создания другой программы (результат действия которой и описан в начале статьи). В памяти компьютера накапливались систематизированные акустические особенности стихотворений поэтов, чьи произведения рассматривались. Вводя в компьютер новые тексты, мы сопоставляем их акустические особенности с теми, которые уже содержатся в его памяти. По результатам сопоставления компьютер называет автора.

Надежна ли наша программа? Мы проверили свыше 80 текстов. Безусловное узнавание имело место в 67,8% случаев. В 19,75% случаев компьютер колебался, например о стихах И. Северянина сообщал: "Хлебников или Северянин". Однако фамилия автора обязательно звучала. Следовательно, можно считать, что определенная степень узнавания была в 87,55% случаев. 12,3% составили случаи, когда компьютер автора "не узнал". Это два стихотворения А. Пушкина, одно - С. Есенина и семь - И. Северянина. Среди неузнанных пушкинских стихов - "Вам восемь лет, а мне семнадцать было..." и "Я Вас любил...". Из стихов И. Северянина компьютер не распознал знаменитое "В шумном платье муаровом", а из стихов С. Есенина - "Низкий дом с голубыми ставнями...". Интересно, что среди неузнанных - наиболее известные стихотворения. Здесь, очевидно, есть над чем подумать. В целом, мы считаем, что процент узнавания, приближающийся к 90, говорит о достаточно высокой степени надежности нашей программы.

Проделанная работа позволяет по-новому подойти к решению некоторых филологических проблем.

Во-первых, специалистов давно волнует проблема определения авторства. Существующие методики "компьютерного" установления авторства включают в себя множество параметров (вспомним хотя бы норвежскую экспертизу произведений М.А. Шолохова, потребовавшую исследования примерно 140 000 самых разнообразных языковых элементов). Наша экспертиза основывается лишь на одной группе признаков, описывающих акустические особенности поэтической речи. Это позволяет говорить о звуке, как о факторе, определяющем индивидуальность поэтической речи.

Например, "Учительская газета" (1990, N 41) опубликовала статью о С. Есенине, где говорилось о сомнениях относительно предсмертного произведения поэта "До свиданья, друг мой, до свиданья...". Мы ввели текст стихотворения в компьютер, и он совершенно определенно ответил: "Автор - Сергей Есенин".

Во-вторых, основной классификацией звуков следует признать акустическую, ибо только с помощью звуков обеспечивается передача различных смыслов, их оттенков и т.д. Тот факт, что при изучении фонетики основное внимание уделяется артикуляции, является ошибкой, так как при этом преобладает чисто внешний признак, не раскрывающий существа явления. Надо сказать, что, разрабатывая свою программу, мы в первом ее варианте тоже отдали дань традиции: построили программу на артикуляционных характеристиках - и не получили никаких результатов: компьютер никого не узнавал.

В-третьих, программа может представить доказательства подражания одного поэта другому, выявить факты творческого влияния. Например, в литературной критике отмечалось, что ранний М. Исаковский стремился противопоставить якобы упадническому духу есенинской поэзии "кипучую" действительность 20-х годов и, пытаясь подражать есенинскому стиху, он наполнял свои произведения духом оптимизма. Действительно ли это так? В память нашего компьютера мы ввели "есенинские" стихи М. Исаковского: "Памяти Н.", "Хорошо в застенчивой прохладце", "Все та же даль..." и некоторые другие. Компьютер "колебался": он ни разу не ответил определенно, что это Есенин, но всегда сообщал, что это Хлебников или Есенин либо что это Есенин или Северянин. Другими словами, точно указать авторство компьютер не решался, но в своих "или... или" всегда называл С. Есенина, что говорит о внешнем сходстве анализируемых стихотворений М. Исаковского со стихами С. Есенина.

И, наконец, компьютер может дать материал для размышлений о степени неповторимости того или иного поэта. Так, при исследовании пушкинских текстов в 77% случаев происходило их узнавание, в 11% - компьютер колебался (кто-то или Пушкин), в 11% (об этом уже говорилось выше) пушкинские тексты совсем не были узнаны. Лишь в 2% случаев, при анализе стихотворений "Фея EIOLE" и "Весенний день горяч и золот..." И. Северянина, Пушкин был назван возможным автором. В то же время В. Хлебников выступает в роли второго автора почти в 30% случаев при анализе текстов С. Есенина и И. Северянина и ни разу при рассмотрении пушкинских стихов. Эти данные подчеркивают неповторимое своеобразие пушкинской поэзии. Подтверждается известная поэтическая фраза: "Пушкин пушкински велик".

В заключение мы вновь обращаемся к вопросу, вынесенному в заголовок статьи. Думаем, что перед исследователем, использующим компьютер, открываются большие возможности. Сознаемся, что, начиная работу, мы ставили перед собой только одну задачу определения авторства. Ход исследования расширил горизонты, и мы неожиданно увидели многое другое.

факты

☐ Издательство Chadwyck-Healey, известное своими академическими изданиями, составило и записало на оптический диск собрание сочинений более 1300 английских поэтов, состоящее почти из 5000 томов (более 1 Гбайт информации)! Многие из этих произведений не перепечатывались с момента их первого опубликования в XVII-XVIII веках. Итак, новая технология предоставила неизвестным поэтам еще одну возможность прославиться.

Режим X: 256-цветовая магия адаптера VGA

М. Эйбраш

Есть известное латинское выражение: "In complexitate est opportunitas" (в сложности кроются возможности), которая приходит на ум при мысли об адаптере VGA. Иллюстрацией ее справедливости может служить странный режим работы адаптера VGA (256 цветов при разрешающей способности 320x240 точек). Программирование адаптера VGA, работающего в этом режиме, - несомненно, трудная работа; он даже не документирован фирмой IBM, но, тем не менее, является одним из лучших (особенно для анимации) режимов работы этого адаптера.

Что делает режим "320x240" особенным?

Пять отличительных черт ставят режим "320x240" (256 цветов) (я называю этот режим режимом X, учитывая его мистический статус в документации фирмы IBM) в особое положение по сравнению с другими режимами работы адаптера VGA.

Во-первых, в этом режиме достигается соотношение горизонтального и вертикального размеров 1:1, т.е. размеры каждого элемента изображения (пикселя) по горизонтали и вертикали одинаковы (на программистском жаргоне пиксель с таким соотношением размеров называется квадратным). Квадратные пиксели позволяют создавать наиболее привлекательные изображения и избегать значительных программистских усилий для подстройки графических примитивов и изображений к горизонтальному и вертикальному размерам пикселя (например, в случае квадратных пикселей круг может быть нарисован как круг, иначе необходимо рисовать его как эллипс, чтобы подстроиться под соотношение горизонтального и вертикального размеров пикселя, что замедляет и усложняет процесс формирования изображения). Для сравнения: режим 13h, единственный документированный режим работы адаптера VGA, обеспечивающий получение 256 цветов, имеет "неквадратную" разрешающую способность 320x200 точек.

Во-вторых, режим X обладает способностью "перевертывания" страниц, что позволяет осуществлять наиболее "гладкую" анимацию. Режим 13h и режим 12h (640x480 точек, 16 цветов), обладающий наивысшей разрешающей способностью, такой возможности не предоставляют.

В-третьих, режим X позволяет странично-ориентированному аппаратному обеспечению адаптера VGA обрабатывать пиксели в параллельном режиме, что увеличивает скорость обновления изображения в четыре раза (по сравнению с режимом 13h).

В-четвертых, подобно режиму 13h (но в отличие от всех остальных режимов), режим X является режимом "байт-на-пиксель" (каждому пикселю соответствует один байт видеопамяти), что исключает медленные операции чтения перед записью и маскирования бит, часто требуемые в 16-цветных режимах. Кроме уменьшения (в два раза) числа обращений к памяти это важно и потому, что схемы кэширования памяти, используемые многими VGA-совместимыми адаптерами, ускоряют операции записи в большей степени, чем операции чтения.

В-пятых, в отличие от режима 13h, в режиме X обеспечивается доступ к большому объему свободной внеэкранной памяти для хранения изображений. Это можно особенно эффективно использовать в сочетании с имеющимися в адаптере "защелками" *(latches)*: "защелки" и внеэкранная память позволяют копировать изображения на экран по 4 пикселя одновременно.

Есть и шестая черта режима X, которая не столь воодушевляет: адаптер, работающий в этом режиме, трудно программировать. Если Вы когда-либо занимались программированием адаптера VGA в 16-цветном режиме, то знаете, насколько трудоемким может оказаться это занятие; программирование адаптера VGA в режиме X зачастую не менее трудоемко и вдобавок подчиняется набору правил, которые существенно отличаются от правил, присущих программированию адаптера VGA в 16-цветном режиме. Программирование для режима X не имеет ничего общего с программированием для элегантного (организованного как "плоская" битовая карта) режима 13h или с программированием плоской, линейной (хотя и разбитой на области) битовой карты, используемой в 256-цветных режимах работы адаптера SuperVGA (я хотел бы подчеркнуть, что в режиме X работают все адаптеры VGA, а не только SuperVGA). Многие программисты, с которыми я разговаривал, любят организацию видеопамяти в виде плоской битовой карты, и я думаю, что это идеальная организация видеопамяти, поскольку она чрезвычайно удобна для программирования. Однако вспомните выражение, с которого я начал статью, - в сложности режима X на самом деле кроется возможность наилучшего сочетания производительности работы графических программ и качества изображения, которое только может быть получено при использовании этого адаптера. Если Вы программируете адаптер VGA в 256-цветном режиме, то (особенно, если Вы занимаетесь анимацией) делаете большую ошибку, если не используете режим X.

Хотя некоторые разработчики пользуются преимуществами режима X, это является исключением из общего правила, поскольку он абсолютно не документирован; только опытный программист, много работавший с адаптером VGA, мог бы обнаружить какой-то намек на существование этого режима, а понять, как заставить адаптер работать в этом режи-

© Dr. Dobb's Journal, July 1991.
All rights reserved. M&T Publishing, Redwood City, CA, USA.

ме (за исключением операций чтения/записи пикселя), достаточно трудно. Я не видел в печати каких-либо упоминаний об этом режиме, кроме моих собственных статей о 256-цветных режимах "320x200", "320x400" и "360x480" в журнале "Programmer's Journal" (январский и сентябрьский номера за 1989 г.). (Однако Джон Бриджез (John Bridges) предоставил для общего пользования код программ для нескольких 256-цветных режимов работы адаптера VGA, и я хотел бы отметить влияние его кода на мою подпрограмму установки режима работы адаптера, представленную в этой статье.)

Учитывая огромные преимущества режима "320x240" перед документированным режимом 13h, я бы хотел, чтобы он использовался как можно большим числом программистов, поэтому в следующих номерах журнала я продолжу изучение этого странного, но весьма полезного режима. Я продемонстрирую программу, устанавливающую этот режим, опишу организацию битовой карты и покажу, как работают базовые операции чтения и записи пикселя. Потом я перейду к "магии": заполнению прямоугольников, очистке экрана, прокрутке, копированию изображений, инвертированию пикселей и (да, да!) заполнению многоугольников (всего лишь другой драйвер), всем быстрым растровым операциям, обеспечиваемым аппаратными средствами, и "перевертыванию" страниц. В конце концов я опубликую работающую программу по анимации, которая демонстрирует многие свойства режима X в действии.

Отправной точкой, естественно, является подпрограмма установки режима работы адаптера.

Выбор 256-цветного режима "320x240"

При желании можно было бы с нуля написать собственную программу установки режима X, но зачем утруждать себя? Вместо этого мы перепоручим большую часть работы системе BIOS - она установит режим 13h, а мы затем преобразуем его в режим X, изменив несколько регистров. Программа 1 (программы см. на с. 44-46) делает именно это.

После установки режима 13h эта программа изменяет значение счетчика числа вертикальных линий и некоторые временные параметры для установки числа видимых строк развертки равным 480 (нет нужды изменять значения, относящиеся к разрешающей способности по горизонтали, так как и в режиме 13h, и в режиме X она составляет 320 пикселей). Однако регистр, задающий максимальное число строк развертки, программируется таким образом, что каждая строка повторяется дважды, и в результате эффективная разрешающая способность по вертикали составляет 240 строк. На самом деле в 256-цветном режиме можно получить 400 или 480 независимых строк (подробности см. в вышеупомянутых статьях); однако в 400-строковом режиме пиксели не квадратные, кроме того, не обеспечивается одновременная поддержка внеэкранной памяти и "перевертывания" страниц, а в 480-строковом режиме "перевертывание" страниц вообще отсутствует (в силу ограничений по памяти). Одновременно программа 1 программирует битовую карту адаптера VGA в плоскостно-ориентированном виде, похожем на тот, который используется в большинстве 16-цветных режимов, и кардинально отличающемся от линейной схемы режима 13h. Курьезная организация битовой карты режима 13h показана на рис.1. Первому пикселю (пиксель в левом верхнем углу экрана) соответствует байт со смещением 0 плоскости 0 (одна благословенная общая особенность режимов X и 13h заключается в том, что каждому пикселю соответствует один байт, что исключает необходимость маскирования отдельных бит видеопамяти). Следующему (находящемуся непосредственно справа от первого) пикселю соответствует байт со смещением 0 плоскости 1, третьему - со смещением 0 плоскости 2, четвертому - со смещением 0 плоскости 3. Пятому пикселю соответствует байт со смещением 1 плоскости 0 и т.д., так что каждой группе из четырех пикселей соответствуют четыре байта, разбросанные по четырем плоскостям, но имеющие одинаковое смещение. Смещение M пикселя N в видеопамяти составляет N/4 (остаток отбрасывается), а номер плоскости P пикселя N определяется так: P = N mod 4. Для операций записи в видеопамять плоскость определяется установкой бита P регистра маски битовой карты (регистра с индексом 2 графического контроллера) в 1, а всех остальных бит - в 0; для операций чтения плоскость определяется установкой регистра чтения карты (регистра с индексом 4) в P.

Нечего и говорить, что такая организация битовой карты весьма неудобна, так как требует довольно больших усилий даже для манипуляции единичным пикселем. Программа 2 записи пикселя должна определить соответствующую ему плоскость и выполнить 16-битозую операцию OUT для выбора этой плоскости (и так для каждого пикселя). Подобные манипуляции надо проводить и при чтении пикселя (см. программу 3). Необходимость выполнения операций вычисления номера и активизации соответствующей плоскости для каждого пикселя вряд ли помогает обеспечению высокой производительности работы.

Однако это не страшно, так как большинство графических программ тратят

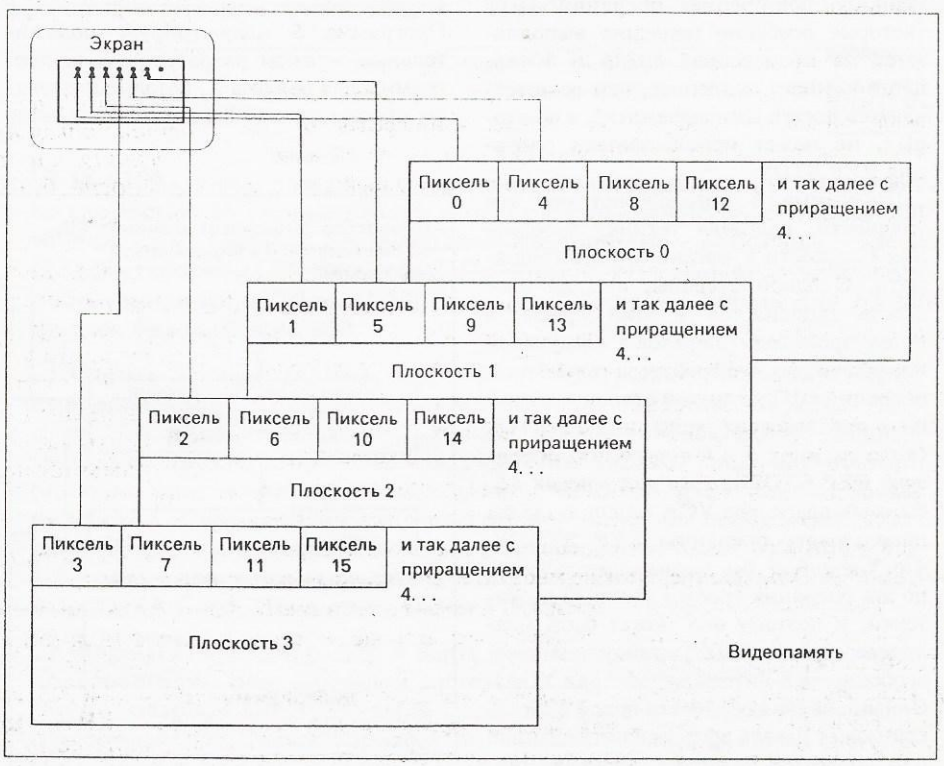


Рис.1. Организация видеопамяти в режиме X (320 x 240 точек, 256 цветов)

относительно мало времени на отображение отдельных пикселей. Я демонстрирую эти программы в качестве базисных примитивов, чтобы Вы могли лучше понять организацию битовой карты, но основными строительными блоками высокопроизводительной графики являются операции заполнения и копирования, а для их выполнении режим X просто незаменим.

Разработка программ с точки зрения возможностей режима X

Программа 4 показывает, как в режиме X заполняется прямоугольник. Для каждого пикселя выбирается соответствующая ему плоскость, и процесс закрашивания циклически охватывает плоскости от нулевой до третьей. Такой алгоритм программирования основан на восприятии задачи, как цепочки операций записи пикселя, и никоим образом не использует уникальных возможностей режима X. Хотя он и выглядит относительно эффективным, на самом деле в результате получается одна из самых медленных подпрограмм.

Программа 4 приведена здесь в иллюстративных целях, а также затем, чтобы иметь исходный вариант для последующих модификаций, использующих возможности режима X, которые позволят значительно ускорить работу программы. Два главных недостатка программы 4 определяются тем, что плоскость выбирается для каждого очередного пикселя. Во-первых, это приводит к бесконечному повторению операций OUT (которые особенно медленно выполняются на процессорах 80386 и 80486; часто намного медленнее, чем осуществляется доступ к видеопамяти), а во-вторых, не может использоваться операция REP STOS.

Программа *5* преодолевает обе эти трудности, подгоняя технику заполнения плоскости к организации видеопамяти. С одной стороны, каждая плоскость заполняется в один прием, а потом происходит переход к следующей плоскости, так что требуется только пять операций OUT; с другой стороны, может быть использована операция REP STOS (в программах *5* и 6 я применял операцию REP STOSB, хотя для многих 16-битовых адаптеров VGA можно было бы использовать операцию REP STOSW, что улучшило бы производительность, но эта операция требует большей подготовки, и поэтому она может быть медленнее для малых прямоугольников, особенно на 8-битовых адаптерах VGA). Одновременная обработка целой плоскости может давать эффект "ниспадания" при обработке больших изображений, поскольку все столбцы одной плоскости обрабатываются до начала обработки столбцов следующей; если это создает проблемы, четыре плоскости можно циклически менять для каждой строки развертки (вместо того, чтобы делать это один раз для всего прямоугольника). Программа *5* работает в 2,5 раза быстрее, чем программа 4 (при очистке экрана компьютера с процессором 80386 и тактовой частотой 20 МГц, кэш-памятью и адаптером VGA фирмы Paradise). Хотя она и несколько медленнее, чем аналогичная программа для режима 13h, разница не столь уж велика. В общем, выполнение операции отдельно по каждой плоскости делает работу почти каждой графической операции в режиме X столь же быстрой (как минимум), как и в режиме 13h (хотя следует признать, что такой стиль программирования в режиме X достаточно непрост). По этой причине полезно организовать структуры данных так, чтобы они учитывали особенности режима X. Например, пиктограммы могли бы быть заранее подготовлены в системной памяти так, чтобы пиксели были организованы в четыре плоскостно-ориентированных набора (или в четыре набора на каждую строку развертки, если необходимо избежать эффекта "ниспадания") для облегчения возможности копирования на экран сразу целой плоскости с помощью операции REP MOVS.

Аппаратная поддержка из неожиданного источника

Программа *5* иллюстрирует положительные аспекты разработки кода программы для режима X, которые позволяют сделать режим X практически столь же быстрым, как режим 13h. Интерес к режиму X вызван такими его особенностями, как квадратные пиксели, "перевертывание" страниц, внеэкранная память, однако высокая производительность была бы, тем не менее, крайне желательным дополнением ко всему этому. Такая производительность в режиме X действительно возможна, хотя она связана с аппаратными средствами адаптера VGA (что довольно странно), которые в общем-то создавались без расчета на работу в 256-цветном режиме.

Все аппаратные средства поддержки адаптера VGA доступны и в режиме X, хотя некоторые и не особенно полезны. Действительно, важной, с точки зрения обеспечения высокой производительности работы графических программ в режиме X, является возможность параллельной обработки данных, соответствующих всем четырем плоскостям; это относится как к "защелкам", так и к возможности выдачи данных в любую плоскость или во все плоскости одновременно. Для операции заполнения прямоугольников достаточно возможности выдачи данных в различные плоскости, поэтому я отложу рассмотрение других аппаратных средств поддержки до следующего раза (между прочим, некоторые возможности, например битовые маски и др., доступны и в режиме 13h, но параллельная обработка данных исключается).

В режимах, основанных на использовании понятия плоскости (к ним относится и режим X), байт, записываемый из процессора в видеопамять, может на самом деле записываться в любое число плоскостей, от нуля до четырех (рис.2). В каждую плоскость, для которой соот-

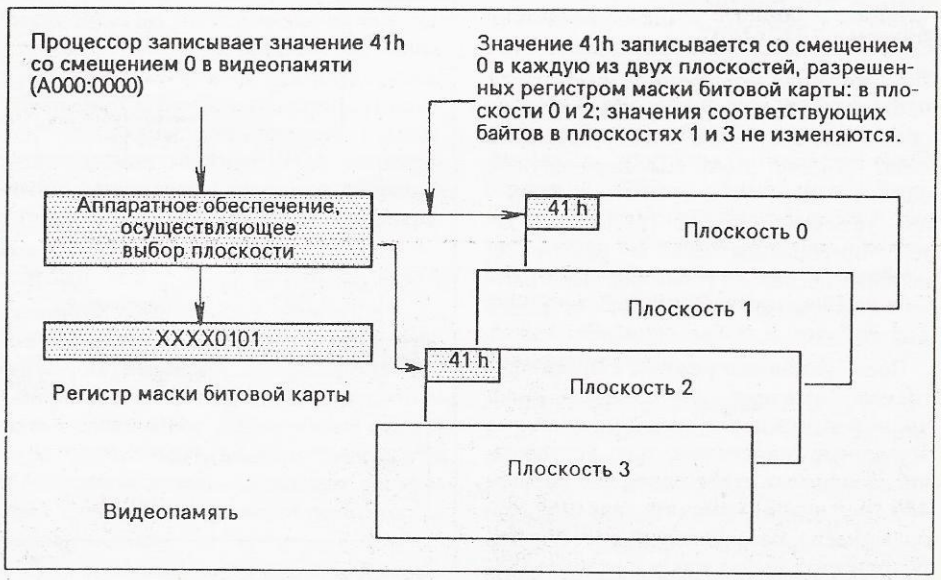


Рис.2. Регистр маски битовой карты выбирает, в какие плоскости осуществляется запись

ветствующий бит в регистре маски битовой карты установлен в 1, пересылается байт от процессора, а те, для которых этот бит установлен в 0, остаются неизменными.

В 16-цветных режимах каждая плоскость содержит четверть каждой группы из восьми пикселей, при этом 4 бита каждого пикселя разбросаны по всем четырем плоскостям. В режиме X ситуация другая. Снова взгляните на рис.1: каждая плоскость полностью содержит один пиксель, при этом по каждому данному адресу находятся четыре пикселя, по одному на плоскость. Несмотря на это регистр маски битовой карты работает в режиме X точно так же, как в и 16-цветных режимах; установите его в 0Fh (все биты в 1), и каждая операция записи будет выполняться для всех четырех плоскостей. Поэтому создается впечатление, что одна операция записи в видеопамять в режиме X может обеспечить установку вплоть до четырех пикселей, и тем самым операция заполнения прямоугольника может быть ускорена в 4 раза.

Программа 6 дает другой пример заполнения прямоугольника; на этот раз используется регистр маски битовой карты для установки до четырех пикселей на каждую операцию STOS. Единственный примененный в ней трюк заключается в том, что левая или правая сторона прямоугольника, не выровненная по кратному четырем столбцу пикселей (т.е. такому, на котором заканчивается один набор из четырех пикселей и начинается другой), должны рисоваться согласно регистру маски битовой карты, поскольку не все пиксели по адресу, соответствующему этой стороне, подлежат модификации. Производительность близка к ожидаемой: программа 6 работает почти в 10 раз быстрее, чем аналогичная программа 4, и почти в 4 раза быстрее, чем программа 5 (а также почти в 4 раза быстрее, чем программа заполнения прямоугольника для режима 13h). Понимание организации битовой карты и средств аппаратной поддержки, работающих в режиме X, действительно помогает.

Для того чтобы Вы лучше почувствовали режим X в действии, я демонстрирую тестовую программу 7, которая устанавливает режим X, а потом рисует ряд прямоугольников. К ней может быть подключена любая из описанных выше программ.

Я надеюсь, что сейчас Вы начинаете понимать, почему мне так нравится режим X. В последующих статьях мы продолжим его изучение, исследуя "чудеса", обеспечиваемые сочетанием "защелок" и аппаратных средств параллельной обработки при проведении таких операций, как прокрутка, заполнение по образцу, копирование и др.

Новости с фронта Edsun

Доходящие до меня сведения свидетельствуют о большом интересе, который проявляют программисты к микросхеме CEG/DAC фирмы Edsun, (см. мои статьи в апрельском и майском номерах журнала "Dr. Dobb's Journal" за 1991 г). Однако все, кому доводилось реально программировать микросхему CEG/DAC, жалуются на то, что это весьма трудно; результаты впечатляют, но процесс их достижения слишком сложен. Тем не менее программирование микросхемы CEG/DAC представляет собой разрешимую проблему, и тот, кому удастся решить ее наилучшим образом, будет выглядеть солидно. Здесь можно провести аналогию с созданием резидентных программ (TSR). Шесть лет назад написание таких программ было чем-то вроде черной магии, и, например, программа Sidekick, достаточно примитивная по современным меркам, принесла своим создателям целое состояние. Сейчас любой новичек может с помощью десятка книг и инструментальных средств написать надежную TSR-программу за несколько часов. Со временем, когда программисты создадут лучшие инструментальные средства и лучше разберутся в микросхеме CEG/DAC, ворчание утихнет и появится соответствующее программное обеспечение. Это еще один случай, когда в сложности одновременно кроется возможность.

Книга месяца

Книгой месяца я считаю книгу "Руководство для программиста по SuperVGA" (Advanced Programmer's Guide to SuperVGA) Сатти и Блэра (Sutty and Blair) (Brady, 1990, ISBN 0-13-010455-8, 44,95 дол.). Для книги в мягкой обложке она несколько дороговата, но к ней прилагается дискета с кодом на языке ассемблера для адаптера SuperVGA. По-моему, эта книга является наилучшим руководством по программированию адаптера SuperVGA (в котором каждый из нескольких дюжин режимов работы имеет различные номера и схемы управления видеопамятью). Поверьте тому, кому пришлось блуждать среди гор книг по микросхемам и руководств по программированию приложений, - эта книга позволит сэкономить немало времени и избавит от многих проблем, если Вам придется непосредственно программировать адаптер SuperVGA.

Однако не все, что я хотел бы видеть в этой книге, в нее попало. Например, авторы описывают только микросхему ET3000 фирмы Tseng Labs, а не широко используемую ныне микросхему ET4000, поддерживающую более высококачественную графику. Конечно, это не вина авторов, а просто отражение невероятного разнообразия возможностей и скорости их изменений в мире SuperVGA.

факты

☐ JX-9700 - новый лазерный принтер фирмы Sharp Electronics, печатает со скоростью 16 страниц/мин и имеет разрешающую способность 300 точек/дюйм. Главные достоинства этого принтера - небольшие масса (22 кг) и габариты (40x42x27,1 см). Принтер JX-9700 может эмулировать работу принтеров HP LaserJet II, Proprinter и Graphics фирмы IBM, FX-80 фирмы Epson, Diablo 630.

☐ Фирма Comsell (США) разработала программу Windows Basics, предназначенную для обучения основам использования среды Windows 3.0 и ее приложений. Кроме всевозможных пояснительных текстов и таблиц обучающийся видит на экране лицо инструктора (конечно же, миловидной женщины!), который дает необходимые пояснения и отвечает на вопросы. Чтобы запустить программу Windows Basics, нужно иметь в своем распоряжении пакет Windows 3.0, проигрыватель для видеодисков и видеоадаптер IBM M-Motion Video Adapter/A Card. Цена программы 1195 дол.

☐ Фирмы Locus Computing и Santa Cruz Operation (SCO) предложили пользователям операционной системы SCO Unix System V программу Merge, разработанную фирмой Locus и предназначенную для эмуляции среды DOS. Таким образом, пользователи операционной системы Unix смогут теперь на своих компьютерах запускать программы, предназначенные для DOS.

ПРОГРАММА 1

```
; Mode X (320x240, 256 colors) mode set routine. Works on all VGAs.
; C near-callable as:
;       void Set320x240Mode(void);
; Tested with TASM 2.0.
; Modified from public-domain mode set code by John Bridges.

SC_INDEX equ    03c4h   ;Sequence Controller Index
CRTC_INDEX equ  03d4h   ;CRT Controller Index
MISC_OUTPUT equ 03c2h   ;Miscellaneous Output register
SCREEN_SEG equ  0a000h  ;segment of display memory in mode X

        .model  small
        .data
; Index/data pairs for CRT Controller registers that differ between
; mode 13h and mode X.
CRTParms label  word
        dw      00d06h  ;vertical total
        dw      03e07h  ;overflow (bit 8 of vertical counts)
        dw      04109h  ;cell height (2 to double-scan)
        dw      0ea10h  ;v sync start
        dw      0ac11h  ;v sync end and protect cr0-cr7
        dw      0df12h  ;vertical displayed
        dw      00014h  ;turn off dword mode
        dw      0e715h  ;v blank start
        dw      00616h  ;v blank end
        dw      0e317h  ;turn on byte mode
CRT_PARM_LENGTH equ     ($-CRTParms)/2)

        code
        public  _Set320x240Mode
_Set320x240Mode proc    near
        push    bp      ;preserve caller's stack frame
        push    si      ;preserve C register vars
        push    di      ; (don't count on BIOS preserving anything)

        mov     ax,13h  ;let the BIOS set standard 256-color
        int     10h     ; mode (320x200 linear)

        mov     dx,SC_INDEX
        mov     ax,0604h
        out     dx,ax   ;disable chain4 mode
        mov     ax,0100h
        out     dx,ax   ;synchronous reset while switching clocks

        mov     dx,MISC_OUTPUT
        mov     al,0e7h
        out     dx,al   ;select 28 MHz dot clock & 60 Hz scanning rate

        mov     dx,SC_INDEX
        mov     ax,0300h
        out     dx,ax   ;undo reset (restart sequencer)

        mov     dx,CRTC_INDEX ;reprogram the CRT Controller
        mov     al,11h  ;VSync End reg contains register write
        out     dx,al   ; protect bit
        inc     dx      ;CRT Controller Data register
        in      al,dx   ;get current VSync End register setting
        and     al,7fh  ;remove write protect on various
        out     dx,al   ; CRTC registers
        dec     dx      ;CRT Controller Index
        cld
        mov     si,offset CRTParms ;point to CRT parameter table
        mov     cx,CRT_PARM_LENGTH ;# of table entries
SetCRTParmsLoop:
        lodsw           ;get the next CRT Index/Data pair
        out     dx,ax   ;set the next CRT Index/Data pair
        loop    SetCRTParmsLoop

        mov     dx,SC_INDEX
        mov     ax,0f02h
        out     dx,ax   ;enable writes to all four planes
        mov     ax,SCREEN_SEG ;now clear all display memory, 8 pixels
        mov     es,ax         ; at a time
        sub     di,di   ;point ES:DI to display memory
        sub     ax,ax   ;clear to zero-value pixels
        mov     cx,8000h ;# of words in display memory
        rep     stosw   ;clear all of display memory

        pop     di      ;restore C register vars
        pop     si
        pop     bp      ;restore caller's stack frame
        ret
_Set320x240Mode endp
        end
```

Конец

ПРОГРАММА 2

```
; Mode X (320x240, 256 colors) write pixel routine. Works on all VGAs.
; No clipping is performed.
; C near-callable as:
;     void WritePixelX(int X, int Y, unsigned int PageBase, int Color);

SC_INDEX equ    03c4h   ;Sequence Controller Index
MAP_MASK equ    02h     ;index in SC of Map Mask register
SCREEN_SEG equ  0a000h  ;segment of display memory in mode X
SCREEN_WIDTH equ 80     ;width of screen in bytes from one scan line
                        ; to the next

parms   struc
        dw      2 dup (?) ;pushed BP and return address
X       dw      ?       ;X coordinate of pixel to draw
Y       dw      ?       ;Y coordinate of pixel to draw
PageBase dw     ?       ;base offset in display memory of page in
                        ; which to draw pixel
Color   dw      ?       ;color in which to draw pixel
parms   ends

        .model  small
        .code
        public  _WritePixelX
_WritePixelX    proc    near
        push    bp      ;preserve caller's stack frame
        mov     bp,sp   ;point to local stack frame

        mov     ax,SCREEN_WIDTH
        mul     [bp+Y]  ;offset of pixel's scan line in page
        mov     bx,[bp+X]
        shr     bx,1
        shr     bx,1    ;X/4 = offset of pixel in scan line
        add     bx,ax   ;offset of pixel in page
        add     bx,[bp+PageBase] ;offset of pixel in display memory
        mov     ax,SCREEN_SEG
        mov     es,ax   ;point ES:BX to the pixel's address

        mov     cl,byte ptr [bp+X]
        and     cl,011b ;CL = pixel's plane
        mov     ax,0100h + MAP_MASK ;AL = index in SC of Map Mask reg
        shl     ah,cl   ;set only the bit for the pixel's plane to 1
        mov     dx,SC_INDEX ;set the Map Mask to enable only the
        out     dx,ax       ; pixel's plane

        mov     al,byte ptr [bp+Color]
        mov     es:[bx],al ;draw the pixel in the desired color

        pop     bp      ;restore caller's stack frame
        ret
_WritePixelX    endp
        end
```

Конец

ПРОГРАММА 3

```
; Mode X (320x240, 256 colors) read pixel routine. Works on all VGAs.
; No clipping is performed.
; C near-callable as:
;     unsigned int ReadPixelX(int X, int Y, unsigned int PageBase);

GC_INDEX equ    03ceh   ;Graphics Controller Index
READ_MAP equ    04h     ;index in GC of the Read Map register
SCREEN_SEG equ  0a000h  ;segment of display memory in mode X
SCREEN_WIDTH equ 80     ;width of screen in bytes from one scan line
                        ; to the next
parms   struc
        dw      2 dup (?) ;pushed BP and return address
X       dw      ?       ;X coordinate of pixel to read
Y       dw      ?       ;Y coordinate of pixel to read
PageBase dw     ?       ;base offset in display memory of page from
                        ; which to read pixel
parms   ends

        .model  small
        .code
        public  _ReadPixelX
_ReadPixelX     proc    near
        push    bp      ;preserve caller's stack frame
        mov     bp,sp   ;point to local stack frame

        mov     ax,SCREEN_WIDTH
        mul     [bp+Y]  ;offset of pixel's scan line in page
        mov     bx,[bp+X]
        shr     bx,1
        shr     bx,1    ;X/4 = offset of pixel in scan line
        add     bx,ax   ;offset of pixel in page
        add     bx,[bp+PageBase] ;offset of pixel in display memory
        mov     ax,SCREEN_SEG
        mov     es,ax   ;point ES:BX to the pixel's address

        mov     ah,byte ptr [bp+X]
        and     ah,011b ;AH = pixel's plane
        mov     al,READ_MAP ;AL = index in GC of the Read Map reg
        mov     dx,GC_INDEX ;set the Read Map to read the pixel's
        out     dx,ax       ; plane

        mov     al,es:[bx] ;read the pixel's color
        sub     ah,ah   ;convert it to an unsigned int

        pop     bp      ;restore caller's stack frame
        ret
_ReadPixelX     endp
        end
```

Конец

ПРОГРАММА 4

```
; Mode X (320x240, 256 colors) rectangle fill routine. Works on all
; VGAs. Uses slow approach that selects the plane explicitly for each
; pixel. Fills up to but not including the column at EndX and the row
; at EndY. No clipping is performed.
; C near-callable as:
;     void FillRectangleX(int StartX, int StartY, int EndX, int EndY,
;         unsigned int PageBase, int Color);

SC_INDEX equ    03c4h   ;Sequence Controller Index
MAP_MASK equ    02h     ;index in SC of Map Mask register
SCREEN_SEG equ  0a000h  ;segment of display memory in mode X
SCREEN_WIDTH equ 80     ;width of screen in bytes from one scan line
                        ; to the next
parms   struc
        dw      2 dup (?) ;pushed BP and return address
StartX  dw      ?       ;X coordinate of upper left corner of rect
```

```
StartY  dw      ?       ;Y coordinate of upper left corner of rect
EndX    dw      ?       ;X coordinate of lower right corner of rect
                        ; (the row at EndX is not filled)
EndY    dw      ?       ;Y coordinate of lower right corner of rect
                        ; (the column at EndY is not filled)
PageBase dw     ?       ;base offset in display memory of page in
                        ; which to fill rectangle
Color   dw      ?       ;color in which to draw pixel
parms   ends

        .model  small
        .code
        public  _FillRectangleX
_FillRectangleX proc    near
        push    bp      ;preserve caller's stack frame
        mov     bp,sp   ;point to local stack frame
        push    si      ;preserve caller's register variables
        push    di

        mov     ax,SCREEN_WIDTH
        mul     [bp+StartY] ;offset in page of top rectangle scan line
        mov     di,[bp+StartX]
        shr     di,1
        shr     di,1    ;X/4 = offset of first rectangle pixel in scan
                        ; line
        add     di,ax   ;offset of first rectangle pixel in page
        add     di,[bp+PageBase] ;offset of first rectangle pixel in
                        ; display memory
        mov     ax,SCREEN_SEG
        mov     es,ax   ;point ES:DI to the first rectangle pixel's
                        ; address
        mov     dx,SC_INDEX ;set the Sequence Controller Index to
        mov     al,MAP_MASK ; point to the Map Mask register
        out     dx,al
        inc     dx      ;point DX to the SC Data register
        mov     cl,byte ptr [bp+StartX]
        and     cl,011b ;CL = first rectangle pixel's plane
        mov     al,01h
        shl     al,cl   ;set only the bit for the pixel's plane to 1
        mov     ah,byte ptr [bp+Color] ;color with which to fill
        mov     bx,[bp+EndY]
        sub     bx,[bp+StartY]  ;BX = height of rectangle
        jle     FillDone        ;skip if 0 or negative height
        mov     si,[bp+EndX]
        sub     si,[bp+StartX]  ;CX = width of rectangle
        jle     FillDone        ;skip if 0 or negative width
FillRowsLoop:
        push    ax      ;remember the plane mask for the left edge
        push    di      ;remember the start offset of the scan line
        mov     cx,si   ;set count of pixels in this scan line
FillScanLineLoop:
        out     dx,al   ;set the plane for this pixel
        mov     es:[di],ah ;draw the pixel
        shl     al,1    ;adjust the plane mask for the next pixel's
        and     al,01111b ; bit, modulo 4
        jnz     AddressSet ;advance address if we turned over from
        inc     di      ; plane 3 to plane 0
        mov     al,00001b ;set plane mask bit for plane 0
AddressSet:
        loop    FillScanLineLoop
        pop     di      ;retrieve the start offset of the scan line
        add     di,SCREEN_WIDTH ;point to the start of the next scan
                        ; line of the rectangle
        pop     ax      ;retrieve the plane mask for the left edge
        dec     bx      ;count down scan lines
        jnz     FillRowsLoop
FillDone:
        pop     di      ;restore caller's register variables
        pop     si
        pop     bp      ;restore caller's stack frame
        ret
_FillRectangleX endp
        end
```

Конец

ПРОГРАММА 5

```
; Mode X (320x240, 256 colors) rectangle fill routine. Works on all
; VGAs. Uses medium-speed approach that selects each plane only once
; per rectangle; this results in a fade-in effect for large
; rectangles. Fills up to but not including the column at EndX and the
; row at EndY. No clipping is performed.
; C near-callable as:
;     void FillRectangleX(int StartX, int StartY, int EndX, int EndY,
;         unsigned int PageBase, int Color);

SC_INDEX equ    03c4h   ;Sequence Controller Index
MAP_MASK equ    02h     ;index in SC of Map Mask register
SCREEN_SEG equ  0a000h  ;segment of display memory in mode X
SCREEN_WIDTH equ 80     ;width of screen in bytes from one scan line
                        ; to the next
parms struc
        dw      2 dup (?) ;pushed BP and return address
StartX  dw      ?       ;X coordinate of upper left corner of rect
StartY  dw      ?       ;Y coordinate of upper left corner of rect
EndX    dw      ?       ;X coordinate of lower right corner of rect
                        ; (the row at EndX is not filled)
EndY    dw      ?       ;Y coordinate of lower right corner of rect
                        ; (the column at EndY is not filled)
PageBase dw     ?       ;base offset in display memory of page in
                        ; which to fill rectangle
Color   dw      ?       ;color in which to draw pixel
parms ends

StartOffset equ -2      ;local storage for start offset of rectangle
Width   equ     -4      ;local storage for address width of rectangle
Height  equ     -6      ;local storage for height of rectangle
PlaneInfo equ   -8      ;local storage for plane # and plane mask
STACK_FRAME_SIZE equ 8

        .model  small
        .code
        public  _FillRectangleX
_FillRectangleX proc    near
        push    bp      ;preserve caller's stack frame
        mov     bp,sp   ;point to local stack frame
        sub     sp,STACK_FRAME_SIZE ;allocate space for local vars
        push    si      ;preserve caller's register variables
        push    di

        cld
        mov     ax,SCREEN_WIDTH
        mul     [bp+StartY] ;offset in page of top rectangle scan line
        mov     di,[bp+StartX]
        shr     di,1
        shr     di,1    ;X/4 = offset of first rectangle pixel in scan
                        ; line
        add     di,ax   ;offset of first rectangle pixel in page
        add     di,[bp+PageBase] ;offset of first rectangle pixel in
                        ; display memory
        mov     ax,SCREEN_SEG
        mov     es,ax   ;point ES:DI to the first rectangle pixel's
        mov     [bp+StartOffset],di ; address
        mov     dx,SC_INDEX ;set the Sequence Controller Index to
        mov     al,MAP_MASK ; point to the Map Mask register
        out     dx,al
        mov     bx,[bp+EndY]
        sub     bx,[bp+StartY]  ;BX = height of rectangle
        jle     FillDone        ;skip if 0 or negative height
        mov     [bp+Height],bx
        mov     dx,[bp+EndX]
        mov     cx,[bp+StartX]
        cmp     dx,cx
        jle     FillDone        ;skip if 0 or negative width
        dec     dx
        and     cx,not 011b
        sub     dx,cx
        shr     dx,1
        shr     dx,1
        inc     dx      ;# of addresses across rectangle to fill
        mov     [bp+Width],dx
        mov     word ptr [bp+PlaneInfo],0001h
                        ;lower byte = plane mask for plane 0,
                        ; upper byte = plane # for plane 0
FillPlanesLoop:
        mov     ax,word ptr [bp+PlaneInfo]
        mov     dx,SC_INDEX+1 ;point DX to the SC Data register
        out     dx,al   ;set the plane for this pixel
        mov     di,[bp+StartOffset] ;point ES:DI to rectangle start
        mov     dx,[bp+Width]
        mov     cl,byte ptr [bp+StartX]
        and     cl,011b ;plane # of first pixel in initial byte
        cmp     ah,cl   ;do we draw this plane in the initial byte?
        jae     InitAddrSet ;yes
        dec     dx      ;no, so skip the initial byte
        jz      FillLoopBottom ;skip this plane if no pixels in it
        inc     di
InitAddrSet:
        mov     cl,byte ptr [bp+EndX]
        dec     cl
        and     cl,011b ;plane # of last pixel in final byte
        cmp     ah,cl   ;do we draw this plane in the final byte?
        jbe     WidthSet ;yes
        dec     dx      ;no, so skip the final byte
        jz      FillLoopBottom ;skip this planes if no pixels in it
WidthSet:
        mov     si,SCREEN_WIDTH
        sub     si,dx   ;distance from end of one scan line to start
                        ; of next
        mov     bx,[bp+Height] ;# of lines to fill
        mov     al,byte ptr [bp+Color] ;color with which to fill
FillRowsLoop:
        mov     cx,dx   ;# of bytes across scan line
        rep     stosb   ;fill the scan line in this plane
        add     di,si   ;point to the start of the next scan
                        ; line of the rectangle
        dec     bx      ;count down scan lines
        jnz     FillRowsLoop
FillLoopBottom:
        mov     ax,word ptr [bp+PlaneInfo]
        shl     al,1    ;set the plane bit to the next plane
        inc     ah      ;increment the plane #
        mov     word ptr [bp+PlaneInfo],ax
        cmp     ah,4    ;have we done all planes?
        jnz     FillPlanesLoop ;continue if any more planes
FillDone:
        pop     di      ;restore caller's register variables
        pop     si
        mov     sp,bp   ;discard storage for local variables
        pop     bp      ;restore caller's stack frame
        ret
_FillRectangleX endp
        end
```

Конец

ПРОГРАММА 6

```
; Mode X (320x240, 256 colors) rectangle fill routine. Works on all
; VGAs. Uses fast approach that fans data out to up to four planes at
; once to draw up to four pixels at once. Fills up to but not
; including the column at EndX and the row at EndY. No clipping is
; performed.
; C near-callable as:
;     void FillRectangleX(int StartX, int StartY, int EndX, int EndY,
;         unsigned int PageBase, int Color);
```

ПРОГРАММА 6

```
SC_INDEX equ    03c4h   ;Sequence Controller Index
MAP_MASK equ    02h     ;index in SC of Map Mask register
SCREEN_SEG equ  0a000h  ;segment of display memory in mode X
SCREEN_WIDTH equ 80     ;width of screen in bytes from one scan line
                        ; to the next
parms   struc
        dw      2 dup (?) ;pushed BP and return address
StartX  dw      ?       ;X coordinate of upper left corner of rect
StartY  dw      ?       ;Y coordinate of upper left corner of rect
EndX    dw      ?       ;X coordinate of lower right corner of rect
                        ; (the row at EndX is not filled)
EndY    dw      ?       ;Y coordinate of lower right corner of rect
                        ; (the column at EndY is not filled)
PageBase dw     ?       ;base offset in display memory of page in
                        ; which to fill rectangle
Color   dw      ?       ;color in which to draw pixel
parms   ends

        .model  small
        .data
; Plane masks for clipping left and right edges of rectangle.
LeftClipPlaneMask       db      00fh,00eh,00ch,008h
RightClipPlaneMask      db      00fh,001h,003h,007h
        .code
        public  _FillRectangleX
_FillRectangleX proc    near
        push    bp      ;preserve caller's stack frame
        mov     bp,sp   ;point to local stack frame
        push    si      ;preserve caller's register variables
        push    di

        cld
        mov     ax,SCREEN_WIDTH
        mul     [bp+StartY] ;offset in page of top rectangle scan line
        mov     di,[bp+StartX]
        shr     di,1    ;X/4 = offset of first rectangle pixel in scan
        shr     di,1    ; line
        add     di,ax   ;offset of first rectangle pixel in page
        add     di,[bp+PageBase] ;offset of first rectangle pixel in
                        ; display memory
        mov     ax,SCREEN_SEG   ;point ES:DI to the first rectangle
        mov     es,ax           ; pixel's address
        mov     dx,SC_INDEX ;set the Sequence Controller Index to
        mov     al,MAP_MASK ; point to the Map Mask register
        out     dx,al
        inc     dx      ;point DX to the SC Data register
        mov     si,[bp+StartX]
        and     si,0003h                ;look up left edge plane mask

        mov     bh,LeftClipPlaneMask[si] ; to clip & put in BH
        mov     si,[bp+EndX]
        and     si,0003h                ;look up right edge plane
        mov     bl,RightClipPlaneMask[si] ; mask to clip & put in BL

        mov     cx,[bp+EndX]    ;calculate # of addresses across rect
        mov     si,[bp+StartX]
        cmp     cx,si
        jle     FillDone        ;skip if 0 or negative width
        dec     cx
        and     si,not 011b
        sub     cx,si
        shr     cx,1
        shr     cx,1    ;# of addresses across rectangle to fill - 1
        jnz     MasksSet ;there's more than one byte to draw
        and     bh,bl   ;there's only one byte, so combine the left
                        ; and right edge clip masks

MasksSet:
        mov     si,[bp+EndY]
        sub     si,[bp+StartY]  ;BX = height of rectangle
        jle     FillDone        ;skip if 0 or negative height
        mov     ah,byte ptr [bp+Color] ;color with which to fill
        mov     bp,SCREEN_WIDTH ;stack frame isn't needed any more
        sub     bp,cx   ;distance from end of one scan line to start
        dec     bp      ; of next
FillRowsLoop:
        push    cx      ;remember width in addresses - 1
        mov     al,bh   ;put left-edge clip mask in AL
        out     dx,al   ;set the left-edge plane (clip) mask
        mov     al,ah   ;put color in AL
        stosb           ;draw the left edge
        dec     cx      ;count off left edge byte
        js      FillLoopBottom ;that's the only byte
        jz      DoRightEdge ;there are only two bytes
        mov     al,00fh ;middle addresses are drawn 4 pixels at a pop
        out     dx,al   ;set the middle pixel mask to no clip
        mov     al,ah   ;put color in AL
        rep     stosb   ;draw the middle addresses four pixels apiece
DoRightEdge:
        mov     al,bl   ;put right-edge clip mask in AL
        out     dx,al   ;set the right-edge plane (clip) mask
        mov     al,ah   ;put color in AL
        stosb           ;draw the right edge
FillLoopBottom:
        add     di,bp   ;point to the start of the next scan line of
                        ; the rectangle
        pop     cx      ;retrieve width in addresses - 1
        dec     si      ;count down scan lines
        jnz     FillRowsLoop
FillDone:
        pop     di      ;restore caller's register variables
        pop     si
        pop     bp      ;restore caller's stack frame
        ret
_FillRectangleX endp
        end
```

Конец

ПРОГРАММА 7

```
/* Program to demonstrate mode X (320x240, 256-colors) rectangle
   fill by drawing adjacent 20x20 rectangles in successive colors from
   0 on up across and down the screen */
#include <conio.h>
#include <dos.h>

void Set320x240Mode(void);
void FillRectangleX(int, int, int, int, unsigned int, int);

void main() {
   int i,j;
   union REGS regset;

   Set320x240Mode();
   FillRectangleX(0,0,320,240,0,0); /* clear the screen to black */
   for (j = 1; j < 220; j += 21) {
      for (i = 1; i < 300; i += 21) {
         FillRectangleX(i, j, i+20, j+20, 0, ((j/21*15)+i/21) & 0xFF);
      }
   }
   getch();
   regset.x.ax = 0x0003;   /* switch back to text mode and done */
   int86(0x10, &regset, &regset);
}
```

Конец

факты

☐ Фирма Southern Peripherals выпустила плату видеоадаптера Tseng Labs Super VGA, обеспечивающую разрешение 1024x768 точек при выводе до 256 цветов из палитры 256000. На уровне регистров новый адаптер совместим с адаптерами CGA, EGA, MDA, VGA и Hercules. Фирма Southern Peripherals предоставляет драйверы для работы адаптера с такими пакетами, как Lotus 1-2-3, Symphony, AutoCAD, Autoshade, Windows, Ventura Publisher, GEM, WordPerfect.

☐ Фирма Cirrus Logic разработала микросхему VGA-контроллера CL-GD6410, предназначенную специально для портативных компьютеров. Эта микросхема - самый миниатюрный и потребляющий минимальную мощность VGA-контроллер из имеющихся на рынке. Новый контроллер может управлять одновременно и жидкокристаллическим монитором, и монитором на электронно-лучевой трубке.

☐ Разработав пакеты Genesis, Circuit Analysis и Circuit Synthesis, фирма MicroSim (США) обеспечила интегрированную среду для проектирования электронных схем, их моделирования и анализа. Пакеты фирмы MicroSim могут работать на компьютерах семейств IBM, Macintosh, VAX, на рабочих станциях фирмы Sun. В зависимости от того, для какой операционной среды предназначены пакеты, их суммарная цена колеблется от 3300 до 36 750 дол.

☐ Поступил в продажу новый автономный телефакс фирмы Hewlett-Packard - Fax-300. В устройстве используется струйная технология, которая обеспечивает получение разрешающей способности 300 точек (12 градаций серого)/дюйм при использовании листовой бумаги. Одна страница текста передается в течение 12 с. В резидентной памяти устройства одновременно могут храниться 28 документов. Цена телефакса 2095 дол.

Научно-технический центр **МОДУЛЬ**

101000 Москва, А/Я 83

Оцифровщик изображений "ПОЛИФЕМ-2"

Предназначен для приема изображений с телекамеры любого стандарта. Может применяться в системах проведения точных измерений на изображениях. Прибор поставляется с базовым комплектом программного обеспечения. Отличается от имеющихся отечественных аналогов низким уровнем шумов, высокой скоростью приема изображений, гибким программным управлением режимами работы. Может быть установлен на любой IBM PC/AT-совместимой ЭВМ.

Основные характеристики:

- тип исполнения: встраиваемая плата;
- размер принимаемого изображения: от 256×256 до 640×512 точек;
- количество градаций яркости: от 2 до 256;
- время фиксации кадра: 40 мсек;
- программное управление яркостью и контрастностью на уровне входного сигнала, встроенная таблица преобразований;
- коррекция геометрических искажений для различных размеров изображения;
- внутренняя и внешняя синхронизация;
- отношение сигнал/шум не хуже 56 дБ.

АРМ "БРОКЕР"

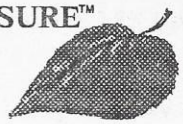
Автоматизированная система биржевой деятельности для брокерских и посреднических контор и фирм

это:

- классификация товаров на базе международной гармонизированной системы описания и кодирования товаров;
- стандартизированные классификаторы единиц измерений количества товаров, типов валют, регионов, условий поставки, характеров сделок и др.;
- банки данных покупателей и продавцов товаров и услуг с реквизитами;
- банки данных предложений на продажу и спроса на покупку товаров;
- быстрое нахождение предложений и спроса по классификатору товаров, по ключевым словам , и по многим другим критериям;
- поиск предложения по спросу и наоборот;
- калькулятор и дневник/календарь;
- система автовызова клиента и передача информации через модемную связь.

А самое главное все это работает в локальной вычислительной сети и обеспечивает одновременный доступ нескольких брокеров к одним и тем же данным.

MEASURE™ Система анализа изображений



Система предназначена для денситометрического и морфологического анализа изображений с 256-ю градациями яркости. Может применяться в медико-биологических исследованиях, материаловедении, геофизике и в других областях, где необходимо проводить измерения на изображениях. Используется совместно с прибором "Полифем-2". Возможно использование с другими приборами, позволяющими создавать файлы изображений в формате Graphic Interchange Format (GIF).

Система "MEASURE" является интегрированной средой для ввода, хранения, проведения измерений и редактирования изображений.

Основные характеристики:

- измерения яркости/плотности по различным областям: точка, сечение, овал, произвольный контур;
- селекция морфологических объектов по группе признаков: площадь, периметр, диапазон яркости;
- калибровка измерений по стандартным образцам;
- полное протоколирование измерений в специализированной базе данных с возможностью повторения измерений и получения отчетов в текстовом виде;
- ведение библиотеки морфологических структур;
- встроенный графический редактор с пошаговым откатом изменений.

Пакет "FILE TRANSFER"

Предназначен для организации прямой связи между рабочими станциями в локальных сетях фирмы Novell. Возможно применение в качестве самостоятельного продукта для связи машин через сетевые адаптеры.

Основные особенности:

- простота и удобство в работе (меню, назначение функциональных клавиш соответствует принятому в программе Norton Commander, возможность пересылки вложенных каталогов);
- высокая скорость;
- возможность одновременной работы для множества пар рабочих станций.

Контактные телефоны в Москве: 152-94-81
152-46-31
152-53-44

О "мышках" и сообщениях

Э. Стивенз

В этой и последующих статьях я рассмотрю несколько вопросов. Основное внимание будет уделено событийно-управляемому программированию *(event-driven (ED) programming)*. Мы познакомимся с двумя аспектами организации ED-программ: во-первых, с тем, как работает диспетчер событий и сообщений, и, во-вторых, с тем, как прикладная программа должна использовать систему сообщений для управления своим выполнением.

В типичной ED-программе события сводятся к нажатию клавиши, перемещению "мыши" или к какому-либо другому внешнему воздействию, которое заставляет программу выполнять определенные действия.

Для иллюстрации воспользуемся программой "вырезания" части изображения на экране в текстовом режиме. Программа запускается из командной строки DOS, и при наличии соответствующего драйвера она может остаться резидентной после завершения (TSR - Terminate and Stay Resident).

Для задания на экране прямоугольника, который программа может записать в виде текстового файла, используется клавиатура или "мышь". Модификацию этой программы я применяю при подготовке документации для вывода на принтер "снимков" *(snapshots)* содержимого экрана.

Хотя большая часть программы приходится на работу с аппаратными средствами "мыши" и на то, чтобы программа была резидентной, основное назначение программы - продемонстрировать, как писать ED-программы на языке С.

Если Вы не хотите возиться с превращением программы в резидентную, можно вызывать ее всякий раз из командной строки. Разумеется, чтобы эту программу копирования содержимого экрана можно было использовать не только в учебных целях, ее нужно сделать резидентной.

© Dr. Dobb's Journal, March 1991.
All rights reserved. M&T Publishing, Redwood City, CA, USA.

Драйверы аппаратуры я писал на Turbo C версии 2.0 и использовал псевдорегистры и другие средства, которые обеспечиваются в этом диалекте языка. Если же Вы хотите использовать другой компилятор, Вам придется модифицировать программу в соответствии с соглашениями этого компилятора относительно управления прерываниями, регистров и т.п., поскольку никакие два различных компилятора не работают совершенно одинаково.

Хотя прикладная часть программы написана на стандартном языке С, для драйверов устройств использованы расширения Turbo C.

События и сообщения

Как же работает ED-программа? Сначала рассмотрим, как можно было бы написать подобную программу, используя традиционный процедурный подход. После окончания инициализации такой программы Вам пришлось бы начать опрос клавиатуры и "мыши". Как только с одним из этих устройств что-либо произойдет, следует ввести информацию с этого устройства и определить, имеет ли она отношение к тому, что нужно Вашей программе, и, если это так, выполнить соответствующее действие и снова вернуться в режим опроса устройств. Если одно из действий пользователя указывает на то, что программа отработала, Вы не возвращаетесь в цикл опроса, а производите завершающую процедуру и выходите из программы. Опрос устройств, считывание с них входных данных и их интерпретация - вот составные части программы.

ED-программа делает все то же самое, но несколько иным способом. Вместо опроса устройств и ответных действий на основе входных данных ED-программа использует функцию диспетчеризации событий, которая при наступлении события производит вызов Ваших программ. Диспетчер посылает сообщение, которое затем интерпретируется Вашей программой. Например, вместо чтения с клавиатуры Ваша программа будет ждать, пока диспетчер не пошлет сообщения, говорящего о том, что пользователь нажал клавишу.

Программное обеспечение, отслеживающее события, будет вместо Вас следить за аппаратурой и накапливать "аппаратные события" в виде очереди сообщений. Диспетчер выбирает сообщения из очереди и рассылает их Вашим программам. Было бы естественно предположить, что Ваши программы могут также помещать сообщения в очередь, а эти сообщения, в свою очередь, будут переданы Вашим прикладным программам через диспетчер.

В чем смысл этого подхода? Почему при этом можно ожидать каких-то преимуществ по сравнению со старым способом?

Во-первых, ED-программа, по-видимому, будет лучше организована. Я понимаю, что Вы это слышали и раньше. То же самое говорится обычно о всякой новой модели поведения, которую кто-то пытается ввести в обиход. Это тот самый случай, когда Вы должны выбрать сами. Преимущества организации ED-программ не были очевидны для меня до тех пор, пока я не переписал по этому образцу обычную программу. При этом многое упростилось. Данный подход не снимает всех проблем программирования, но в ряде случаев он облегчает разработку и написание программы, поскольку оказывает благотворное воздействие на ее структуру. А чем больше структурированность, тем лучше.

Во-вторых, если Вы уже программировали в среде Windows, то подход к ED-программированию покажется Вам очень удобным. В среде Windows Вы создаете окно и связанную с ним собственную функцию обработки. Диспетчер окон посылает сообщения функциям обработки окон, как только происходит событие, которое может оказаться важным для какого-то из окон. Например, когда Вы перемещаете "мышь" внутри окна, диспетчер посылает в окно соответствующее сообщение. Когда Вы нажимаете клавишу клавиатуры, диспет-

чер посылает сообщение активному в этот момент окну.

Подход к управлению через внешние события особенно хорош для программирования в среде Windows. Например, выбрать команду из меню в среде Windows пользователь может вручную одним из нескольких возможных способов: с помощью "мыши"; в результате нажатия "горячей" клавиши; в результате открытия меню и нажатия соответствующей клавиши, наконец, с помощью перемещения курсора по меню к требуемой команде и нажатия клавиши Enter. Однако Вашей прикладной программе до этого нет никакого дела. Об этом позаботится администратор сообщений. Если пользователь выбрал команду, то Ваша программа получит об этом итоговое сообщение, одно и то же сообщение, независимо от того, каким способом пользователь осуществил свой выбор. Сообщение укажет Вашей программе, какую команду выбрал пользователь. Более того, в другом месте Вашей программы можно создать имитацию такого же выбора команды, всего лишь послав это же сообщение.

Драйверы аппаратуры

Для построения ED-программы нам потребуется, прежде всего, определенное программное обеспечение, позволяющее распознавать события (чтобы их можно было превратить в сообщения). Программой копирования содержимого экрана должны управлять клавиатура и "мышь". Хотелось бы, чтобы прикладная программа не зависела от аппаратуры, а всем этим ведала программа-администратор событий. Хотя большинство программных сред, обеспечивающих управление через события, поставляются уже в комплекте с драйверами аппаратуры, нам, тем не менее, потребуется написать свои собственные. Помните, однако, что главными здесь являются само событие и код сообщения, поступающий вслед за этим.

Клавиатура и курсор

В программах 1 и 2 (программы см. на с.*52-55*) представлены функции keys.h и getkey.c, которые управляют клавиатурой и курсором на экране. Подобные функции могли Вам встречаться и в других программах. Файл keys.h определяет некоторые значения клавиш, которые будут использоваться программой. В ней также содержатся прототипы функций управления клавиатурой и кусором. В файле getkey.c содержатся две функции управления клавиатурой и несколько функций управления курсором. Функция keybit возвращает значение ИСТИНА, если была нажата клавиша. Функция getkey считывает код клавиши клавиатуры и преобразует функциональные ключи в значения от 128 и выше. Описания клавиш в файле keys.h соответствуют этому. Функции управления курсором обеспечивают программе следующие возможности: задавать позицию курсора, считывать его текущую позицию, прятать курсор и возвращать его на экран, а также сохранять и восстанавливать атрибуты курсора для программ, выполнение которых было прервано.

"Мышь"

В программах 3 и 4 представлены драйверы mouse.h и mouse.c для манипулятора "мышь" персонального компьютера. В файле mouse.h определены прототипы функций и ряд макросов. В файле mouse.c содержатся функции, которые позволяют программе определять наличие "мыши", считывать или задавать координаты курсора "мыши" на экране, реагировать на нажатие кнопки "мыши", включать и выключать курсор, а также запоминать и восстанавливать атрибуты курсора для программ, прерванных TSR-программой. Имеются и другие операции с "мышью", которые обеспечиваются этими функциями. Я остановился только на тех, которые нужны в нашей программе. В справочном руководстве "Microsoft Mouse Programmer's Reference", выпущенном агентством Microsoft Press, описано, как осуществляются все операции с "мышью".

Драйвер сообщений

В программах *5* и 6 представлены программные средства драйвера сообщений: файлы message.h и message.c. В файле message.h описываются коды сообщений. В данном примере предусмотрено только 16 сообщений. Мы на них еще остановимся впоследствии. В прикладных программах можно добавить к этому списку дополнительные сообщения.

Имеется всего три функции, к которым должна обращаться программа для организации управления через события с помощью драйвера сообщений. Функция dispatch_message представляет собой модуль диспетчеризации сообщений. Вы сообщаете ей адрес Вашей программы обработки сообщения, которая представляет собой функцию типа void, рассчитанную на получение трех параметров типа "целое". Первым параметром должен быть код сообщения, а два других являются родовыми параметрами, которые могут использоваться в сообщениях произвольным образом.

Для внесения сообщений в очередь сообщений используется функция post_message. Диспетчер будет посылать сообщения в той очередности, в которой они расположены в очереди. Сообщения, которые регистрируются функцией post_message, не будут посланы немедленно. Чтобы сообщение было послано немедленно, нужно обратиться к функции send_message. Эту функцию следует использовать в том случае, когда сообщение имеет возвращаемое значение или же когда оно должно вызвать немедленные действия.

Сообщения

Сообщение START посылается диспетчером Вашей программе в начале работы, а Ваша программа посылает сообщение STOP, чтобы полномочная программа завершила цикл диспетчеризации сообщений.

В этом простом примере управления через события обработка сообщения происходит сразу же и единственным образом. В более сложных системах, подобных среде Windows, сообщений будет намного больше, имеются разные категории сообщений, и их можно посылать различным функциям обработки. Например, на экране может быть несколько окон, и каждое из них может иметь свои собственные сообщения, предписывающие начать и закончить работу.

Диспетчер посылает сообщение KEYBOARD в том случае, когда пользователь нажал клавишу. Первый из двух параметров будет содержать значение, соответствующее нажатой клавише.

Сообщения CURRENT_KEYBOARD_CURSOR и KEYBOARD_CURSOR посылаются прикладной программой. Сообщение CURRENT_KEYBOARD_CURSOR возвращает текущие координаты курсора в виде двух параметров. Первый параметр представляет собой адрес для координаты X, второй параметр - адрес для координаты Y. Сообщение KEYBOARD_CURSOR изменяет позицию курсора. Первый параметр - это новая координата X, а второй параметр - новая координата Y.

Диспечтер посылает сообщения RIHGT_BUTTON или LEFT_BUTTON, когда Вы нажимаете правую или левую кнопку "мыши" соответственно. Пока Вы удерживаете кнопку нажатой, диспетчер будет продолжать посылать сообщения. Диспетчер посылает сообщение MOUSE_MOVED, когда Вы двигаете "мышь", независимо от того, была нажата кнопка или нет. Когда Вы отпускаете кнопку, диспетчер посылает сообщение BUTTON_RELEASED. В этих четырех сообщениях, связанных с "мышью", первый параметр будет содержать координату столбца на экране (X), а второй параметр - координату строки (Y), в которой находился курсор

в момент наступления соответствующего события.

Сообщения CURRENT_MOUSE_CURSOR, MOUSE_CURSOR, SHOW_MOUSE и HIDE_MOUSE посылаются прикладной программой. Сообщение CURRENT_MOUSE_CURSOR возвращает текущие экранные координаты курсора "мыши" в виде двух параметров. Первый параметр - это адрес памяти, по которому должно быть помещено значение координаты X, второй параметр - адрес координаты Y. Сообщение MOUSE_CURSOR вызывает изменение положения курсора "мыши". Первый параметр - новая координата X, второй параметр - новая координата Y.

Сообщение HIDE_MOUSE приводит к тому, что курсор "мыши" убирается с экрана, а сообщение SHOW_MOUSE снова выводит его на экран. В этих двух сообщениях оба параметра равны нулю.

Имеются и другие сообщения, которые может послать драйвер "мыши". Например, может оказаться полезной возможность определять координаты курсора в результате двойного нажатия кнопки. Драйвер клавиатуры может передавать через второй параметр статус клавиш Shift, Alt и Ctrl. Такой ограниченный набор сообщений и событий в данном примере выбран только для того, чтобы продемонстрировать сам принцип, но в реальных системах с управлением через события в Вашем распоряжении их будет намного больше.

Из прикладной программы можно послать сообщение VIDEO_CHAR, чтобы получить символ на экране. Параметрами являются координаты X и Y, а сообщение возвращает соответствующий этим координатам символ.

В файле message.h определяется структура RECT, которая содержит координаты X/Y верхнего левого и нижнего правого углов экрана. С помощью сообщений GET_VIDEORECT и PUT_VIDEORECT осуществляется пересылка данных между экраном и буфером Вашей программы. Первый параметр - это адрес структуры RECT, которая описывает координаты прямоугольника, второй параметр - адрес буфера.

Отметим, что тип данных PARAM, определенный в файле message.h для параметров сообщений, представляет собой тип "целое". Если в сообщении подразумеваются адреса, Вы должны привести их к типу PARAM. Если же Вы работаете с большими числами, Вам следует изменить определение типа PARAM на "длинное целое".

Очередь сообщений

Функции файла message.c формируют циклическую очередь сообщений. Функция post_message добавляет в эту очередь новую запись. Записи в очереди состоят из кода сообщения и двух параметров. Функция collect_message опрашивает аппаратуру о появлении событий и помещает сообщения в очередь. Эта функция также распознает случай, когда она выполняется в первый раз, и помещает в очередь сообщение START. Функция collect_message возвращает значение ИСТИНА, если очередь существует и в очереди имеются сообщения. Она возвращает значение ЛОЖЬ, если очередь пуста.

Функция dispatch_message предназначена для вызова прикладных программ в процессе диспетчеризации сообщений из очереди. Она вызывает функцию collect_message, чтобы поставить в очередь все незарегистрированные события и чтобы проверить, есть ли в очереди сообщения, которые можно послать прикладным программам. Если сообщение стоит в начале очереди, функция dispatch_message извлекает его из очереди и вызывает функцию send_message для посылки сообщения прикладной функции обработки сообщения. Функция send_message к тому же выполняет определенные действия для сообщений типа MOUSE_CURSOR, которые посылаются прикладными программами для управления аппаратурой. Заметим, что, когда функция send_message вызывает прикладную функцию обработки сообщений, она рассматривает сообщение только в том случае, если код возврата функции обработки сообщения имеет значение ИСТИНА. Это позволяет прикладной программе избегать стандартной процедуры обработки сообщений. В нашем примере эта возможность ни разу не используется, но она типична для систем с управлением через события. Функция send_message может возвращать значение вызвавшей ее программе. Например, для сообщения VIDEO_CHAR она возвращает значение символа на экране.

Функция dispatch_message возвращает вызывающей программе значение ИСТИНА независимо от того, было или не было найдено сообщение для посылки. Однако, если она определит, что ею была произведена диспетчеризация сообщения STOP, то вместо этого она возвратит значение ЛОЖЬ. Поэтому прикладная программа должна последовательно вызывать функцию dispatch_message до тех пор, пока будет возвращаться значение ИСТИНА. Программа должна завершать сеанс в том случае, когда функция dispatch_message возвращает значение ЛОЖЬ.

Функция mouse_event вызывает функции работы с "мышью" для формирования сообщений о событиях, связанных с манипулятором "мышь". Она вызывается функцией collect_message. Функция mouse_event считывает позицию "мыши" и фиксирует координаты X и Y в виде глобальных переменных. Функция collect_message использует эти переменные при формировании параметров сообщений, помещаемых в очередь. Если с момента последнего вызова функции mouse_event позиция "мыши" изменилась, то функция возвращает сообщение MOUSE_MOVED. Если после последней проверки Вы отпустили кнопку "мыши", то функция вернет сообщение MOUSE_RELEASED. Если правая или левая кнопка "мыши" нажата, то функция возвращает сообщение RIGHT_BUTTON или LEFT_BUTTON соответственно.

Программа копирования содержимого экрана

Как видите, сообщения и функции обработки событий невелики по объему. Для реализации простой среды, обеспечивающей управление через события, не потребуется большого количества программных кодов. Эффективность подхода зависит от того, как Вы сумеете им воспользоваться. ED-программа 7, представляющая собой файл copyscrn.c, иллюстрирует использование программных средств администратора событий и сообщений.

Программа начинает с создания на диске текстового файла grab.dat. После этого она производит циклические вызовы функции dispatch_message с передачей адреса функции message_proc. Эти вызовы продолжаются до тех пор, пока функция dispatch_message не вернет значение ЛОЖЬ; при этом функция copyscrn закрывает текстовый файл и возвращает управление.

Теперь посмотрим на эту программу, как на систему, управляемую через события. В системе может что-то произойти только в том случае, если пользователь сделает что-то с клавиатурой или "мышью". Эти события будут трансформированы в сообщения, которые принимает и обрабатывает внутрипрограммная функция message_proc.

Сообщение START

Если вернуться к файлу message.c, то Вы увидите, что, когда программа copyscrn вызывает функцию dispatch_message в первый раз, функция collect_message помещает в очередь сообщение START, которое будет послано функции message_proc. Сообщение START используется ею, чтобы приступить к обработке. При этом она посылает сообщения, которые приводят к запоминанию позиций курсора клавиатуры и

"мыши", а также к переводу обоих курсоров в верхний левый угол экрана.

Сообщения клавиатуры

Теперь программа выполняется, причем курсоры клавиатуры и "мыши" расположены в верхнем левом углу экрана, а текстовый файл открыт. В это время в прикладной программе ничего не происходит, поскольку пользователь не совершает никаких действий. Диспетчер работает, следя за клавиатурой и "мышью".

Предположим теперь, что пользователь нажал клавишу клавиатуры. Снова обратимся к файлу message.c. Функция collect_message вызывает функцию драйвера клавиатуры keybit, чтобы определтить, была ли нажата какая-нибудь клавиша. Если функция keybit возвращает значение ИСТИНА, то функция collect_message помещает в очередь сообщение KEYBOARD, а значение, возвращенное функцией getkey, заносит в качестве первого параметра. Поскольку выполняется цикл диспетчеризации сообщений, то в конце концов функция send_message перешлет это сообщение функции message_proc в исходный файл copyscrn.c. Функция message_proc обрабатывает сообщение KEYBOARD, выполняя различные действия в зависимости от того, какая клавиша была нажата. Она отслеживает нажатие клавиш "Вверх", "Вниз", "Вправо" и "Влево", а также клавиш Esc, Enter и F2. Нажатие всех остальных клавиш игнорируется.

Клавиши "Вверх", "Вниз", "Вправо" и "Влево" можно использовать для перемещения курсора по всему экрану. Если нажать клавишу F2, то Вы дадите знать программе о том, что хотите приступить к описанию прямоугольника на экране. Позиция курсора в момент нажатия клавиши F2 будет соответствовать одному из углов прямоугольника. Если после этого Вы будете перемещать курсор, то программа будет обозначать прямоугольник, изменяя цвета символов внутри прямоугольника. Вторичным нажатием клавиши F2 Вы сообщите программе, что Вас такой прямоугольник не устраивает. Изображение прямоугольника будет убрано, и Вы сможете выбрать новую начальную точку и нажать клавишу F2. Когда подходящий прямоугольник будет определен, нужно нажать клавишу Enter. Чтобы аннулировать весь сеанс, нужно нажать клавишу Esc.

Функции forward, upward, backward и downward управляют положением курсора. Функция highlight рисует и перерисовывает прямоугольник в соответствии с перемещением курсора. Функция setstart включает и отключает маркировку на экране.

Сообщение STOP

Рассмотрим, как функция message_proc обрабатывает нажатие клавиш Enter ('\r') и Esc. Вспомним, что их нажатие вызывает завершение программы, которое либо сопровождается записью параметров прямоугольника на диск, либо аннулирует его построение. При поступлении сообщений о нажатии этих клавиш функция message_proc вызывает функцию post_message для помещения в очередь сообщения STOP. Первый параметр будет иметь значение ИСТИНА в случае нажатия клавиши Enter и значение ЛОЖЬ в случае нажатия клавиши Esc. Это сообщение, как и все остальные, в конечном итоге также будет передано функции message_proc. Теперь посмотрим, как функция message_proc обрабатывает сообщение STOP. Если блок выделен, то функция вызывает функцию highlight для отмены выделения блока. Если первым параметром является ИСТИНА, то функция вызывает функцию writescreen для записи блока в виде файла на диске. В любом случае функция посылает сообщение для восстановления позиций курсоров "мыши" и клавиатуры, которые они занимали к моменту начала работы программы. Вспомним, что функция dispatch_message использует сообщение STOP как признак того, что нужно возвратить значение ЛОЖЬ вызвавшей ее функции с целью прекращения выполнения программы, но она сделает это только после того, как Ваша программа обработки сообщений завершит работу на сообщении STOP.

Сообщения "мыши"

Работа с помощью "мыши" аналогична работе с помощью клавиатуры. Вы можете перемещать "мышь", пока ее курсор не окажется в одном из углов (произвольном) прямоугольника, который Вы хотели бы "вырезать". Нажмите левую кнопку "мыши" и удерживайте ее нажатой, пока перемещаете курсор, - программа будет отслеживать перемещения. Отпустите кнопку, и задание прямоугольника будет окончено.

Если Вам не понравился полученный прямоугольник, "переместитесь" в другой угол и снова нажмите левую кнопку. Для записи параметров полученного прямоугольника на диск нажмите клавишу Enter или правую кнопку "мыши". Нажатие клавиши Esc аннулирует прямоугольник и завершает работу программы.

Когда Вы нажимаете левую кнопку "мыши", функция message_proc получает сообщение LEFT_BUTTON. Это сообщение будет продолжать поступать, пока Вы удерживаете кнопку нажатой, так что его нужно обрабатывать только в первый раз. Если программа не производит в настоящий момент маркировку блока с помощью "мыши", то это сообщение заставляет программу приступить к работе: запомнить координаты и вызвать функцию setstart.

Функция message_proc получает сообщение MOUSE_MOVED при любом перемещении "мыши". Если в этот момент не происходит маркировки прямоугольника, то данное сообщение игнорируется. Если же маркировка происходит, то функция message_proc вызывает одну из тех же самых функций (forward, upward, backward или downward), которые используются в процессе задания прямоугольника после поступления сообщения KEYBOARD.

Когда поступает сообщение BUTTON_RELEASED, функция message_proc регистрирует окончание маркировки блока с помощью "мыши". Когда поступает сообщение RIGHT_BUTTON, функция производит те же действия, что и в случае поступления сообщения KEYBOARD от клавиши Enter: отправляет в очередь сообщение STOP с первым параметром, имеющим значение ИСТИНА, т.е. такое сообщение STOP, которое вызывает запись параметров прямоугольника с экрана на диск.

Программа позволяет задать прямоугольник с помощью "мыши", а затем нажать клавишу F2 и задать другой прямоугольник с помощью клавиатуры. Она позволяет также в процессе задания прямоугольника с клавиатуры нажать кнопку "мыши", чтобы отменить этот прямоугольник и начать задание нового с помощью "мыши".

Заключение

Структура рассмотренной ED-программы типична для большинства систем с управлением через события. В ней задействовано большинство необходимых компонент, хотя и в меньшем объеме. Для того чтобы использовать управление через события, совсем не обязательно, чтобы эти события инициировались аппаратными средствами. Можно использовать обычные функции ввода-вывода и организовывать "программные события", которые управляли бы процессом выполнения программы. И все же это будет уже другая технология, которая, во-первых, заставляет по-иному взглянуть на программирование и, во-вторых, по-видимому, обещает упорядочить сложные вопросы программирования.

ПРОГРАММА 1

```
/* ----------- keys.h ------------ */

#define TRUE  1
#define FALSE 0

#define ESC      27
#define F2      188
#define UP      200
#define FWD     205
#define DN      208
#define BS      203

int getkey(void);
int keyhit(void);
void curr_cursor(int *, int *);
void cursor(int, int);
void hidecursor(void);
void unhidecursor(void);
void savecursor(void);
void restorecursor(void);
void set_cursor_type(unsigned);
#define normalcursor() set_cursor_type(0x0607)
```

Конец

ПРОГРАММА 2

```
/* ----------- getkey.c ---------- */

#include <bios.h>
#include <dos.h>
#include "keys.h"

#define KEYBOARD 0x16
#define ZEROFLAG 0x40
#define SETCURSORTYPE 1
#define SETCURSOR     2
#define READCURSOR    3
#define HIDECURSOR 0x20

static unsigned video_mode;
static unsigned video_page;
static int cursorpos;
static int cursorshape;

/* ---- Test for keystroke ---- */
int keyhit(void)
{
    _AH = 1;
    geninterrupt(KEYBOARD);
    return (_FLAGS & ZEROFLAG) == 0;
}

/* ---- Read a keystroke ---- */
int getkey(void)
{
    int c;
    while (keyhit() == 0)
        ;
    if (((c = bioskey(0)) & 0xff) == 0)
        c = (c >> 8) | 0x80;
    else
        c &= 0xff;
    return c;
}

static void videoint(void)
{
    static unsigned oldbp;
    _DI = _DI;
    oldbp = _BP;
    geninterrupt(0x10);
    _BP = oldbp;
}

void videomode(void)
{
    _AH = 15;
    videoint();
    video_mode = _AL;
    video_page = _BX;
    video_page &= 0xff00;
    video_mode &= 0x7f;
}

/* ---- Position the cursor ---- */
void cursor(int x, int y)
{
    videomode();
    _DX = ((y << 8) & 0xff00) + x;
    _AX = 0x0200;
    _BX = video_page;
    videoint();
}

/* ---- get cursor shape and position ---- */
static void near getcursor(void)
{
    videomode();
    _AH = READCURSOR;
    _BX = video_page;
    videoint();
}

/* ---- Get current cursor position ---- */
void curr_cursor(int *x, int *y)
{
    getcursor();
    *x = _DL;
    *y = _DH;
}

/* ---- Hide the cursor ---- */
void hidecursor(void)
{
    getcursor();
    _CH |= HIDECURSOR;
    _AH = SETCURSORTYPE;
    videoint();
}

/* ---- Unhide the cursor ---- */
void unhidecursor(void)
{
    getcursor();
    _CH &= ~HIDECURSOR;
    _AH = SETCURSORTYPE;
    videoint();
}

/* ---- Save the current cursor configuration ---- */
void savecursor(void)
{
    getcursor();
    cursorshape = _CX;
    cursorpos = _DX;
}

/* ---- Restore the saved cursor configuration ---- */
void restorecursor(void)
{
    videomode();
    _DX = cursorpos;
    _AH = SETCURSOR;
    _BX = video_page;
    videoint();
    set_cursor_type(cursorshape);
}

/* ----------- set the cursor type ------------- */
void set_cursor_type(unsigned t)
{
    videomode();
    _AH = SETCURSORTYPE;
    _BX = video_page;
    _CX = t;
    videoint();
}
```

Конец

ПРОГРАММА 3

```
/* ------------ mouse.h ------------- */

#ifndef MOUSE_H
#define MOUSE_H

#define MOUSE 0x33

int mouse_installed(void);
int mousebuttons(void);
void get_mouseposition(int *x, int *y);
void set_mouseposition(int x, int y);
void show_mousecursor(void);
void hide_mousecursor(void);
int button_releases(void);
void intercept_mouse(void);
void restore_mouse(void);
void resetmouse(void);

#define leftbutton()  (mousebuttons()&1)
#define rightbutton()  (mousebuttons()&2)
#define waitformouse() while(mousebuttons());

#endif
```

Конец

ПРОГРАММА 4

```
/* ------------ mouse.c ------------ */

#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <string.h>
#include "mouse.h"

static void mouse(int m1,int m2,int m3,int m4)
{
    _DX = m4;
    _CX = m3;
    _BX = m2;
    _AX = m1;
    geninterrupt(MOUSE);
}

/* ---------- reset the mouse ---------- */
void reset_mouse(void)
{
    mouse(0,0,0,0);
}

/* ----- test to see if the mouse driver is installed ----- */
int mouse_installed(void)
{
    unsigned char far *ms;
    ms = MK_FP(peek(0, MOUSE*4+2), peek(0, MOUSE*4));
    return (ms != NULL && *ms != 0xcf);
}
```

```
/* ------ return true if mouse buttons are pressed ------- */
int mousebuttons(void)
{
    int bx = 0;
    if (mouse_installed())    {
        mouse(3,0,0,0);
        bx = _BX;
    }
    return bx & 3;
}

/* ---------- return mouse coordinates ---------- */
void get_mouseposition(int *x, int *y)
{
    if (mouse_installed())    {
        int mx, my;
        mouse(3,0,0,0);
        mx = _CX;
        my = _DX;
        *x = mx/8;
        *y = my/8;
    }
}

/* -------- position the mouse cursor -------- */
void set_mouseposition(int x, int y)
{
    if(mouse_installed())
        mouse(4,0,x*8,y*8);
}

/* --------- display the mouse cursor -------- */
void show_mousecursor(void)
{
    if(mouse_installed())
        mouse(1,0,0,0);
}

/* --------- hide the mouse cursor ------- */
void hide_mousecursor(void)
{
    if(mouse_installed())
        mouse(2,0,0,0);
}

/* --- return true if a mouse button has been released --- */
int button_releases(void)
{
    int ct = 0;
    if(mouse_installed())    {
        mouse(6,0,0,0);
        ct = _BX;
    }
    return ct;
}

static int mx, my;

/* ----- intercept the mouse in case an interrupted program is using it ---- */
void intercept_mouse(void)
{
    if (mouse_installed())    {
        _AX = 3;
        geninterrupt(MOUSE);
        mx = _CX;
        my = _DX;
        _AX = 31;
        geninterrupt(MOUSE);
    }
}

/* ----- restore the mouse to the interrupted program ----- */
void restore_mouse(void)
{
    if (mouse_installed())    {
        _AX = 32;
        geninterrupt(MOUSE);
        _CX = mx;
        _DX = my;
        _AX = 4;
        geninterrupt(MOUSE);
    }
}
```

Конец

ПРОГРАММА 5

```
/* ---------- message.h ------------ */

#ifndef MESSAGES_H
#define MESSGAES_H

#define MAXMESSAGES 50

/* --------- event message codes ----------- */
enum messages {
    START,
    STOP,
    KEYBOARD,
    RIGHT_BUTTON,
    LEFT_BUTTON,
    MOUSE_MOVED,
    BUTTON_RELEASED,
    CURRENT_MOUSE_CURSOR,
    MOUSE_CURSOR,
    SHOW_MOUSE,
    HIDE_MOUSE,
    KEYBOARD_CURSOR,
    CURRENT_KEYBOARD_CURSOR,
    VIDEO_CHAR,

    PUT_VIDEORECT,
    GET_VIDEORECT
};

/* ------- defines a screen rectangle ------ */
typedef struct {
    int x, y, x1, y1;
} RECT;

/* ------ integer type for message parameters ----- */
typedef int PARAM;

void post_message(enum messages, PARAM, PARAM);
int send_message(enum messages, PARAM, PARAM);
int dispatch_message(int (*)(enum messages, PARAM, PARAM));
RECT rect(int, int, int, int);

#endif
```

Конец

ПРОГРАММА 6

```
/* --------- message.c---------- */

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include "mouse.h"
#include "keys.h"
#include "message.h"

static int mouse_event(void);

static int px = -1, py = -1;
static int mx, my;
static int first_dispatch = TRUE;

static struct msgs {
    enum messages msg;
    PARAM p1;
    PARAM p2;
} msg_queue[MAXMESSAGES];

static int qonctr;
static int qoffctr;
static int (*mproc)(enum messages,int,int);

/* ----- post a message and parameters to msg queue ---- */
void post_message(enum messages msg, PARAM p1, PARAM p2)
{
    msg_queue[qonctr].msg = msg;
    msg_queue[qonctr].p1 = p1;
    msg_queue[qonctr].p2 = p2;
    if (++qonctr == MAXMESSAGES)
        qonctr = 0;
}

/* --------- clear the message queue --------- */
static void clear_queue(void)
{
    px = py = -1;
    mx = my = 0;
    first_dispatch = TRUE;
    qonctr = qoffctr = 0;
}

/* ------ collect events into the message queue ------ */
static int collect_messages(void)
{
    /* ---- collect any unqueued messages ---- */
    enum messages event;
    if (first_dispatch)    {
        first_dispatch = FALSE;
        reset_mouse();
        show_mousecursor();
        send_message(START,0,0);
    }
    if ((event = mouse_event()) != 0)
        post_message(event, mx, my);
    if (keyhit())
        post_message(KEYBOARD, getkey(), 0);
    return qoffctr != qonctr;
}

int send_message(enum messages msg, PARAM p1, PARAM p2)
{
    int rtn = 0;
    RECT rc;
    if (mproc == NULL)
        return -1;
    if (mproc(msg, p1, p2))    {
        switch (msg)    {
            case STOP:
                hide_mousecursor();
                clear_queue();
                mproc = NULL;
                break;
            /* -------- keyboard messages ------- */
            case KEYBOARD_CURSOR:
                unhidecursor();
                cursor(p1, p2);
                break;
            case CURRENT_KEYBOARD_CURSOR:
                curr_cursor((int*)p1,(int*)p2);
                break;
```

```
            /* -------- mouse messages -------- */
            case SHOW_MOUSE:
                show_mousecursor();
                break;
            case HIDE_MOUSE:
                hide_mousecursor();
                break;
            case MOUSE_CURSOR:
                set_mouseposition(p1, p2);
                break;
            case CURRENT_MOUSE_CURSOR:
                get_mouseposition((int*)p1,(int*)p2);
                break;
            /* ---------- video messages ---------- */
            case VIDEO_CHAR:
                gettext(p1+1, p2+1, p1+1, p2+1, &rtn);
                rtn &= 255;
                break;
            case PUT_VIDEORECT:
                rc = *(RECT *) p1;
                puttext(rc.x+1, rc.y+1, rc.x1+1, rc.y1+1,(char *) p2);
                break;
            case GET_VIDEORECT:
                rc = *(RECT *) p1;
                gettext(rc.x+1, rc.y+1, rc.x1+1, rc.y1+1,(char *) p2);
                break;
            default:
                break;
        }
    }
    return rtn;
}

/* ---- dispatch messages to the message proc function ---- */
int dispatch_message(
    int (*msgproc)(enum messages msg,PARAM p1,PARAM p2))
{
    mproc = msgproc;
    /* ------ dequeue the next message ----- */
    if (collect_messages()) {
        struct msgs mq = msg_queue[qoffctr];
        send_message(mq.msg, mq.p1, mq.p2);
        if (++qoffctr == MAXMESSAGES)
            qoffctr = 0;
        if (mq.msg == STOP)
            return FALSE;
    }
    return TRUE;
}

/* ---------- gather and interpret mouse events -------- */
static int mouse_event(void)
{
    get_mouseposition(&mx, &my);
    if (mx != px || my != py) {
        px = mx;

        py = my;
        return MOUSE_MOVED;
    }
    if (button_releases())
        return BUTTON_RELEASED;
    if (rightbutton())
        return RIGHT_BUTTON;
    if (leftbutton())
        return LEFT_BUTTON;
    return 0;
}

/* ---------- make a RECT from coordinates --------- */
RECT rect(int x, int y, int x1, int y1)
{
    RECT rc;
    rc.x = x;
    rc.y = y;
    rc.x1 = x1;
    rc.y1 = y1;
    return rc;
}
```

Конец

ПРОГРАММА 7

```
/* ---------------- copyscrn.c -------------- */
#include <stdio.h>
#include <stdlib.h>
#include "keys.h"
#include "message.h"

static int message_proc(enum messages, int, int);
static void near highlight(RECT);
static void writescreen(RECT);
static void near setstart(int *, int, int);
static void near forward(int);
static void near backward(int);
static void near upward(int);
static void near downward(int);
static void init_variables(void);

static int cursorx, cursory;    /* Cursor position       */
static int mousex, mousey;      /* Mouse cursor position */

static RECT blk;
static int kx = 0, ky = 0;
static int px = -1, py = -1;
static int mouse_marking = FALSE;
static int keyboard_marking = FALSE;
static int marked_block = FALSE;
static FILE *fp;

#ifdef TSR
#define main tsr_program
#endif

/* ---------- enter here to run screen grabber --------- */
void main(void)
{
    fp = fopen("grab.dat", "wt");
    if (fp != NULL)    {
        /* ----- event message dispatching loop ---- */
        while(dispatch_message(message_proc))
            ;
        fclose(fp);
    }
}


/* --------- event-driven message processing function ------- */
static int message_proc(
    enum message message,   /* message */
    int param1,             /* 1st parameter */
    int param2)             /* 2nd parameter */
{
    int mx = param1;
    int my = param2;
    int key = param1;

    switch (message)    {
      case START:
      init_variables();
      post_message(CURRENT_KEYBOARD_CURSOR,(PARAM) &cursorx, (PARAM) &cursory);
      post_message(KEYBOARD_CURSOR, 0, 0);
      post_message(CURRENT_MOUSE_CURSOR,(PARAM) &mousex, (PARAM) &mousey);
      post_message(MOUSE_CURSOR, 0, 0);
          break;
      case KEYBOARD:
      switch (key)    {
                case FWD:
                    if (kx < 79)    {
                        if (keyboard_marking)
                            forward(1);
                        kx++;
                    }
                    break;
                case BS:
                    if (kx)    {
                        if (keyboard_marking)
                            backward(1);
                        --kx;


                    }
                    break;
                case UP:

                    if (ky)    {
                        if (keyboard_marking)
                            upward(1);
                        --ky;
                    }
                    break;
                case DN:
                    if (ky < 24)    {
                        if (keyboard_marking)
                            downward(1);
                        ky++;
                    }
                    break;
                case F2:
                    mouse_marking = FALSE;
                    setstart(&keyboard_marking, kx, ky);
                    break;
                case '\r':
                    post_message(STOP, TRUE, 0);
                    break;
                case ESC:
                    post_message(STOP, FALSE, 0);
                    break;
      }
      send_message(KEYBOARD_CURSOR, kx, ky);
      break;
      case LEFT_BUTTON:
            if (!mouse_marking)    {
                px = mx;
                py = my;
                keyboard_marking = FALSE;
                setstart(&mouse_marking, mx, my);
      }
      break;
      case MOUSE_MOVED:
            if (mouse_marking)    {
                if (px < mx)
                    forward(mx-px);
                if (mx < px)
                    backward(px-mx);
                if (py < my)
                    downward(my-py);
                if (my < py)
                    upward(py-my);
                px = mx;
                py = my;
      }
      break;
      case BUTTON_RELEASED:
            mouse_marking = FALSE;
            break;
      case RIGHT_BUTTON:
            post_message(STOP, TRUE, 0);
            break;
```

```
        case STOP:
        if (marked_block)    {
                highlight(blk);
                if (param1)
                    writescreen(rect(min(blk.x, blk.x1),min(blk.y, blk.y1),
                                     max(blk.x, blk.x1),max(blk.y, blk.y1)));
            }
            send_message(MOUSE_CURSOR, mousex, mousey);
            send_message(KEYBOARD_CURSOR, cursorx, cursory);
            init_variables();
            break;
        default:
            break;
    }
    return TRUE;
}

/* ------- set the start of block marking ------- */
static void near setstart(int *marking, int x, int y)
{
    if (marked_block)
        highlight(blk);       /* turn off old block */

    marked_block = FALSE;
    *marking ^= TRUE;
    blk.x1 = blk.x = x;   /* set the corners of the new block */
    blk.y1 = blk.y = y;
    if (*marking)
        highlight(blk);   /* turn on the new block */
}

/* ----- move the block rectangle forward one position ----- */
static void near forward(int n)
{
    marked_block = TRUE;
    while (n-- > 0)    {
        if (blk.x < blk.x1)
            highlight(rect(blk.x,blk.y,blk.x,blk.y1));
        else
            highlight(rect(blk.x+1,blk.y,blk.x+1,blk.y1));
        blk.x++;
    }
}

/* ---- move the block rectangle backward one position ----- */
static void near backward(int n)
{
    marked_block = TRUE;
    while (n-- > 0)    {
        if (blk.x > blk.x1)
            highlight(rect(blk.x,blk.y,blk.x,blk.y1));
        else
            highlight(rect(blk.x-1,blk.y,blk.x-1,blk.y1));
        --blk.x;
    }
}

/* ----- move the block rectangle up one position ----- */
static void near upward(int n)
{
    marked_block = TRUE;
    while (n-- > 0)    {
        if (blk.y > blk.y1)
            highlight(rect(blk.x,blk.y,blk.x1,blk.y));
        else
            highlight(rect(blk.x,blk.y-1,blk.x1,blk.y-1));
        --blk.y;
    }
}

/* ----- move the block rectangle down one position ----- */
static void near downward(int n)
{
    marked_block = TRUE;
    while (n-- > 0)    {
        if (blk.y < blk.y1)
            highlight(rect(blk.x,blk.y,blk.x1,blk.y));
        else
            highlight(rect(blk.x,blk.y+1,blk.x1,blk.y+1));
        blk.y++;
    }
}

/* ------ write the rectangle to the file ------- */
static void writescreen(RECT rc)
{
    int vx = rc.x;
    int vy = rc.y;
    while (vy != rc.y1+1)    {
        if (vx == rc.x1+1)    {
            fputc('\n', fp);
            vx = rc.x;
            vy++;
        }
        else    {
            fputc(send_message(VIDEO_CHAR, vx, vy), fp);
            vx++;
        }
    }
}

/* ------- simple swap macro ------ */
#define swap(a,b) {int s=a;a=b;b=s;}

/* -------- invert the video of a defined rectangle ------- */
static void near highlight(RECT rc)
{
    int *bf, *bf1, bflen;
    if (rc.x > rc.x1)
        swap(rc.x,rc.x1);
    if (rc.y > rc.y1)
        swap(rc.y,rc.y1);
    bflen = (rc.y1-rc.y+1) * (rc.x1-rc.x+1) * 2;
    if ((bf = malloc(bflen)) != NULL)    {
        send_message(HIDE_MOUSE, 0, 0);
        send_message(GET_VIDEORECT, (PARAM) &rc, (PARAM) bf);
        bf1 = bf;
        bflen /= 2;
        while (bflen--)
            *bf1++ ^= 0x7700;
        send_message(PUT_VIDEORECT, (PARAM) &rc, (PARAM) bf);
        send_message(SHOW_MOUSE, 0, 0);
        free(bf);
    }
}

/* ---- initialize global variables for later popup ---- */
static void init_variables(void)
{
    mouse_marking = keyboard_marking = FALSE;
    kx = ky = blk.x = blk.y = blk.x1 = blk.y1 = 0;
    px = py = -1;
    mouse_marking = FALSE;
    keyboard_marking = FALSE;
    marked_block = FALSE;
}
```

Конец

факты

☐ Фирма Tiny Computers заявила, что ее специалисты создали семейство самых маленьких в мире настольных *(desktop)* ПК с размерами корпуса 210x73x133 мм. Две модели этого семейства имеют следующие характеристики:

TC1 (младшая модель): центральный процессор V30 фирмы NEC с тактовой частотой 10 МГц, оперативная память емкостью 640 Кбайт, один дисковод для 3,5-дюймовых дисков емкостью 1,44 Мбайт;

TC3 (старшая модель): центральный процессор 80286 с тактовой частотой 12 МГц, оперативная память емкостью 1 Мбайт (может быть расширена до 4 Мбайт), один дисковод для 3,5-дюймовых дисков емкостью 1,44 Мбайт и жесткий диск емкостью 40 Мбайт.

Компьютеры снабжены модулятором, позволяющим подключать к ним как стандартный монитор, так и цветной телевизор системы PAL.

☐ Немецкая фирма Neos International Computer Products выпустила универсальное устройство ввода Neos I-Beam для компьютеров семейства Macintosh. Это устройство можно использовать как оптическую "мышь", ручной сканер (300 точек/дюйм) и цифрователь. Устройство позволяет обрабатывать изображения формата A0 (84x120 см). Цена 2500 марок ФРГ.

☐ Фирма Mouse Systems выпустила очень легкую и миниатюрную оптическую "мышь" Little Mouse/PC, которая позволяет работать со в 100 раз более высокой точностью, чем аналогичные устройства, основанные на механических принципах. Использование оптической технологии позволило избавиться в этом устройстве от движущихся частей и повысить таким образом его надежность.

Новый взгляд на TSR-программы

Э. Стивенз

В предыдущей статье (см. с. 48-55), мы обсуждали событийно-управляемое программирование на языке С и написали небольшую программу, которая позволяет использовать клавиатуру и "мышь" для записи в виде файла состояния экрана в текстовом режиме. Эта программа может пригодиться для формирования документации. Вы можете "вырезать" на экране прямоугольную область, записать ее содержимое на диск и после этого обработать это содержимое с помощью редактора текста. Затем Вы можете вывести его на печатающее устройство или включить в текстовый файл.

Созданная программа запускается из командной строки DOS, что ограничивает область ее применения. Часто бывает нужно получить "мгновенные снимки" *(snapshots on-the-fly)* состояния экрана в процессе выполнения документируемой программы. Большинство программ не обеспечивают сохранения состояния экрана при выходе из них в DOS, а если и обеспечивают, то часть экрана будет занята командной строкой DOS, в результате чего, возможно, будет вытеснена та его часть, которая Вам нужна. Кроме того, надоедает возня с переходами из DOS в программу и обратно, поэтому для "вырезания" требующейся области экрана программа копирования должна быть резидентной.

Мы рассмотрим, как превратить программу, запускаемую из командной строки, в программу, остающуюся резидентной после завершения (TSR - Terminate and Stay Resident). О TSR-программах написано множество книг и журнальных статей. Я сам писал на эту тему в трех книгах и снова обращаюсь к ней, поскольку хочу завершить программу, приведенную в предыдущей статье, а также потому, что было бы нечестно обрушить на Вас лавину кодов без всяких объяснений.

© Dr. Dobb's Journal, April 1991.
All rights reserved. M&T Publishing, Redwood City, CA, USA.

TSR-программы существуют, поскольку разработчики DOS не догадались предусмотреть простейшего механизма переключения между задачами, одновременно находящимися в памяти.

TSR-программы - это уродцы. Они существуют, поскольку разработчики DOS не догадались предусмотреть простейшего механизма переключения между задачами, одновременно находящимися в памяти, - не многозадачности, а простого переключения между задачами. TSR-программы стали возможны, поскольку тем же самым разработчикам потребовалось "пристегнуть" спулер к однопользовательской однозадачной DOS. Они сделали в DOS определенные вставки, которые позволили спулеру работать в фоновом разделе таким образом, чтобы это не мешало выполнению основной задачи. Они не опубликовали подробностей об этих вставках, вероятно, из-за того, что этот способ и ненадежный, и некрасивый. Другие программисты разобрались с кодом спулера и создали "всплывающие" утилиты, подобные входящим в состав пакета SideKick. В конце концов эта технология стала общеупотребительной, и TSR-программы расцвели пышным цветом. Между прочим, Вы можете использовать операцию Paste подпрограммы Notepad пакета SideKick для выполнения той же задачи, которую выполняет учебная программа копирования содержимого экрана, за исключением того, что в пакете SideKick нельзя использовать "мышь" для "вырезания" нужной части экрана.

Теперь в двух словах о том, как работают "всплывающие" TSR-программы. Вы можете их запустить из командной строки DOS. Они связываются с определенными векторами прерываний и используют специальный вызов DOS для завершения программы без освобождения занимаемой памяти. Отсюда и название "резидентная после завершения". Когда Вы выполняете другие программы, DOS загружает их в память после TSR-программы. Если Вы нажимаете "горячую" клавишу TSR-программы, то программа обработки прерывания (которую сама TSR-программа подключила к вектору прерываний от клавиатуры) воспринимает нажатие этой клавиши и выполняет определенные действия, вызывающие "всплытие" TSR-программы.

TSR-программа, содержащая обращения к DOS, не может быть вызвана в произвольный момент времени. DOS не всегда в состоянии осуществить надлежащие действия. Если Вы прерываете выполнение программы в процессе обращения к DOS и запускаете другую программу, которая сама обращается к DOS, то DOS терпит крах, поэтому TSR-программа должна установить индикатор, свидетельствующий о ее желании "всплыть", и затем ждать, пока DOS не даст "добро". DOS осуществляет это двумя способами. Она формирует нечто, называемое признаком INDOS, - индикатор того, что DOS работает и ее работа не может быть прервана. До тех пор, пока признак INDOS установлен, нельзя переключаться на другую программу, обращающуюся к DOS. Хитрость состоит в том, что для ввода с клавиатуры программа COMMAND.COM обращается к DOS. Пока на экран выведен запрос ввода команды, DOS функционирует и признак INDOS установлен. Если бы признак INDOS давал единственный способ для прерывания работы DOS, то в ситуации, когда на экран выведен запрос ввода команды, никакое "всплы-

тие“ не было бы возможно. Чтобы обойти эту западню, в DOS изобретено другое уродство. Пока DOS “циклится“ в ожидании ввода с клавиатуры, периодически производятся действия, обеспечивающие возможность прервать работу DOS. Когда клавиша нажата, DOS выполняет прерывание 0x28. TSR-программа подключается к подпрограмме обработки прерывания INT 28 и при выполнении последней определяет, что может “всплыть“.

Обобщим сказанное: TSR-программа подключается к подпрограмме обработки прерывания от клавиатуры для отслеживания нажатия “горячей“ клавиши, к подпрограмме обработки прерывания INT 28, чтобы определить, когда DOS разрешает запуск, и к подпрограмме обработки прерывания от таймера для наблюдения за всем остальным. Когда ее подпрограмма обработки прерывания от клавиатуры обнаруживает, что Вы нажали “горячую“ клавишу, она устанавливает признак, свидетельствующий об этом. Когда выполняется подпрограмма обработки прерывания от таймера (18,2 раза в секунду), она следит за признаком “горячего“ пуска. Если признак “горячего“ пуска установлен, а признак INDOS - нет, то подпрограмма обработки прерывания от таймера запускает TSR-программу. Если при выполнении подпрограммы обработки прерывания INT 28 обнаруживается, что установлен признак “горячего“ пуска, то уже эта подпрограмма запускает TSR-программу.

Перед “всплытием“ TSR-программа должна переключить контекст прерываемой программы на свой. В настоящей многозадачной среде сама операционная система заботится о переключении контекста между задачами. В уродливом мире TSR-программ каждая резидентная программа должна делать это самостоятельно. Одна из причин того, что ситуация с TSR-программами является ненадежной, связана с существованием различных способов для переключения контекста; не все программы делают это одинаково, и не каждая программа выполняет при этом все необходимые действия. Что я вкладываю в понятие переключения контекста? Чтобы переключить контекст, необходимо заменить стек прерываемой программы на стек TSR-программы, исхитриться заставить DOS считать TSR-программу своей единственной задачей, переключив ее указатель на префикс сегмента (PSP) запускаемой программы, заменить область обмена с диском (DTA) прерываемой программы на соответствующую DTA-область TSR-программы, временно избавиться от реакции DOS в случае критической ошибки и нажатия пользователем комбинаций клавиш Ctrl-Break и Ctrl-C; если TSR-программа собирается использовать “мышь“, то следует сохранить текущий контекст “мыши“ - ведь прерываемая программа также может ее использовать.

В зависимости от того, что делает TSR-программа, Вам может понадобиться также управлять режимом работы экрана. Если TSR-программа настолько гибка, что может определить текущий режим работы экрана и функционировать в этом режиме, то Вам не нужно менять контекст экрана. Однако, если TSR-программа может, например, работать только в текстовом режиме, Вам придется запомнить текущий режим и, может быть, содержимое видеопамяти и изменить режим работы экрана на тот, который нужен TSR-программе. В случае программы копирования содержимого экрана можно об этом не беспокоиться, поскольку она копирует только состояние экрана в текстовом режиме, и мы вправе рассчитывать, что Вы станете запускать ее только в этом режиме. Манипулирование режимами работы экрана на ПК - непростое дело. Если прерываемая программа назначает режим в результате непосредственной адресации управляющих регистров видеотерминала, Вы можете оказаться не в состоянии определить текущий режим, тем более его сохранить, а затем снова на него переключиться.

После того как Вы выполните все действия по переключению контекста, можете запускать TSR-программу. TSR-программа, которая выполняется при нулевом признаке INDOS, может выполнять почти все вызовы DOS. TSR-программа, которая выполняется по прерыванию INT 28, может выполнять большинство вызовов DOS, следующих за функцией 0x0c. Функции от 0 до 0x0c являются функциями обслуживания экрана и клавиатуры, и для них существует опасность краха. Но не волнуйтесь. Эти функции так себя ведут и имеют такую производительность, что большинством TSR-программ не используются. Большинство TSR-программ для ввода с клавиатуры используют BIOS, а отображение на экран осуществляют путем прямой записи в видеопамять. TSR-программам следует также избегать обращений к DOS для распределения памяти или запуска других программ с помощью функции EXEC этой операционной системы.

После завершения работы TSR-программа должна восстановить контекст, т.е. привести его в то состояние, в котором он был перед “всплытием“ программы. Заметим, что программа, описанная в данной и предыдущей статьях, будет корректно выполняться как TSR-программа только для версий DOS, начиная с 3.0. Функции, с помощью которых получают доступ и формируют префикс PSP, в ранних версиях DOS надлежа-

факты

☐ Английский журнал PC USER регулярно публикует "двадцатку" лучших пакетов программ. Предлагаем вниманию читателей таблицу, составленную нынешней осенью.

Место	Название пакета	Фирма-поставщик	Тип пакета
1	WordPerfect 5.1	WordPerfect	Текстовый процессор
2	Lotus 1-2-3 Rel 3.1	Lotus	Электронная таблица
3	SuperCalc 5	Computer Associates	Электронная таблица
4	Word for Windows	Microsoft	Текстовый процессор
5	QEMM 386	Quarterdeck	Утилита
6	Excel 3.0	Microsoft	Электронная таблица
7	Lotus 1-2-3 Rel 2.2	Lotus	Электронная таблица
8	Xtree Pro Gold	Executive Systems	Утилита
9	Lotus 1-2-3 Rel 2.3	Lotus	Электронная таблица
10	PC Tools Deluxe 6	Central Point Software	Утилита
11	Norton Utilities 5	Symantec	Утилита
12	MS-Works DOS	Microsoft	Интегрированный пакет
13	Freelance 4.0	Lotus	Графический пакет
14	MS-Word 4 Mac	Microsoft	Текстовый процессор
15	WordStar 6	WordStar	Текстовый процессор
16	Laplink PC	Traveling Software	Коммуникационный пакет
17	Display Write 4	IBM	Текстовый процессор
18	dBase IV 1.1	Ashton-Tate	База данных
19	Smartware II	Informix	Интегрированный пакет
20	DataEase 4.2	Sapphire	База данных

щим образом не работают. Для использования PSP в DOS версии 2.0 существует ряд изощренных приемов, которые описаны в литературе, приведенной в конце этого раздела.

Приведенная на с. 59 программа представляет собой драйвер tsr.c, который превращает обычные программы в TSR-программы. Драйвер использует некоторые расширения Turbo C для чтения и записи регистров и выполнения прерываний. В других компиляторах для выполнения этих нестандартных операций предусмотрены иные способы, поэтому, если Вы используете не Turbo C, то Вам придется преобразовать коды драйвера, относящиеся к аппаратной части, в соответствии с соглашениями, принятыми в Вашем компиляторе. В большинстве компиляторов нет аналога псевдорегистрам сегмента стека и указателя стека, которые имеются в Turbo C, поэтому Вам может потребоваться написать на языке ассемблера подпрограмму для чтения и изменения значений регистров SS и SP. В большинстве компиляторов для ПК имеется набор функций для обслуживания прерывания int86, использующих структуры REGS и SREGS, и во всех они реализованы сходным образом, так что эта часть преобразования не вызовет затруднений.

Вы можете применять данный драйвер для создания других TSR-программ из обычных программ на языке C, если эти программы не используют вызовов DOS для ввода-вывода с клавиатуры. Три оператора описания в начале программы (см. с.59) связывают драйвер с TSR-программой. Символические имена KEYMASK и SCANCODE определяют "горячую" клавишу, а tsr_program определяет имя прикладной функции, которую вызывает драйвер в момент нажатия пользователем "горячей" клавиши. В данном примере для "горячей" клавиши используются скан-код 52 и маска 8. Эти значения соответствуют комбинации клавиш Alt-"точка".

Драйвер TSR-программ управляет всем их "хозяйством": и "всплытием", и тем, как производить смену контекста между выполняющейся программой и TSR-программой. Этот драйвер включает в себя главную функцию программы, которую он поддерживает. Чтобы программа не стала TSR-программой, для нее нужно предусмотреть собственную главную функцию, которая либо вызывает функцию tsr_program, либо сама ею является. Фактически это простейший путь для отладки TSR-программы. Поскольку в таком виде она является обычной программой DOS, Вы можете отлаживать ее с помощью отладчика программ на исходном языке и компоновать с драйвером TSR-программ уже после того, как программа заработает должным образом. В предыдущей статье в программу был включен условный оператор стадии компиляции. Если Вы определите это глобальное имя на стадии компиляции, то в программе будет изменено имя ее главной функции на tsr_program - имя функции, которая вызывается драйвером для запуска TSR-программы.

TSR-программы есть и будут объектами, несущими опасность.

Есть еще много того, что стоило бы знать о TSR-программах. Дополнительную информацию Вы можете получить из книг, которые перечислены ниже, а может быть, Вы решите, что знаете уже достаточно и будете использовать драйверы, подобные описанному в этой статье. Правда, описанный здесь драйвер не является вполне законченным. Вы не сможете использовать его с коммерческими TSR-программами, поскольку он не может работать под управлением DOS 2.0 и будет выводить на экран неизвестно что, если Вы вздумаете его запустить в графическом режиме. В литературе описан ряд других драйверов, в том числе и часть из моих собственных, где предусмотрено, как обойти некоторые из указанных здесь ограничений. Имеются коммерческие библиотеки, которые обеспечат управление Вашими TSR-программами. Лучшим выбором является бесплатно распространяемый пакет TesSeRact, который может управлять большинством TSR-программ в присутствии большей части остальных. Вы можете получить пакет TesSeRact через многие сервисные центры или же заказать у фирмы Innovative Data Concepts по телефону 215-443-9705. Можно раздобыть также и исходные коды. (Я бы использовал этот пакет намного чаще, если бы было проще набирать его название.)

Список литературы по TSR-программам

Вот список книг, в которых описывается механизм функционирования TSR-программ или обсуждаются вопросы их написания:

- Stevens A. Extending Turbo C Professional. - MIS Press, 1989
- Microsoft Mouse Programmer's Reference. - Microsoft Press, 1989
- The Waite Groupe. MS-DOS Developer's Guide, 2nd Edition. - Howard W. Sams, 1989
- Duncan R. et al. The MS-DOS Encyclopedia - Microsoft Press, 1988
- Tischer M. PC System Programming for Developers. - Abacus, 1989
- Young M. J. Performance Programming Under MS-DOS. - Sybex, 1987
- Wilton R. Programmer's Guide to PC & PS/2 Video Systems. - Microsoft Press, 1987

Заключение

Теперь, после того как я написал и даже опубликовал TSR-программу, я вынужден пойти на попятный и сказать, что, с моей точки зрения, TSR-программы есть и будут объектами, несущими опасность. Сначала оправданием их существования было то обстоятельство, что на ПК не обеспечивалась многозадачность. Здравый смысл заставляет склоняться к мысли о том, что, поскольку существуют мириады добрых старых ПК на базе микропроцессора 8088 с их жалкими 640 Кбайтами памяти, мы обязаны продолжать разрабатывать для них программное обеспечение.

Зачем? Можно приобрести микропроцессор 386 с огромной памятью за сумму, много меньшую той, которую приходилось платить в 1981 г. за ПК с 64 Кбайтами памяти и без жесткого диска.

Скажем прямо: "Остается либо использовать программное старье, либо раскошелиться и купить современную технику". В настоящее время в многозадачных средах, таких, как Windows или DesqView, шутя решаются щекотливые проблемы TSR-программ для тех, кто согласен использовать современное оборудование. Напишите программу, какую захотите. Забудьте о прерывании INT 28, признаке INDOS, фатальных ошибках, обработчиках прерываний, переключении контекста и всей этой чепухе. Запустите эту программу под управлением DesqView и вызывайте ее на выполнение, когда вздумается. А если Вы запустили ее, когда программа передачи данных занята пересылкой по линии связи большого файла, что тогда? Все будет как надо. А если кто-то из консерваторов не может жить без Вашей потрясающей новой программы, но при этом не хочет отказаться от старья, напомните ему, что существует командная строка DOS.

В результате, я решил больше никогда не связываться с TSR-программами.

ПРОГРАММА

```
/* -------- tsr.c --------- */

/*
 * A Terminate-and Stay Resident (TSR) engine
 */

#include <dos.h>
#include <stdlib.h>
#include <stdio.h>
#include "mouse.h"
#include "keys.h"

void tsr_program(void);

#define KEYMASK  8
#define SCANCODE 52

extern unsigned _stklen = 1024;
extern unsigned _heaplen = 8192;

/* ------- the interrupt function registers -------- */
typedef struct {
    int bp,di,si,ds,es,dx,cx,bx,ax,ip,cs,fl;
} IREGS;

/* --- vectors ---- */
#define DISK    0x13
#define CTRLBRK 0x1b
#define INT28   0x28
#define CRIT    0x24
#define CTRLC   0x23
#define TIMER   8
#define KYBRD   9
#define DOS     0x21

unsigned highmemory;

/* ------ interrupt vector chains ------ */
static void (interrupt *oldtimer)(void);
static void (interrupt *old28)(void);
static void (interrupt *oldkb)(void);
static void (interrupt *olddisk)(void);

/* ------ ISRs for the TSR ------- */
static void interrupt newtimer(void);
static void interrupt new28(void);
static void interrupt newdisk(IREGS);
static void interrupt newkb(void);
static void interrupt newcrit(IREGS);
static void interrupt newbreak(void);
static unsigned sizeprogram; /* TSR's program size     */
unsigned dossegmnt;          /* DOS segment address    */
unsigned dosbusy;            /* offset to InDOS flag   */
static int diskflag;         /* Disk BIOS busy flag    */
unsigned mcbseg;             /* address of 1st DOS mcb */
static char far *mydta;      /* TSR's DTA              */
int hotkeyhit = FALSE;
int tsrss;          /* TSR's stack segment */
int tsrsp;          /* TSR's stack pointer */

/* ------------- context for the popup ---------------- */
unsigned intpsp;            /* Interrupted PSP address   */
int running;                /* TSR running indicator     */
char far *intdta;           /* interrupted DTA           */
unsigned intsp;             /*      "       stack pointer */
unsigned intss;             /*      "       stack segment */
unsigned ctrl_break;        /* Ctrl-Break setting        */
void (interrupt *oldcrit)(void);
void (interrupt *oldbreak)(void);
void (interrupt *oldctrlc)(void);

/* ------- local prototypes -------- */
static void resident_psp(void);
static void interrupted_psp(void);
static void popup(void);

void main(void)
{
    unsigned es, bx;
    /* ---------- compute memory parameters ------------ */
    highmemory = _SS + ((_SP + 256) / 16);
    /* ------ get address of DOS busy flag ---- */
    _AH = 0x34;
    geninterrupt(DOS);
    dossegmnt = _ES;
    dosbusy = _BX;
    /* ---- get the seg addr of 1st DOS MCB ---- */
    _AH = 0x52;
    geninterrupt(DOS);
    es = _ES;
    bx = _BX;
    mcbseg = peek(es, bx-2);
    /* ----- get address of resident program's dta ----- */
    mydta = getdta();
    /* ------------ prepare for residence ------------ */
    tsrss = _SS;
    tsrsp = _SP;
    oldtimer = getvect(TIMER);
    old28 = getvect(INT28);
    oldkb = getvect(KYBRD);
    olddisk = getvect(DISK);

    /* ----- attach vectors to resident program ----- */
    setvect(KYBRD, newkb);
    setvect(DISK, newdisk);
    setvect(TIMER, newtimer);
    /* ------ compute program size ------- */
    sizeprogram = highmemory - _psp + 1;
    /* ----- terminate and stay resident ------- */
    _DX = sizeprogram;
    _AX = 0x3100;
    geninterrupt(DOS);
}

/* ---------- break handler ------------ */
static void interrupt newbreak(void)
{
    return;
}

/* -------- critical error ISR ---------- */
static void interrupt newcrit(IREGS ir)
{
    ir.ax = 0;              /* ignore critical errors */
}

/* ------ BIOS disk functions ISR ------- */
static void interrupt newdisk(IREGS ir)
{
    diskflag++;
    (*olddisk)();
    ir.ax = _AX;          /* for the register returns */
    ir.cx = _CX;
    ir.dx = _DX;
    ir.es = _ES;
    ir.di = _DI;
    ir.fl = _FLAGS;
    --diskflag;
}
/* ----- keyboard ISR ------ */
static void interrupt newkb(void)
{
    static unsigned char kbval;

    kbval = inportb(0x60);
    if (!hotkeyhit && !running)
        if ((peekb(0, 0x417) & 0xf) == KEYMASK)
            if (SCANCODE == kbval)    {
                hotkeyhit = TRUE;
                /* --- reset the keyboard ---- */
                kbval = inportb(0x61);
                outportb(0x61, kbval | 0x80);
                outportb(0x61, kbval);
                outportb(0x20, 0x20);
                return;
            }
    (*oldkb)();
}
/* ----- timer ISR ------- */
static void interrupt newtimer(void)
{
    (*oldtimer)();
    if (hotkeyhit && (peekb(dossegmnt, dosbusy) == 0) &&
            !diskflag)
        popup();
}

/* ----- 0x28 ISR -------- */
static void interrupt new28(void)
{
    (*old28)();
    if (hotkeyhit)
        popup();
}

/* ------ switch psp context from interrupted to TSR ----- */
static void resident_psp(void)
{
    intpsp = getpsp();
    _AH = 0x50;
    _BX = _psp;
    geninterrupt(DOS);
}

/* ---- switch psp context from TSR to interrupted ---- */
static void interrupted_psp(void)
{
     _BX = intpsp;
     _AH = 0x50;
    geninterrupt(DOS);
}
/* ------ execute the resident program ------- */
static void popup(void)
{
    running = TRUE;
    hotkeyhit = FALSE;
    intsp = _SP;
    intss = _SS;
    _SP = tsrsp;
    _SS = tsrss;
    oldcrit = getvect(CRIT);    /* redirect critical err  */
    oldbreak = getvect(CTRLBRK);
    oldctrlc = getvect(CTRLC);
    setvect(CRIT, newcrit);
    setvect(CTRLBRK, newbreak);
    setvect(CTRLC, newbreak);
    ctrl_break = getcbrk();     /* get ctrl break setting */
    setcbrk(0);                 /* turn off ctrl break    */
    intdta = getdta();          /* get interrupted dta    */
    setdta(mydta);              /* set resident dta       */
    resident_psp();             /* swap psps              */
    intercept_mouse();          /* intercept the mouse    */

/* ------ save the video cursor configuration ------- */
savecursor();
normalcursor();
unhidecursor();
enable();
tsr_program();              /* call the TSR C program */
disable();
/* ------ restore the video cursor configuration ----- */
restorecursor();
restore_mouse();            /* restore the mouse      */
interrupted_psp();          /* reset interrupted psp  */
setdta(intdta);             /* reset interrupted dta  */
setvect(CRIT, oldcrit);     /* reset critical error   */
setvect(CTRLBRK, oldbreak);
setvect(CTRLC, oldctrlc);
setcbrk(ctrl_break);        /* reset ctrl break       */
disable();
_SP = intsp;                /* reset interrupted stack*/
_SS = intss;
running = FALSE;            /* reset semaphore        */
```

Конец

Скорее всего, Вы пока не вполне знаете, что означает параметр FILES в файле CONFIG.SYS...

А. Ю. Абакин

Это неудивительно, если даже в таком грамотно составленном справочнике, как "TECH HELP", информации об этом параметре не хватает. Естественно, что в отечественной литературе тем более вряд ли удастся отыскать его приемлемое толкование.

Учитывая, что мы еще на долгие времена "повязаны" с компьютерами семейства IBM PC, а значит и с MS-DOS и что подобная информация достается нашему программисту с великим трудом, надеюсь, что изложенный ниже материал будет полезен многим другим программистам, которым не придется в подобной ситуации тратить недели на исследование столь малозначимого предмета.

Итак, параметр FILES влияет на операции системы стандартного ввода-вывода, осуществляемые через каналы доступа к данным (HANDLER). Чаще всего он описывается фразой примерно следующего содержания: "Это максимальное число одновременно открытых файлов в системе". Формально верно, однако такое определение позволяет предполагать, что для любой отдельно взятой задачи доступны все в настоящее время свободные описатели файлов... и очень сильно заблуждаться. Каким бы большим ни было значение параметра FILES, на самом деле текущая задача никогда не сможет открыть более 20 таких файлов, пять из которых к тому же являются зарезервированными и открываются всегда (поэтому для задачи остается лишь 15 HANDLER'ов). Это ограничение можно обойти не совсем корректным путем (а чего в этой операционной системе можно добиться корректно?), который станет очевидным после рассмотрения структуры таблиц MS-DOS, связанных с обеспечением стандартного ввода-вывода. Кстати, именно этим путем пошли разработчики СУБД FoxBase.

Все начинается с известной (в узком кругу) функции MS-DOS под номером 52H, которая "портит" регистры ES и BX. Она возвращает по адресу ES:BX блок параметров операционной системы (описание функции 52H заимствовано из справочника "TECH HELP"):

Смещение	Длина		
-2H	2	baseMCB	ES:[BX-2] - сегментный адрес первого блока управления памятью (Memory Control Block - MCB);
+0H	4	Смещение, Сегмент	адрес первого блока параметров диска (Disk Parameter Block - DPB);
+4H	4	Смещение, Сегмент	адрес первой таблицы файлов (то, что нас интересует);
+8H	4	Смещение, Сегмент	адрес заголовка драйвера устройства CLOCK;
+0cH	4	Смещение, Сегмент	адрес заголовка драйвера устройства CON;
+10H	2	maxSect	максимальный размер сектора для блочных устройств;
+12H	4	Смещение, Сегмент	адрес заголовка буферов ввода-вывода;
+16H	4	Смещение, Сегмент	адрес списка текущих каталогов всех дисков;
+1aH	4	Смещение, Сегмент	адрес таблицы файлов ввода-вывода через блок управления файлом (File Control Block - FCB);
+1eH	2		размер таблицы файлов (ввод-вывод через FCB);
+20H	1		driveCnt - число дисководов в системе;
+21H	1		lastDrive - соответствует значению LASTDRIVE в файле CONFIG.SYS.

Таким образом, по адресу со смещением +4 расположен "длинный" адрес первого элемента списка таблиц открытых файлов. Каждый такой элемент имеет следующую незамысловатую структуру:

Смещение	Длина		
+0H	4	Смещение, Сегмент	адрес следующего элемента списка (в конце списка смещение равно ffffH);
+4H	2	number	число описателей открытых файлов в этой таблице;
+6H		Описатель файла	число описателей равно number.
		Описатель файла	
		. . . . .	
		Описатель файла	

В системах MS-DOS 3.30 и 4.00 при загрузке обычно имеется два элемента списка. Первый находится внутри тела MS-DOS и рассчитан на пять описателей, а все остальные описатели расположены во втором элементе списка.

Описатель файла имеет следующую структуру:

+00H	2	Counter	счетчик HANDLER'ов, связанных с данным файлом;
+02H	2	OpenMod	режим открытия файла (Open Mode);
+04H	1	Att	атрибуты файла (каталоговые);
+05H	2	Flags	слово флагов (комментарий см. ниже);
+07H	4	Смещение / Сегмент	адрес блока параметров диска (DPB);
+0bH	2	TotClus	число кластеров, занимаемых файлом;
+0dH	2	Ftime	время создания файлов;
+0fH	2	Fdate	дата создания файла;
+11H	4	Fsize	длина файла;
+15H	4	Foffset	текущее смещение в файле;
+19H	2	LstClus	номер последнего кластера файла (используется при открытии файла для записи);
+1bH	2	1stClus	номер начального кластера файла;
+1dH	2	DirSect	номер начального сектора каталога;
+1fH	1	Ent	номер элемента каталога, относящегося к данному файлу;
+20H	11	F I L E N A M E E X T	
+2bH	2	?????	похоже, что эти поля зарезервированы, во всяком случае ненулевыми я их не видел;
+2dH	2	?????	
+2fH	2	?????	
+31H	2	PSP	префикс программного сегмента задачи (Program Segment Prefix - PSP), открывшей файл;
+33H	2	?????	используется, когда загружена утилита SHARE;

для MS-DOS 3.30:

+35H - длина описателя файла;

для MS-DOS V4.0:

+35H	2	?????	назначение мне неизвестно;
+37H	2	?????	
+39H	2	?????	

+3bH - длина описателя файла.

О поле FLAGS известно, что для файлов на диске в трех младших битах хранится номер дисковода. Что касается остальных бит, то это чисто флаговые биты, смысл которых не исследован.

Понятно, что число назначенных HANDLER´ов в общем случае больше или равно числу открытых файлов и не связано с параметром FILES (как это утверждается при описании прерывания 67H в справочнике "TECH HELP V4.0"). Таким образом, открытый HANDLER и открытый файл - не одно и то же, так как одному открытому файлу может быть поставлен в соответствие более чем один HANDLER (после описателя файла со смещением +00).

Это все, что касается таблиц открытых файлов.

Чтобы отобразить какие-то реальные номера HANDLER´ов на таблицу файлов, используется таблица HANDLER´ов, находящаяся в PSP активного процесса, т.е. того PSP, который может быть получен вызовом функции 62H операционной системы. Для этого используется область PSP, которая обычно обозначается как зарезервированная для DOS. Сама таблица находится со смещением 18H в PSP и занимает 20 байт. Соответственно, задача может получить номера HANDLER´ов с 0 по 19. Значением каждого байта является номер открытого файла в таблице открытых файлов. Значение ffH соответствует свободному HANDLER´у.

Таким образом, задача не может использовать более двадцати HANDLER´ов, как это уже отмечалось. Вероятно, разработчики MS-DOS не думали, что такое ограничение останется навсегда, ибо в PSP со смещением 32H находится размер таблицы HANDLER´ов (14H - 20(10)), который MS-DOS проверяет при манипуляциях с таблицей.

Обойти это крайне неприятное ограничение (особенно для разработчиков СУБД) можно очень просто. Достаточно создать внутри задачи свою расширенную таблицу HANDLER´ов. После открытия файла необходимо переписать в свою таблицу сохраненный DOS номер входа в таблицу открытых файлов. Затем, перед любой операцией ввода-вывода нужно скопировать этот номер в какой-нибудь фиксированный элемент таблицы HANDLER´ов MS-DOS и выполнить операцию, используя этот HANDLER. Например, в СУБД FoxBase для этих целей используется последний HANDLER - номер 13H, и через него осуществляется весь ввод-вывод.

В механизме стандартного ввода-вывода есть еще один "подводный камень", который может привести в недоумение творцов резидентных программ (TSR), осуществляющих операции с файлами. Активизация таких процессов происходит в произвольный момент времени и, естественно, PSP не этой задачи будет учтен операционной системой как активный. Соответственно, если TSR-программа при активизации пытается использовать ранее открытый файл (точнее, HANDLER), то может произойти все что угодно, кроме того, что планировал программист. Ситуация несколько лучше, если файл открывается и закрывается каждый раз при активизации данной резидентной программы. Все будет работать, если не возникнет достаточно маловероятная ситуация: у активного (с точки зрения MS-DOS) процесса окажется переполненной таблица HANDLER´ов. Лучше всего использовать недокументированную функцию 51H: установить активный PSP до и после выполнения ввода-вывода. Сначала необходимо установить свой PSP, потом восстановить предыдущее значение PSP. Пользоваться недокументированными функциями крайне нежелательно, но что делать, если, официально разрешив задачам оставаться резидентными (функция 21H, AH = 31H), разработчики MS-DOS не снабдили программистов достаточной информацией о том, как себя вести, сделав программу резидентной. Впрочем, необходимость переключения активного PSP, возможно, возникает только при организации ввода-вывода, но тогда появляется проблема нереентерабельности рассматриваемой операционной системы. Данные же о том, какие функции MS-DOS можно прерывать, а какие нет, также являются нелегальными. Кроме того, при выполнении операций ввода-вывода возникают и другие нюансы. Фирма Microsoft в каком-то смысле вполне последовательна, хотя это создает дополнительные сложности для ее клиентов.

Остается открытым вопрос о том, что произойдет, если число файлов в системе превысит значение 255. Можно воспользоваться прерыванием 67H, но непонятно, как должна видоизмениться таблица HANDLER´ов в PSP, чтобы можно было указать номер файла, больший 255.

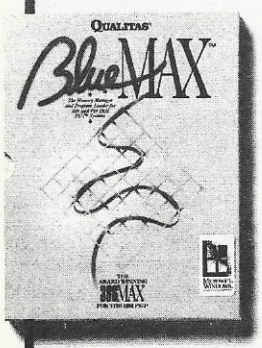
Экономия памяти экономит деньги

А. А. Петров

Тех, кто работает с программными средствами, требующими больших объемов оперативной памяти, могут заинтересовать программные продукты фирмы Qualitas (США):

386MAX®

BlueMAX™



- 386MAX 5.1 - драйвер расширенной памяти, диспетчер памяти, загрузчик драйверов и резидентных программ для IBM PC-совместимых компьютеров на базе процессоров 80386SX, 80386 и 80486;
- BlueMAX 5.1 - драйвер расширенной памяти, диспетчер памяти, загрузчик драйверов и резидентных программ для компьютеров семейства PS/2 на базе процессоров 80386SX, 80386 и 80486;
- MOVE'EM - загрузчик драйверов и резидентных программ для IBM PC-совместимых компьютеров, созданных на базе процессоров 80286 и 80386 и оснащенных платой дополнительной памяти, спроектированной в соответствии с требованиями спецификации EMS 4.0.

386MAX и BlueMAX обеспечивают управление дополнительной памятью в соответствии со спецификациями EMS 4.0, XMS 2.0, VCPI и VDS.

386MAX и BlueMAX освобождают основную память компьютера, автоматически загружая драйверы устройств и резидентных программ в область памяти от 640 Кбайт до 1 Мбайт - так называемую память High DOS. Встроенная в 386MAX и BlueMAX оптимизирующая процедура MAXIMIZE позволяет наилучшим образом использовать объем фрагментированной памяти High DOS, обеспечивая максимально возможное освобождение основной памяти. Оригинальный алгоритм загрузки резидентных программ позволяет загрузить в память High DOS даже те программы, длина кода инициализации которых превышает размер фрагмента памяти High DOS (при этом, разумеется, имеется в виду, что длина кода резидентной части программы не превышает размера соответствующего фрагмента).

Кроме того, BlueMAX производит сжатие кода базовой системы ввода-вывода (BIOS) благодаря исключению кодов процедур, не используемых DOS, что позволяет дополнительно освободить более 80 Кбайт памяти.

По сравнению со средствами управления памятью операционной системы DOS 5.0 рассматриваемые программные средства 386MAX и BlueMAX предоставляют следующие дополнительные возможности:

- позволяют перемещать коды процедур из "медленного" ПЗУ в быстродействующее ОЗУ, что увеличивает скорость выполнения операций;
- в среде Windows 3.0 обеспечивают работу многих резидентных программ (таких, как ANSI.SYS, SideKick и др.) одновременно в нескольких окнах DOS;
- с целью повышения производительности могут "переместить" наиболее "быструю" память в область адресов 1-го Мбайта; наибольший выигрыш это может дать при использовании усовершенствованных компьютеров семейства IBM PC на основе процессоров 80386 или 80486.

Загрузчик драйверов и резидентных программ MOVE'EM, освобождающий основную память компьютера путем перемещения драйверов устройств и резидентных программ в область памяти High DOS, обладает следующими возможностями:

- на компьютерах с объемом памяти менее 640 Кбайт увеличивает объем основной памяти до 640 Кбайт благодаря использованию дополнительной памяти;
- на компьютерах с видеоадаптерами MDA и CGA обеспечивает дополнительный объем памяти в 64 или 96 Кбайт соответственно;
- специальная процедура определяет оптимальную последовательность загрузки драйверов и резидентных программ, обеспечивая наиболее полное использование памяти.

Русский алфавит и Clipper 5.0

При работе с пакетом Clipper 5.0 подсистема ввода пропускает только символы с численными значениями ASCII-кодов, меньшими 128. Эту проблему легко решить, так как исходные тексты подсистемы ввода поставляются вместе с пакетом. Однако, как оказалось, Clipper 5.0 работает не со всеми драйверами русской клавиатуры. Дело в том, что он вызывает функцию INT 16, занося в AH код 10h,11h, в то время как на это рассчитаны не все драйверы. С помощью приведенной программы проблему можно устранить, хотя, конечно, лучше учесть указанную особенность при написании драйвера. Программа должна быть запущена после загрузки драйвера.

Такая же "недружелюбность" обнаружена у редактора Multi-Edit 4.00.

С. В. Аксиненко
г. Хабаровск

ПРОГРАММА

```
prog segment
        org 100h
assume  cs:prog,ds:prog,es:prog
start:
        jmp vhod                    ;переход на начало программы
klav:
        and ah,0FH                  ;обнуление старших четырех бит
        db 0eah                     ; JMP на старый обработчик INT 16
smesh:  dw 0                        ;
baza:   dw 0                        ;
vhod:
        mov ah, 35h                 ; получить адрес обработчика INT 16
        mov al,16h
        int 21h
        mov word ptr baza,es        ;сохранить сегмент в baza
        mov word ptr smesh,bx       ;смещение в smesh
        mov dx,offset klav          ;переустановить обработчик INT 16
        mov al,16h                  ;на klav
        mov ah,25h
        int 21h
        mov dx,11h                  ;конец работы
        mov ah,31h
        mov al,03h
        int 21h
prog ends
END START
```


ПОДПИСНОЙ КУПОН НА 1992 ГОД

Организациям и частным лицам,
желающим подписаться на

«ЖУРНАЛ Д-РА ДОББА»,

следует заполнить обратную сторону подписного купона
и вместе с копией платежного поручения *(для организаций)*
или квитанции почтового перевода *(для частных лиц)*
отправить по адресу:

125190, Москва, А-190, а/я 240.

Цена подписки (6 номеров) 34 руб.
Деньги на подписку переводить

издательству «ТРИАДА»
на р/с № 467909
ОПЕРУ Главного управления
Центрального банка РСФСР
(МФО 201779 Н6) г. Москва

Уважаемая редакция!

В первом номере "Журнала д-ра Добба" только треть объема носит практический характер, к тому же уровень не всегда достаточно высок. Например, статья А.И. Масаловича по графическим форматам на мой взгляд является негативным примером. Не говоря уже об ошибках набора (на рис.1 вместо формата GIF дан еще раз формат TIFF), даже приведенная информация не дает представления о формате TIFF: не указана структура тега, кодирование типов и пр.

Для того, чтобы статья представляла какой-либо практический интерес, необходимо по крайней мере приводить описания структур данных на C или Паскале. Для остальных форматов не приводятся даже блок-схемы.

Совет по перехвату и использованию изображений в своих собственных программах трудно реализуем: описания формата PIC автор не приводит, а пакет Halo явно избыточен в небольших программах, требует отдельных драйверов устройств, к тому же лицензия фирмы Media Cybernetics на его использование в профессиональных разработках стоит $17000. В результате, практическая польза от интересной на первый взгляд статьи оказывается равна нулю.

В.Н. Васильев
г.Москва

Уважаемый тов. Васильев!

С большим интересом прочитал Ваше письмо, проникнутое искренним желанием помочь нашему журналу. Благодарен за Ваш интерес к статье "Форматы графических данных". Присоединяюсь к извинениям редакции за ошибку в наборе. Хотел бы, однако, по-иному расставить акценты в рассуждении о полноте и полезности подобных статей.

Многолетнее сотрудничество с журналом "Интеркомпьютер" и опыт общения с зарубежными авторами показывают - никакая статья не может вместить всей информации, необходимой для работы. Да и не должна. Давая общее представление о том или ином вопросе, обзорная статья, как правило, становится началом переписки и дискуссии между автором и заинтересованными читателями. "Журнал д-ра Добба" открыт для таких дискуссий, так что если у Вас есть конкретные вопросы по статье - с удовольствием готов на них ответить. Однако сразу предостерегаю: даже самое внимательное чтение популярных и специальных статей не избавляет от необходимости изучения пухленьких томов документации. Если Вы, конечно, хотите работать профессионально.

С уважением
А.Масалович

Журнал Д-ра Добба

ПОДПИСНОЙ КУПОН НА 1992 ГОД

Фамилия, имя, отчество или название организации, фирмы *(полностью)* _____

_____

_____

Адрес, по которому Вы хотели бы получать «Журнал д-ра Добба» _____

(обязательно указывать почтовый индекс)

_____

_____

Количество комплектов годовой подписки _____

Цена подписки (6 номеров) 34 руб.

Фирма "Бюро коммерческой связи" предлагает услугу:

ГОДОВАЯ ПОДПИСКА НА 1992 ГОД

НА ПЕРИОДИЧЕСКИЕ ИЗДАНИЯ

(издания рассылаются подписчикам в конвертах)

№№ п/п	Наименование, издатель	Аннотация	Число номеров в год	Цена, руб.	
				для организаций	для частных лиц
1.	Издание "Дубль" Издательское предприятие "Телер", г.Москва	Еженедельник. Публикация абсолютно всей рекламы и коммерческой информации, прошедшей по центральному телевидению и всесоюзному радио. Уникальная возможность для деловых людей быть в курсе всех коммерческих новостей. Подписка на "Дубль" - это успех в бизнесе!	48	860	860
2.	Журнал "Мир ПК" СП"Информэйшн Компьютер Энтерпрайз" (ICE) (СССР-США) г.Москва	Несомненно, один из лучших отечественных сборников по вычислительной технике и информатике. Интересен профессионалам, полезен коммерсантам, доступен новичкам в мире персональных компьютеров. Сведения о современных иностранных и отечественных программных продуктах и компьютерах. Реклама, маркетинг. Качество издания не ниже мировых стандартов! Аналог журнала "PC World".	10	135	80
3.	Газета "Компьютеруорлд СССР" СП ICE (СССР-США), г.Москва	Еженедельник для предпринимателей и специалистов в области информационной технологии. Аналог всемирно известной американской газеты "Computer-World".	48	660	190
4.	Журнал "Сети" СП ICE (СССР-США), г.Москва	Иллюстрированный аналог известного журнала "Network World". Вопросы создания и развития вычислительных сетей. Коммерческие предложения, результаты научных разработок.	6	53	35
5.	"Журнал д-ра Добба" Издательство "Триада", г.Москва	Переняв все лучшее от своего предшественника "Интеркомпьютера", этот новый компьютерный журнал пополнился переводами статей из американских журналов "Dr. Dobb's Journal" (все о программировании), "DBMS" (все о базах данных), "LAN Technology" (все о ЛВС), "Personal Workstation" (все о рабочих станциях). Отличается конкретностью и ориентированностью на практику.	6	34	34
6.	Журнал "Байтик" Издательство "Триада", г.Москва	Первый в стране иллюстрированный журнал для молодежи, посвященный бытовым и школьным компьютерам. Публикует системные и прикладные программы, компьютерные игры, информацию о компьютерных клубах, рекламу, объявления.	6	24	24
7.	Журнал "Друг" Издательство "Триада", г.Москва	Первый российский иллюстрированный журнал для любителей и владельцев собак. История, породы, воспитание, советы ветеринара. Публикации из зарубежных журналов. Реклама.	4	33	33
8.	Журнал "Бизнес: Восток-Запад" СП "Интерэксперт" (СССР-ФРГ), г.Москва	Несомненный лидер среди коммерческих изданий для деловых людей. Экономическая, юридическая и коммерческая информация, практические консультации. Реклама услуг и продукции.	6	42	42
9.	Журнал "Московский наблюдатель" Союз театральных деятелей РСФСР, г.Москва	Лучшее иллюстрированное издание о театре. Распространяется в СССР, Великобритании, Израиле и США. Материалы об актерах, театральных премьерах. Проблемные статьи. Реклама.	12	49	49

Для оформления заказа необходимо перечислить стоимость подписки фирме "Бюро коммерческой связи" на р/с 3461463 в Измайловском отд. Мосбизнесбанка (МФО 201423) и выслать по адресу 115408, Москва, а/я 1:

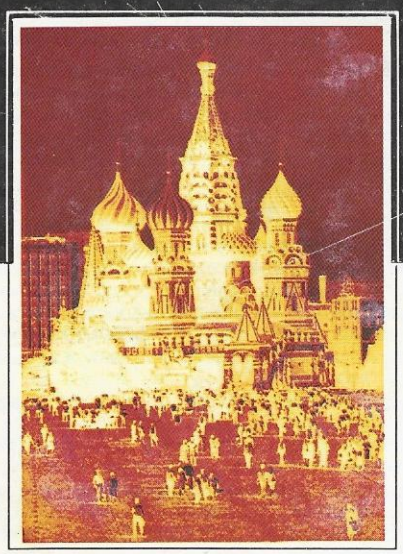
- копию платежного поручения (для организаций) или квитанцию почтового перевода (для частных лиц);
- заявку, в которой указать название издания, количество экземпляров, почтовый индекс, адрес, ФИО подписчика, телефон.

Заявки, направленные после 15.12.1991 г., и гарантийные письма не принимаются.

Предлагаем услуги:

- издательствам - по организации подписки и рассылке коммерческих изданий;
- рекламодателям - по рассылке проспектов, буклетов и других рекламных материалов подписчикам перечисленных изданий (прямая адресная рассылка - это самый эффективный вид рекламы).

THIRD INTERNATIONAL COMPUTER FORUM (ICF)



COME TO MOSCOW!

- ☐ The ICF brings together people that influence and determine the major trends of information technologies in the USSR and world-wide
- ☐ The ICF lets western companies obtain top-quality, first-hand information about the real and potential computer and software markets in the USSR
- ☐ The ICF provides Soviet computer professionals with better access to products and technologies from the rest of the world

Scheduled Speakers:

- Philippe Kahn (Borland)
- Jerry Kaplan (Go)
- Greg Herrick (Zeos)
- Dick Williams (Digital Research)
- Esther Dyson (EDventure Holdings)
- Stewart Alsop (InfoWorld)
- Ken Wasch (Software Publishers Association)
- Fred Langa (Byte)
- and others.

Topics:

- Status of the USSR computer market; business opportunities
- Future of PC market
- Workstations market
- Macintosh
- UNIX applications
- LAN policy
- LAN practice
- Windows and OS/2 market
- Pen-based technologies
- etc.

If you're a part of the emerging Soviet and east European computer industry, or if you want to learn more about doing business with that rapidly-growing market, *you need to attend the Third Annual International Computer Forum!*

SAVE MONEY BY PRE-REGISTERING

The registration fee covers access to keynotes, sessions, presentations, exhibitions, cultural programs, 3 lunches and 3 receptions. Register before March 31, 1992 and receive the special discount rate of just US$760. After March 31, 1992 the normal fee of US$840 applies.

RESERVE YOUR EXHIBITION SPACE

Price includes exhibition space itself, labor for building the booth, power supply (220 volt, 50Hz), and furniture (tables, chairs, etc.). The fee is US$3400 for each 9 sq.m. ($2800 for the members of ICC).

FOR MORE INFORMATION

Contact the ICC at:

Mail: 101813 USSR,
Moscow, Proyezd Serova, 4.
Fax/phone: 7-095-921-09-02.
MCI ID: 439-1034.
Internet: levon@staff.icc.msk.su.
San Francisco-Moscow
Teleport: ICC.

WORLD TRADE CENTER, MOSCOW, USSR
JUNE 15-18, 1992