

ЖУРНАЛ Д-ра Дობба

RUSSIAN
DR. DOBB'S
JOURNAL

№ 1' 1993

СУБД ДЛЯ ПРОФЕССИОНАЛОВ

**ПРОЕКТИРОВАНИЕ
СЛОЖНЫХ СИСТЕМ**

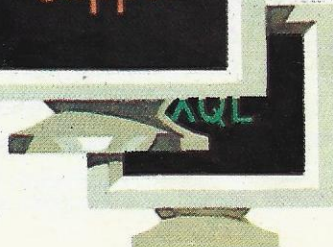
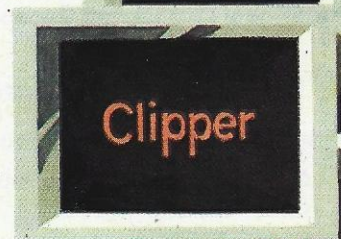
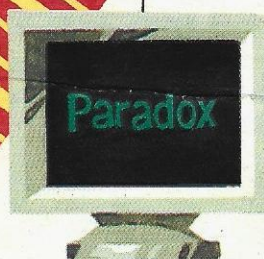
УПРАВЛЕНИЕ ЦВЕТОМ

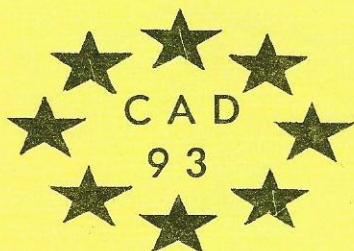
ТРЕХМЕРНАЯ АНИМАЦИЯ

**КАК НАПИСАТЬ
ПРОГРАММУ
БЕЗ ОШИБОК**

**Novell NetWare:
НОВЫЕ ВЕРСИИ**

ВЕРИФИКАЦИЯ ФАЙЛОВ





XX Jubilean International Conference and School
for young scientists and professionals
within the framework of
"Informatization of RUSSIA" Program
(Information Networking of RUSSIA Program)

CAD-93:
New Information Technologies
for Science, Education and Business
Yalta (Gurzuf), May 4-13, 1993

Sponsored by the Russian Academy of Science, Higher Education
and Technical Policy of RUSSIA; Russian Research Institute of Information Technology and CAD;
Microelectronics Research Institute "Progress"; Research Institute of IPIAN;
Research Center on Computer Technology; Zaporozhye State University; Voronesh Politechnic Institute;
Joint Venture "Software Operating Systems", Russian Dr. Dobb's Journal;
Moscow Instrumentation Institute

The Russian School for young design automation professionals was founded in 1974 as a non-profit association of Russian researchers and professionals, including leading scientists in the field of Artificial Intelligence and CAD. The School will also include extensive software demonstrations and exhibitions.

Submissions are invited on substantial, original, and previously unpublished research in all aspects of Computer-Aided Design and Artificial Intelligence, including, but not limited to:

- ☐ Artificial Intelligence for Analysis and Synthesis in CAD Systems
- ☐ Computer Networks
- ☐ Automation of Computer Design

- ☐ Expert Systems for Science, Education and Business
- ☐ Principles of Information Networking
- ☐ Databases for Information Networking
- ☐ Information Networking and Market Economics
- ☐ Social Impact of Information Networking
- ☐ Documentation Support Technologies
- ☐ Neurocomputers Theory and Application

The International Conference will be held from May 10 through May 13, 1993. On May 13 participants may choose, at an additional cost, to depart for Moscow to begin a 4-day tour of "Golden Ring of RUSSIA" in which they will visit the beautiful old Russian cities of Suzdal, Vladimir, Yaroslavl and Sergiev Posad.

International Organizing Committee address

Prof. Eugenie L. Gloriov
Moscow Instrumentation Institute
20, Stromynka Street, Moscow, 107076,
RUSSIA
"International Conference CAD-93"
Tel: 095/297-17-19 • Fax: 095/288-18-61

CAD-93 Representative In the US and Europe

Dr. Michael A. Shanblatt
Dept. of Electrical Engineering
Michigan State University
East Lansing, MI 48824-1226, USA.
Tel: 517/353-7249 • Fax: 517/353-1980
E-mail: mas@ee.msu.edu

С о д е р ж а н и е

СОБЫТИЯ

4

КРУГОЗОР

9

М.С. Каменнова Системный подход к проектированию сложных систем
System Approach to Complex Systems Design by M.S. Kamennova
Methods, methodologies and tools for complex systems design

МНЕНИЯ

16

А.Н. Козырев Интересы индустрии программирования и свобода личности
Programming Industry Interests and Personal Freedom by A.N. Kozurev
Division of copyright for software designed as a matter of official functions execution.

СУБД

21

Л.Л. Войтенко, А.А. Искра СУБД для профессионалов. Что выбрать?
DBMS for Professionals. What is the best choice? by L.L. Voytenko and A.A. Iskra
DBMS and the other tools for workstation software design.

А.А. Барчук Clipper 5.01. Русификация заголовков столбцов
Clipper 5.01: Field Name Russification by A.A. Barchuk
The author uses TBrowse class of Clipper 5.01 for database data looking-up and updating.

СЕТИ

30

М.Н. Беленькая, А.А. Бухман, Т.В. Карпова, К.И. Сумм, А.Н. Есаулов
Новые версии сетевой ОС Novell NetWare.
Сравнение функциональных характеристик
New Versions of Novell Netware by M.N. Belenkaya, A.A. Buhman, T.V. Karpova, K.I. Summ, A.N. Esaulov
Comparison of NetWare 2.2, NetWare 3.11, and NetWare Lite 1.0 performances.

КОПИЛКА ОПЫТА

38

С.Н. Кучеров Создание надежных программ
Reliable Programming by S.N. Kucherov
New dynamic memory administrator of Turbo Pascal 6.0.
М. Эйбраш Быстродействующие адаптеры VGA и трехмерная анимация
Potato Heads, Fast VGAs, and More by Michael Abrash
Н.Г. Краснокутский Управление цветом в адаптере VGA
Color Management in VGA by N.G. Krasnokutsky
Color creation in 16- and 256-color modes.
Д. Кодис Контрольные суммы Флетчера
Fletcher's Checksum by John Kodis
М. Нельсон Верификация файлов
File Verification Using CRC by Mark R. Nelson

ПОЧТОВЫЙ ЯЩИК

59

Д-ра Добба

1/93

Компьютерный
журнал для всех



Журнал зарегистрирован
Министерством печати
и массовой информации РСФСР
Свидетельство
о регистрации №791
Учредитель -
трудовой коллектив
издательства "Триада"
Директор А.А. Эйдес
Главный редактор М.Л. Сальников
Зав.редакцией
О.В. Ксендацкая
Редактор Е.Ф. Товстенко
Художественный редактор
С.Я. Грозовская
Обложка художника
А.Д. Кузнецова
Компьютерная верстка:
М.М. Егорова

Т Р И А Д А

Рукописи не рецензируются
и не возвращаются.

За правильность текстов рекламы,
подготовленной заказчиками,
редакция ответственности не несет.

Адрес для переписки:
125190, Москва,
А-190, а/я 240
Телефон: (095) 124-98-49
Электронная почта: @elbi.msk.su



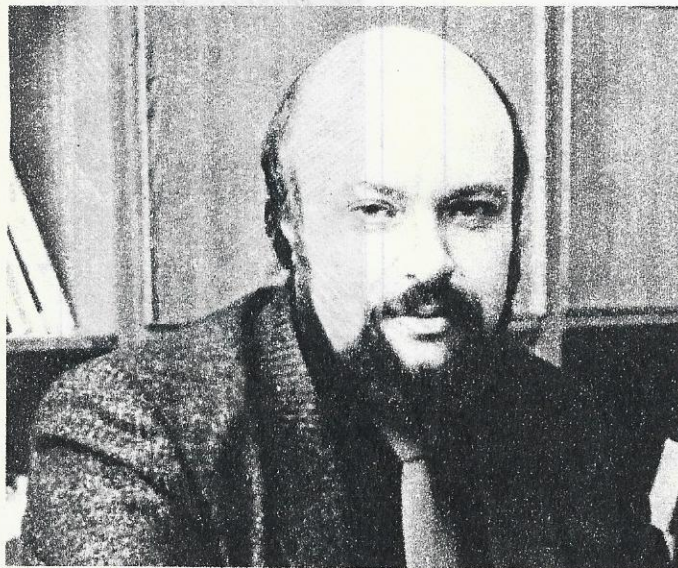
Information & Publishing Firm
"MACAU & Co"
Begovaya St. 14

Заказ № 2390/3 Объем 8 п.л.
Тираж 50 000 Цена свободная

3 типография
© «Журнал д-ра Добба», 1993

КОМПЬЮТЕРНАЯ ПРЕССА.

Нужно ли



Новый главный редактор "Журнала д-ра Добба" Михаил Сальников - профессиональный системный программист, специализирующийся в области методов создания программного обеспечения и оценки его качества. В издательской среде М. Сальников известен как автор аналитических и научно-популярных статей в различных компьютерных изданиях, как автор-переводчик книг по компьютерной тематике, как член научно-редакционного совета издательства "Компьютер", как эксперт по программному обеспечению и консультант ряда изданий и фирм, специализирующихся в области информатики.

Газета "Софт Маркет", №41 (57), 3 ноября 1992 г.

Один мой знакомый специально решил подсчитать, сколько же периодических изданий по компьютерной тематике выходит у нас в стране. Оказалось, что только в Москве существует более 20 (!) таких изданий. Среди них известные популярные журналы, имеющие подобно "Журналу д-ра Добба" иностранного партнера, добротные оригинальные издания, не имеющие "западной поддержки", и совсем простые, мало кому известные издания. Все вместе это и можно назвать компьютерной прессой.

Сотрудничая со многими компьютерными изданиями не один год, я составил определенное представление о роли, задачах и месте отечественной компьютерной прессы в российской информатике. Некоторыми соображениями на эту тему мне и хотелось бы поделиться с читателями.

Безусловно, основная (но не главная) задача компьютерной прессы - объективная подача информации о программных и технических средствах, представленных на мировом компьютерном рынке.

На первый взгляд она решается сравнительно просто. Наш рынок наводнен программными средствами (в основном, конечно, пиратскими копиями), нам доступны западные компьютерные издания и, наконец, представители иностранных фирм регулярно встречаются с журналистами, чтобы поделиться последней информацией о своих достижениях и представить свое очередное изделие. Однако поток информации очень велик и обработать ее в рамках одного издания крайне сложно. Прежде всего, как минимум, нужно иметь укомплектованную высоко-

профессиональными кадрами лабораторию тестирования. Но этого мало, поскольку сотрудники лаборатории могут охватить только основные направления информатики, поэтому необходимо привлекать к работе специалистов узкого профиля (для этого их, конечно, нужно сначала найти).

Работая с представителями иностранных фирм и поставляемыми ими материалами, следует помнить, что продвигая свою продукцию на рынке, они "отрабатывают свой номер", согласно давно проверенной схеме, в которой мы, возможно, еще не вполне разбираемся (это вовсе не значит, что нас снабжают искаженной информацией, просто она готовится в соответствии с задачами, стоящими перед фирмой). Из этого следует важный вывод - к контактам с представителями различных фирм (не только иностранных) следует тщательно готовиться. Попросту говоря, компьютерная пресса должна быть достаточно "зубастой", чтобы на любой пресс-конференции или представлении продукции (слово "презентация", как и словосочетание "контактный телефон", мой язык выговаривать просто отказывается) можно было бы задать множество жестких и возможно неприятных вопросов, как это принято на подобных мероприятиях во всем мире. (Сегодня же, да простят меня коллеги, компьютерная пресса скорее служит передаточным звеном между фирмами и потребителем, не слишком влияя на их взаимоотношения. Речь не идет о прямой рекламе, за содержание которой издание ответственности не несет, хотя и здесь должны существовать определенные этические нормы. Было бы нелепо, если бы скажем "Журнал д-ра Добба"

делить рынок?

поместил на своих страницах рекламу подобную рекламе из одной веселой телепередачи: ""Агдам"! Завтрашнее похмелье уже сегодня..."")

Понятно, что подготовка к контактам с представителями фирм очень трудоемка и требует значительных людских ресурсов. Наиболее простой элемент такой подготовки - это, например, анализ информации телеконференций, проходящей по международным компьютерным сетям.

Сказанного вполне достаточно, чтобы понять, что объективная подача информации требует значительных вложений (в первую очередь денежных). И это сегодня, когда пресса (компьютерная, в частности), испытывает значительные материальные трудности. Многие издания просто приостанавливают выпуск, многие выходят нерегулярно, а некоторые вообще заканчивают свое существование...

Выше я неоднократно употреблял слово "рынок", хотя, строго говоря, рынка компьютерных средств, программного обеспечения и средств связи у нас еще нет. Его формирование только-только началось. В формировании такого рынка я и вижу главную задачу компьютерной прессы. (Хотя понятно, что какой-то отдельный рынок компьютерных средств, программного обеспечения и средств связи, независимый от рынка вообще, сам по себе построен быть не может.) Не секрет, что в нашей стране работают не только фирмы, корректно ведущие дела, но, к сожалению, появилось множество авантюристов, готовых вывезти из страны все, что угодно, поставив взамен нам то, без чего мы вполне могли бы обойтись - залежалый товар, устаревшие технологии и т.д. Но это только половина беды, другая половина в том, что с нашей стороны у подобных "бизнесменов" достаточно "деловых партнеров". Компьютерная пресса и должна помочь потребителям на рынке компьютерных средств, программного обеспечения и средств связи отделить "зерна от плевел".

Даже если не говорить об авантюристах, а рассматривать лишь уважаемые фирмы, задача остается той же. Выходя на рынок, любая фирма планирует получить доход, продавая свой товар или услуги, потребитель же заинтересован оптимальным образом израсходовать имеющиеся у него средства. Особенно это важно сейчас, когда мы, по моему мнению, потенциально сказочно богаты (ресурсами, идеями, умами...), а фактически невероятно бедны. Мне приходилось слышать прямо противоположные мнения, касающиеся политики западных фирм на нашем рынке. С одной стороны, мол демпинго-

вые цены на программное обеспечение, предлагаемое ведущими западными фирмами, - величайшее благо, поскольку обеспечивают нашим потребителям легальный доступ к прогрессивным программным продуктам, а с другой, - мол, демпинговые цены окончательно загубят еще не народившуюся отечественную индустрию программного обеспечения. Истина же, наверное, не в крайностях. Компьютерная пресса и должна помочь разобраться, что к чему, тем более, что пришли в нее главным образом высококлассные специалисты.

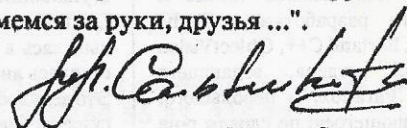
Компьютерная пресса может и должна добросовестно ориентировать не только потребителей, но и коммерческие структуры, способные вложить средства в развитие отечественной информатики. Это важнейшая часть формирования рынка, поскольку без собственной индустрии в этой области нам не обойтись и было бы наивно думать, что иностранные фирмы сами по себе будут развивать эту отрасль в нашей стране.

Это главное (но далеко не все), что хотелось бы сказать о задачах компьютерной прессы. Если у читателей появится интерес к поднятой теме, я готов продолжить разговор на страницах нашего журнала.

Теперь же нужно отметить, что рынок компьютерных изданий также находится в фазе формирования и, казалось бы, есть конкуренция и есть за что бороться и что делить. Высказывания в таком ключе мне приходилось слышать неоднократно, особенно от людей, давно работающих в компьютерной прессе, стоявших у ее истоков.

Подобный подход представляется мне абсолютно неверным. Главная задача - формирование рынка компьютерных средств, программного обеспечения и средств связи может решаться только объединенными усилиями всех изданий, работающих в этой области. Нужна концентрация ресурсов. Люди, работающие в компьютерной прессе, должны рассматривать коллег, как партнеров по общему делу, а не как конкурентов. Требуются и совместные вложения (в простейшем случае, например, при обработке поступающей от фирм информации), и организационные структуры, и многое другое. Работа в этом направлении уже ведется, хотя говорить подробно об этом еще рано.

Сказанное мной, конечно, не откровение. Есть и положительные результаты в первых попытках сотрудничества. Так что, "возьмемся за руки, друзья..."



Михаил Сальников
Главный редактор

ПРЕСС-КОНФЕРЕНЦИЯ ФИРМЫ BORLAND

22 сентября 1992 г. в Центральном доме туриста (Москва) состоялась пресс-конференция фирмы Borland. В конференции приняли участие Вице-президент и Генеральный директор Европейского отделения фирмы Франсуа Миколь, Директор по развитию бизнеса в странах Восточной Европы Юрий Парад, специалисты Европейского отделения и представители фирмы в СНГ. Прессу представляли примерно три десятка сотрудников ведущих компьютерных изданий России и Украины. Длительная около двух часов пресс-конференция была посвящена открытию Российского отделения фирмы - акционерного общества "Борланд А/О", характеристике технологической и рыночной стратегий фирмы Borland, а также представлению новых программных продуктов фирмы.

Выступивший в начале конференции г-н Миколь сделал обзор пройденного фирмой пути и ее сегодняшнего положения. Фирма Borland, насчитывающая более 2000 сотрудников и имеющая годовой доход свыше 480 млн. дол. (с учетом приобретенной фирмы Ashton-Tate), входит в тройку крупнейших в мире производителей программной продукции. Более половины всех продаж фирма осуществляет за пределами США, ее представительства действуют в 90 государствах. Одним из последних, как с грустной иронией заметил г-н Миколь, было открыто Московское представительство.

В настоящее время фирма Borland состоит из четырех основных подразделений: прикладных разработок (Quattro Pro и Sidekick), языков (dBase и средства разработки - Turbo Pascal, Borland C++, ObjectVision и т.п.), отдела, ведающего СУБД Paradox, и небольшого, но являющегося, по словам г-на Миколя, жемчужиной фирмы,

отдела, который занимается сервером баз данных Interbase.

Г-н Миколь отметил два основных стратегических принципа фирмы Borland в области технологии: нейтральную позицию по отношению к операционной среде и полную и безоговорочную приверженность объектно-ориентированному программированию. Первый принцип выражается в стремлении создавать параллельные разработки для DOS и Windows, не обходя вниманием OS/2 (имеются версии ObjectVision и редактор Brief для OS/2), а также Unix и VAX/VMS (dBase и Interbase). С приобретением фирмы Ashton-Tate фирма Borland получила несколько Apple-ориентированных разработок. Что касается второго принципа, то фирма Borland стала одним из первых крупных производителей программного обеспечения, целиком перешедших на использование объектно-ориентированного подхода в своих разработках. В настоящее время все ее компиляторы, базы данных и электронные таблицы написаны на языке C++. Некоторые фирмы, например Lotus и WordPerfect, по утверждению г-на Миколя, также используют компилятор Borland C++ при создании своих продуктов.

Все это, как заявил г-н Миколь, позволило фирме Borland создать лучшие в своем классе программные продукты и завоевать 65% мирового рынка в области средств разработки и 70% рынка СУБД.

Вслед за Ф. Миколем выступил г-н Драммонд, специализирующийся на электронных таблицах. Он представил новую версию среды для работы с электронными таблицами Quattro Pro 4.0 для DOS и версию Quattro Pro для Windows. Как заявил выступающий, презентация пакета Quattro Pro для Windows откладывалась в США семь раз и состоялась лишь две недели назад. Это делалось из перфекционистских соображений: новый пакет для работы с электронными таб-

лицами в среде Windows должен был стать лучшим в мире! "И он стал таковым," - просто и скромно констатировал докладчик. Теперь, по его словам, фирме Borland принадлежат лучшие в мире средства для работы с электронными таблицами - как в среде DOS (Quattro Pro 4.0), так и в среде Windows (Quattro Pro for Windows). В числе преимуществ пакета Quattro Pro for Windows фирма Borland видит организацию данных в форме файла-"блокнота" из 256 поименованных "страниц", объединяющего несколько электронных таблиц. Другое достоинство пакета - удобный интерфейс пользователя, включающий в себя оперативную панель (набор кнопок, соответствующих наиболее употребительным командам), управление сложными макросами с помощью кнопок, "меню-потребования" (возможность указать "мышью" на любой объект в пределах экрана и нажатием кнопки мыши вызвать всплывающее меню, связанное с данным конкретным объектом). Пакет позволяет также манипулировать двух- и трехмерной графикой и обеспечивает совместимость с многочисленными форматами данных.

Выступление г-на Рейна, представившего СУБД Paradox 4.0, прошло под девизом "Самая мощная база данных теперь стала еще и самой быстрой". По сравнению с предыдущей версией производительность версии Paradox 4.0 возросла более чем в 10 раз. Для этого потребовалось переписать свыше 70% кода пакета. Г-н Рейн долго описывал преимущества СУБД Paradox 4.0 по сравнению с наиболее мощными из конкурирующих продуктов - FoxPro, приводя сравнительные диаграммы, полученные, по его словам, в результате независимой экспертизы, проведенной американской Национальной лабораторией тестирования программных продуктов. Судя по этим диаграммам, производительность СУБД Paradox, как в

однопользовательском, так и в многопользовательском режиме, при запросе данных в одной или нескольких таблицах и при выполнении других операций в большинстве случаев многократно превосходит аналогичные показатели СУБД FoxPro. Г-н Рейн продемонстрировал несколько приемов работы с СУБД Paradox, сделав, в частности, выборку по указанному признаку из базы с 1 млн. записей и получив таблицу примерно с 1 тыс. записей. Это действительно заняло чуть больше 1 с (размер записей, правда, был невелик: база содержала некие простые фиктивные данные, и, кроме того, не было уточнено, на каком процессоре выполнялись эти действия).

Надо сказать, что представители фирмы ожидали скептического подхода к своим выступлениям, и эти ожидания отчасти оправдались. В ответ на громкие заявления о безоговорочном приоритете продуктов фирмы Borland в зале неоднократно раздавались смешки и звучали каверзные вопросы. Однако в целом все прошло спокойно, а г-н Рейн, выступление которого было не лишено артистизма, даже сорвал аплодисменты.

Последовавшие за выступлениями вопросы касались лицензионной политики фирмы и доступности ее продуктов, в частности русифицированных, в СНГ. В ответ было заявлено, что фирма Borland намерена лоббировать законодательные действия российского правительства по защите интеллектуальной собственности, но, как и прежде, намеревается бороться не с одиозными незарегистрированными пользователями, а, в первую очередь, с теми, кто пытается нажиться на продаже ворованных программ и незаконно копированной документации. Представители фирмы и прессы были единодушны в мнении о необходимости повышения культуры пользователей в нашей стране. Способствовать этому должны как законодательство, так и об-

Ы Т И Я

разование, причем не последнюю роль должно сыграть сохранение цен на программы на минимальном уровне.

В ответ на вопросы о методике испытаний, позволивших получить столь выигрышные для продуктов фирмы Borland диаграммы, журналисты были отосланы к опубликованным описаниям методики, которые можно приобрести всего за 25 дол.!

Что касается поставок любых разработок фирмы в страны СНГ и "локализации" (перевода программ и документации на местные языки), дело, по словам представителей фирмы, за заказчиками: будут заказы - будут и поставки.

А.Ф. Силонов

НОВЫЙ КОМПИЛЯТОР ФИРМЫ BORLAND

11 декабря 1992 г. в Москве фирма Borland представила новейшие программные продукты: Borland Pascal with Objects 7.0 и Turbo Pascal 7.0. Новый компилятор языка Pascal - Borland Pascal with Objects 7.0 пополнил серию компиляторов фирмы Borland для профессиональных программистов, к которой до сих пор относился только пакет Borland C++. В очередной раз фирма подтвердила готовность развивать языки программирования Pascal и C, предоставляя пользователям все большие возможности для создания более совершенных программ.

Пакет Borland Pascal with Objects 7.0 - новая интегрированная среда, позволяющая профессиональным программистам разрабатывать как прикладные DOS-программы, функционирующие в реальном и защищенном (DPMI) режимах, так и Windows-программы.

Реализуются эти разработки с помощью входящих в пакет компиляторов Turbo Pascal for Windows 1.5, Turbo Pascal 7.0 и утилит.

Пакет Turbo Pascal 7.0 - последняя версия компилятора Turbo Pascal, помогающая программистам быстро осваивать объектно-ориентированное программирование в реальном режиме в среде DOS.

НОВОЕ В КОМПИЛЯТОРЕ BORLAND PASCAL WITH OBJECTS 7.0

Компилятор защищенного режима DOS. Теперь вы можете создавать программы, работающие в режиме DPMI и использующие до 16 Мбайт оперативной памяти компьютера - преодолел 640-килобайтный барьер, доставлявший программистам столько неприятностей.

DLL-библиотеки. Компилятор позволяет создавать DLL-библиотеки в формате Windows для поддержки программирования защищенного режима DOS с возможностью использования процедур, написанных на языках C и C++.

WordPerfect™ 5.1

Оригинальный текстовый процессор фирмы WordPerfect™ Corporation, признанный во всем мире, поставляется с комплектом шрифтов ParaType™ фирмы ParaGraph®

Пакет полностью русифицирован, включает:

- программу проверки правописания в русском тексте
- программу переноса слов по правилам русского языка
- документацию на русском языке

WordPerfect™ 5.1 позволяет:

- объединять в одном документе текст и графику
- редактировать уравнения, работать с широким набором математических и специальных символов
- работать с табличной графикой

103051 Москва, Цветной бульвар, 3, телефоны: (095) 200 25 66, (095) 924 17 81, телефакс: (095) 928 27 68
Дилер: 194223 Санкт-Петербург, ул. Пушкинская, 2, телефоны: (812) 164 56 33, 315 73 12

PARAGRAPH

Расширение синтаксиса. Фирма Borland продолжает совершенствовать язык Turbo Pascal. В версии 7.0 добавлены ключевые слова Public и Private для описания полей и методов объектов. Ключевое слово Inherited может использоваться для обозначения родительского объекта при обращении к его методам. Наконец-то добавлены ключевые слова Break и Continue для прерывания и продолжения циклов.

Новые способы передачи параметров. Теперь вы можете описать параметр процедуры или функции как массив неопределенной длины. Диапазон значений индексов массива передается неявно и доступен внутри процедуры при использовании функций High и Low. Кроме того, добавлена возможность описания параметров-констант, что позволяет уменьшить размер генерируемого кода и повысить быстродействие его выполнения.

Возможность работы со строками, оканчивающимися нулем. Тип PChar, впервые введенный в Turbo Pascal для среды Windows, теперь доступен и в DOS-программах.

Использование 32-битовых инструкций для процессора Intel 80386. Ваша программа сама определит тип процессора, на котором она выполняется, и выберет оптимальные функции для умножения и деления 32-битовых чисел.

Оптимизация кода. Оптимизированные операции чтения и записи текстовых файлов будут выполняться в 1,3-2,0 раза быстрее, чем в предыдущей версии компилятора. Оптимизированы также операции над множествами и указателями.

Неограниченное число уровней вложенности модулей. В версии 7.0 снято существовавшее ранее ограничение до 15 этого числа. К тому же компилятор более корректно работает в ситуациях с циклической вложенностью модулей.

Расширение Turbo Vision. Введены новые типы объектов для манипулирования иерархическими данными, добавлена возможность проверки диапазонов значений в полях ввода.

Вывод на печать из Windows-программы. Для тех, кто пытался обеспечить работу с принтером для Windows-программ на языке Turbo Pascal, будет приятным сюрпризом новый тип объекта TPrinter, который берет на себя выполнение этой работы.

Новая интегрированная среда разработки программ. Интегрированная среда разработки DOS-программ для защищенного режима позволяет использовать при отладке весь объем памяти компьютера. Встроенный текстовый редактор дает возможность редактировать файлы размером до 25 Мбайт.

Выделение синтаксиса цветом. Ключевые слова, идентификаторы, строковые константы и комментарии текста программ окрашены различным цветом - это производит прекрасное впечатление.

Неограниченный откат. Возможность отменить, а затем, если нужно, восстановить изменения в тексте программы ограничена только объемом свободной памяти.

Инспектор объектов. Это мощное средство отладки, позволяющее обследовать дерево объектов, поля и методы каждого из них и быстро найти место обращения к этим методам в тексте программы.

Компилятор справочников. Вы можете не только создавать справочники для своих программ, но и расширять справочник интегрированной среды Borland Pascal with Objects 7.0.

Меню Tools. Возможность добавлять в меню интегрированной среды любые программы, что позволяет, например, компилировать файлы на языке ассемблера, не выходя в DOS.

Новые утилиты. Утилиты WinSight и WinSpector расширяют возможности отладки Windows-программ.

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Для работы пакета Borland Pascal with Objects 7.0 необходимо следующее:

- процессор Intel 80286 или более мощный;
- объем оперативной памяти 1,5 Мбайт;
- ОС DOS 3.3 или старше;
- Microsoft Windows 3.0 или 3.1;
- монитор не хуже EGA.

Объем памяти, занимаемой пакетом на жестком диске:

- 27 Мбайт при полной установке;
- 6 Мбайт для реального режима DOS;
- 6 Мбайт для защищенного режима DOS;
- 15 Мбайт для системы Windows.

С.Н. Кучеров

МИССИЯ UNIX SYSTEM LABORATORIES

24-25 сентября 1992 г. в МИ-ЭМ состоялась конференция-семинар по использованию операционной системы (ОС) UNIX.

"Мы пришли к вам с благой вестью о возможностях новой версии ОС UNIX", - заявили представители фирмы UNIX System Laboratories (USL), явившейся одним из организаторов этой конференции. В ее организации также участвовали: нью-йоркский бизнесмен А. Фенси, российская ассоциация "Информационные технологии в образовании" (ИНТО) и SUUG (ассоциация пользователей ОС UNIX).

Основным событием конференции было представление новых программных продуктов фирмы USL: ОС UNIX System V Release 4.2 (UNIX SVR 4.2) и системы TUXEDO, управляющей взаимодействием разнородных элементов сетей.

Новая версия ОС UNIX, снабженная развитым графическим интерфейсом, является попыткой добиться совместимости

всех предшествующих версий ОС UNIX, XENIX, SunOS, графических интерфейсов MOTIF и OPEN LOOK. Кроме того, ОС UNIX SVR 4.2 также поддерживает программы, предназначенные для работы в средах MS-DOS и Windows. По мнению представителей фирмы USL, такие широкие возможности новой операционной системы, а также ее полноценное функционирование в многопользовательском, многозадачном режиме, наличие в ней большого числа мощных утилит и средств разработки программ должны обеспечить ОС UNIX беспорный успех на рынке компьютерных технологий. Привлекательность ОС UNIX для программистов дополняется уникальной открытостью системы: все ее компоненты доступны в исходных текстах.

Для заокеанских участников конференции явилась одним из шагов в рамках планируемого широкого наступления фирмы USL на российский рынок: значительная часть приглашенных состояла из представителей банков и бирж. "В качестве конечных пользователей нас интересуют прежде всего крупные предприниматели, т.е. люди, решающие масштабные задачи, а не те, которые используют компьютеры в основном для игр или распечатки документов", - сказал А. Фенси, занимающийся созданием сети дистрибьютеров программных и аппаратных средств в СНГ. Западные участники конференции проявили заинтересованность в привлечении программистов из бывшего СССР к сотрудничеству в создании новых программных средств. Была также выражена традиционная обеспокоенность западных фирм отсутствием защиты интеллектуальной собственности в странах СНГ, хотя, как признался консультант USL по странам Центральной и Восточной Европы Н. Брига, такие страны, как Испания и Италия, по уровню программ не уступают России.

А.Ф. Силонов

НОВАЯ КОМАНДА ФИРМЫ APPLE НА РЫНКЕ СНГ

В рамках проходившей в Москве с 25 по 28 ноября 1992 г. выставки "TRAINING'92" компьютерной прессе была официально представлена новая команда (уже третья на нашей памяти), приступившая к продвижению на рынке бывшего СССР продукции фирмы Apple Computer.

Официальным дистрибьютером фирмы Apple стала новая независимая маркетинговая компания RUI, о чем Европейским отделением фирмы Apple было объявлено еще 6 октября 1992 г. Главный менеджер RUI Джеймс Гайди уже имеет большой и успешный опыт продвижения продукции фирмы Apple в Восточной Европе: за его плечами создание одной из наиболее процветающих дистрибьютерских фирм - фирмы TIS (Чехословакия, 1990 г.). С российской стороны административное звено возглавляет Николай Федулов, бывший ранее президентом Merisel.

В дальнейшем новый дистрибьютер фирмы Apple планирует активно задействовать коллективы разработчиков "на местах" и приглашает всех заинтере-

ресованных лиц к широкому сотрудничеству. Именно такая стратегия, по мнению руководства RUI, может привести к реальным достижениям на нашем рынке.

Уместно напомнить, что до сих пор положение фирмы Apple на рынке СНГ было более чем скромным, хотя ее успехи на мировом рынке производят большое впечатление. Достаточно сказать, что по результатам третьего квартала 1992 г. фирма Apple впервые за все время своего существования вышла на первое место в мире по числу продаж персональных компьютеров.

Адрес компании RUI:
Россия, 103772,
Москва, ул. Москвина, 8.
Тел.: (095) 229-11-36,
(095) 229-74-14.

Т. Ять

ФИЛИАЛ ФИРМЫ MICROSOFT В РОССИИ ОТКРЫТ!

15 декабря 1992 г. в отеле "Балчуг" состоялась пресс-конференция, посвященная историческому для компьютерного мира событию - открытию филиала фирмы Microsoft в России, состоявшемуся после трехлетнего исследования фирмой потенци-

ального рынка. В процессе проведения исследований фирма Microsoft проводила конференции разработчиков программного обеспечения, изучала мнения пользователей и разработчиков, проводила бета-тестирование своих программных продуктов и т.д. По результатам работ фирма пришла к заключению о целесообразности открытия филиала в России.

Возглавил филиал Роберт Клаф (Robert Clough), известный в нашей стране как сотрудник фирмы Nuntucket, оказавший решающее влияние на продвижение продуктов этой фирмы на нашем рынке (вот и рассуждай после это о номенклатуре, как о типично советском явлении). Кроме того, Роберт Клаф - президент созданной в России Ассоциации поставщиков программных продуктов, соучредителями которой являются и некоторые отечественные фирмы.

Полный штат филиала должен составить 11 человек. В России и странах бывшего СССР фирма Microsoft планирует создать мощную организацию, предоставляющую потребителям программное обеспечение, сопутствующие продукты, систему обслуживания и поддержки.

Деятельность фирмы Microsoft в России будет развиваться по четырём основным направлениям: семейство Windows, прикладные программы, рабочие группы и комплексные решения. В этом смысле работа фирмы в России ничем не будет отличаться от ее работы в других странах.

В задачи Российского филиала будет входить:

- создание инфраструктуры, содействующей развитию новой компьютерной индустрии. Инфраструктура будет включать в себя дистрибьютеров, дилеров, обучающие центры, группы адаптации и разработки программного обеспечения;
- непосредственная поддержка разработчиков программного обеспечения "на местах";
- обеспечение двусторонней связи пользователей с фирмой Microsoft для более полного удовлетворения потребностей пользователей в получении информации, а также поддержке и приобретении новых версий программных продуктов.

Адрес филиала фирмы:
Россия, 103064, Москва,
ул. Старая Басманная, 14.
Тел.: (095) 262-12-13,
(095) 262-10-13.
Факс: (095) 262-23-51.

Т. Ять

ПОДПИСКА на "Журнал д-ра Добба"

Редакция журнала оформляет подписку на 1993 г. (6 номеров).

Для оформления подписки необходимо перечислить 3900 руб. (для организаций) издательству "Триада" на расчетный счет № 467128 в коммерческом банке "Родина", кор. счет 161366 в РКЦ ГУ ЦБ РФ по г. Москва, МФО 201791.

Копию платежного поручения вместе с заявкой (в которой указать: название журнала, почтовый индекс, адрес и название организации) следует направить по адресу:

125190, Москва, А-190, а/я 240.

Желающим оформить подписку на вторую половину 1993 г. следует обращаться в отделения связи.

По всем вопросам, связанным с оформлением подписки, звоните по телефону

(095) 124-98-49

ИНФОРМАЦИЯ О НОВЫХ ВИРУСАХ

Мы продолжаем публикацию информации о новых вирусах в формате антивирусной программы AVSP 2.0, начатую в предыдущем номере (№4'92) "Журнала д-ра Добра". Там же приведено описание формата представления информации о вирусах.

Программу AVSP 2.0 распространяет Центр "ДИАЛОГ-МГУ".

"ДИАЛОГ-МГУ" обращается ко всем читателям с просьбой присылать информацию об обнаруженных новых вирусах или программы, зараженные новыми вирусами.

Читатели, первыми нашедшие какой-либо новый вирус, о котором отсутствует информация в программе AVSP, получают приз от Центра "ДИАЛОГ-МГУ".

Для ввода приведенной ниже информации в Вашу программу войдите в пункт меню "Данные о файлах и вирусах", подменю "Изменение информации о вирусах". Для более подробного описания можно обратиться к документации (пункт "Ввод информации о вирусах из журналов").

Имя вируса: Jews-2339
Номер: 77.1-37197
Тип вируса: PROGRAM
Размер: 2339
Шаблон: 008C8448008B844200A30600EB
Процедуры: *.SYS { Opt(St:27) Mov(End:-5,Int:0,5) Mov(St:66,Head:6,2) Tr(St:0) }
*.COM { Opt(End:-2312) Mov(End:-5,Beg:0,3) Tr(End:-2339) }
Справка: Содержит строку "Jews-2 Virus. MSU 1991". Заражает драйверы (SYS файлы).

Имя вируса: V-330
Номер: 191.1-9524
Тип вируса: PROGRAM
Размер: 330
Шаблон: # mov ax,% / mov cx,% / mov si,% / push ax / xor byte ptr [si],%
Процедуры: *.COM
Справка: Вирус не имеет постоянного шаблона. Поиск возможен только с помощью AVSP версии 1.5 и выше. Содержит закодированную строку :
"(C)1991 1.05 DM". Лечение возможно для всех файлов, о которых запомнена информация.

Имя вируса: V-198
Номер: 210.1-40092
Тип вируса: PROGRAM
Размер: 198
Шаблон: E800005E83EE0350B4E9CD21
Процедуры: *.COM { Opt(End:-198) Mov(End:-169,Beg:0,4) Tr(End:-198) }
Справка: Только размножается.

Имя вируса: Terror
Номер: 221.1-44005
Тип вируса: PROGRAM
Шаблон: 2E8C1E410550B859ECCD213BE8753E

По вопросам приобретения AVSP 2.0 обращайтесь в "ДИАЛОГ-МГУ" по телефонам (095) 939-38-90, 939-38-91.

Адрес: 119899, г. Москва, Ленинские горы, Центр СП "ДИАЛОГ" в МГУ

Системный подход к проектированию сложных систем

Проектирование больших, сложных систем требует дисциплины, особых методов и инструментальных средств, т.е. системного подхода.

М. С. Каменнова

Объем и сложность современных технических систем и систем программного обеспечения постоянно растут. Успешное функционирование таких систем решающим образом зависит от нашей способности предвидеть их создание и внедрение описанием всего комплекса проблем, связанных с дальнейшей работой: указанием функций системы, которые должны быть автоматизированы, определением интерфейсов человек-компьютер, описанием способов взаимодействия системы с ее окружением. Иными словами, этап проектирования является критическим для создания эффективных и надежных систем.

Системное проектирование - это процесс, который включает в себя формулировку требований к системе и определение ограничений, влияющих на ее функционирование, разложение системы на подсистемы, выделение на каждом уровне разложения системных компонент и описание связей между ними.

Уже в конце 60-х годов специалисты, занимавшиеся созданием сложных систем широкого профиля, стали осознавать необходимость большей упорядоченности процесса разработки таких систем. Дальнейшая формализация этого процесса привела к разбиению его на несколько этапов:

- анализ - определение функций системы;
- проектирование - определение подсистем и их интерфейсов;
- реализация - разработка подсистем и их интерфейсов;
- интеграция - соединение подсистем в единое целое;
- тестирование - проверка работы системы;
- инсталляция - введение системы в действие;
- эксплуатация - использование системы.

Однако при этом на каждом этапе применялись традиционные подходы, что обуславливало возникновение многих проблем после внедрения систем. Эксплуатационные расходы существенно превышали расходы на создание систем и продолжали стремительно расти по мере увеличения их сложности.

Многие эксперты справедливо связывали рост эксплуатационных расходов с природой ошибок, допущенных в процессе создания системы. Исследования показали, что больше всего ошибок "закладывается" на этапах анализа и проектирования и гораздо меньше их возникает на этапах реализации и тестирования. Стоимость обнаружения и исправления ошибок увеличивается по мере прохождения этапов разработки (чем более поздний этап, тем выше стоимость). Например, исправление ошибки на этапе проектирования может стоить в 2 раза, на этапе тестирования - в 10 раз и на этапе эксплуатации - в 100 раз дороже, чем на этапе анализа. На обнаружение ошибок, допущенных на этапах анализа и проектирования, тратится примерно в 2 раза меньше времени, а на их исправление - примерно в 5 раз меньше времени, чем на те же процедуры в случае ошибок, допущенных на более поздних стадиях.

Часто ошибки, возникающие на ранних этапах создания системы, являлись следствием неполноты функциональных спецификаций или несогласованности между спецификациями и проектом, выполненным по ним. Однако эксперты понимали, что это не неизбежность, а следствие неадекватности методов создания систем. Был выдвинут тезис: "Совершенствование методов проектирования и анализа есть ключ к созданию высокопроизводительных и надежных систем". В результате были разработаны новые методы, специально предназначенные для использования на ранних этапах процесса разработки сложных систем широкого профиля. Наиболее известные из них:

- метод функционального проектирования SADT Росса;
- методы, ориентированные на потоки данных: методы Йордана, ДеМарко, Гейна и Сарсона;
- методы структурирования данных: методы Джексона-Уорнера, Орра, ER (Entity-Relationship)-диаграмм Чена;
- метод декомпозиционных диаграмм для логического моделирования.

Инструментальные средства проектирования

Появление новых методов проектирования поставило задачу создания программного обеспечения, позволяющего автоматизировать их использование для проектирования больших систем. От идеи до реализации прошло 10 лет, и к середине 80-х годов сформировался рынок программных средств, названных CASE-системами.

Что же такое CASE? Сегодня этот термин трактуется весьма широко. Первоначально он расшифровывался, как Computer Aided Software Engineering (компьютерная поддержка проектирования программного обеспечения), и связано это было с автоматизацией разработки программных систем. В дальней-

Мария Сергеевна Каменнова - кандидат технических наук,

директор фирмы "МетаТехнология".

Области интересов: применение формальных методов и инструментальных средств для проектирования и анализа сложных систем и поддержки широкомасштабных проектов;

использование информационных технологий

для бизнес-анализа деятельности банков,

бирж и других коммерческих структур,

совершенствования регионального управления.

Телефон: 237-98-75

шем понятие CASE приобрело новый смысл, все чаще аббревиатуру стали расшифровывать, как Computer Aided System Engineering (компьютерная поддержка проектирования систем), а CASE-средства все больше стали ориентироваться на создание спецификаций, проектирование и моделирование сложных систем широкого назначения.

Современный рынок программных средств насчитывает более 100 CASE-продуктов, однако они занимают на нем особое положение, и прежде всего из-за их высокой стоимости. Конечно, можно встретить пакеты, называемые их авторами CASE-продуктами, по цене 300-400 дол. (чаще всего это просто редакторы диаграмм потоков данных). CASE-продукты, заслуживающие внимания, стоят не менее 3-4 тыс. дол. а некоторые - и сотни тысяч долларов. В настоящее время рынок CASE-продуктов быстро развивается и, по оценкам специалистов, в 1992 году составит более 2 млрд. дол.

Проблемы, связанные с созданием и внедрением CASE-продуктов, стали темами нескольких европейских проектов ESPRIT. С 1983 по 1991 годы Европейское Сообщество вложило около 700 млн. дол. в научно-исследовательские работы, проводимые в данном направлении. Более 200 млн. дол. выделено для аналогичного японского проекта Sigma, причем 50% объема финансирования приходится на промышленные и финансовые организации, остальная часть средств предоставлена государством.

Применение CASE-продуктов позволяет отслеживать процесс принятия решений при разработке больших проектов, систематизировать информацию о проекте и его компонентах, упрощая тем самым верификацию проекта и сопутствующей ему документации. Важным аспектом CASE-подхода является поддержка коллективной работы, при которой каждый из разработчиков создает свою подсистему, причем в любой момент времени эти подсистемы могут быть объединены в общий проект.

Применение CASE-продуктов требует от потенциальных пользователей специальной подготовки и обучения. Опыт показывает, что внедрение этих продуктов осуществляется медленно. Однако по мере приобретения пользователями практических навыков и повышения общей культуры проектирования эффективность применения этих средств резко возрастает.

Большинство из предлагаемых на рынке CASE-продуктов ориентированы на разработку программного обеспечения и охватывают обширную область поддержки многочисленных технологий программирования: от простейших графических средств документирования систем до полномасштабных средств автоматизации, обеспечивающих поддержку всего жизненного цикла создания программного продукта.

В дальнейшем мы будем исходить из предположения, что программная система - это частный случай системы вообще. Процесс создания программного обеспечения хотя и имеет свои особенности, но включает в себя практически те же стадии жизненного цикла, что и системы общего назначения. Поэтому в данной статье мне бы хотелось более подробно описать методологии и программные продукты, ориентированные на общесистемный подход.

С середины 70-х годов правительство и военные ведомства США финансировали многочисленные проекты, ориентированные на разработку методов описания и моделирования сложных систем. Одним из них явился проект ICAM (Integrated Computer-Aided Manufacturing), цель которого состояла в разработке подходов, обеспечивающих повышение эффективности производства благодаря систематическому внедрению компьютерных технологий. В соответствии с проектом ICAM было разработано семейство методологий IDEF (ICAM DEFinition), которое состоит из трех самостоятельных

методологий моделирования различных аспектов функционирования производственной среды или системы:

- IDEF0 - методология создания функциональной модели производственной среды или системы (основана на методе SADT Росса);
- IDEF1 - методология создания информационной модели производственной среды или системы (основана на реляционной теории Кодда и использовании ER-диаграмм Чена);
- IDEF2 - методология создания динамической модели производственной среды или системы.

Позднее BBC США в рамках проекта ICAM начали разработку проекта IISS (Integrated Information System Support), целью которого было создание технологии объединения (логического и физического) в сеть неоднородных вычислительных систем. Одним из практических результатов проекта IISS стало создание методологии семантического моделирования данных IDEF1X - расширения методологии IDEF1.

Методологии IDEF0 и IDEF1X являются стандартизованными. IDEF0- и IDEF1X-модели, будучи независимыми, дают адекватное и достаточно полное представление о сложной системе.

Элементы методологии IDEF0

Методология IDEF0 предназначена для структурированного представления функций системы и анализа системных требований. Она является одной из самых известных и широко используемых методологий проектирования.

Д. Росс сформулировал основные положения метода SADT почти 20 лет назад, и с тех пор системные аналитики всего мира используют этот подход для решения широкого спектра проблем разработки программного обеспечения, анализа телекоммуникационных сетей и управления ими, долгосрочного и стратегического планирования, управления автоматизированным производством, проектирования архитектур, системной поддержки и диагностики вычислительных систем, создания встроенного программного обеспечения для оборонных систем, обучения персонала, бизнес-анализа, управления финансами и материально-техническими ресурсами.

В рамках проекта ICAM метод SADT и некоторые аспекты его применения были стандартизованы, после чего получили название методологии IDEF0. С 1981 года BBC США требовали, чтобы все фирмы, конкурирующие за заключение контрактов с этим ведомством, представляли и обосновывали свои предложения в терминах IDEF0.

Почему методология IDEF0 получила столь широкое распространение? Главным образом потому, что это единственная методология, с помощью которой легко представляются такие аспекты функционирования систем, как управление, обратная связь и механизм исполнения. Это объясняется тем, что она возникла при проектировании систем более общего назначения, в отличие от других структурных методов, появившихся при проектировании систем программного обеспечения.

В терминах IDEF0 любая система представляется в виде комбинации блоков и дуг. Основной конструкцией IDEF0-модели является функциональный блок (рис.1). Блоки представляют собой функции или активные элементы системы, дуги - множество различных объектов системы. Между объектами и функциями возможны четыре вида отношений: ВХОД, УПРАВЛЕНИЕ, ВЫХОД и МЕХАНИЗМ. Каждое из этих отношений изображается дугой, связанной с определенной стороной блока.

В основе методологии IDEF0 лежат следующие правила:

- функциональный блок (или Функция) преобразует ВХОДЫ в ВЫХОДЫ (т.е. входную информацию в выходную);

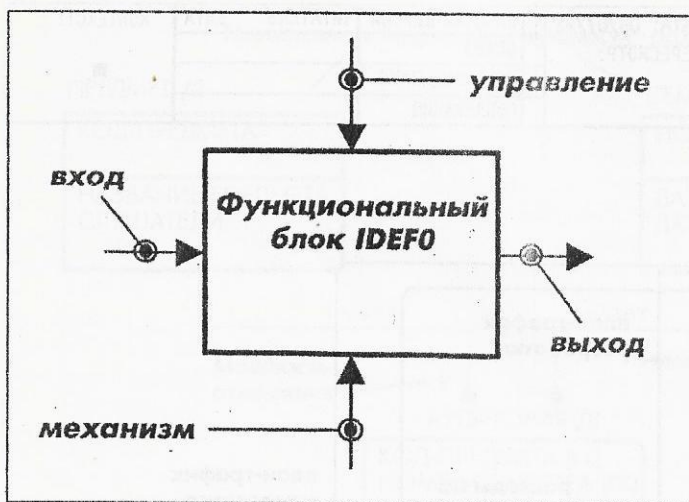


Рис.1

- **УПРАВЛЕНИЕ** определяет, когда и как это преобразование может или должно произойти;
- **МЕХАНИЗМ** непосредственно осуществляет это преобразование.

Дуги связывают блоки вместе и отображают взаимодействие и взаимное влияние блоков, которые могут выражаться либо в передаче выходной информации от одной функции к другой для дальнейшего преобразования, либо в выработке управляющей информации, которая предписывает, что должна делать другая функция. Отсюда следует, что IDEF0-модели - это ни блок-схемы, ни просто диаграммы потоков данных, а предписывающие диаграммы, которые представляют входные/выходные преобразования, а также указывают правила этих преобразований.

Каждый функциональный блок IDEF0-диаграммы может быть декомпозирован, т.е. представлен в виде совокупности других взаимосвязанных функциональных блоков, которые детально описывают исходный блок. В свою очередь, некоторые из новых блоков могут быть декомпозированы далее. Входные и выходные дуги блока определяют интерфейс между блоком и его декомпозицией (рис.2).

IDEF0-модели состоят из набора иерархически связанных IDEF0-диаграмм. Каждая диаграмма обычно содержит 3-5 функциональных блоков и является подробным описанием функционального блока, расположенного на предшествующем уровне иерархии. Пример типичной IDEF0-диаграммы, описывающей фрагмент процесса разработки программного продукта, показан на рис.3. Модель создана с помощью пакета Design/IDEF.

Элементы методологии IDEF1X

Традиционный подход к построению информационных систем основан на определении данных с двух различных точек зрения: пользователя и компьютера. Определение данных с точки зрения пользователя (т.е. внешняя схема представления данных) сводится к отображению их в различном виде на бумаге или экране компьютера, а с точки зрения компьютера (внутренняя схема представления) - к определению данных в терминах структур файлов, в которых эти данные хранятся и обновляются.

Обычно внутренняя схема представления и хранения данных, принятая для некоторой прикладной программы, не может быть без изменения использована в последующих ее модификациях, что в результате приводит к избыточ-

ным и часто противоречивым определениям одних и тех же данных.

Осознание этой проблемы привело к выводу о том, что для создания идеальной среды управления данными необходима третья точка зрения. Такую точку зрения дает концептуальная схема представления данных, которая сводится к единому (интегрированному) определению данных в рамках одного предприятия (отрасли, системы и т.д.). Определение данных в соответствии с концептуальной схемой не связано ни с каким конкретным использованием данных и не зависит от способа хранения данных и/или доступа к ним.

Важнейшая цель такого представления заключается в непротиворечивой интерпретации данных и взаимосвязей между ними, что необходимо для интеграции, совместного использования и управления целостностью данных.

Введение понятия "концептуальной схемы" привело к появлению методологии семантического моделирования данных, т.е. к определению их значений в контексте взаимосвязей с другими данными.

Методология IDEF1X - один из подходов к семантическому моделированию данных, основанный на ER-концепции (концепции "Сущность-Отношение"). Информационная модель, построенная с помощью IDEF1X-методологии, является дополнением функциональной IDEF0-модели и детализирует объекты, которыми манипулируют функции системы. Концептуально IDEF1X-модель можно рассматривать как проект логической схемы базы данных для проектируемой системы.

Компонентами IDEF1X-модели являются:

- сущности;
- отношения между сущностями;
- атрибуты сущностей.

Сущности. Каждая сущность представляет собой множество реальных или абстрактных объектов (людей, мест, событий, состояний, идей, пар предметов и т.д.), обладающих общими атрибутами или характеристиками. Отдельный элемент этого множества называется экземпляром сущности.

Сущность может иметь любое количество отношений с другими сущностями. Она называется независимой, если

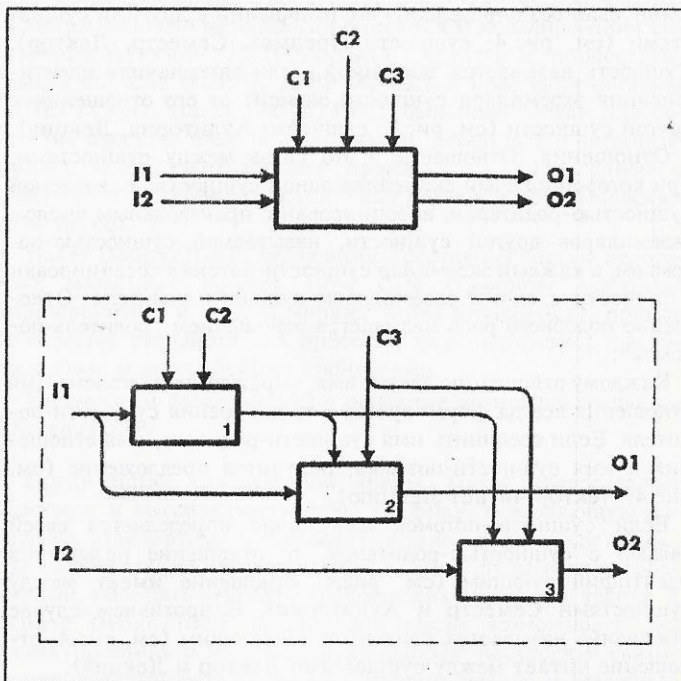


Рис.2

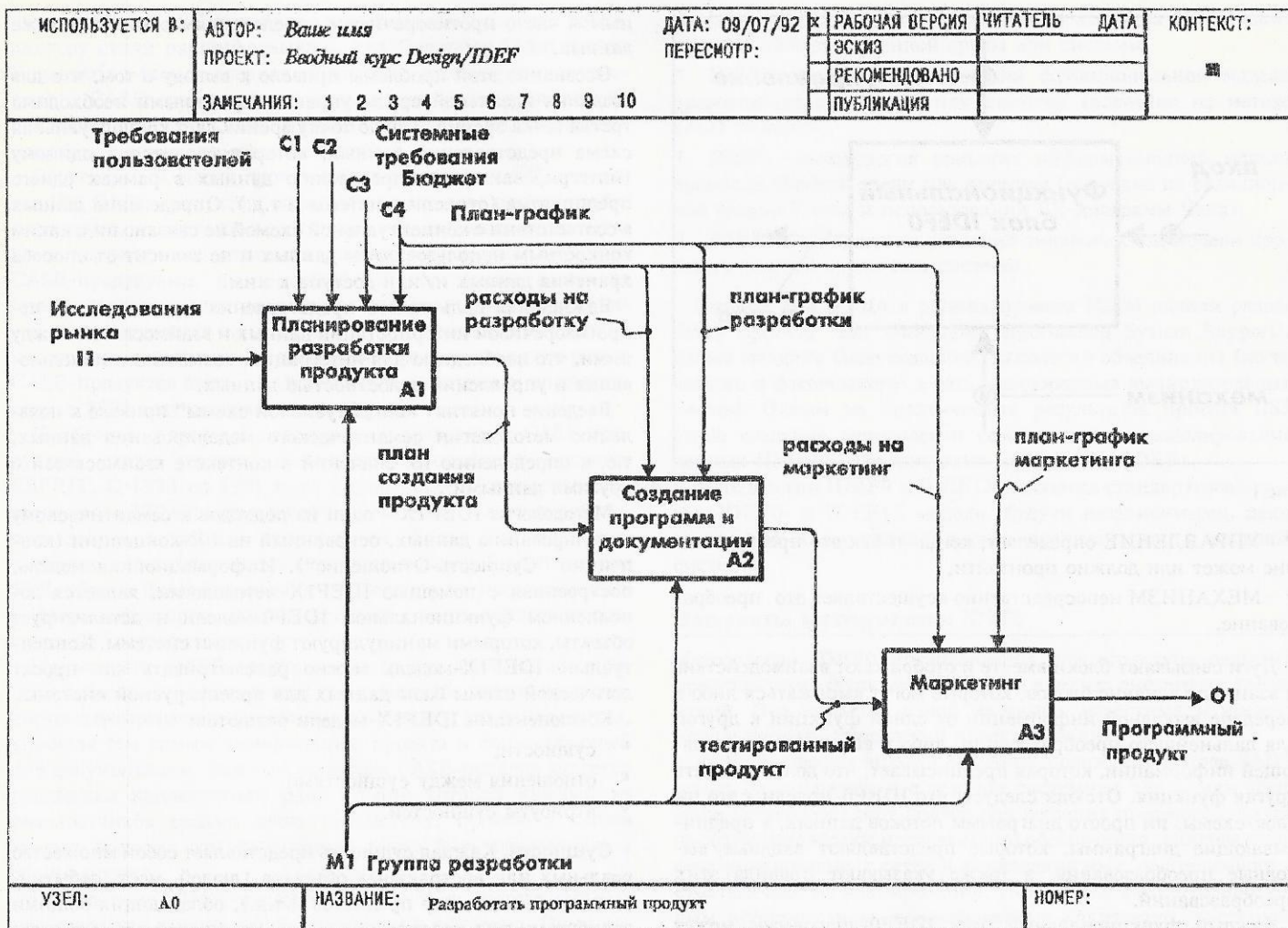


Рис.3

каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями (см. рис.4: сущности Предмет, Семестр, Лектор). Сущность называется зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности (см. рис.4: сущности Аудитория, Лекция).

Отношения. Отношение - это связь между сущностями, при которой каждый экземпляр одной сущности, называемой сущностью-родителем, ассоциирован с произвольным числом экземпляров другой сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Отношение подобного рода называется отношением "родитель-потомок".

Каждому отношению дается имя, выражаемое глаголом. Имя отношения всегда формируется с точки зрения сущности-родителя. Если соединить имя сущности-родителя, имя отношения и имя сущности-потомка, получится предложение (см. рис.4: Лектор читает Лекцию).

Если сущность-потомок однозначно определяется своей связью с сущностью-родителем, то отношение называется идентифицирующим (см. рис.4: отношение имеет между сущностями Семестр и Аудитория). В противном случае отношение называется неидентифицирующим (см. рис.4: отношение читает между сущностями Лектор и Лекция).

Отношение связи изображается линией, соединяющей сущность-родителя с сущностью-потомком, с точкой на кон-

це линии у сущности-потомка. Идентифицирующее отношение изображается сплошной линией, неидентифицирующее - пунктирной.

Так как некоторые реально существующие объекты являются категориями других реально существующих объектов, то и сущности, в некотором смысле, должны быть категориями других сущностей. Для выражения такой связи было введено "отношение категоризации", т.е. отношение между двумя или более сущностями, при котором каждый экземпляр одной сущности, называемой общей, связан в точности с одним экземпляром одной и только одной из других сущностей, называемых сущностями-категориями.

Связь родитель-потомок и отношения категоризации рассматриваются как специфические отношения, поскольку они точно определяют, как экземпляры одной сущности связаны с экземплярами другой сущности. Неспецифическое отношение, называемое также отношением "многие-ко-многим", характеризует связь между двумя сущностями, при которой каждый экземпляр первой сущности связан с произвольным (в том числе нулевым) числом экземпляров второй сущности, а каждый экземпляр второй сущности связан с произвольным (в том числе нулевым) числом экземпляров первой сущности.

Атрибуты. Сущность имеет один или несколько атрибутов, которые либо являются собственными для сущности, либо наследуются через отношение. Атрибуты однозначно идентифицируют каждый экземпляр сущности. Каждый атрибут имеет уникальное имя. Атрибуты изображаются в виде списка

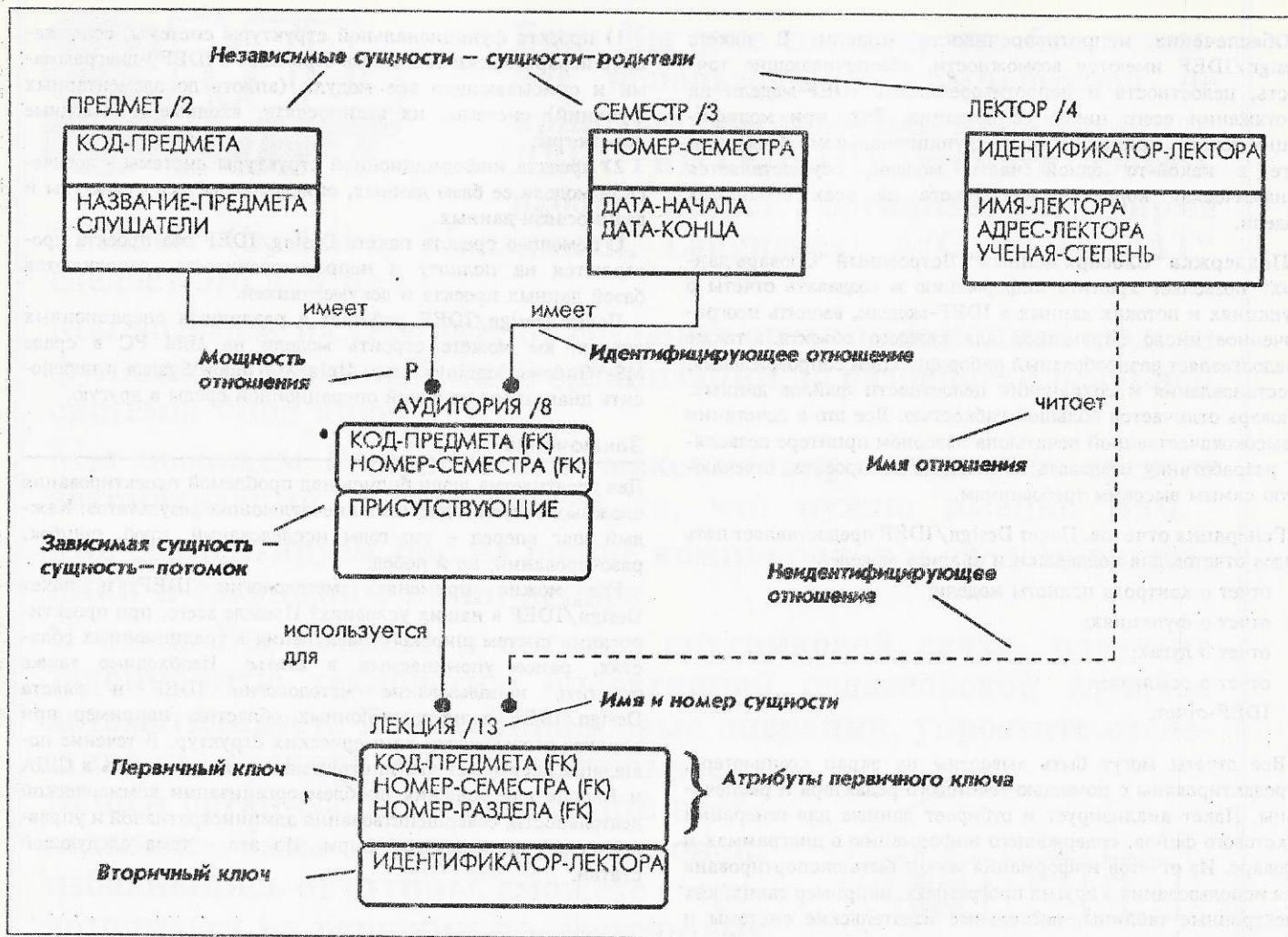


Рис.4

их имен внутри блока сущности, причем каждый атрибут занимает отдельную строку (см. рис.4).

Техника создания IDEF1X-модели заключается в выделении сущностей, определении их атрибутов и неспецифических ("многие-ко-многим") отношений между сущностями. Следуя определенным правилам, модель детализируют таким образом, чтобы все сложные виды отношений типа "многие-ко-многим" были раскрыты и заменены соответствующим множеством простых (специфических) отношений. При этом необходимо следить за правильностью наследования первичных ключей, описанием вторичных ключей и их атрибутов.

IDEF1X-модель можно считать завершенной, если:

- каждая сущность полностью определена;
- каждая сущность однозначно идентифицируется через ее атрибуты;
- все отношения преобразованы в бинарные;
- сущность появляется в модели только один раз.

CASE-пакет Design/IDEF

Методологии IDEF послужили основой для создания нескольких CASE-продуктов. Ниже приведено краткое описание пакета Design/IDEF, разработанного фирмой Meta Software (США). Выбор пакета Design/IDEF определен тем, что это один из немногих CASE-продуктов, предлагаемых на нашем рынке.

Пакет Design/IDEF - графическая среда для проектирования и моделирования сложных систем широкого назначения, поддерживающая методологии описания и моделирования системных функций (IDEF0/SADT), структур и потоков данных в системе (IDEF1, IDEF1X, ER) и поведения системы (IDEF/CPN). Пакет Design/IDEF был использован для создания проектов сложнейших систем, связанных с автоматизацией и компьютеризацией производства, управлением и контролем в телекоммуникационных сетях и аэрокосмических комплексах.

Пакет Design/IDEF, являющийся составной частью некоторых известных пакетов типа CIM (Computer Integrated Manufacturing) и CAE (Computer Aided Engineering), принят в качестве стандарта для проектов, финансируемых американскими и европейскими спонсорами.

Рассмотрим более подробно основные возможности пакета Design/IDEF.

Организация графической среды. Пакет предоставляет быструю и высококачественную графику, позволяющую создавать стандартные и пользовательские графические объекты, осуществлять выравнивание и манипулирование объектами, выбирать атрибуты графических объектов и текста. В пакете Design/IDEF реализованы также возможности для редактирования и моделирования данных: построение связывающих линий типа "резинка", маршрутизация и сглаживание дуг и т.д.

Обеспечение непротиворечивости модели. В пакете Design/IDEF имеются возможности, обеспечивающие точность, целостность и непротиворечивость IDEF-модели на протяжении всего цикла ее создания. Так, при модификации текста, принадлежащего функциональному блоку или дуге в какой-то одной части модели, осуществляется динамическая корректировка текста на всех страницах модели.

Поддержка "Словаря данных". Встроенный "Словарь данных" позволяет хранить информацию и создавать отчеты о функциях и потоках данных в IDEF-модели, вводить неограниченное число параметров для каждого объекта, а также предоставляет разнообразный набор функций сопровождения, восстановления и сохранения целостности файлов данных. Словарь отличается большой гибкостью. Все это в сочетании с высококачественной печатью на лазерном принтере позволяет разработчику создавать документацию проекта, отвечающую самым высоким требованиям.

Генерация отчетов. Пакет Design/IDEF предоставляет пять видов отчетов для поддержки и анализа моделей:

- отчет о контроле полноты модели;
- отчет о функциях;
- отчет о дугах;
- отчет о ссылках;
- IDEF-отчет.

Все отчеты могут быть выведены на экран компьютера, отредактированы с помощью текстового редактора и распечатаны. Пакет анализирует и отбирает данные для генерации текстового файла, содержащего информацию о диаграммах и словаре. Из отчетов информация может быть экспортирована для использования в других программах, например таких, как электронные таблицы, настольные издательские системы и текстовые редакторы.

Организация коллективной работы. Пакет Design/IDEF поддерживает работу многочисленной группы разработчиков, создающих одновременно большую и сложную IDEF-модель. Каждый разработчик проектирует свою подмодель. Подмодели легко интегрируются в одну большую модель.

Моделирование данных (IDEF1-, IDEF1X- и ER-методологии). Пакет Design/IDEF позволяет создавать информационные модели, которые представляют как собственно данные, так и связи между ними в системе.

Информация, содержащаяся в IDEF-моделях, экспортируется в любую базу данных, а сами модели могут быть экспортированы в Design/CPN - пакет динамического моделирования и анализа сложных систем.

Как CASE-продукт по разработке программного обеспечения пакет Design/IDEF поддерживает первые этапы создания программного продукта, которыми являются:

- формулировка требований и целей проекта;
- разработка спецификаций (формализованное описание требований);
- создание проекта (определение подсистем и их взаимодействий);
- документирование проекта (создание базы данных проекта, текстовое описание составных частей проекта);
- анализ проекта (проверка проекта на полноту и непротиворечивость).

Результатом работы пакета Design/IDEF является проект программной системы, состоящий из двух частей:

1) проекта функциональной структуры системы, содержащего иерархически связанные страницы с IDEF0-диаграммами и описывающего все модули (вплоть до элементарных функций) системы, их взаимосвязи, входные и выходные параметры;

2) проекта информационной структуры системы - логической модели ее базы данных, описывающей все структуры и взаимосвязи данных.

С помощью средств пакета Design/IDEF оба проекта проверяются на полноту и непротиворечивость, дополняются базой данных проекта и документацией.

Пакет Design/IDEF работает в различных операционных средах: вы можете строить модели на IBM PC в среде MS-Windows, Macintosh или Unix X Window System и переносить диаграммы из одной операционной среды в другую.

Заключение

Два десятилетия люди бились над проблемой проектирования сложных систем и достигли впечатляющих результатов. Каждый шаг вперед - это годы исследований, проб, ошибок, разочарований, но и побед.

Где можно применять методологии IDEF и пакет Design/IDEF в наших условиях? Прежде всего, при проектировании систем широкого назначения в традиционных областях, ранее упоминаемых в статье. Необходимо также отметить использование методологии IDEF и пакета Design/IDEF в нетрадиционных областях, например при анализе деятельности коммерческих структур. В течение последних десяти лет эти инструменты использовались в США и Европе для решения проблем организации коммерческой деятельности, совершенствования административной и управленческой деятельности фирм. Но это - тема следующей статьи.

факты

- Клод Леглиз, глава отдела маркетинга средств Multimedia и суперкомпьютеров фирмы Intel, заявил, что программное обеспечение, разработанное фирмой Xing Technology, в сочетании с процессором i750 представляет собой достаточно дешевое средство для показа на большом экране видеофайлов в формате MPEG с частотой 30 кадр/с.
- 10 ноября 1992 г. СП ПараГраф объявило о подписании им лицензионного соглашения с американской фирмой Adobe Systems. Данное соглашение предоставляет СП ПараГраф право распространять на территории СНГ последнюю версию пакета ATM (Adobe Type Manager) в составе пакетов, созданных в СП ПараГраф. Это соглашение продолжает линию сотрудничества между фирмой Adobe Systems и СП ПараГраф. По заключенному ранее соглашению фирма Adobe Systems лицензировала у СП ПараГраф утилиту ParaWin - средство поддержки многоязычной работы в среде Windows.



Если Вы уже поняли, что бессистемное использование компьютеров не приносит Вам прибыли и/или решили **ДЕЙСТВИТЕЛЬНО** автоматизировать работу Вашей организации, фирма

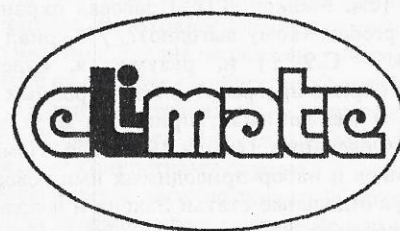
SMLsoft (Systems, Managment, Languages) А/О "КЛИМАТ" сделает это для Вас.

Мы возьмем на себя все: от создания оптимизационной модели функционирования Вашей организации до разработки реальной системы "под ключ".

Мы поможем Вам получить максимальную отдачу от средств автоматизации и приобрести то, что нужно именно Вам.

Мы сориентируем Вас на рынке компьютеров и средств оргтехники и сэкономим Ваши деньги.

С нами Вы войдете в систему спутниковой связи, получите законченную технологию оперативной издательской службы, автоматизируете Ваши финансовые операции, упростите отслеживание "узких" мест в деятельности Вашей организации, приобретете необходимые базы данных и, наконец, избавившись от рутины, сможете сосредоточиться на решении по-настоящему содержательных вопросов.



Наш адрес:

117261, Москва, Ленинский проспект, 72

Телефон: (095) 939-59-93 • Факс: (095) 939-59-91

Электронная почта: sml@klimat.msk.su



ИЗДАТЕЛЬСТВО "ТРИАДА" ПРИГЛАШАЕТ РЕКЛАМНЫХ АГЕНТОВ
И ВСЕХ ЖЕЛАЮЩИХ

ПРИНЯТЬ УЧАСТИЕ В РАСПРОСТРАНЕНИИ ЕГО ТИРАЖЕЙ.

ОПЛАТА ДОГОВОРНАЯ.

ТЕЛЕФОН: (095) 124-98-49 • ФАКС: (095) 124-70-21

Интересы индустрии программирования и свобода личности

А. Н. Козырев

Наилучшей из хозяйственных форм, как бы она ни называлась и какую бы комбинацию капитализма и социализма, частной и общественной собственности она ни определяла, является та, которая наиболее обеспечивает свободу как от природной бедности, так и от социальной неволи.

о. С. Булаков. Православие. Очерки учения православной церкви

Разделение прав на программный продукт, созданный в порядке выполнения служебных обязанностей, - ключевой вопрос в законодательстве о правовой охране программного обеспечения (ПО). Решив его в пользу авторов или работодателей, законодатель неизбежно подвергается ожесточенной критике, причем каждый раз "потерпевшая" сторона утверждает, что в создавшихся условиях отечественной индустрии программирования грозит деградация или гибель. Так было с разделом 4 "Авторское право" "Основ гражданского законодательства" [см. Малков Л.П. Правовая охрана программ в СССР: cui prodest (кому выгодно)? // Журнал д-ра Добба. - 1991. - №3. - С.9-11] и, разумеется, будет с Законом "О правовой охране программ для электронных вычислительных машин и баз данных", принятым Верховным Советом Российской Федерации 14 мая 1992 года. Изменяется только состав критиков и набор приводимых ими доводов.

Анализируя отдельные статьи Закона и доводы противоборствующих сторон, я хочу показать, что наихудший вариант охраны ПО - отсутствие какого-либо действующего законодательства в данной области, причем такая альтернатива вполне реальна. Вопрос же о разделении прав выходит за рамки чисто экономической проблематики, но именно для эффективности отрасли значение его не столь уж велико.

Противоречия интересов, экономическая теория и право

Начнем с высших ценностей, составляющих смысл всякого дела, чтобы не превратить в фетиш сиюминутные интересы программистов, предпринимателей или всей индустрии программирования и не упустить из поля зрения более высокие цели. Здесь я вполне солидарен с о. С. Булгаковым, считающим высшей ценностью свободу личности, правовую и хозяйственную, и хочу в свете этой ценности сравнить разные хозяйственные формы и правовые нормы. Надо заметить, что сходной позиции придерживались английские либералы - "старые виги", никогда не ставившие экономическую эффек-

тивность выше личной свободы [см. Хайек Ф.А. Пагубная самонадеянность. - М.: Новости, 1992].

Первое, на что следует обратить внимание, - это симметрия в отношении к программному продукту со стороны его авторов, работающих по найму, и работодателя. Как авторы, так и работодатель заинтересованы в сотрудничестве, причем конфликт и разрыв между ними в большинстве случаев означают, что данный продукт просто не будет создан. Объединившись вместе на договорной основе, авторы и работодатель располагают полным объемом прав на программный продукт, который тем больше, чем шире определен объект охраны. В этих рамках любое расширение прав авторов оборачивается ущемлением прав работодателя и наоборот. Аналогичная симметрия, замеченная Р. Коузом [см. Coase R.H. The problem of social cost // The Journal of Law and Economics. - October 1960, Vol.3. - P.1-44] в большом количестве не связанных между собой (на первый взгляд) примеров, подтолкнула его к пересмотру взглядов на отношения собственности и общественные блага. Работа Р. Коуза послужила исходной точкой для создания современной теории собственности, за которую в 1991 году он получил Нобелевскую премию. Впрочем, даже без обращения к теории, а основываясь лишь на соображениях симметрии, можно догадаться, что процветание индустрии программирования зависит не от распределения прав внутри отрасли, а от их общего объема и умения сторон договориться.

Ключевую роль в теории прав собственности играет тезис, получивший название "Теорема Коуза", хотя в строгом смысле слова теоремой он, конечно, не является. Смысл Теоремы Коуза состоит в том, что распределение прав собственности между конкретными физическими и юридическими лицами не имеет значения для эффективности общественного производства в целом, а влияет только на распределение доходов между этими лицами. Вместе с тем отсутствие собственника обычно приводит к негативным последствиям.

Применительно к экономике "знаний" теория прав собственности [см. Adelstein R.P., Peretz S.I. The Competition of technologies in markets for ideas: Copyright and fair use in evolutionary perspective // International Review of Law and Economics. - 1985. - №5. - P.209-238] отвергает как популярную в прошлом идею свободного распространения интеллектуальных продуктов, в том числе ПО, так и прямое государственное вмешательство. Она позволяет пойти еще дальше, придавая первостепенное значение всевозможным ограничениям свободного распространения ПО и отводя второстепенную роль вопросу о распределении имущественных прав

Анатолий Николаевич Козырев - старший научный сотрудник ЦЭМИ АН СССР, кандидат физико-математических наук. Диссертацию по математической теории экономического равновесия защитил в 1984 г. Член Эконометрического общества. Автор более 30 научных работ по кибернетике и экономике. Научное хобби - социально-экономические и правовые аспекты программирования.

между лицами, причастными к его созданию, т.е. между авторами и работодателями. Только искусственно созданные препятствия для свободного распространения ПО, одним из которых можно считать правовую охрану, позволяют создать рынок таких продуктов, обеспечивающий эффективное разделение труда и возмещение инновационных затрат. Более того, согласно данной теории эффективность отрасли и экономики в целом не зависит от конкретного распределения имущественных прав. Важен только сам факт, что у каждого продукта есть правообладатель, наделенный правом разрешать или запрещать использование этого продукта другими экономическими агентами.

Применительно к экономике программирования Теорема Коуза означает, что для эффективного функционирования данной отрасли необходимо распределить имущественные права на производимый продукт между различными участниками производства, причем сам факт наличия прав важнее их распределения. Иными словами, наибольшее из возможных зол - свободное распространение программного продукта (разумеется, исключая отсутствие всякого распространения). Все другие варианты более предпочтительны, причем достоинство их тем выше, чем шире сфера договорных отношений по поводу имущественных прав и чем уже сфера свободного распространения.

Вывод из сказанного почти очевиден. Наибольшее значение для развития индустрии программирования имеют те статьи Закона, которые определяют область его применения и объем распределяемых прав, а также статьи, определяющие меру ответственности за нарушение этих прав. Льготы отдельным категориям пользователей, ограничения сферы применения Закона и желанная для многих поэтапность его введения в действие объективно снижают общий положительный эффект, хотя и позволяют решить многие частные вопросы.

Однако такой вывод правомерен лишь для индустрии программирования в целом, а не для какой-то национальной экономики. Он был бы правомерен и для российской индустрии программирования, если бы она была изолирована от индустрии программирования других стран, либо доминировала на мировом рынке ПО, как американская индустрия программирования.

Тем не менее первостепенное значение имеет объем распределяемых прав, который зависит от определения объекта правовой охраны и от наличия в Законе дополнительных указаний, позволяющих толковать Закон более или менее широко.

Объект охраны

Основной объект правовой охраны (программа для ЭВМ) определен в Законе как "объективная форма представления совокупности данных и команд, предназначенных для функционирования электронных вычислительных машин (ЭВМ) и других компьютерных устройств с целью получения определенного результата". Термин "программа для ЭВМ" включает в себя также "подготовительные материалы, полученные в ходе ее разработки, и порожденные ею аудиовизуальные отображения" (статья 1, пункт 1). В цитируемом определении программы фактически обозначена только охраняемая Законом часть определяемого объекта, т.е. объективная форма представления. Другим объектом правовой охраны является база данных, определяемая как "объективная форма представления и организации совокупности данных ...", т.е. определение строится по тому же принципу, что и определение программы.

В определениях подчеркивается, что Закон охраняет объективную форму представления данных и команд, но не сами данные или команды. Столь явный упор на "объективную

форму представления" вполне оправдан, так как не допускает размытия границ авторского права, т.е. не оставляет ложных иллюзий по поводу охраноспособности заложенных в программах идей, алгоритмов и системотехнических решений. Тем самым в корне пресекаются попытки расширительно толковать Закон, целиком лежащий в русле авторского права и конкретизирующий его нормы применительно к новому объекту, но не меняющий сути самого авторского права. В остальном же объект охраны определен предельно широко, т.е. настолько широко, насколько позволяют границы авторского права.

Авторское право распространяется на любые программы для ЭВМ и базы данных, как выпущенные, так и не выпущенные в свет, представленные в объективной форме, независимо от их материального носителя и достоинства (статья 3, пункт 1). Наличие творческой деятельности автора предполагается до тех пор, пока не доказано обратное, т.е. основные условия охраноспособности в рамках авторского права априори считаются выполненными.

Предоставляемая Законом "охрана не распространяется на идеи и принципы, лежащие в основе программы для ЭВМ, базы данных или какого-либо их элемента, в том числе на идеи и принципы организации интерфейса и алгоритма, а также языки программирования" (статья 3, пункт 5). Именно таким образом границы охраноспособности программ определены в "Директиве ЕС по правовой охране компьютерных программ" (DIRECTIVE ON THE LEGAL PROTECTION OF COMPUTER PROGRAMS), принятой 14 мая 1991 года. Существенно здесь то, что явно признаны неохраноспособными лишь идеи алгоритмов. Об охраноспособности самих алгоритмов ни в Директиве, ни в Законе не сказано ничего. Предельно деликатный вопрос об охраноспособности объективной формы представления алгоритмов оставлен на усмотрение судов, которым предстоит разбирать конкретные дела и выносить решения по каждому конкретному случаю.

Рассуждая теоретически, логично считать объективной формой представления заложенного в программе алгоритма саму программу. При такой интерпретации вопроса ответ на него будет положительным, причем речь может идти не только о программе в объектных кодах или в виде исходного текста, но и о подготовительных материалах. Последнее фактически означает охраноспособность алгоритма, формализованного до такой степени, когда написание программы сводится к кодированию. Считать программиста-кодировщика единственным автором программы, разумеется, было бы ошибкой. Не меньше оснований считаться автором программы у того, кто готовит задания для кодировщиков. Однако провести точную границу между программированием-кодированием и программированием как творчеством не так-то просто. Следовательно, вопрос об охраноспособности формы представления алгоритма естественно перерастает в вопрос о соавторстве. Его однозначное и окончательное решение в рамках Закона в принципе невозможно. В конечном счете это всегда предмет соглашения заинтересованных сторон либо предмет конфликта и судебного разбирательства между ними.

Разумеется, вопрос о соавторстве теряет какую-либо актуальность, если все права на создаваемый программный продукт принадлежат работодателю. В определенном смысле такое решение оптимально, так как вполне согласуется с Теоремой Коуза и, кроме того, снимает вопрос о соавторстве вместе с возможными конфликтами на его почве.

Несколько сужают сферу применения Закона льготы, предоставляемые Постановлением о введении Закона в действие отдельным категориям пользователей.

Согласно статье 4 Постановления "до 1 января 1994 года на территории Российской Федерации использование в научно-

исследовательских, учебно-просветительских и личных целях без извлечения прибыли программ для электронно-вычислительных машин и баз данных, выпущенных в свет (опубликованных) до вступления в силу указанного Закона, допускается без согласия автора (правообладателя) и без выплаты ему вознаграждения". Таким образом, несанкционированное использование ПО без извлечения прибыли (в некоторых случаях) оказывается даже законным.

В силу ограниченности срока действия предоставляемых льгот и распространения их только на бесприбыльное использование ПО, причем в ограниченных целях, можно говорить лишь о попытке или о "тени" компромисса между желанием ввести правовую охрану ПО и желанием избежать резких изменений.

Реальным компромиссом при введении Закона в действие следует считать путь, по которому пошли Китай и Польша. В этих странах копии программ, созданные до введения правовой охраны ПО, можно продолжать использовать и далее. Таким образом смягчаются негативные последствия введения правовой охраны, но ослабляются позитивные. Чтобы оценить приемлемость такого компромисса для различных экономических агентов, т.е. для фирм-разработчиков ПО, программистов и различных категорий пользователей, необходим достаточно подробный анализ реального использования ПО. Если возможности для такого анализа нет, то приходится делать выводы на основании субъективных оценок, базирующихся только на личном опыте. Впрочем, и при таком подходе многие последствия достаточно очевидны. Кроме того, очевидны технические сложности при выяснении того, когда изготовлена каждая конкретная копия программы, опубликованной до введения Закона в действие.

Пиратство наоборот и другие законные способы заработать

Жесткость мер, применяемых к "пиратам", - обязательное условие действенности Закона. В этом отношении российский Закон выгодно отличается от "Основ гражданского законодательства", упоминавшихся выше.

Согласно пункту 1 статьи 18 Закона в случае нарушения прав автора с целью извлечения прибыли вместо возмещения убытков он может требовать выплаты компенсации в сумме не менее 100 тыс. руб. и не более 2 млн. руб., определяемой по усмотрению суда, арбитража или третейского суда. Важно, что извлечение прибыли усугубляет возможную меру ответственности. Постановление о введении Закона в действие также содержит указание на возможность различного подхода к несанкционированному использованию программ с целью извлечения прибыли и с другими целями.

Выявление правонарушителей, извлекающих прибыль, и применение к ним более жестких санкций позволяют сделать "рентабельной" борьбу с компьютерным пиратством. Вполне вероятно появление в ближайшем будущем специальных фирм, охраняющих права поставщиков ПО. Такие фирмы могут за приемлемую плату выявлять нарушителей и заставлять их приобрести используемое ПО законным путем у правообладателя или его агентов. Фирма данного профиля может совмещать свои основные функции с функциями агента по распространению ПО, получая таким образом дополнительную прибыль. Оплата услуг фирмы должна строиться по сделному принципу, т.е. в виде долей от получаемых штрафов и комиссионных от проданного ПО. Речь идет в сущности о пиратстве наоборот, которым вполне могут заняться сегодняшние "пираты" [см. Шершульский В. О духе и букве одного Закона // Компьютеруолд - Москва. - 1992. - №16(28). - С.1, 8-10]. Поскольку в настоящее время среди используемых на

территории России копий программ не менее 95% пиратские, работа таких фирм может быть на первых порах очень эффективной. В результате может резко сократиться число фирм, использующих пиратские копии для извлечения прибыли, следовательно, несколько расширится рынок ПО. По отношению к остальным незаконным пользователям аналогичные меры не могут быть применены, так как их труднее обнаружить.

В статье 4 Постановления о введении Закона в действие указаны случаи, в которых можно использовать ПО без разрешения автора и без выплаты ему вознаграждения. Речь идет о достаточно коротком периоде (до 1 января 1994 года), что само по себе снижает значимость компромисса. Кроме того, речь идет о ПО, опубликованном к моменту введения Закона в действие, причем об использовании его только в определенных целях (научно-исследовательских, учебно-просветительских или личных) без извлечения прибыли. Такие льготы не могут привести к сколько-нибудь существенному сужению рынка охраняемого ПО в целом, но могут повлиять на ситуацию в некоторых специфических его секторах.

То обстоятельство, что изготовление новых копий ранее опубликованных программ не считается правонарушением, если его целью не является извлечение прибыли, можно считать курьезом. Фактически разрешено вплоть до 1994 года бесплатно распространять ранее опубликованные программы, нанося ощутимый вред производителям ПО. В частном порядке такая деятельность, вероятно, будет продолжаться и после 1 января 1994 года, когда перестанет быть законной. Однако суть курьеза не в этом, а в принципиальной возможности извлечения прибыли из внешне аналогичной деятельности. Возможно появление фирм, "бесплатно" распространяющих чужие обучающие программы и оказывающих сопутствующие услуги за достаточно высокую цену, причем с оплатой заказчиком командировочных расходов. При действующих ценах на школьные обучающие программы такой бизнес может обеспечивать вполне приемлемый доход, не меньше доходов фирм и коллективов, разрабатывающих оригинальные программы. Другим курьезом является то, что от ответственности за несанкционированное использование ранее полученного ПО не освобождаются правительственные организации, органы Верховного Совета и т.д. Однако все эти органы и организации вполне могли позаботиться о своем лице заранее, так как использование ПО без санкции правообладателя давно уже официально осуждается в нашей стране, хотя и не преследовалось вплоть до принятия Закона.

Практический выход из щекотливого положения, в котором оказались многие бюджетные организации, включая властные структуры и различные координирующие органы, следует искать на пути компромисса в области цен. По всей видимости, бюджетным организациям придется обращаться к правообладателям и получать лицензии на приемлемых для обеих сторон условиях.

Расширение спроса на ПО, вызванное быстрым введением Закона и его жесткостью, наиболее выгодно тем фирмам, которые ранее ориентировались на разработку лицензионно чистого ПО. Даже с учетом того, что таких фирм у нас абсолютное меньшинство, для программистов быстрое введение Закона в действие может быть выгодно. Спрос на их услуги должен увеличиться даже в случае разорения большинства ныне действующих фирм.

Распределение имущественных прав на ПО и свобода

По российскому Закону имущественные права на ПО, созданное в порядке выполнения служебных обязанностей или

по заданию работодателя, принадлежат работодателю, если в договоре между ним и автором не предусмотрено иное (статья 12, пункт 1). Такой подход полностью согласуется с "Директивой ЕС" и духом проводимых в стране реформ, но расходится с характерной для советского авторского права традицией приоритета прав личности. Не вызывает сомнений необходимость сближения российского законодательства с законодательствами стран-членов ЕС, но было бы наивно и даже не вполне добросовестно не замечать очевидных негативных последствий разрыва с традицией, не совсем обоснованно именуемой "доморощенной" [см. Малков Л.П. Закон о правовой охране программ как вызов отрасли // Журнал д-ра Добра. - 1992. - №3. - С.8-12].

Опасность всеобщего разброда и развала как результат слабой защищенности работодателей в индустрии программирования действительно существует и осознается самими программистами, а не только экономистами и администрацией фирм. Обсуждая этот вопрос с П.Н. Меняйло (основным разработчиком пакета "Five star"), Е.Н. Веселовым и другими достаточно известными программистами, я был даже немного удивлен их готовностью идти на значительные уступки в области личных прав, если это идет на пользу качеству продукта. Например, Е.Н. Веселов считает, что "Мастер" и "Лексикон" могли бы быть существенно лучше, если бы эти продукты были отчуждены от него (автора) на достаточно ранней стадии и дорабатывались коллективом программистов с различной специализацией. Слишком ограниченные бывшим советским законодательством права фирм не позволяли создать устойчивый работоспособный коллектив. Следовательно, расширение прав работодателя в целом соответствует интересам развития отрасли.

Опасность, исходящая с другой стороны, обычно осознается на стадии конфликта между фирмой-работодателем и увольняемым программистом. При заключении трудового контракта и приеме на работу любое упоминание о возможном конфликте в будущем воспринимается как бестактность и недоверие к партнеру, поэтому принимаются либо типовые для данной фирмы условия, либо нормы Закона, действующие по умолчанию. Срабатывает же статья 12 только тогда, когда у одной из сторон возникнут претензии к партнеру. Российский закон в этом случае оказывается на стороне работодателя, причем не только закон как таковой.

Следует также иметь в виду разницу возможностей нанимателя и нанимаемого в случае конфликта, обеспечивающую первому огромные преимущества. Как правило, наниматель располагает существенными материальными ресурсами, штатными юристами и консультантами по различным вопросам. Наемный работник после увольнения сразу оказывается в достаточно тяжелом положении, которое быстро ухудшается, если он не направляет усилия на поиски нового места работы. Потребовать сверх "разумной меры" за ранее оплаченную (не полностью) работу достаточно опасно, так как может последовать встречное обвинение в вымогательстве. К тому же интересы фирмы (работодателя) в любом судебном разбирательстве будет отстаивать профессиональный юрист, тогда как уволенный программист всюду должен поспевать сам. Оценить тяготы подобного состязания способен тот, кто испытал его на себе.

Правом может воспользоваться далеко не всякий, а реально пользуются в основном сильные, причем даже в очень демократичной и цивилизованной стране [см. Foster E. Can the legal system do what the industry cannot? (Надежда только на юристов) // Компьютеруолд - Москва. - 1992. - №16(28). - С.1,11]. Судебные процессы между крупнейшими американскими фирмами демонстрируют отнюдь не только непрерывные поиски все более правильных решений, но и стремление

монополизировать отдельные сегменты рынка ПО, не особенно заботясь ни об интересах потребителя, ни о средствах. Законодательство США об авторском праве, позволяющее отчуждать даже само авторство, вполне благоприятствует устремлениям гигантов индустрии программирования и, разумеется, ими поддерживается.

Опасность злоупотребления со стороны работодателя своей реальной властью - это угроза свободе личности программиста, тогда как опасность злоупотребления свободой со стороны программиста - угроза власти и экономическому могуществу работодателя. В моей иерархии ценностей свобода личности выше экономического могущества, поэтому сводить вопрос к интересам отрасли я не намерен. И все же к этому вопросу придется вернуться еще раз в связи с понятием естественной монополии и проблемой личных неимущественных прав, к числу которых относятся право на имя и на целостность (неприкосновенность) произведения.

Естественная монополия и личные неимущественные права

Один из основных доводов в пользу отчуждаемости личных неимущественных прав - огромные успехи индустрии программирования в США, где отчуждается даже само авторство. Для такой точки зрения имеются определенные основания, хотя наивно видеть причину чужого успеха лишь в более совершенном законодательстве, как и в любом другом факторе, вырванном из контекста. Правильнее говорить лишь о наблюдаемых тенденциях и субъективных оценках. При такой постановке вопроса элементарная добросовестность требует указать на целый ряд противоречий, органически присущих любой наукоемкой отрасли, но особенно ярко проявившихся именно в индустрии программирования.

В глобальном плане это противоречие между тенденциями к демополизации и к естественной монополизации отрасли, которое проявляется в частных противоречиях. Стремление к демополизации проявляется не только в дроблении рынка и появлении конкурирующих программных продуктов, но и в разделении прав на любой программный продукт, причем разделение прав может быть как добровольным, так и принудительным. Противоположная тенденция также имеет два аспекта. Один из них (внешний) правовой, другой (более глубокий) экономический.

Термин "естественная монополия" не случаен и широко используется в западной экономической литературе для обозначения монополии, возникающей в силу объективной необходимости. Такой причиной может быть, например, исключительное право на распространение данного программного продукта или патент на производство какого-то устройства, т.е. источником монополии оказывается право интеллектуальной собственности.

Среди экономических причин естественной монополии можно выделить по крайней мере три момента: 1) возрастание эффективности при увеличении масштабов производства; 2) наличие внешних влияний в потреблении продукта; 3) потребность в стандартизации интерфейса и совместимости различных продуктов между собой.

Все три момента присутствуют в индустрии программирования, причем достаточно легко идентифицируются и увязываются с нормами авторского права. Возрастающая эффективность индустрии программирования объясняется большими затратами на разработку программного продукта при относительно низких затратах на его тиражирование.

Чтобы покрыть расходы на разработку и получить прибыль, необходимы либо высокие цены, либо большие тиражи, что достигается наилучшим образом при полной монополизации рынка одной фирмой. Исключительное право фирмы на программный продукт, если оно обеспечивается законодательством, решает именно эту задачу.

Разделение прав между фирмой и реальными авторами продукта ослабляет положение фирмы, но увеличивает потенциальную возможность конкуренции. В частности, появляется возможность создания конкурирующего продукта теми же авторами, но под другой вывеской. Например, они могут создать собственную фирму. Успех новой фирмы едва ли не в первую очередь будет зависеть от того "капитала", с которым она встанет на путь конкурентной борьбы, причем существенную часть "капитала" может составить известное имя. И, разумеется, это может быть только имя программиста, а не имя фирмы.

Внешние влияния наиболее заметны при покупке популярного программного продукта, когда есть конкурирующий продукт, еще не успевший стать популярным. Дополнительными стимулами к покупке популярного продукта оказываются такие соображения, как возможность проконсультироваться у приятеля, уже имеющего этот продукт, и т.п. Еще более ощутимы внешние влияния при эксплуатации компьютерных сетей и соответствующего программного обеспечения. Полностью реализовать положительный эффект внешних влияний в потреблении можно либо при монополизации рынка одной фирмой, либо при полной стандартизации интерфейсов.

Стандартизация может быть альтернативой монополии, как и ее следствием. В случае распространения интеллектуальной собственности на человеко-машинный интерфейс возникает противоречие между интересами фирмы правообладателя и общими интересами отрасли в распространении данного интеллектуального продукта в качестве общеотраслевого фактического стандарта [см. Besen S.M. and Raskind L.J. An introduction to the law and economics of intellectual property// Journal of Economic Perspectives. - Winter 1991. - Vol.5. - N1. - P.3-27]. Таким образом, даже поверхностный анализ проблемы стандартизации обнаруживает противоречивость интересов отрасли и отдельных фирм.

О губительном влиянии монополий на развитие высокотехнологичных отраслей написано столько, что нет смысла развивать эту тему. Для меня по целому ряду причин было важно показать именно положительную сторону монополизации в программировании. Во-первых, это подводит некоторую базу под внешне парадоксальный прием, когда в интересах развития отрасли запрещается свободное использование чужих интеллектуальных достижений. А именно таким образом стимулирует развитие программирования охрана компьютерных программ авторским правом. Во-вторых, выявляется множество частных противоречий, разрешимых только через компромисс, который может быть найден лишь в результате специальных исследований. Пока же рано давать оценки правовым нормам. И в-третьих, еще раз иллюстрируется старая истина о "благих намерениях, которыми вымощена дорога в ад", на этот раз в ад всеобщей и полной монополизации.

Перспективы

В момент завершения этой статьи ситуация с правовой охраной ПО все еще не вполне ясна. Пакет из четырех законов об интеллектуальной собственности, куда входят законы Российской Федерации "О правовой охране программ для ЭВМ и баз данных", "О правовой охране топологий интегральных микросхем" и "О товарных знаках, знаках обслуживания и мест

происхождения товаров", принятые 14 мая 1992 года, а также Патентный закон, принятый 18 июня 1992 года, был возвращен Б.Н. Ельциным в Верховный Совет для повторного рассмотрения. Причина - противоречие порядка введения всех четырех законов недавно принятой поправке к Конституции, относящей вопросы интеллектуальной собственности к совместному ведению Российской Федерации и входящих в нее республик.

Повторное рассмотрение пакета законов в июле 1992 года не состоялось, так как депутатам не хватило времени. Обсуждение вопроса было перенесено на осень, и 23 сентября 1992 года все четыре закона были приняты повторно с минимальными (но достаточными, чтобы удовлетворить формальные претензии Президента Российской Федерации) поправками. Последнее вовсе не означает, что борьба завершилась. Перед повторным рассмотрением вопроса в Верховном Совете в "Независимой газете" от 15.09.92 была опубликована заметка "Закон пока не работает", где Ассоциация поставщиков программных продуктов в лице исполнительного директора С. Фенева обвинила РАИС в противодействии вступлению в силу Закона "О правовой охране программ для ЭВМ и баз данных". Опровержение было напечатано в "Независимой газете" 23.09.92, в день повторного принятия Закона. Выступивший с опровержением Политический советник Генерального директора РАИС А. Черкизов утверждал, что "...РАИС не знаком с текстом этого Закона и никогда не высказывал никаких замечаний по его содержанию". Между тем, в письме № 1-8-10/176 от 24.08.92 Генерального директора РАИС М.А. Федотова Президенту Б.Н. Ельцину содержится просьба еще раз воздержаться от подписания указанного Закона и на основании Положения о РАИС направить его в Агентство для подготовки официального расширенного заключения. При этом Генеральный директор ссылается на статью 12 Закона, согласно которой имущественные права на программы для ЭВМ и базы данных, созданные по служебному заданию, "принадлежат работодателю, если в договоре между ним и автором не предусмотрено иное". По мнению М.А. Федотова здесь "вместо охраны в рамках авторского права имеет место охрана особого рода". А это, как утверждается, противоречит международным обязательствам России.

Подписание (или неподписание) Закона Президентом Российской Федерации состоит гораздо раньше, чем будет опубликована эта статья, и осложнения (если они возникнут) вновь будут носить процедурный характер. Однако суть дела не в сроках и не в деталях процедур.

Меня смущает возобладавший стиль обсуждения достаточно тонких вопросов, когда ошеломляющие заголовки и заявления [см., например, Малков Л.П. Тихое решение на 10 000 000 000//PC Magazine/ Russian Edition. - 1991. - №2. - P.9, 10] сочетаются с полной апатией массовой аудитории к любому не вступившему в силу закону. Вздрогнув в очередной раз от резкого заявления, общественность остывает не только к сделанному заявлению, но и к самой теме правовой охраны программ. На таком фоне противоборствующие стороны постоянно апеллируют к интересам страны и отрасли, преследуя на деле лишь свои корпоративные интересы. Страсти кипят в основном вокруг охраны программ и Патентного закона, причем для полноты понимания ситуации их следует представлять себе в совокупности. В этом случае можно догадаться, как проходило бы обсуждение Закона о правовой охране программ, будь программисты так же организованы, как изобретатели.

Я не удивлюсь, если отсрочки и появление технических затруднений с введением правовой охраны программ будут продолжаться и после повторного рассмотрения Закона Верховным Советом.

СУБД для профессионалов. Что выбрать?

*Анализ использования СУБД и других инструментальных средств
для разработки программного обеспечения автоматизированных рабочих мест (АРМ)
различного назначения.*

Л. Л. Войтенко, А. А. Искра

СУБД и локальные АРМ

При разработке локальных АРМ основными характеристиками, оптимальное сочетание которых необходимо обеспечить, являются:

- 1) удобство интерфейса пользователя;
- 2) быстрота разработки и универсальность программной оболочки;
- 3) возможность быстрой модификации программной оболочки для удовлетворения требований, возникающих у пользователей в процессе эксплуатации АРМ;
- 4) объемно-временные характеристики, требования к вычислительным ресурсам;
- 5) мобильность;
- 6) возможность работы в сети;
- 7) лицензионная чистота разработки.

В зависимости от условий, в которых предстоит эксплуатировать локальные АРМ, преобладающее значение приобретают те или иные характеристики, однако в любом случае очень важны характеристики 4, 1, 2.

Локальные АРМ могут создаваться для коммерческого распространения, для информационного обслуживания организаций, которыми они разрабатываются, а также как элемент при создании более сложных информационных систем.

Для коммерчески распространяемых АРМ дополнительно должны быть учтены следующие условия: в большинстве случаев они будут эксплуатироваться на ПК семейства IBM PC AT (или даже PC XT) под управлением MS-DOS при предъявлении жестких требований к используемым объемам оперативной памяти.

Разработчик подобных АРМ ориентируется прежде всего на создание EXE-файлов, не требующих наличия у пользователей собственных пакетов соответствующих СУБД и не нуждающихся при выполнении в объемах памяти, превышающих 300-400 Кбайт. Структура баз данных и алгоритмы обработки информации в АРМ коммерческого назначения, как правило, являются достаточно простыми и понятными. Учитывая отсутствие сложных алгоритмов обработки, разработчик стремится к созданию простого и удобного интерфейса пользователя. Для подобных целей наиболее подходящими являются СУБД реляционного типа.

Обсуждая локальные АРМ коммерческого назначения, отметим следующее:

- стоимость собственно информации, хранящейся в базах данных, может превышать стоимость программной оболочки;

- жесткие требования предъявляются к временным характеристикам, получаемым в процессе эксплуатации АРМ, удобству обработки строковых данных, легкости освоения работы с ними конечными пользователями;
- в последнее время все большее значение приобретает возможность эксплуатации АРМ в локальной вычислительной сети либо в режиме удаленного доступа;
- хорошим стилем при разработке АРМ является создание интерфейса, удовлетворяющего стандарту CUA (Common User Access) и обеспечивающего работу в графическом режиме в среде Windows, а также реализация возможности обработки видео- и аудиоинформации.

Среди программных продуктов, предназначенных для подобных разработок и частично удовлетворяющих этим условиям, можно отметить часто используемую СУБД Clipper 5.0.

Эта СУБД реляционного типа по производительности уступает только СУБД Paradox 4.0 и FoxPro 2.0. Благодаря наличию механизма динамических оверлеев Clipper 5.0 позволяет создавать EXE-файлы, для выполнения которых требуются сравнительно небольшие объемы оперативной памяти (хотя это достигается за счет некоторого ухудшения временных характеристик). В СУБД Clipper 5.0 имеются примитивы, позволяющие реализовать всплывающие окна, горизонтальные и вертикальные меню с развитыми возможностями, редакторы текстов и баз данных. К сожалению, стандартный язык не предусматривает средств работы с графикой и мышью. Эти возможности реализуются другими пакетами.

Программисту, работающему с СУБД Clipper 5.0, представляются такие средства автоматизации программирования,

*Леонид Леонидович Войтенко -
кандидат технических наук,
ведущий научный сотрудник РосНИИИС.
Область профессиональных интересов -
языковые и программные средства для разработки
пользовательского интерфейса, применяемые при создании
информационных систем.
Александр Анатольевич Искра -
старший научный сотрудник РосНИИИС.
Область профессиональных интересов -
проектирование информационных систем,
оценка и выбор системного программного обеспечения
для их разработки.
Телефон: 284-43-16*

как встроенные функции (примерно 200 функций), библиотека функций Clipper Tools One (примерно 300 функций), библиотека предкомпоновки, встроенный препроцессор, утилиты поддержки проекта RMAKE, резидентный справочник программиста. Имеется механизм интеграции с базами данных, форматы которых оличаются от dBase-формата. Для этого используется вызов EXE-программ, написанных, например, на языках C, PASCAL и обеспечивающих интерфейс с базами данных других форматов (например, с базами данных СУБД Paradox). При этом, однако, не исключено возникновение проблем, связанных с нехваткой оперативной памяти. В печати появилась информация о том, что планируется создать заменяемые драйверы баз данных, позволяющие из Clipper-программ работать с базами данных других СУБД.

Одним из преимуществ использования СУБД Clipper 5.0 является хорошая поддержка зарегистрированных пользователей фирмой-разработчиком. Выпущена полностью русифицированная версия этой СУБД - Clipper 5.01. Среди системных возможностей Clipper 5.0 необходимо отметить возможность работы Clipper-приложений в среде MS Windows, а также под управлением сетевых ОС, использующих средства MS-DOS (например, Novell Netware, 3Com и др.). Для несложных приложений обычно не вызывает затруднений реструктурирование базы данных.

Конечно же, Clipper 5.0 не является идеальным инструментальным средством. При разработке сложных приложений, имеющих разветвленную структуру данных, возникают проблемы с обработкой большого числа файлов, в том числе индексных. Интерфейс с пользователем не всегда отвечает самым современным требованиям. Отсутствует встроенный графический интерфейс. При работе с большими базами данных (сотни тысяч записей) быстродействие хуже, чем у некоторых других СУБД, например FoxPro 2.0 и FoxBase 2.0 [см. Гетманский А.Д., Топчян Б.Х. СУБД: Так какая же быстрее? // Мир ПК. - 1992. - №2. - с.59].

Однако для создания не очень сложных программных продуктов, предназначенных для коммерческого распространения, Clipper 5.0 является хорошим инструментальным средством.

По нашему мнению, для разработки АРМ коммерческого назначения приходится полностью исключить использование таких известных СУБД, как dBase, FoxBase, Paradox, более мощных СУБД Oracle, R:base for DOS и многих других, обладающих определенными преимуществами, но не отвечающих жестким требованиям по использованию оперативной памяти и созданию EXE-файлов.

При отсутствии жестких требований к оперативной памяти в качестве достойного конкурента СУБД Clipper 5.0 для разработки данного класса приложений можно рассматривать СУБД FoxPro 2.0. По трудоемкости автоматизированной разработки приложений, возможности создания современного интерфейса с пользователем (в том числе графического), хорошим временным характеристикам при обработке баз данных, особенно большого объема, FoxPro 2.0 явно превосходит Clipper 5.0. Однако при этом размер откомпилированного EXE-файла очень велик: обычно 500-600 Кбайт, но может достигать до 1 Мбайт и более.

В FoxPro 2.0 реализованы мощные средства создания экранного интерфейса с конечным пользователем. В их основе лежит развитый оконный интерфейс с возможностью передачи сообщения от одного элемента окна к другому и связыванием всех окон в единую рабочую среду. Кроме того, реализованы функции поддержки "мыши", ряд примитивов для наиболее популярных элементов экранного интерфейса (системы выпадающих меню, ключей активации, диалоговых

окон, областей редактирования текста и т.д.), автоматический проектировщик экранов.

Оптимально спроектированная и индексируемая база данных FoxPro 2.0 может содержать до 500 000 записей в одном файле. При таком объеме базы данных и использовании ПК 386 (33 МГц) возможна практически прозрачная для пользователя обработка его запросов. Объем исходного текста, написанного на языке СУБД FoxPro, примерно в 2,5 раза меньше объема текста на языке Clipper, предназначенного для решения аналогичной задачи. Минимальный объем памяти, необходимый для нормальной работы FoxPro 2.0, составляет 512 Кбайт. FoxPro 2.0 активно использует как дополнительную, так и расширенную память.

При наличии вычислительных средств с необходимыми ресурсами оперативной памяти (1 Мбайт и более) СУБД FoxPro 2.0 может служить прекрасным инструментальным средством для создания локальных АРМ, предназначенных для информационного обслуживания той организации, в которой осуществляется разработка, так как эта СУБД позволяет создавать хороший интерфейс, ориентированный на конкретного пользователя, быстро вносить изменения в разрабатываемые продукты и функционально наращивать АРМ в процессе их эксплуатации.

Кроме того, в FoxPro 2.0 реализованы средства для создания гипертекстовых справочников, объектно-ориентированный подход к разработке приложений, мощная оболочка для создания и корректировки баз данных, отчетов.

Разрабатываемые с помощью FoxPro 2.0 приложения обладают хорошей мобильностью (исходный текст программы с незначительными изменениями может быть перенесен для функционирования в среде MS Windows, Macintosh, UNIX).

Если разработчик хочет создать конкретное приложение в максимально короткие сроки, то мы советуем ему выбрать FoxPro 2.0.

Из-за высоких требований к объему оперативной памяти использование СУБД FoxPro 2.0 при организации работы в локальной вычислительной сети ограничено.

Опытным разработчикам, пишущим программы на языке C, для создания АРМ, с нашей точки зрения, лучше всего использовать СУБД, имеющие интерфейс с включающим языком C. Обеспечивая все необходимые средства манипулирования данными, в том числе для работы в сети, они в то же время оставляют программисту заботы об организации интерфейса с пользователем. С одной стороны, это дополнительно повышает трудоемкость разработки программного продукта, а с другой - позволяет создавать наиболее удобную и гибкую информационную среду для пользователя. Применение объектно-ориентированных средств программирования, подобных пакету Turbo Vision и языку C++, позволяет создавать современный экраный интерфейс, соответствующий стандарту CUA.

Для СУБД Paradox 3.5 подобным средством является пакет Paradox Engine 2.0. Сохраняя все достоинства СУБД Paradox, пакет Paradox Engine 2.0 предоставляет опытному программисту возможность придать процессу разработки необходимую гибкость.

Совместное использование пакетов C++, Turbo Vision и Paradox Engine позволяет профессиональному разработчику создавать широкий класс приложений: от локальных АРМ до интегрированных информационных систем на основе локальных вычислительных сетей. Наличие SQL-интерфейса (Paradox SQL Link) при использовании SQL-сервера дает возможность перейти от архитектуры обслуживания файла к архитектуре обслуживания клиента (client-server architecture - CSA).

Несколько слов о СУБД Paradox. Версия Paradox 3.5, безусловно, являющаяся одной из наиболее функционально мощных и эффективных СУБД общего назначения, неоднократно признавалась лучшей по результатам регулярно проводимых в США оценок СУБД для ПК (появившаяся на нашем рынке программных продуктов версия Paradox 4.0 имеет еще более лестные оценки). Однако необходимость применения пакета Paradox Run Time на ПК пользователей и невозможность использования включающих языков программирования (C, Pascal и т.п.) делают ее лучшей лишь для разработки приложений непрофессионалом, так как непрограммируемый дружественный интерфейс не позволяет создавать сложных приложений. Использование же интерпретатора языка PAL, требующего уже профессиональных навыков, не позволяет создавать высокоэффективных приложений, реализующих сложные модели данных.

С нашей точки зрения, СУБД Paradox 3.5 может быть использована для создания исходных моделей АРМ, реализуемых в последующем с помощью программ на языке C++ и пакета Paradox Engine, а также может служить для дополнительного информационного обслуживания пользователей, в том числе в сетях. СУБД Paradox 3.5 способна работать в среде подавляющего большинства сетей: Novell, 3Com, Token Ring, Banyan и т.д., - использующих средства MS-DOS, поддерживает практически все известные SQL-серверы, обеспечивает блокирование таблиц и записей, имеет развитую систему защиты информации и авторизации доступа. Использование разработанного фирмой Borland средства управления распределением памяти VROOMM (Virtual Runtime Object-Oriented Memory Manager) позволило нам добиться 30%-ной экономии памяти при выполнении приложений, разработанных с помощью СУБД Paradox 3.5. А применение пакета Turbo Drive на ПК с процессором 286, 386 или 486 обеспечивает повышение быстродействия.

СУБД Paradox 3.5 действительно обладает многими прекрасными качествами, однако она не для профессионалов. Работая с этой СУБД, профессионал окажется в "плелу" идей фирмы Borland.

Для создания локальных АРМ, не отличающихся большой сложностью структур данных и алгоритмов обработки и предназначенных для информационного обслуживания организации-работчика, безусловно, подойдут широко используемые СУБД семейства dBASE: dBASE IV и производительная FoxBase+. Они имеют развитый интерфейс, автоматизируют решение многих проблем, но уступают СУБД Clipper 5.0 и FoxPro 2.0.

Наш, быть может, небогатый опыт использования R:base for DOS продемонстрировал наряду с функциональной мощью этой СУБД ее "тяжеловесность" в ординарных приложениях для ПК на основе процессора 286.

Несколько слов о типах данных, хранимых в базах и моделях (структурах) данных. Большинство программистов, начавших свою карьеру с работы на ПК, на практике не сталкивались с СУБД, основанными не на реляционной модели данных. Для них может показаться "экзотикой" использование СУБД, реализующих иерархические и сетевые модели данных. Однако практическая необходимость создания сложных приложений, требующих установления информационных связей между значительным числом разнотипных объектов (до нескольких десятков), показывает, что это далеко не так. Использование СУБД Clipper 5.0 для разработки приложений, заставляющих работать более чем с полусотней файлов (с учетом индексных), легко убеждает в этом. Простота и естественность применения реляционной модели данных при проектировании и разработке приложений, не требующих учета информационных связей между многими типами объек-

тов, теряются при реализации сложных интегрированных систем. Конечно, опытный разработчик может найти приемы, позволяющие разбить полное информационное множество на подмножества, для которых реляционная модель работает удовлетворительно. Однако, в конечном счете, расплачиваться за это придется пользователю, причем расплачиваться невозможностью полностью удовлетворить свои информационные потребности.

Относительно типов данных, хранение и обработка которых предусмотрены в популярных СУБД, наблюдается более или менее полное единодушие. Разногласия возникают в основном по поводу длины хранимых строковых данных, возможности работы с числами с плавающей точкой, возможности различения неопределенного значения (NULL-Value) поля от строки пробелов или числового нуля. Постоянно растущие информационные потребности пользователей уже сегодня предполагают хранение и обработку аудио- и видеoinформации, однако число универсальных СУБД, отвечающих этим требованиям, весьма невелико. На сегодняшний день создано достаточно много конкретных приложений, предназначенных для хранения и обработки специализированной аудио- и видеoinформации, но мы говорим о большем - о возможности хранения и обработки подобной информации в единой базе данных с помощью универсальной СУБД наряду с хранением и обработкой "привычных" для нее типов данных.

Одной из немногих СУБД, позволяющих работать с базами данных, основанными на сетевой модели данных, хранить и обрабатывать (конечно, с использованием включающего языка C) информацию подобного типа, а также программировать операторы реляционных языков и примитивы манипулирования с фреймами в базах знаний, является СУБД db_Vista III. О других возможностях этой СУБД мы еще поговорим при обсуждении вопроса о создании информационных систем на основе локальных вычислительных сетей.

Последняя версия СУБД Paradox - Paradox 4.0, позволяет создавать поля двоичного формата и хранить в них аудио- и видеoinформацию.

СУБД и автоматизированные информационные системы

При проектировании и разработке автоматизированных информационных систем перед прикладными и системными программистами встает множество проблем, связанных с выбором инструментальных средств. Мы не ставим своей целью их всестороннее рассмотрение. Наша задача - обсудить проблему выбора СУБД и некоторых других средств программирования, позволяющих, с одной стороны, решить задачу информационного обслуживания пользователей, а с другой - органически "вписаться" в локальную вычислительную сеть. Мы затронем только некоторые из возникающих проблем:

- особенности работы в сети и трудоемкость ее организации;
- построение однородных и неоднородных сетей и распределенных баз данных.

При обсуждении данных проблем мы рассмотрим не только особенности работы конкретных СУБД в сетях, но и возможные направления их использования. При этом мы исходим из следующих предположений:

- для разработки отдельных частей автоматизированных информационных систем могут использоваться различные инструментальные средства;
- информационная система может создаваться наращиванием функциональных возможностей АРМ;
- подобные разработки выполняет, как правило, коллектив программистов.

Все рассмотренные выше СУБД имеют средства организации коллективного доступа к базам данных и ориентированы на работу в сетях, однако для обеспечения такой работы программист должен предпринимать различные усилия. Большинство СУБД реализуют распределенный (сетевой) доступ к централизованной базе данных. Конечно, разработчик может успешно сочетать работу как с централизованными, так и с локальными базами данных.

При эксплуатации локальных АРМ, разработанных с помощью СУБД Clipper 5.0, в локальных сетях не нужны дополнительные программные средства на каждом АРМ. Однако при разработке соответствующих приложений программист должен позаботиться о правильном блокировании доступа к файлам и записям для обеспечения монопольного их использования абонентом сети при выполнении им некоторых операций. Блокирование записей необходимо при любых их изменениях и добавлении новых записей, а блокирование файлов - при выполнении некоторых групповых операций. Блокирование возможно и при выполнении операций чтения. Отсутствие механизма транзакций и возможные ошибки в Clipper-программе могут приводить к нарушению целостности базы данных. Сохранение целостности данных при аварийном завершении работы требует дополнительных усилий. Clipper-приложения могут функционировать под управлением сетевых ОС, использующих средства MS-DOS.

Весьма тонкие механизмы блокирования ресурсов предусмотрены в СУБД Paradox. Они автоматически реализуют многопользовательский доступ к базам данных и записям баз данных при выполнении операций чтения и обновления информации без специального программирования. Автоматическое блокирование файлов и записей обеспечивает целостность базы данных. Автоматическое обновление экрана позволяет видеть изменения, сделанные в базе данных другими пользователями в реальном масштабе времени. СУБД Paradox способна работать в сетях Novell, 3Com, Token Ring, Torus, Typestry, Starlan, Banyan и других сетях, использующих средства MS-DOS. Таким образом, охвачено и большинство аппаратных стандартов, в том числе Arcnet, Ethernet, Token Ring. Paradox поддерживает также практически все известные SQL-серверы (фирм IBM, Oracle, Microsoft). Это справедливо и для приложений, разработанных с помощью пакета Paradox Engine 2.0.

Приложения, созданные с помощью СУБД FoxPro 2.0, могут работать практически во всех популярных сетях (их примерно 30): от Office Lan и Apple-Talk до 3Com и Novell NetWare. Обеспечена также обработка SQL-запросов.

СУБД db_VISTA III реализована для работы в среде более чем 25 типов операционных систем (в том числе MS-DOS, MS Windows, Macintosh, VAX/VMS, а также для всех основных разновидностей ОС UNIX: UNIXSystemV, BSDUNIX, SCOUNIX, XENIX, SunOS, ULTRIX). На IBM-совместимых ПК db_VISTA III может работать в локальных вычислительных сетях, поддерживающих интерфейс NetBIOS. db_VISTA III предоставляет широкий выбор структур сетевой информационной системы. Возможна организация работы с файло-сервером и без него. Базы данных и файлы базы данных могут размещаться в сети произвольным образом. Файлы одной базы данных могут размещаться в различных узлах локальной сети, что очень полезно для создания больших и сложных интегрированных информационных систем. В этом отношении db_VISTA III является уникальной среди других СУБД для ПК. Учитывая прекрасную производительность db_VISTA III при обработке информации, в том числе для баз данных большого объема, можно говорить о хороших возможностях по использованию этой СУБД коллективами профессионалов,

применяющих язык С при создании серьезных прикладных систем, предназначенных для обработки распределенной информации (включая специфическую видео- и аудиоинформацию).

Говоря о разработке сложных информационных систем, мы хотим отметить следующее.

- Подобные системы имеют свою историю создания. У некоторых из них по разным соображениям (отсутствие финансов, необходимой техники, постепенное "созревание" заказчика) мог отсутствовать исходный системный проект, и они могли возникать как "надстройка" над первоначально разрозненными базами данных, разработанными с помощью разных СУБД. Безусловно, это не лучший вариант. Хотя мы не отрицаем возможности использования различных СУБД в одной интегрированной информационной системе, однако считаем, что, во-первых, это разнообразие нежелательно и, во всяком случае, должно быть минимально (не больше двух-трех СУБД), во-вторых, используемые СУБД, безусловно, должны "уметь" функционировать под управлением одной сетевой операционной системы, если, конечно, база данных не используется локально на одном АРМ, и, в-третьих, желательно, чтобы эти СУБД были "родственниками", т.е. либо работали с базами данных одного и того же формата (как, например, СУБД семейства dBASE), либо предоставляли доступ к своим ресурсам из одного и того же включающего языка. Безусловно, речь идет о языке С, и здесь выбор СУБД достаточно ограничен: Paradox 3.5 (пакет Paradox Engine 2.0), Btrieve и db_VISTA III.

- Следует иметь в виду бурное развитие инструментальных средств и операционных сред, в которых они функционируют, появление все новых пакетов, направленных на автоматизацию программирования на языке С и на разработку новых стандартов интерфейса пользователя и изменение "моды" на стиль программирования.

- В процессе создания систем у каждого разработчика в отдельности и коллектива в целом появляются опыт и стремление двигаться от простого к сложному. Важно, чтобы в процессе этого движения не приходилось каждый раз начинать сначала.

Так что же все-таки выбрать?

Мы далеки от мысли выдать профессиональным программистам какой-либо один рецепт для выбора СУБД и других инструментальных средств.

Во-первых, уважающий себя профессионал сам уже сделал выбор, и только появление чего-то нового и действительно стоящего либо изменение сферы приложения усилий заставит его изменить свой выбор.

Во-вторых, для разработки приложений разного назначения и разной степени интеграции наиболее удобными и эффективными могут оказаться разные СУБД.

В-третьих, приходится учитывать наличие соответствующих технических средств как у разработчика, так и у пользователя конечного продукта, а также возможность применения программистом тех или иных инструментальных средств.

В-четвертых, бурное развитие как самих СУБД, так и других инструментальных средств заставляет нас говорить скорее о направлении, по которому должна идти разработка программного обеспечения и которое дает коллективу программистов возможность успешно создавать локальные АРМ и интегрированные информационные системы.

Анализируя тенденции разработки программных продуктов ведущими фирмами, можно сделать следующие выводы:

- в настоящее время лидеры в области создания программного обеспечения (например, такие, как фирма Borland) переходят к выпуску продуктов, работающих в графическом режиме в среде Windows;

- в ближайшем будущем продукты, работающие в графическом режиме в среде Windows, будут наиболее конкурентоспособными на рынке программного обеспечения;

- происходит интенсивный процесс осознания преимуществ объектно-ориентированного программирования.

Таким образом, если разработчик программного обеспечения не хочет через некоторое время оказаться в "отцепленном вагоне", ему следует немедленно переориентироваться на создание соответствующих программных продуктов. Однако для практической реализации такой переориентации имеется ряд препятствий:

- нужны более мощные технические средства, чем ПК на основе процессора 286 с тактовой частотой 12 МГц и 40-Мбайтным жестким диском;

- требуется изменить характер своего мышления при разработке программ, поскольку идеология объектно-ориентированного программирования в графическом режиме для среды Windows в корне отличается от традиционной, а учиться негде и не у кого.

Нам представляется целесообразным в качестве первого этапа такой переориентации разрабатывать программные продукты на языке C++ с использованием пакета Turbo Vision для создания современного экранного интерфейса.

По своей сути Turbo Vision - это "мини-Windows" для текстового режима работы адаптера. Так же, как и Windows, Turbo Vision предоставляет программисту для создания современного экранного интерфейса средства, соответствующие стандарту CUA. Интерфейс конструируется из следующих основных объектов:

ПАНЕЛЬ - прямоугольная часть экрана, внутри которой размещается информация, связанная с отдельным актом диалога.

МЕНЮ - строка, размещающаяся в верхней части панели и состоящая из элементов, активизация которых может вызывать появление выпадающих (*pull-down*) подменю. Подменю, появляющееся при выполнении команды меню, называется всплывающим (*pop-up*).

КЛАВИША - элемент нижней строки панели, связывающий команду с определенной клавишей клавиатуры. Выполнение команды осуществляется нажатием на клавишу клавиатуры либо выбором ее курсором "мыши".

КНОПКА - текст с названием команды. Располагается внутри панели и позволяет выполнить команду путем выбора текста курсором "мыши".

ПОЛЕ ВВОДА - область панели для ввода и редактирования строки литер.

ПОЛЕ ВЫБОРА - группа вариантов значения поля, из которых можно выбрать один или несколько.

ПАЛИТРА - сочетание цветов элементов панели.

Язык C++ является перспективным объектно-ориентированным языком программирования, в полном объеме реализующим концепции инкапсуляции, полиморфизма и наследования типов.

Для разработки приложений на языке C++ с использованием пакета Turbo Vision достаточно (правда, на пределе возможностей) ПК с процессором 286, тактовой частотой 12 МГц, ОЗУ емкостью 2 Мбайт и 40-Мбайтным жестким диском.

Идеологически Turbo Vision и Windows очень близки, поэтому работа с исходными текстами Turbo Vision является

хорошим "фундаментом" для разработки программных продуктов в среде Windows.

В качестве СУБД может быть использована любая, предоставляющая доступ к своим ресурсам из языка C. Можно одновременно применять несколько таких СУБД, используя преимущества каждой из них для более эффективной реализации отдельных функций. Практически сейчас существует три таких СУБД: Paradox 3.5 (пакет Paradox Engine 2.0), Btrieve и db_Vista III.

Для разработки приложений, в которых использование реляционной модели данных дает удовлетворительные результаты (это могут быть как локальные АРМ, так и информационные системы на основе локальных вычислительных сетей либо их части), наиболее целесообразно применить пакет Paradox Engine, поскольку, с одной стороны, имеется опыт работы с ним, а с другой - он позволяет достаточно полно и эффективно реализовать требуемые функции.

Для создания интегрированных систем со сложными информационными связями между объектами, реализуемых на основе однородных и неоднородных локальных вычислительных сетей, в том числе с распределенными базами данных, целесообразно использовать СУБД db_Vista III или Btrieve.

При таком подходе разработанному продукту будут присущи следующие преимущества:

- соответствие требованиям стандарта CUA, а следовательно, конкурентоспособность в настоящее время и в ближайшем будущем;

- отсутствие жесткой привязки к используемой СУБД (в случае необходимости всегда можно будет перейти от использования СУБД Paradox 3.5 к использованию любой другой, предоставляющей доступ к своим ресурсам из языка C);

- лицензионная чистота EXE-модулей, не требующих для своего выполнения наличия других программных продуктов, что существенно для коммерческого распространения;

- будет подготовлена благоприятная почва для перехода к созданию программных продуктов, работающих в графическом режиме в среде Windows с применением объектно-ориентированных языков программирования;

- откроется путь к использованию разрабатываемых программных продуктов в среде перспективной ОС UNIX.

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ОБЪЕДИНЕНИЕ

TRAVERS



ТРАВЕРС

УНИВЕРСАЛЬНЫЕ ПРОГРАММАТОРЫ

ИМС ППЗУ, ПЛМ, ОЭВМ, ПЛИС...

Телефоны: (095) 531-33-42, 531-54-80

Факс: (095) 531-15-62

Подробная информация в журнале "Мир ПК", 1992, №6

CLIPPER 5.01

Русификация заголовков столбцов

А. А. Барчук

Все программисты рано или поздно сталкиваются с проблемой нехватки времени, когда в "экстремальных" условиях необходимо "сострять", а порой и отладить программу. Естественно, о высоком качестве результатов такой работы говорить не приходится. Однако есть немало способов увеличить эффективность процесса программирования. Один из них - создание гибких модулей, которые легко подключаются и адаптируются к любой новой системе. Несмотря на то что такие программы получают немного больше и некоторые функции могут оказаться избыточными, продолжительность их разработки существенно сокращается. Более того, качество такой системы может быть значительно выше, так как все включаемые в нее модули являются своего рода аккумуляторами идей и удачных находок. Рассмотрим пример такого программного модуля.

Как известно, почти во всех прикладных системах просмотр и корректировка справочной информации, хранящейся в базах данных, осуществляются в режиме BROWSE.

Рассмотрим, как для этих целей можно использовать класс TBrowse в СУБД CLIPPER 5.01. Компилируем файл \CL5.1\SOURCE\SAMPLE\TBdemo.prd (любезно предоставленный фирмой Nantucket) с помощью командной строки:

```
clipper tbdemo /m/n
```

После компоновки:

```
tlink tbdemo,,nul,clipper extend terminal
dbfntx
```

или

```
RTlink FI tbdemo LIB clipper, extend,
terminal, dbfntx
```

получим готовый к использованию файл tbdemo.exe.

Rec #	KOD_DET	NAME_DET	KG	MIN_PLAN
35	35	Болт 921-19.384	F	0.00
34	34	Болт ЛФ8.920.067	F	0.00
195	207	Винт ЛФ8.900.016	F	0.00
202	85	Винт МЗ*10	F	0.00
99	101	Винт МЗ*6	F	0.00
208	218	Винт М5*8	F	0.00
205	215	Винт М8*7.5	F	0.00
190	202	Воронка ЛФ8.654.009	F	0.00
172	180	Втулка 932-19.711-01	F	0.00
168	176	Втулка 967-19.245	F	0.00
75	75	Втулка ЕИУА.711141.034	F	0.00
28	28	Втулка ЕИУА.712744.001	F	0.00
175	184	Втулка ЕИУА.713341.000	F	0.00
158	163	Втулка ЛФ8.220.038-01	F	0.00
180	192	Втулка ЛФ8.220.128	F	0.00
186	198	Втулка ЛФ.220.150	F	0.00
138	142	Втулка ЛФ8.220.162	F	0.00
179	191	Втулка ЛФ8.223.122	F	0.00

Рис.1

Структура DBF_RUS.DBF			
Field Name	Type	Width	Dec
NAME_DBF	Character	10	
KOL_FIKS	Numeric	1	0
P11_N	Character	30	
P11_S	Logical	1	
P11_P	Numeric	2	0
P12_N	Character	30	
P12_S	Logical	1	
P12_P	Numeric	2	0
P13_N	Character	30	
P13_S	Logical	1	
P13_P	Numeric	2	0
P14_N	Character	30	
P14_S	Logical	1	
P14_P	Numeric	2	0
P15_N	Character	30	
P15_S	Logical	1	
P15_P	Numeric	2	0
P16_N	Character	30	
P16_S	Logical	1	
...	...	...	...

Группа I

Группа II

Группа III

Группа IV

Группа V

Группа VI

Рис.2

Rec #	NAME_DBF	KOL_FIKS	P11_N	P11_S	P11_P	P12_N	P12_S
1	SDet	1	Наименование детали	T	2	Код	T
...	...	...	...	...	...	...	...

Продолжение

P12_P	P13_N	P13_S	P13_P	P14_N	P14_S	P14_P	...
1	Kr	T	3	Min план	T	4	...
...	...	...	...	...	...	...	...

Рис.3

Rec #	Наименование детали	Код	Kr	Min план
35	Болт 921-19.384	35	F	0.00
34	Болт ЛФ8.920.067	34	F	0.00
195	Винт ЛФ8.900.016	207	F	0.00
202	Винт МЗ*10	85	F	0.00
99	Винт МЗ*6	101	F	0.00
208	Винт М5*8	218	F	0.00
205	Винт М8*7.5	215	F	0.00
190	Воронка ЛФ8.654.009	202	F	0.00
172	Втулка 932-19.711-01	180	F	0.00
168	Втулка 967-19.245	176	F	0.00
75	Втулка ЕИУА.711141.034	75	F	0.00
28	Втулка ЕИУА.712744.001	28	F	0.00
175	Втулка ЕИУА.713341.000	184	F	0.00
158	Втулка ЛФ8.220.038-01	163	F	0.00
180	Втулка ЛФ8.220.128	192	F	0.00
186	Втулка ЛФ8.220.150	198	F	0.00
138	Втулка ЛФ8.220.162	142	F	0.00
179	Втулка ЛФ8.223.122	191	F	0.00

Рис.4

Компоновку лучше выполнять первым способом, так как он значительно быстрее и полученный EXE-файл поддается упаковке такими программами, как PKLITE.EXE и LZEXE.EXE.

Запуск файла производится с помощью строки

tbdemo.exe Sdet.dbf Sdet.ntx

где Sdet.dbf - некоторый (произвольный) файл базы данных (в нашем случае это "Справочник деталей"), Sdet.ntx - его индексный файл (указывать необязательно).

На экране монитора появится изображение содержимого этой базы данных в режиме BROWSE (рис.1).

Первый столбец - номера записей. Если загрузить файл без индекса, то записи будут располагаться по возрастанию номеров.

Над каждым последующим столбцом стоит наименование соответствующего поля. К сожалению, наименования полей выводятся на английском языке и длина их не превышает 10 символов. Это неудобство можно обойти, внося некоторые изменения в текст программы и создав дополнительную базу данных (назовем ее DBF_RUS.DBF) со структурой, представленной на рис.2.

Число групп полей должно быть не меньше общего числа полей в русифицируемой (в дальнейшем будем называть ее рабочей) базе данных. Иначе

ЛИСТИНГ

```

****
* Tbrus.prg
* Illustration of TBROWSE and GET objects.
* Copyright (c) 1990-1991 Nantucket Corp. All rights reserved.
* Добавления внес А.А. Барчук, г.Херсон
*
* Compile: CLIPPER Tbrus /m/n
* Link:  RTLINK FILE Tbdemo
* Execute: Tbrus <dbf> [<ntx>] (Без расширений)
*

#include "Inkey.ch"
#include "Setcurs.ch"
#include :Error.ch

// These #defines use the browse's "cargo" slot to hold the
// "append mode" flag for the browse. The #defines make it
// easy to change this later (e.g. if you need to keep
// several items in the cargo slot).
#define TURN_ON_APPEND_MODE(b) (b:cargo := .T.)
#define TURN_OFF_APPEND_MODE(b) (b:cargo := .F.)
#define IS_APPEND_MODE(b) (b:cargo)

// Separator strings for the browse display
#define MY_HEADSEP "----"
#define MY_COLSEP " | "

***
* Tbdemo <dbf> [<Index>]

```

PROCEDURE Tbdemo(dbf, Index)

```

LOCAL bSaveHandler, error
LOCAL cScreen

//Добавить описание переменных
LOCAL AI, Ss := .f.,Fc

//Убрать расширение в имени базы данных
dbf:=STUFF(dbf,IF(AT(".",dbf)>0,AT(".",dbf),10),4,"")

//Lazy man's error checking
bSaveHandler := ERRORBLOCK({|x| BREAK(x)})

BEGIN SEQUENCE
    USE (dbf) INDEX (Index)

RECOVER USING error
    IF error : genCode == EG_OPEN
        ?? "Error opening file(s)"

    ELSE
        //Assume it was a problem with the params
        ?? "Usage: Tbrus <dbf> [<Index>]"
    ENDIF

    QUIT
END

//Restore the default error handler

```


говоря, одна группа русифицирует одно поле.

Проиндексируем базу данных по ключу: UPPER(NAME_DBF), и присвоим индексному файлу имя DBF_RUS.NTX.

Теперь внесем некоторые изменения в текст программы, предварительно переименовав ее, например, в Tbrus.prg (см. листинг).

Порядок работы с программой:

1. В поле NAME_DBF базы DBF_RUS.DBF ввести наименование базы данных, которую вы хотите русифицировать.

2. Если некоторые столбцы нужно зафиксировать, то в поле KOL_FIKS следует ввести цифру, отличную от 0.

3. В поле P11_N ввести заголовок первого столбца на русском языке.

4. В поле P11_S ввести .E..

Это поле предназначено для быстрой перенастройки. Если в него ввести .F., то соответствующее поле рабочей базы данных не будет выводиться на экран.

5. В поле P11_P указать порядковый номер поля рабочей базы данных.

6. Повторить указанные в пунктах 3 - 5 действия для всех столбцов.

Далее можно добавлять неограниченное число русифицируемых баз данных.

Пример работы с программой приведен на рис.3.

Для заполнения базы данных DBF_RUS.DBF можно использовать эту же программу, но без русификации (т.е. база данных сама себя не русифицирует).

Если русификация не нужна, то просто измените или удалите содержимое поля NAME_DBF.

Заполнение базы данных DBF_RUS.DBF необходимо проводить в режиме использования индекса. Это можно сделать с помощью, например, такой команды:

```
tbrus.exe DBF_RUS.DBF DBF_RUS.NTX
```

После заполнения базы данных DBF_RUS.DBF остается только запустить программу:

```
tbrus.exe Sdet.dbf Sdet.ntx
```

На рис.4 показано, что примерно получится на экране.

Как видно, заголовки столбцов представлены на русском языке, и по сравнению с примером, приведенным на рис.1, первые два столбца с содержанием "Справочника деталей" переставлены местами, причем наименования деталей зафиксированы и при горизонтальной прокрутке будут все время оставаться на экране, что очень удобно при просмотре.

```
//Определение числа полей в загруженной базе данных
Fc:=FCOUNT()
```

```
//Запомнить ALIAS текущей рабочей зоны
Al:=ALIAS()
```

```
//Открытие БД 'DBF_RUS.dbf', содержащей
//информацию о заголовках столбцов
IF File("DBF_RUS.dbf").and.File("DBF_RUS.NTX")
USE DBF_RUS INDEX DBF_RUS NEW
//Определение числа возможных столбцов
Fc:=MIN((FCOUNT()-2)/3,Fc)
```

```
//Поиск в 'DBF_RUS' имени загруженной базы данных
SEEK UPPER((dbf))
```

```
//Переменная Ss определяет, как прошел поиск
Ss:=IF(.NOT.EOF().AND.dbf="DBF_RUS"..T.,Ss)
```

```
//Восстанавливает предыдущую рабочую зону с загруженной БД
Select (Al)
ENDIF
```

```
//Save screen, set color, etc.
cScreen := SAVESCREEN()
SETCOLOR("N/BG")
CLEAR SCREEN
```

```
//Добавить передачу переменных Ss и Fc
MyBrowse(3, 6, MAXROW()-2, MAXCOL()-6,Fc,Ss)
```

```
//Put things back
SET COLOR TO
@ MAXROW(), 0
RESTSCREEN(,,,cScreen)
```

```
QUIT
```

```
***
```

```
* MyBrowse()
* Create a Tbrowse object and browse with it.
*
```

```
//Добавить прием переменных Ss и Fc
STATIC PROCEDURE MyBrowse(nTop, nLeft, nBottom, nRight,Fc,Ss)
```

```
LOCAL browse //The TBrowse object
LOCAL cColorSave, nCursSave //State preservers
LOCAL nKey //Keystroke
LOCAL lMore //Loop control
```

```
//Make a "stock" Tbrowse object for the current workarea
//Добавить передачу переменных Ss и Fc
browse := StockBrowseNew(nTop, nLeft, nBottom, nRight, Fc, Ss)
```

```
...
```

```
...
```

```
...
```

```
...
```

```
...
```

```
...
```

Далее без изменений до

```
***
```

```
* StockBrowseNew()
* Create a "stock" Tbrowse object for the current workarea.
*
```

```
//Добавить прием переменных Ss и Fc
STATIC FUNCTION StockBrowseNew(nTop, nLeft, nBottom, nRight, Fc, Ss)
```

```
LOCAL browse
```



```

* //Вместо:
LOCAL n, column, cType
//Сделайте так:
LOCAL column, cType

//Start with a new browse object from TBrowseDB()
browse := TBrowseDB(nTop, nLeft, nBottom, nRight)

//Add a column for each field in the current workarea
FOR n := 1 TO Fc && Здесь вместо FCount() - > переменная Fc

//Определение заголовка столбца
IF Ss && Если в "DBF_RUS" найдено имя загруженной БД
VP_n := "DBF_RUS->P"+STR(n+10,2)+"_n"
VP_p := "DBF_RUS->P"+STR(n+10,2)+"_p"
VP_s := "DBF_RUS->P"+STR(n+10,2)+"_s.AND.&VP_p> 0"
ELSE
VP_n := "FieldName(n)"
VP_s := ".T."
VP_p := "n"
ENDIF

IF &VP_s && Если разрешено "высвечивание" текущего столбца
//Вместо строки:
//Make a new column
column := TBColmnNew(
* Field(n),
* FieldWBlock(Field(n), Select())
* )

//Сделать такую:
/* make the new column */
column := TBColmnNew( TRIM(&VP_n), ;
FieldWBlock(Field(n), Select()))
//Add the column to the browse
browse:addColumn(column)
ENDIF

NEXT
//Определение числа фиксируемых столбцов
browse:freeze := IIF(Ss, dbf_rus->Kol_fiks+1,1)

RETURN browse

...
...
... Далее без изменений до
...
...
***
** AddRecno()
* Insert a frozen column at the left that shows current record number
*

STATIC PROCEDURE AddRecno(browse)

LOCAL column

//Create the column object
column := TBColmnNew( " Rec #", {11 RECNO()})

//Insert it as the leftmost column
browse:InsColumn(1, column)

//Freeze it at the left
//Эту строку можно убрать
//browse:freeze := 1

RETURN

**
* Конец листинга программы Tbrus.prg

```

ПРЕДПРИЯТИЕ

ОВИМЭКС

ПРОДАЕТ:

- Компьютеры IBM PC/AT-286, 386, 486
- Сети, сетевое оборудование ARCNET
- Мониторы, принтеры, дисководы, платы
- Манипуляторы "мышь", блоки питания
- Клавиатуры, кабели
- Интерфейсы, адаптеры
- Вилки и розетки с заземлением
- Дискеты: 3,5", 5,25", чистящие диски
- Коробки для хранения дискет
- Кассеты с лентами для принтера: EPSON-1050, STAR NX-1500, OKI-183
- Микрокалькуляторы
- Пишущие машинки и телефаксы
- Множительные аппараты

ПРЕДЛАГАЕТ:

- ☐ Услуги по замене красящей ленты
 - ☐ Русификацию принтеров
 - ☐ Гарантийное и послегарантийное обслуживание
 - ☐ Разовый ремонт ПЭВМ
- а также
- ☐ Рассмотрение предложений коммерсантов по открытию совместных магазинов

ПРИГЛАШАЕТ:

- Специалистов для создания филиалов по техобслуживанию и ремонту компьютеров в городах СНГ с обеспечением запчастями
- Торговых агентов

КУПИТ:

- * Компьютеры IBM PC, бывшие в эксплуатации или неисправные

НАШ АДРЕС:

РОССИЯ, 620014,
г. Екатеринбург, ул. 8 Марта, 24

ТЕЛЕФОНЫ:

Коммерческий отдел:
(3432) 22-76-23
Отдел сбыта: (3432) 22-62-86
Факс: (3432) 517-456

Новые версии сетевой ОС Novell NetWare

Сравнение функциональных характеристик

М. Н. Белевская, А. А. Бухман,
Т. В. Карпова, К. И. Сумм, А. Н. Эсаулов

1991 год подтвердил лидерство фирмы Novell на рынке сетевых системных продуктов: ей принадлежат примерно 60% всех покупаемых в мире сетевых операционных систем (ОС). За последнее время широкое распространение получили новые версии сетевых ОС фирмы Novell: NetWare 2.2, NetWare 3.11, NetWare Lite 1.0.

Общая характеристика

Для ОС Novell NetWare 2.2 в качестве файл-сервера используют компьютеры, совместимые с ПК семейства IBM PC AT на базе микропроцессоров 80286, 80386, 80486 фирмы Intel или с компьютерами семейства PS/2. Предназначена она для небольших организаций, в которые входят рабочие группы, объединенные общим направлением деятельности. Отличаясь богатыми возможностями, ОС NetWare 2.2 удобна и при установке, и при использовании, и при администрировании.

Фирма поставляет модификации ОС NetWare 2.2, рассчитанные на 5, 10, 50 и 100 пользователей. Причем, если версии ранних сетевых ОС фирмы Novell (ELS I, ELS II, Advanced NetWare), предназначенные для малого числа пользователей, имели несравнимо меньшие возможности, чем рассчитанная на 100 пользователей ОС SFT NetWare, то модификации NetWare 2.2 для разного числа пользователей - абсолютно одна и та же ОС; различие этих модификаций заключается только в максимальном числе одновременно подключенных станций и в цене.

Операционная система NetWare 2.2 предоставляет возможность для использования в качестве рабочих станций ПК семейств IBM PC и PS/2, работающих под управлением DOS, Windows и OS/2, а также ПК семейства Macintosh. Причем все эти рабочие станции могут быть объединены в одну сеть.

В отличие от NetWare 2.2, ОС Novell NetWare 3.11 разработана специально для решения проблем, возникающих в больших и сложных сетях. Для этой мощной 32-разрядной сетевой ОС файл-сервер создается на базе микропроцессоров 80386 или 80486 фирмы Intel. Разные модификации ОС NetWare 3.11 рассчитаны на 10, 20, 100 и 250 пользователей. Как и для ОС NetWare 2.2, кроме числа одновременно подключенных станций и цены, эти модификации абсолютно ничем не различаются.

Очень существенным отличием NetWare 3.11 от NetWare 2.2 является то, что первая ОС состоит из ядра, постоянно функционирующего на файл-сервере, и дополнительных модулей (NetWare Loadable Modules - NLM), которые можно легко подгружать или удалять. Эти модули могут представлять собой драйверы, стеки протоколов, утилиты файл-сервера или прикладные программы, предназначенные для работы на файл-сервере.

Поддерживая многие широко распространенные коммуникационные протоколы, ОС NetWare 3.11 позволяет объединять в одну сеть ПК, работающие под управлением DOS, OS/2, Windows, UNIX, ПК семейства Macintosh, мини- и большие ЭВМ. Рабочие станции разных типов, функционирующие в сети под управлением NetWare 3.11, могут создавать на файл-сервере файлы с именами, присущими ОС, под управлением которых они работают, при этом форматы представления файлов также соответствуют ОС рабочих станций.

Проблемы установки сетевой ОС

На практике установка ОС NetWare оказывается делом совсем не элементарным. Это связано с тем, что фирма Novell ужесточила требования, предъявляемые к оборудованию, предназначенному для использования в локальной

сети, работающей под управлением новых версий ОС NetWare. Фирма Novell регулярно публикует перечень фирм, оборудование которых ею сертифицировано, т.е. гарантируется его пригодность в качестве соответствующих элементов файл-сервера или рабочей станции. Предполагается, что именно на такой технике будет создаваться локальная сеть. Именно поэтому для ОС NetWare 3.11 резко ограничен выбор драйверов сетевого адаптера.

К сожалению, отечественный рынок средств вычислительной техники таков, что оборудование, сертифицированное фирмой Novell, доступно немногим. При установке ОС NetWare разных версий нам приходилось сталкиваться с большим количеством самого различного оборудования. Накопленный нами опыт позволяет расширить перечень сертифицированного фирмой Novell оборудования с учетом особенностей нашего компьютерного рынка. Этот опыт обобщен в выпускаемых нами бюллетенях, аналогичных бюллетеням фирмы Novell. Фрагмент такого бюллетеня приведен в табл.1.

*Марина Наумовна Белевская,
Алла Ароновна Бухман,
Татьяна Валентиновна Карпова,
Константин Игоревич Сумм,
Александр Николаевич Эсаулов -
сотрудники совместного
советско-британского предприятия
"Компьютерные системы
и технологий".
Работают с ОС Novell NetWare
с 1987 года.
Занимаются поддержкой
локальных сетей,
функционирующих под управлением
Novell NetWare.
Публиковали статьи по этому вопросу
в журналах "Интеркомпьютер",
"Журнал д-ра Добра",
"КомпьютерПресс".
Телефон: 923-10-86.*

Т а б л и ц а 1. Результаты проверки работы контроллеров дисков на файл-сервере

Фирма	Тип контроллера	Версия ОС	Результат тестирования
JOINCOM	ESDI JC 1610 REV.A	2.15	+
JOINCOM	ESDI JC 1610 REV.A	2.2	-
JOINCOM	ESDI JC 1610 REV.B	2.15, 2.2	+
Adaptec	ACB-2322 ESDI	2.15, 3.11	+
Adaptec	1642B SCSI	2.15, 2.2, 3.11	+
WD	WD 1007A	2.2	+
Data Tec.	DTC 6280	2.2	+

Освоение ОС NetWare

С нашей точки зрения, документация, поставляемая с ОС NetWare 2.2 и 3.11, является более полной и удачнее организованной, чем документация на предыдущие версии. Кроме того, в версиях NetWare 2.2 и 3.11 появилась утилита HELP, дающая пользователю возможность в интерактивном режиме и в удобной форме получать информацию об утилитах ОС, системных сообщениях и основных понятиях ОС NetWare.

Однако для начинающего администратора или пользователя сети незаменимую помощь могут оказать обучающие программы фирмы Novell. (Аналоги таких программ созданы и на русском языке.)

Администрирование

После того как локальная сеть установлена, сразу же возникают проблемы, связанные с ее администрированием, к функциям которого относятся:

- "прописка" пользователей в сети;
- организация каталогов в томах файл-сервера;
- обучение пользователей и контроль за их действиями, обеспечение надлежащей технологии работы пользователей;
- изменение конфигурации ОС (в случае необходимости);
- учет ресурсов пользователя;
- диагностика сбоев и ошибок;
- защита от несанкционированного доступа;
- тестирование, поддержка целостности, сохранение и восстановление информации.

Для начинающего администратора сети в ОС NetWare 2.2 предусмотрена новая, дополнительная утилита NWSETUP, предназначенная для "прописки" пользователей в сети.

К остальным средствам "прописки" пользователя и определения соответствующих ему параметров (утилиты SYSCON и MAKEUSER), имевшимся в ОС NetWare 2.15, а также в ОС NetWare 2.2 и 3.11, добавилась новая ути-

лита - USERDEF, предназначенная для "прописки" большого количества пользователей.

Организация каталогов в томах файл-сервера, как и в предыдущих версиях NetWare, осуществляется средствами MS-DOS с рабочей станции. Для просмотра каталогов и выполнения различных операций над ними помимо средств MS-DOS можно пользоваться утилитами NDIR, LISTDIR, RENDER, FILER. По сравнению с версией NetWare 2.15 здесь изменений нет.

Для организации работы с различными объектами сети и ее поддержки в ОС NetWare существует внутренняя база данных BINDARY. Ее структура в версии NetWare 3.11 изменилась: эта база данных теперь состоит не из двух файлов, а из трех.

В новых версиях NetWare - 2.2 и 3.11, появились средства для изменения некоторых параметров сетевой ОС и драйвера сетевого адаптера: утилиты DCONFIG, ECONFIG и JUMPERS.

Утилита DCONFIG позволяет менять ряд параметров, определяющих конфигурацию файл-сервера, внешнего моста или рабочей станции.

Утилита ECONFIG применяется в сетях типа Ethernet для изменения протокола, используемого драйвером сетевого адаптера, с Ethernet II на IEEE 802.3 и наоборот.

Утилита JUMPERS служит для программного изменения конфигурации драйвера сетевого адаптера на файл-сервере или на рабочей станции.

Средства учета ресурсов пользователей в NetWare 2.2 и NetWare 3.11, как и в NetWare 2.15, дают администратору сети возможность накапливать следующую информацию: число блоков, считанных и записанных на сетевой диск, число блоков, записанных на сетевой диск за день, число запросов на ресурсы файл-сервера. Эти средства предоставляются утилитой SYSCON.

Новая утилита - PAUDIT, позволяет просматривать журнал всех обращений пользователей к файл-серверу, а новая утилита - ATOTAL, выдает итоги ис-

пользования ресурсов файл-сервера за день и неделю.

Средства сбора статистической информации

В версии NetWare 2.2 эти средства практически не отличаются от используемых в версии 2.15. Простейшую статистическую информацию можно получить с помощью консольной команды MONITOR. Эта информация содержит такие сведения, как процент времени, используемого процессором файл-сервера для выполнения сетевых запросов, за последнюю секунду (обновляется каждую секунду); число кэш-буферов, информация из которых еще не переписана на диск; активность каждой рабочей станции и состояние открытых файлов. Более подробную статистическую информацию, как и раньше, позволяет получить утилита FCONSOLE ОС NetWare 2.2.

Кроме того, в версиях NetWare 2.2 и 3.11 существуют консольные команды RESET ROUTER, TRACK ON и TRACK OFF, позволяющие отслеживать пересылку пакетов по сети. Они использовались и в версии 2.15, но не были там описаны.

Команда DISPLAY NETWORKS позволяет увидеть, какие сети соответственно поддерживает файл-сервер, а команда DISPLAY SERVERS - какие файл-серверы доступны данному файл-серверу.

Однако наиболее полную статистическую информацию о работающей сети можно получить с помощью дополнительного продукта фирмы Novell - NetWare Care, позволяющего увидеть на экране план любой ветви сети, в том числе все серверы и мосты. Таким образом, можно следить за изменением трафика по сети от узла к узлу. NetWare Care выдает подробную статистическую информацию как по использованию ОС NetWare ресурсов файл-сервера, так и по работе каждого узла сети. Возможно выполнение тестов на пересылку пакетов между двумя произвольными узлами и их получение.

Для сбора статистической информации могут быть использованы и аппаратные средства. Таким средством является, например, включаемое в локальную сеть устройство LANalyzer, которое анализирует каждый передаваемый по сети пакет. Результат может выдаваться на экран либо на печать. Кроме того, LANalyzer позволяет получить статистические данные о производительности работы файл-сервера и мостов, а также о загрузенности сети.

Для сбора статистической информации в NetWare 3.11 служат следующие средства:

Консольные команды, являющиеся частью программы `server.exe`, работающей на файл-сервере:

- **MEMORY** - позволяет узнать общий объем адресуемой памяти, установленной на файл-сервере;
- **SPEED** - определяет скорость работы процессора и позволяет правильно задать ее с помощью аппаратных средств;
- **SET** - дает возможность определить текущие параметры операционной системы. Большинство из них не требуют переопределения, но некоторые могут быть изменены для получения большей производительности (что команда **SET** и позволяет сделать).

Утилита **MONITOR**, работающая под управлением ядра ОС на файл-сервере. Это загружаемый модуль, который позволяет получить информацию о распределении памяти для загруженных в нее NLM-модулей, а также выяснить, освободились ли ресурсы после завершения работы загружаемого модуля. Администратор сети может получить некоторые общие параметры, характеризующие производительность системы:

- число первоначально загруженных кэш-буферов и число доступных в настоящий момент буферов;
- число запросов, которые ждут своей очереди быть записанными на диск;
- число "серверных процессов", т.е. число ссылок на задачи для выполнения запросов станции;
- число открытых станций и число открытых операционной системой и рабочими станциями файлов;
- состояние жесткого диска: его размер, область переопределения блоков и ее статус; можно также определить, какие тома и их сегменты находятся на испорченном диске;
- оценку загруженности кабельной системы (по информации, выдаваемой сетевым драйвером);
- оценку занимаемых ресурсов;
- информацию о том, как часто используется файл, как он используется, сколько пользователей с ним работает и был ли он заблокирован.

Для ваших нужд ОС NetWare 3.11 будет выделять дополнительную память. Она предупредит, если оставшегося объема памяти будет мало, и предупредит также, если не сумеет выделить для ресурса дополнительную память. Использование ресурсов можно планировать, если иметь информацию об использовании кэш-буферов (*cache buffers*);

буферов пересылаемых пакетов (*packet receive buffers*); сервисных процессов (*file service processes*), а также о распределении памяти для ненадолго загружаемых запросов (*allocation short term memory*). Однако данная информация нигде не накапливается, поэтому трудна для анализа.

Диагностика сбоев

К сожалению, как и в предыдущих версиях ОС NetWare, в версиях 2.2 и 3.11 средства диагностики сбоев оборудования и программного обеспечения практически отсутствуют. Интерес представляет утилита **COMCHECK**, предназначенная для проверки кабельной системы.

Для диагностики сбоев дополнительно созданы как аппаратные, так и программные средства. Таким аппаратным средством является, например, упомянутое выше устройство LANalyzer, позволяющее получать информацию о неверно сформированных пакетах, "столкновениях" пакетов, нарушениях кабельной системы и т.д.

Исключительно удобным средством диагностики для NetWare 2.2 является также NetWare Care. Этот продукт позволяет определить следующее:

- наличие потерянных входных или выходных пакетов на файл-сервере;
- нехватку памяти на мосте для обеспечения необходимого сервиса;
- наличие "столкновений" при попытке передачи пакетов рабочей станцией вследствие перегруженности сегмента сети;
- наличие в сети слишком большого количества мостов или порчу таблицы маршрутизации (*router table*) на одном из мостов;
- число возвращений рабочей станции сообщения "Сервер занят" вместо выполнения запроса файл-сервером;
- неверное формирование пакетов в сети;
- число слишком больших и слишком маленьких пакетов, сформированных прикладными программами, и другие сбои.

Существуют и другие средства диагностики (не фирмы Novell): **EMONITOR**, **ARCMONITOR**, **NETPROBE**, и др. Эти средства очень полезны для организации сервиса и поддержки.

Защита от несанкционированного доступа

Средства защиты от несанкционированного доступа в NetWare 2.2 и 3.11 обеспечивают те же четыре уровня защиты, что и в NetWare 2.15:

- присвоение пользователю пароля и введение других ограничений для получения доступа к ресурсам сети;
- предоставление пользователю различных прав доступа к разным каталогам на сетевом диске;
- закрытие от пользователей некоторых каталогов на сетевом диске;
- назначение файлам атрибутов, регламентирующих работу пользователя с этими файлами.

Обеспечиваемые ОС NetWare 2.2 возможности защиты от несанкционированного доступа принципиально не отличаются от аналогичных возможностей NetWare 2.15, за исключением, пожалуй, того, что некоторые атрибуты теперь можно присвоить не только файлам, но и каталогам.

Использование в ОС NetWare 2.2 и 3.11 новой команды - **SECURITY**, позволяет администратору сети узнать о возможных нарушениях существующей организации защиты от несанкционированного доступа.

Для блокировки консоли файл-сервера, работающего под управлением NetWare 2.2 (так же, как и в случае применения ОС NetWare 2.15) на нем может быть использован VAP-процесс (Value Added Process) **LOCK.VAP**.

Немного изменился набор предоставляемых NetWare 3.11 прав и атрибутов. Кроме того, изменился механизм защиты от несанкционированного доступа к каталогам на сетевом диске.

Защита консоли файл-сервера, работающего под управлением NetWare 3.11, может осуществляться двумя способами:

- утилита **MONITOR** позволяет полностью блокировать консоль файл-сервера;
- утилита **SECURE CONSOLE** ограничивает использование операций, которые можно выполнить с консоли файл-сервера.

Несмотря на некоторые изменения в организации защиты от несанкционированного доступа в NetWare 3.11, на технологию работы в сети они принципиально не влияют.

В сетях, работающих под управлением NetWare 2.2 или NetWare 3.11, помимо старых типов пользователей (супервизора, оператора консоли, оператора печати и обычного пользователя) появились новые: менеджер группы, менеджер ресурсов и оператор сервера, обеспечивающего разделение ресурсов печати (*print server*).

Однако эти новые типы пользователей требуются для очень большого числа пользователей в сети (более 300).

В сети, работающей под управлением NetWare 2.2 или NetWare 3.11, можно использовать средство NetWare Name Service (NNS), позволяющее создать домен серверов в сети, где работают несколько файл-серверов. При этом создается общая база данных ресурсов домена, что существенно упрощает администрирование многосерверной сети и позволяет использовать любые ресурсы домена без выполнения команд login и attach для различных файл-серверов.

Технология работы в сети

Залогом стабильности, надежности и эффективности использования ЛВС является правильно организованная технология работы в сети: выбор структуры каталогов сетевого диска, создание групп пользователей, четкое распределение прав и обязанностей пользователей, организация защиты от несанкционированного доступа, создание графика регламентных работ и четкое его выполнение. Рекомендации, которые фирма Novell дает по организации технологии, с нашей точки зрения, недостаточны, однако эта проблема довольно серьезна и требует специального рассмотрения, что мы и предполагаем сделать в одном из последующих номеров "Журнала д-ра Добба".

Особенности использования рабочей станции

У рабочей станции, функционирующей в сети под управлением ОС NetWare 2.2 или 3.11, появляются новые возможности. Если на компьютере установлена дополнительная (*expanded*) или расширенная (*extended*) память, то она может быть использована для работы сетевой оболочки. Таким образом экономится примерно 34 Кбайт основной памяти, используемых под прикладные программы. Однако при этом теряется производительность. Новая утилита - WUPDATE, предназначена для обновления файлов, если они отличны от файлов с теми же именами на сетевом диске. Эта утилита позволяет, например, автоматически обновлять на компьютере сетевую оболочку.

Защита целостности, восстановление и сохранение информации

Средства защиты целостности, предоставляемые ОС NetWare 2.2 и NetWare 3.11, обеспечивают высокую надежность сети. Как и NetWare 2.15 SFT, версии NetWare 2.2 и NetWare 3.11 обеспечивают уровень II защиты от сбоев, предусматривающий выполнение таких операций, как:

- дублирование FAT (*fat allocation table*) и DET (*directory entries table*) на разных цилиндрах сетевого диска;
- верификация информации непосредственно после ее записи на диск;
- динамическая переадресация информации, попавшей на "плохие" блоки диска (HOT FIX);
- "зеркальная" запись информации на два диска;
- запись информации на два диска при использовании разных контроллеров;
- применение системы поддержки транзакций (*transaction tracking system - TTS*), осуществляющей их откат;
- поддержка источника бесперебойного питания, позволяющая корректно завершить работу в сети в случае нарушения энергоснабжения.

Дублирование FAT и DET и верификация информации осуществляются операционной системой независимо от параметров ее установки. Размер области HOT FIX обычно составляет 2% от физического объема диска. Если известно, что поверхность диска содержит большое число дефектных блоков, то размер области HOT FIX можно увеличить при установке операционной системы. Дублирование информации на двух дисках является довольно дорогим способом защиты, причем требующим более квалифицированной поддержки. Пользоваться им целесообразно только в том случае, если речь идет о дисках, имеющих большие объемы и содержащих очень ценную информацию. Поддерживать сеть, в которой использован такой способ защиты, должен достаточно грамотный администратор. Необходимым условием обеспечения защиты целостности информации, содержащейся на сетевом диске, является использование TTS. Работа TTS подробно описана в статье С.А. Астафьева и др. "СУБД Btrieve в среде Novell" (КомпьютерПресс, 1991, №8). Не рекомендуется также использовать невыделенный файл-сервер, хотя NetWare 2.2 и позволяет это сделать. Работа в сети с невыделенным файл-сервером крайне ненадежна: не может использоваться TTS, а зависание прикладной программы, работающей на файл-сервере, вызывает зависание всей сети. Разделение памяти файл-сервера между операционной системой и прикладной программой нетехнологично.

В NetWare 3.11 хранятся четыре копии таблицы, содержащей информацию о томах и составляющих их сегментах. ОС NetWare 3.11 создает и поддержива-

ет новые файлы - TT\$LOG.ERR и VOL\$LOG.ERR. Файл TT\$LOG.ERR представляет собой журнал работы TTS, где фиксируется инициализация TTS, полная информация о каждом случае незавершения транзакции, произведенных откатах и о завершении работы TTS. В файл VOL\$LOG.ERR, создаваемый для каждого тома, заносятся все системные сообщения, касающиеся данного тома.

Несмотря на то что в рассматриваемых версиях ОС NetWare имеются мощные средства защиты от потери информации, ОС предоставляет администратору сети дополнительные возможности для создания архивных копий.

В версиях NetWare 2.2 и 3.11 появилась утилита NBACKUP, позволяющая сохранить и восстановить копию как сетевого, так и локального диска.

В NetWare 2.2, как и в некоторых предыдущих версиях ОС NetWare, для архивирования содержимого сетевого диска на стример и соответствующего восстановления можно запустить VAP-процесс BACKUP на любом из файл-серверов сети или на внешнем мосте. С помощью утилит NBACKUP и BACKUP.VAP можно скопировать как всю информацию, содержащуюся на файл-сервере, в том числе информацию об объектах сети (*bindary*) и правах пользователей (*trustee*), так и отдельные тома, каталоги или файлы сетевого диска. Необходимо отметить, что эти утилиты дают возможность архивировать файлы, в которых используются соглашения об именах, присущих DOS или ОС Macintosh.

В NetWare 3.11 имеется утилита SBACKUP.NLM, работающая под управлением ядра ОС. Эта утилита позволяет архивировать созданные на файл-сервере файлы, в которых используются соглашения об именах, присущих DOS, ОС Macintosh, OS/2 или UNIX.

Удаленный доступ

В NetWare 3.11 появилось новое средство для администратора удаленной сети - Remote Console. С его помощью администратор сети, запустив NLM-модули на файл-сервере, может на рабочей станции или отдельно стоящем компьютере эмулировать консоль файл-сервера и управлять с данной консоли его работой. При этом реальные консоль и клавиатура файл-сервера по-прежнему остаются доступными. В любой момент администратор сети может прекратить действие Remote Console. Это средство работает как при прямом доступе к файл-серверу, так и через асин-

хронный модем, т.е. вы можете получить доступ к консоли файл-сервера, находящегося в одной локальной сети с компьютером администратора или в локальной сети, связанной с вашей ЛВС через мост (внешний или внутренний), но без модемной связи, а можете иметь доступ к консоли файл-сервера, соединенного с вашей рабочей станцией через асинхронный модем, совместимый с модемом фирмы Hayes.

Помимо управления консолью файл-сервера Remote Console позволяет также установить ОС на новом файл-сервере с удаленной консоли. Для этого с помощью утилиты RSETUP создается загрузочная дискета, которую затем передают специалисту, устанавливающему ее в дисковод файл-сервера, где будет работать ОС NetWare; файл-сервер загружается с этой дискеты, а все дальнейшие шаги по установке ОС NetWare выполняются с удаленной консоли. Вы можете также копировать файлы с диска компьютера администратора сети, где происходит эмуляция консоли файл-сервера, на диск файл-сервера, а также перезагружать файл-сервер. К сожалению, ряд ошибок, приводящих к зависаниям, затрудняет интенсивное использование средства Remote Console.

Другим средством удаленного доступа, часто используемым при работе с ОС NetWare, является NetWare Access Server. Это дополнительный продукт фирмы Novell, поставляемый отдельно от ОС NetWare 2.2 и 3.11. Мы рассматриваем его, так как он представляет практический интерес, но мало известен на нашем рынке.

NetWare Access Server позволяет реализовать мультизадачность и администрирование удаленных рабочих станций. Функционирует это средство на специально выделенном компьютере (Access Server) на основе микропроцессоров 80386 или 80486. Работая на удаленном ПК семейств IBM PC или Macintosh либо на терминале, вы можете иметь доступ к файлам, хранящимся в локальной сети, и серверам, обеспечивающим разделение ресурсов печати, используя телефонные каналы связи, линии связи с коммутацией пакетов X.25, коммуникационный сервер (NetWare Asynchronous Communication Server) или соединения сетей. Выполнение заказанной пользователем работы происходит в памяти Access Server. При этом пользователь может открывать несколько сессий, выполнять заранее сформированные задания, а администратор сети может контролировать действие сети или с

консоли Access Server, или с консоли любой рабочей станции. Для работы Access Server нужны сетевой адаптер и специальный коммуникационный адаптер, позволяющий использовать несколько последовательных портов.

На рабочих станциях связь с Access Server осуществляется с помощью коммуникационной программы OnLAN/PC или программы эмуляции терминала, для которых не требуется ни установки сетевых адаптеров, ни сетевой оболочки. Для работы Access Server необходимы по крайней мере 640 Кбайт оперативной памяти, расширенная оперативная память для организации виртуальных сессий, дисководы на 1,44 или 1,2 Мбайт, коммуникационные адаптеры для подключения нескольких устройств через последовательный порт. Access Server не поддерживает работу ни "мыши", ни цифрователя. С разрешения администратора сети пользователи удаленных станций могут работать с дисками, установленными на Access Server, но не могут использовать принтеры, подключенные непосредственно к этому компьютеру.

Создание приложений

В локальной сети приложения могут функционировать на рабочей станции или на файл-сервере. Работа последних более эффективна и надежна, поскольку такие приложения обеспечивают следующие преимущества:

- уменьшение сетевого трафика (меньший объем данных должен передаваться по сети);
- сокращение числа запросов на файл-сервер;
- экономия памяти на рабочей станции;
- повышение надежности защиты от сбоя.

Таким образом, очевиден повышенный интерес к проектированию и созданию приложений, функционирующих непосредственно на файл-сервере. Эти приложения должны быть организованы как VAP-процессы в версиях ОС NetWare, ориентированных на процессор 80286, или как NLM-модули в версиях ОС NetWare, ориентированных на процессор 80386. Средства для разработки таких приложений отсутствуют в стандартной поставке ОС NetWare и предоставляются фирмой Novell дополнительно: Novell C Interface for DOS, Novell C Compiler 268, Novell C Compiler 368.

Создание VAP-процессов делает возможной разработку в среде ОС NetWare программно-технологических

комплексов для передачи сообщений, обработки прерываний, ведения журнала ОС и т.д.

К сожалению, сетевая ОС не позволяет запускать отдельные VAP-процессы - либо запускаются все имеющиеся в системе VAP-процессы, либо не запускается ни один из них. Прекращение работы VAP-процессов осуществляется командой DOWN файл-сервера, приводящей к перезагрузке системы, либо специальной командой конкретного VAP-процесса, но без возможности последующей перезагрузки данного процесса.

Ядро сетевой ОС управляет каждым VAP-процессом как независимой операцией. Готовый к выполнению процесс сетевой ОС помещает в очередь на обработку. После выполнения процесс "засыпает" и остается в таком состоянии до тех пор, пока "не изъявит желания проснуться", затем повторяется. Сетевая ОС планирует выполнение процессов в соответствии с их приоритетами.

Программирование VAP-процессов на языке высокого уровня (например, языке C фирмы Microsoft) сопряжено с набором ограничений, налагаемых самой сетевой ОС. Это, прежде всего, отказ от применения стандартных функций, использующих различные опции прерывания 21h DOS (sprintf(), settextposition() и т.д.), а также функций динамического распределения памяти (malloc(), calloc() и пр.). Необходимо иметь в виду опасность такой предоставляемой компилятором языка C фирмы Microsoft возможности, как проверка стека (STACK CHECK). Пользоваться этой возможностью не нужно. Программирование VAP-процессов трудоемко и потребовало от нас значительных усилий.

В отличие от VAP-процессов, NLM-модули, являясь частью сетевой ОС, способны более широко использовать ее ресурсы. Если VAP-процессы взаимодействуют с ОС NetWare только с помощью ограниченного набора функций, то NLM-модули способны, кроме того, напрямую, без дополнительного интерфейса обращаться к внутренним программам сетевой ОС.

Среди наиболее важных достоинств NLM-модулей следует отметить следующие:

- возможность их динамического запуска и завершения работы;
- способность распределения и перераспределения памяти;
- способность взаимодействия с другими NLM-модулями.

Таблица 2. Сравнение новых версий ОС Novell NetWare

Характеристики	Версия ОС			
	2.15 SFT	2.2	3.11	Lite 1.0
ОС, которая может быть установлена на рабочей станции				
DOS 2.X	+	+	+	-
DOS 3.X	+	+	+	+
DOS 4.X	+	+	+	+
Windows/386	+	+	+	+
OS/2 V1.1	+	+	+	-
Macintosh	+	+	+	-
Поддержка NETBIOS	+	+	+	+
Поддержка протоколов SPX/IPX	+	+	+	+
Динамическое изменение конфигурации	-	-	+	-
Администрирование группы пользователей	-	+	+	-
Кодирование паролей	-	+	+	-
Поддержка NETBIOS через мосты	+	+	+	-
Поддержка асинхронного удаленного моста через порты COM1/COM2	+	+	+	-
Учет ресурсов	+	+	+	-
Расширенная защита от несанкционированного доступа	+	+	+	-
Уровень защиты от сбоев	II	II	II	-
Управление источником бесперебойного питания	+	+	+	-
Работа с системой поддержки сообщений (MHS)	+	+	+	-
Эмуляция терминала 3270 на рабочей станции	+	+	+	-
Тип файл-сервера (В - выделенный, Н - невыделенный), микропроцессор	В 80286 Н 80286	В 80286 Н 80286	В 80386	В 80286 Н 80286
Тип контроллера жестких дисков	MFM, ESDI, SCSI	IDE, MFM ESDI, SCSI	IDE, MFM ESDI, SCSI	Все контроллеры, работающие с DOS
Максимальное число пользователей	100	5, 10, 50, 100	10, 20, 100, 250	25
Объем оперативной памяти файл-сервера, Мбайт	1-15	2,5-12	4-4·10 ³	*
выделенный файл-сервер				
невыделенный файл-сервер	1,5-15	2,5-8		*

Все это позволяет реализовывать на базе NLM-модулей более смелые, с точки зрения проектировщика, проекты.

Было бы упрощением полагать, что NLM-модули представляют собой резидентные программы, которые выполняются на файл-сервере и пользуются частью его ресурсов. В действительности дело обстоит немного иначе: NLM-модуль, однажды загруженный в память файл-сервера, становится частью сетевой ОС и, естественно, способен более тесно взаимодействовать с ресурсами файл-сервера. В этом, кстати, и кроется причина коренных отличий NLM-модулей от VAP-процессов.

Следовательно, NLM-модуль способен пользоваться целым рядом ранее недоступных сетевых ресурсов и, кроме того, изменять некоторые элементы сетевой ОС, в частности такие, как стек протоколов (*protocol stack*). При этом доступ NLM-модуля к системным ресурсам осуществляется на основе прямого вызова внутренних системных программ ОС NetWare.

Важным качеством NLM-модулей является возможность их динамического запуска на файл-сервере. Запуск осуществляется по команде LOAD ОС NetWare, при этом могут быть запущены не все имеющиеся в системе NLM-модули, а только некоторые из них. Завершение работы загруженного NLM-модуля также производится динамически командой UNLOAD. Все это повышает гибкость программного обеспечения, реализованного на основе NLM-модуля.

Процесс создания NLM-модуля можно условно разбить на следующие этапы:

- написание программы при использовании компилятора Watcom C или языка ассемблера. При этом процесс программирования NLM-модулей практически является написанием обычной программы. Это выгодно отличает NLM-модули от VAP-процессов, программирование которых сопряжено с соблюдением целого ряда ограничений;
- трансляция исходной программы, получение OBJ-файла;
- редактирование OBJ-файла с помощью редактора NLM-LINK и получение загружаемого модуля, т.е. файла со стандартным расширением .LAN, .DSK, .NLM или .NAM в зависимости от типа NLM-модуля.

NetWare Lite v1.0

В ряду сетевых ОС фирмы Novell ОС NetWare Lite занимает особое место. Это простая дешевая сетевая ОС, позволяющая взаимодействовать несколькими

пользователям персональных компьютеров, которым требуется только базисный набор сетевых функций и возможностей.

Эта система, реализованная как надстройка над DOS, предоставляет пользователям возможность совместно использовать файлы, программные средства и принтеры. Система может работать в среде Microsoft Windows.

Для объединения персональных компьютеров в сеть достаточно вставить в них платы, предназначенные для использования в сети ARCNET, Ethernet или Token Ring, и соединить их между собой кабелем. Сеть может содержать от 2 до 25 компьютеров, на каждом из которых должна быть установлена своя ОС NetWare Lite, причем количество пользователей может расти по мере расширения сети. Для установки ОС NetWare Lite даже непрофессионалам требуется всего несколько минут. Так же мало времени займет изменение конфигурации доступных ресурсов сети.

Каждый компьютер сети может быть объявлен сервером, сервером-клиентом или клиентом. Сервер может использовать свои ресурсы (файлы на жестком диске, подключенные к компьютеру принтеры) совместно с другими компьютерами сети. При этом сам он является выделенным, т.е. не может служить рабочей станцией.

Для одновременного использования одного компьютера в качестве сервера и рабочей станции его следует объявить сервером-клиентом. В этом случае компьютеру будут доступны ресурсы других серверов и он сможет разделять свои собственные ресурсы между пользователями сети.

Если подключенный к сети компьютер объявлен клиентом, он может функционировать лишь как рабочая станция, т.е. пользоваться ресурсами

Характеристики	Версия ОС			
	2.15 SFT	2.2	3.11	Lite 1.0
Максимальная емкость памяти на диске, Гбайт	2	2	$32 \cdot 10^3$	*
Максимальный размер тома, Мбайт	255	255	$32 \cdot 10^6$	
Максимальное число томов на один файл-сервер	32	32	64	
Максимальный размер файла, Мбайт	256	256	$4 \cdot 10^3$	*
Максимальное число открытых файлов	10^3	10^3	10^5	*
Максимальное число одновременно открытых индексных файлов	10^3	10^3	10^5 (при установке ограничения не задаются)	
Максимальное число активных транзакций	200	200	10^4	
Возможный размер блока, Кбайт	4	4	4, 8, 16, 32, 64	4
Возможность соединения нескольких сетей	+	+	+	-

* Ограничения налагаются внешней средой.

серверов, не имея возможности предоставить свои ресурсы другим компьютерам сети.

Для защиты от несанкционированного доступа к сетевым ресурсам предусмотрены специальные средства: присвоение пользователям паролей и наборов полномочий, а также разграничение доступа пользователей.

Синтаксис команд и утилит NetWare Lite и описанных выше версий ОС

NetWare одинаков. Поэтому ОС NetWare Lite, прежде всего, полезна в том случае, если пользователи, уже имеющие опыт работы в больших сетях, функционирующих под управлением ОС Novell NetWare, должны обеспечить действие небольшой сети (5-6 компьютеров).

В табл.2 обобщены результаты сравнительного анализа новых версий сетевых ОС, выпущенных фирмой Novell.

МОСКОВСКИЙ МЕЖДУНАРОДНЫЙ ФЕСТИВАЛЬ КОМПЬЮТЕРНОЙ ГРАФИКИ И АНИМАЦИИ

Это форум для теле- и кинокомпаний, коммерческих предприятий, научно-исследовательских и учебных институтов, рекламных агентств, анимационных центров и студий компьютерной графики.

Программа фестиваля:

- конкурс кино- и видеоработ (компьютерная графика и анимация)
- информационный и ретроспективный показы
- выставка видеокомпьютерных технологий и мультимедиа
- выставка компьютерного искусства
- конференции, семинары, круглые столы
- видеокомпьютерные уроки

ПРИГЛАШАЮТСЯ УЧАСТНИКИ, СПОНСОРЫ, РЕКЛАМОДАТЕЛИ

Наш адрес: 129128, Россия, Москва, ул. Будаевская, 3, ком. 217—220

Телефон: (095) 187-75-60 • Факс: (095) 187-75-60

СИТЕК — КОМПЛЕКСНАЯ ПОДДЕРЖКА ВАШЕГО БИЗНЕСА

- **КОНЪЮНКТУРА РЫНКА:** валютные курсы, биржевые цены, прогнозы, аналитическая информация
- **ТелеБАЗАР** - торговля без посредников, тысячи прямых коммерческих предложений
- **БАЗЫ ДАННЫХ:** "Продажа недвижимости", "Производители и потребители", "Зарубежный партнер", "Банки", "Биржи"
- **РЕКЛАМА** Ваших товаров и услуг
- **ЭЛЕКТРОННЫЕ ИЗДАНИЯ:** экономика, финансы, право, менеджмент, маркетинг
- **КОММУНИКАЦИИ:** электронная почта, факсимильная, телексная, телеграфная связь
- **ЭЛЕКТРОННЫЙ РАСЧЕТНЫЙ ЦЕНТР**
- **СЛУЖБЫ БЕЗОПАСНОСТИ:** электронная подпись, криптозащита



Став абонентом СИТЕК, Вы
сможете делать свой бизнес
не выходя из офиса или квартиры,
используя лишь ПЭВМ,
модем и телефон

Вы не работали в сетях?
Вам помогут курсы
"СИТЕК от А до Я" в учебно-научном
центре "МГТУ-МАСТАК"

**СИТЕК - ЭТО ДВУХЛЕТНИЙ ОПЫТ В БИЗНЕСЕ,
ЛИЦЕНЗИЯ МИНСВЯЗИ РОССИИ, ТЫСЯЧИ АБОНЕНТОВ,
СОТНИ ГОРОДОВ**

ПРИГЛАШАЕМ АГЕНТОВ, ДИЛЕРОВ, ДИСТРИБУТОРОВ
ДЛЯ ВЗАИМОВЫГОДНОГО СОТРУДНИЧЕСТВА И
ОТКРЫТИЯ РЕГИОНАЛЬНЫХ ПРЕДСТАВИТЕЛЬСТВ



Научно-производственная фирма «МАСТАК»
111020, Москва, а/я 100
Тел.: (095) 360-8638, 360-8527, 360-6909
Факс: (095) 360-7874
Эл. почта: secret@mastak.msk.su

Запрашивайте
бюллетень
услуг СИТЕК
— вышлём
немедленно
и бесплатно

Создание надежных программ

*Новый администратор динамической памяти
компилятора Turbo Pascal 6.0.*

С. Н. Кучеров

Можете ли вы гарантировать, что в программе, которую вы написали и отладили, больше нет ни одной ошибки? На этот вопрос принято отвечать: "Нет", - хотя каждый программист, долго и тщательно тестируя свою программу, стремится ответить: "Да". Однако известно, что тестирование помогает обнаружить в программе ошибки, а не доказать их отсутствие. Единственный способ написать программу без ошибок (надежную программу) - с самого начала писать ее правильно. Другими словами, нужно выполнять ряд условий и правил, которые, хотя и не обязательны, с точки зрения языка программирования, но помогают предотвратить появление многих ошибок.

Перефразируя классика, можно сказать, что редкая ошибка доживет до бета-версии, если при компиляции в программу были включены процедуры контроля диапазонов, переполнения стека и ввода-вывода. Это утверждение не относится, однако, к большинству ошибок, связанных с использованием динамических переменных. Компилятор Turbo Pascal версии 6.0 предоставляет программистам мощные средства управления динамической памятью, позволяющие предотвратить появление в программе подобных ошибок. В данной статье вашему вниманию представляется обзор этих возможностей.

Структура программы

В заголовке получаемого в процессе компиляции EXE-файла содержится информация о максимальном и минимальном размерах блока памяти, отводимого для данной программы. Эти размеры можно установить директивой компилятора \$M, которая должна выглядеть так:

`{$M Стек, Минимум, Максимум}`

Здесь "Стек" - размер стека программы (от 0 до 64 Кбайт), "Минимум" и "Максимум" - соответственно минимальный и максимальный размер блока памяти, выделяемого для этой программы. Если при запуске программы размер свободной памяти меньше, чем указано в параметре "Минимум", то будет выдано сообщение `Program too big to fit in memory` (программа слишком велика, чтобы поместиться в памяти). В противоположном случае программе будет выделен блок, размер которого задается параметром "Максимум".

При загрузке EXE-файла в память его заголовок заменяется на блок PSP (Program Segment Prefix). После него в блоке памяти располагаются элементы самой программы в порядке, указанном на рис.1.

Все типизированные константы располагаются в сегменте типизированных констант, размер которого определяется на стадии компиляции. Переменные, объявленные в разделе VAR вне процедур и функций (глобальные переменные), располагаются в сегменте глобальных переменных, размер которого также определяется на стадии компиляции. Типизированные константы и глобальные переменные являются самыми надежными элементами программы, но, к сожалению, часто нельзя обойтись без использования динамической памяти.

Устройство динамической памяти

Динамической память называется потому, что в разное время работы программы одна и та же область памяти может принадлежать разным переменным, которые по этой причине называются динамическими переменными. Именно это обстоятельство повышает вероятность ошибок при работе с такой памятью.

Рассмотрим механизм распределения динамической памяти. Когда вы хотите отвести память под динамическую переменную, вы вызываете процедуру `New` или `GetMem`. Первая отличается от второй лишь тем, что сама устанавливает размер выделяемого блока и вызывает указанный конструктор, если память отводится под объект. Первая динамическая переменная размещается в самом начале динамической памяти (там, куда указывал указатель `HeapPtr`). Значение указателя `HeapPtr` увеличивается на размер выделенного блока. Следующая переменная занимает место сразу за предыдущей. Если расстояние между последней переменной и вершиной динамической памяти (указатель `HeapEnd`) уменьшится настолько, что для следующей переменной не хватит места, то администратор памяти обратится за советом к функции, адрес которой хранится в указателе `HeapError`. Обращение к этой функции выглядит так:

`function HeapErrorFunc(Size: Word): Integer;`

Параметр `Size` показывает, сколько памяти нужно отвести под динамическую переменную. Если функция вернет значение, равное 0, то администратор памяти остановит выполнение программы и выдаст сообщение об ошибке номер 203 ("Heap overflow error..."). Если функция вернет значение, равное 1, то администратор не прервет выполнение программы, а присвоит указателю значение NIL. По этому признаку вызывающая процедура сможет определить, что требуемой памяти не хватило. Если же функция вернет значение, равное 2, то администратор памяти попытается выделить память еще раз.

Кроме этого, администратор памяти вызывает функцию `HeapError` с параметром `Size`, равным 0, каждый раз, когда выделение памяти прошло успешно. Как мы увидим позже,

Сергей Николаевич Кучеров - системный программист.
Область интересов -
объектно-ориентированное программирование.
Телефон: 236-21-89.

значение функции `HeapError` очень велико, так как, перехватив ее в своей программе, вы можете получить полный контроль за работой администратора динамической памяти.

Для того чтобы освободить часть памяти, можно опустить указатель `HeapPtr` ниже, воспользовавшись, например, процедурами `Mark` и `Release`. Первая запоминает текущее значение `HeapPtr` в переданном ей указателе, а вторая восстанавливает запомненное значение. Следует отметить, что такой опасный способ освобождения динамической памяти не следует использовать в программе, если вы хотите всегда быть уверенными в ее надежности. Для более надежного освобождения памяти существует другой способ.

Свободные блоки

Как показывает опыт, использование процедур `Mark` и `Release` не только опасно, но и неудобно. Особенно отчетливо примитивность этого метода проявляется с точки зрения объектно-ориентированного подхода. Каждый объект, который занимает динамическую память, должен "убрать за собой" самостоятельно. Для этих целей используют процедуры `Dispose` и `FreeMem`, освобождающие именно тот блок, который был занят. Использование этих процедур гораздо логичнее, но приводит к появлению пустых промежутков в динамической памяти.

Для наиболее эффективного заполнения динамической памяти администратор памяти компилятора Turbo Pascal 6.0 поступает следующим образом: каждый раз, когда процедура `New` или `GetMem` заказывает динамическую память, размер требуемого блока выравнивается в сторону большего значения, достигая кратности 8 байт. Другими словами, если вы закажете от 0 до 8 байт, администратор выделит 8 байт. Если вы закажете от 9 до 16 байт, администратор выделит 16 байт. И так далее. При этом преследуются две цели. Во-первых, любой свободный блок имеет размер не меньше 8 байт, что необходимо для размещения в нем специальной записи. Во-вторых, такая организация блоков памяти позволяет избежать чрезмерного дробления динамической памяти при частом отведении и освобождении небольших блоков памяти.

Как только занятый блок памяти освобождается, администратор памяти записывает в его начало данные, структура которых может быть представлена в виде следующей записи:

```
PFreeBlock = ^TFreeBlock;
TFreeBlock = record
  Next  : PFreeBlock;
  Tail  : word;
  Parag : word;
end;
```

Поле `Next` указывает на следующий пустой блок динамической памяти или содержит значение указателя `HeapPtr`, если этот блок последний. Поля `Parag` и `Tail` определяют размер свободного блока, который может быть вычислен по формуле $Size := Parag \cdot 16 + Tail$.

Адрес первого свободного блока хранится в переменной `FreeList`. Таким образом, указатель `FreeList` указывает на связанный список свободных блоков, последний из которых содержит указатель на вершину самого верхнего занятого блока. На последнее обстоятельство я хотел бы обратить особое внимание программистов. В начале верхней незанятой части динамической памяти не содержится записи `TFreeBlock`, и попытка, например, определить размер этой области, ссылаясь на структуру `TFreeBlock`, может ввести вас в заблуждение.

Как только процедура `New` или `GetMem` требует выделить область в динамической памяти, администратор памяти начинает просмотр свободных блоков, начиная с `FreeList` и

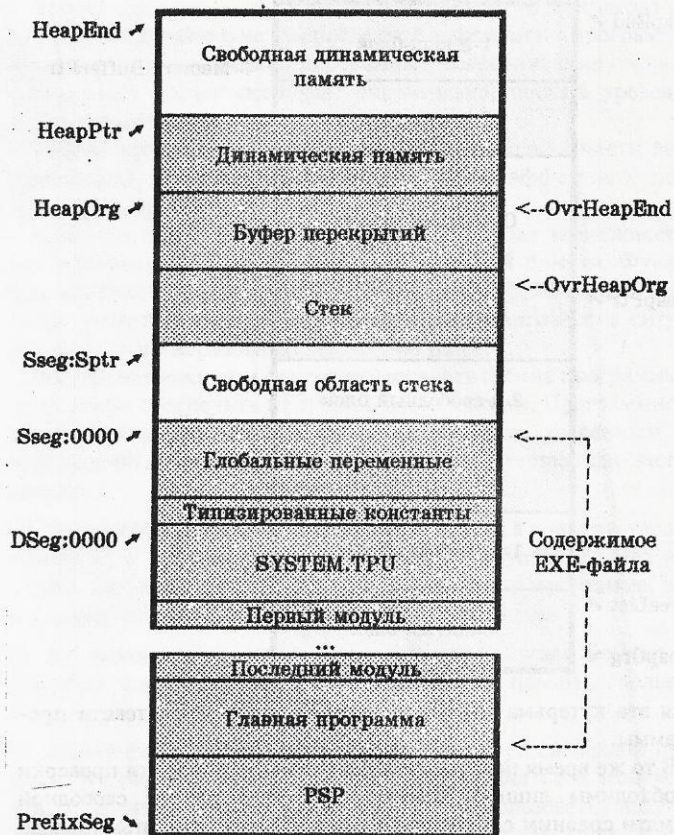


Рис. 1

кончая `HeapPtr`. В первом же подходящем по размеру блоке будет отведено место под динамическую переменную.

Если вы в своих программах активно используете динамическую память, то наверняка убедились, что администратор динамической памяти компилятора Turbo Pascal 6.0 прекрасно справляется с работой, и я не советую в нее вмешиваться без особой необходимости. Если же стандартные возможности компилятора вас не устраивают, попробуйте использовать пакет Turbo Vision.

Turbo Vision

Пакет Turbo Vision, поставляемый вместе с компилятором Turbo Pascal 6.0, является не просто удобной библиотекой для создания многооконного интерфейса. В этом пакете реализовано много новых идей, которые способны перевернуть укоренившиеся представления о методах программирования на языке Паскаль.

В данной статье нас будут интересовать новые подходы к управлению памятью и ее распределению. Разработчики пакета Turbo Vision значительно продвинулись вперед в направлении повышения безопасности программирования и эффективности использования динамической памяти.

Поставляемый в составе пакета Turbo Vision модуль `Memory` содержит ряд функций, дополняющих работу администратора памяти компилятора Turbo Pascal. Прежде чем перейти к рассмотрению этих функций, приведу небольшой пример. Представьте, что вы строите в памяти сложный объект, состоящий из множества подобъектов и десятки раз вызывающий процедуры `New` или `GetMem`. В соответствии с традиционными методами перед каждым вызовом процедур `New` и `GetMem` вам придется проверять суммарный размер свободной памяти и в случае ее нехватки останавливать работу конструктора, освобождать уже занятую память и выдавать сообщение об ошибке.

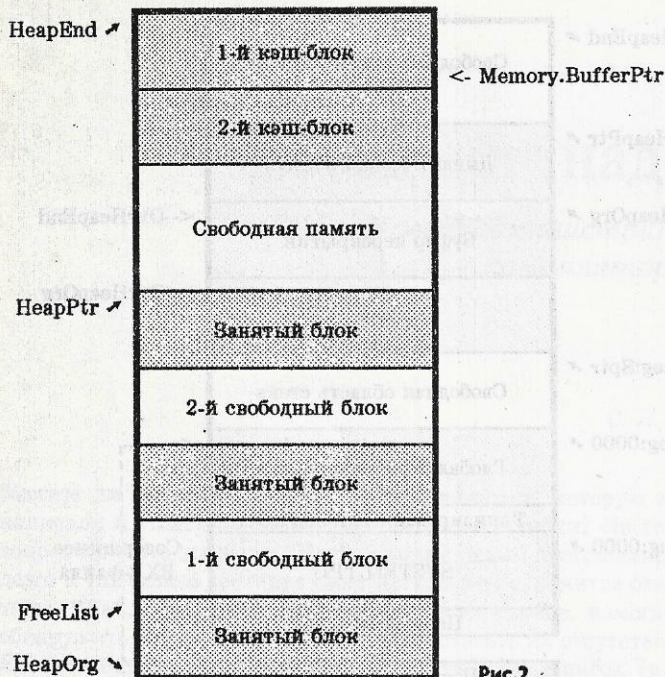


Рис.2

Вся эта кутерьма займет уйму времени и строк текста программы.

В то же время нетрудно предположить, что все эти проверки необходимы лишь в том случае, когда размер свободной памяти сравним с размером самого объекта. При этом гораздо удобнее следить за тем, чтобы размер свободной памяти не опускался до опасного предела. В этом случае число проверок можно сделать минимальным и ограничиться одной проверкой на каждый крупный объект. Для этого в модуле Memory имеется маленькая функция LowMemory, которая возвращает значение TRUE, если суммарный размер свободной памяти меньше, чем LowMemSize * 16 байт. Значение константы LowMemSize, описанной в модуле Memory, по умолчанию равно 256, что соответствует 4 Кбайт. Если максимальный размер объектов в вашей программе больше или меньше этого значения, то вы можете изменить значение константы LowMemSize.

В пакете Turbo Vision функция LowMemory применяется следующим образом: во время инициализации объекта проверка размера свободной памяти не производится, так как предполагается, что памяти хватит. После того как объект построен, вызывается функция LowMemory, которая проверяет, хватит ли памяти для следующего объекта. Если размер памяти меньше 4 Кбайт, то только что созданный объект убирается из памяти, для него вызывается деструктор и выдается сообщение о нехватке памяти. При этом главное отличие от традиционного способа контроля размера свободной памяти заключается в том, что процесс инициализации объекта не прерывается. Это особенно важно, если объект имеет сложную структуру.

Разумеется, данный метод годится только для объектов, занимающих в памяти ограниченное место. Если же вы создадите, например, большой динамический массив, то контроль наличия памяти все же придется проводить традиционными методами.

"Тайники" и "зазеркалье"

Модуль Memory реализует еще одну оригинальную идею. Мы знаем, что быстродействие программы и экономное использование памяти всегда противостоят друг другу. Так, в пакете Turbo Vision было необходимо обеспечить

максимально быстрый вывод на экран, что и удалось сделать с помощью буферизации. Каждый выводимый на экран объект может иметь буфер, т.е. область памяти, в которой он "строит" свое изображение. Таким образом, нет необходимости каждый раз "строить" изображение, а можно просто скопировать на экран содержимое буфера. При этом значительно возрастает скорость перерисовки содержимого всего экрана.

Все это так, но если каждый объект будет отводить в памяти место под свой буфер, то для всего экрана ее может потребоваться слишком много. Целесообразно пожертвовать быстродействием программы и обойтись без буферизованного вывода, но как это сделать? Разработчики пакета Turbo Vision предложили оригинальный механизм использования для таких целей верхней части динамической памяти. Применяя процедуру GetBufMem, вы можете создать "тайник" (cache) в верхней, не используемой до поры части динамической памяти. Поскольку администратор памяти модуля System не заметит этого, то результат, возвращаемый функциями MemAvail и MaxAvail, не изменится. Верхняя область памяти (я буду называть ее "зазеркальем") почти всегда свободна, хотя в любой момент может быть занята. Это означает, что размещенные в ней "тайники", которые я буду называть кэш-буферами, в любой момент могут быть обнаружены и уничтожены. Тем не менее, несмотря на свою "криминальную" сущность, кэш-буферы могут значительно повысить эффективность программы.

Для обращения к отведенному в "зазеркалье" кэш-блоку служат те же средства, что и для обращения к любой другой области памяти, с той разницей, что для обращения к кэш-блоку можно использовать только тот указатель, который был передан в процедуру GetBufMem. Причину этого вы поймете, когда мы будем рассматривать механизмы работы "зазеркалья".

А сейчас нас интересует, что же произойдет, если, продолжая занимать динамическую память и не зная о существовании кэш-блоков, администратор памяти доберется до ближайшего из них и ответит в нем память под обычную динамическую переменную. В этом случае администратор "зазеркалья" просто удалит мешающий кэш-блок и указателю, используемому для доступа к этому блоку, присвоит значение NIL.

Такой механизм позволяет для ускорения работы программ использовать свободную динамическую память в то время, пока эта память не используется по прямому назначению. В этом случае фрагмент программы может выглядеть примерно так:

```

If BufPtr<nil then begin
  { Использовать буфер BufPtr^ для повышения быстродействия }
end else begin
  { Буфер потерян, попробуем обойтись без него... }
end;

```

Естественно, что кэш-блок можно освободить и вручную, вызвав процедуру FreeBufMem.

Устройство "зазеркалья"

Теперь поподробнее рассмотрим механизм работы администратора "зазеркалья". Модуль Memory в разделе реализации содержит указатель на последний кэш-блок, называемый BufferPtr (он описан в разделе реализации, поэтому не пытайтесь обратиться к нему из своей программы). Когда вы вызываете процедуру InitMemory, этому указателю присваивается начальное значение, равное значению указателя

HeapEnd. Это означает, что в динамической памяти нет кэш-блоков. Процедура InitMemory присвоит указателю HeapError значение адреса своей внутренней функции HeapNotify.

После этого процедура GetBufMem может зарезервировать кэш-блок. Сначала она проверит, достаточно ли памяти между вершиной динамической памяти и вершиной самого верхнего занятого блока. Если памяти недостаточно, то указателю присваивается значение NIL.

Процедура GetBufMem отводит на 6 байт больше, чем требует параметр Size. Лишние 6 байт нужны ей, чтобы разместить информацию о размере блока и указателе на него:

```
PBuffer = ^TBuffer;
```

```
TBuffer = record
```

```
  Size : Word;
```

```
  Master : ^Pointer;
```

```
  Data : record end;
```

```
end;
```

Master - это указатель на указатель. Процедура GetBufMem помещает в него адрес указателя, который был передан ей в качестве параметра. После каждого перемещения блока его значение будет изменяться, а после удаления ему будет присвоено значение NIL.

Размер блока указывается в параграфах, другими словами, его нужно умножить на 16, чтобы получить размер в байтах. Сразу за указателем Master размещается блок данных. После этого процедура GetBufMem изменяет значение указателя BufferPtr: теперь он указывает на запись TBuffer кэш-блока (рис.2).

Следующие кэш-блоки будут размещаться подряд. Следует заметить, что, в отличие от "легальной" динамической памяти, "зазеркалье" не фрагментируется. За порядком в "зазеркалье" следит процедура FreeBufMem. Если вы решили освободить кэш-блок и вызвали для этого процедуру FreeBufMem, то после освобождения блока все низлежащие блоки вместе со своим содержимым будут перемещены выше на освободившееся место. Указатели на эти блоки будут соответственно изменены.

Что же произойдет, когда верхний "легальный" блок доберется до нижнего "тайника"? В этом случае нам поможет процедура HeapNotify, которая, как вы помните, заменяет стандартную процедуру HeapError.

Когда очередной вызов New или GetMem займет часть "зазеркальной" памяти, администратор памяти этого не заметит, но он вызовет процедуру HeapError (HeapNotify) с параметром 0. Процедура HeapNotify знает о существовании "зазеркалья". Она выяснит, какие кэш-блоки мешают "легальному" блоку, и вызовет процедуру FreeBufMem для освобождения части из них.

Таким образом будет достигнута цель: максимальное использование всей области динамической памяти.

Так как же написать надежную программу?

Компилятор Turbo Pascal 6.0 предоставляет очень широкие возможности для управления памятью. При этом можно выделить три уровня доступа к памяти.

Первый уровень - абсолютная адресация, в частности через псевдомассивы Mem, MemW. В качестве индекса используется абсолютный адрес памяти, например:

```
W := MemW[0040:0000];
```

что эквивалентно записи

```
W := word(Ptr($40,0)^);
```

Второй уровень - динамическая память. Администратор памяти самостоятельно распределяет выделенную программе память под динамические переменные. Выравнивание границ выделяемых блоков позволяет значительно снизить уровень фрагментации динамической памяти.

Третий уровень - кэш-буферизация в свободной части динамической памяти, позволяющая наиболее эффективно использовать временно незанятую память.

Администратор памяти предоставляет также возможность контролировать распределение динамической памяти. Функция контроля, адрес которой хранится в переменной HeapError, позволяет реагировать не только на критические ситуации, но и на нормальное выделение памяти.

Все эти возможности помогают создавать гибкие программы, но не могут обеспечить их полной безопасности. Программист обязан соблюдать этикет, чтобы всегда быть уверенным в надежности своих программ. Вот некоторые правила этого этикета.

□ Занимайте динамическую память только в конструкторах объектов, а освобождайте - только в деструкторах. Это не только сократит время поиска опасных мест в программе, но и сделает ее подлинно объектно-ориентированной.

□ Не используйте явных констант в процедурах GetMem и FreeMem, чтобы освобождать всегда столько памяти, сколько было занято.

□ Выберите одну стратегию контроля размера свободной памяти и придерживайтесь этой стратегии для всех модулей вашей программы. Если вы решили использовать модуль Memory из пакета Turbo Vision, то пользуйтесь функциями этого модуля и не перехватывайте указатель HeapError.

□ Не используйте процедуры Mark и Release в сложных программах. Эти процедуры предназначены для совместимости со старыми версиями компилятора и для специальных задач.

□ Помните, что, в отличие от языка C, Turbo Pascal предоставляет вам большое число встроенных типов, таких, как строки, множества, перечисления, диапазоны и многомерные массивы. Пакет Turbo Vision добавляет к этому списку многофункциональные динамические массивы. Использование всех этих возможностей значительно повысит надежность ваших программ.

Соблюдение этих несложных правил поможет вам не только быстро написать надежную программу, но и лучше понять объективные законы программирования.

Центр "ИНТЕРФЕЙС" предлагает уникальные пакеты программ для IBM PC/XT/AT

FORGRAF - новая графическая библиотека для Фортрана-77 (научная, деловая, интерактивная графика);
GRAPH - призер конкурса "Борланд-Контакт-91" (самая мощная графическая библиотека для Си, поставка в исходных текстах);
ISP - диалоговая система моделирования и цифровой обработки сигналов (поставка в исходных текстах на Си);
PROTECT - пакет программ защиты от несанкционированного копирования (поставка в исходных текстах на Си).

Стоимость пакетов сравнима с зарплатой инженера или научного работника. Частным лицам скидка до 50%.

Высокое качество и доступные цены - стиль нашей работы!

По запросу вышлем описание, условия поставки, демодискету.

142432, Черногоровка, Моск. обл., а/я 33, "Интерфейс",
В.Н. Гайдуллин

Быстродействующие адаптеры VGA и трехмерная анимация

М. Эйбраш

Не так давно я получил письмо, побудившее меня рассказать вам о высокопроизводительных методах трехмерной компьютерной анимации. Мой корреспондент просил обсудить в числе прочих проблему перемещения скрытых поверхностей объемных тел, осуществляемого в реальном времени и, желательно, в режиме X.

16-битовый адаптер VGA и монохромный монитор

Серьезному программисту, работающему в области компьютерной графики, необходима система из двух мониторов (с адаптером VGA и с монохромным адаптером), поскольку требуется отображать отладочную информацию, не нарушая работы графической программы. Конечно, можно использовать второй компьютер, но это дороже и слишком замедляет работу, поэтому почти все известные мне разработчики графических пакетов избрали двухмониторный вариант.

К сожалению, применение 8-битового монохромного адаптера переводит любой адаптер VGA в режим работы с 8-битовыми операциями (см. статью "Demistifying 16-bit VGA" в журнале "Dr. Dobbs' Journal", May 1990). В таком режиме при каждом обращении к видеопамети передается лишь 1 байт, как если бы у адаптера VGA был 8-битовый вход. Из-за этого по меньшей мере вдвое снижается скорость последовательных обращений к видеопамети VGA, например в случае использования операции REP STOSW для закрасивания прямоугольников. Кроме того, многие шины воспринимают обращения к видеопамети как обращения к памяти процессора, что переводит его в состояние ожидания, тем самым снижая скорость доступа до ничтожной скорости работы шины процессора - отсюда двукратное

замедление всех обращений к видеопамети VGA. Больше того, это двукратное снижение скорости мультиплицируется, и время доступа к видеопамети VGA может возрасти вчетверо. В результате подключения 8-битового монохромного адаптера высокопроизводительная система на основе процессора 80386 или 80486 с 16-битовым адаптером Super-VGA будет "ползти" не быстрее, чем при использовании 8-битового адаптера VGA, - практически со скоростью работы видеоадаптера, первоначально разработанного фирмой IBM.

Выход очевиден: приобрести 16-битовый монохромный адаптер. Но, увы, такого не существует. Производство монохромных адаптеров - захудалая отрасль, и никому не придет в голову разрабатывать их 16-битовую разновидность в угоду крошечной группке пользователей (а я с радостью заплатил бы за такой лишние 50 дол.).

Итак, аппаратного решения нет. Однако есть программное - утилита SETBUS (см. листинг 1), предложенная Ч. Марслеттом (Ch. Marslett). Программа SETBUS обеспечивает любому адаптеру VGA, созданному на основе микросхемы ET4000 фирмы Tseng Labs, даже при подключенном монохромном адаптере 16-битовый режим работы. После запуска этой программы скорость работы с видеопаметью VGA на моей двухмониторной системе возросла в 4,3 раза по сравнению с нормой. Правда, чтобы не впадать в эйфорию, следует помнить, что реальное ускорение работы конкретных VGA-ориентированных графических программ будет зависеть от частоты обращений к видеопамети, от использования двухбайтовых операндов и т.п., - в общем, скорость работы возрастет меньше, а подчас намного меньше, чем в четыре раза. Так, система Windows большую часть времени тратит не на обработку обращений к видеопамети, поэтому после запуска программы SETBUS 16 скорость ее ра-

боты, конечно, увеличивается, но даже не в два раза. Как бы то ни было, предложенная утилита немного ускоряет выполнение программ, работающих с адаптером VGA, а ее стоимость намного ниже стоимости высокоскоростных аппаратных средств.

Конечно, у программы SETBUS есть недостатки. После ее запуска монохромный монитор будет "сбиваться" при каждом обращении к его видеопамети, так как адаптер VGA будет вмешиваться в этот процесс. Я даже не гарантирую, что все у вас будет в целости и сохранности, хотя сам делал это неоднократно и без последствий. Короче - испытайте программу SETBUS на свой страх и риск, только не пытайтесь одновременно использовать оба адаптера. Тонкость состоит в написании командных файлов таким образом, чтобы адаптеры, VGA и монохромный, подключались поочередно, по мере необходимости. Например, в начало командного файла, с помощью которого вы запускаете систему Windows или другие VGA-ориентированные программы, можно поместить вызов программы SETBUS 16, а в конец этого файла - вызов программы SETBUS 8. Тогда графические программы будут работать на полной скорости, используя адаптер VGA, а монохромный адаптер сможет нормально выполнять свои функции в оставшееся время.

Да, программа SETBUS немного неудобна, но она оправдывает себя: какой смысл покупать 16-битовый адаптер VGA, если он всегда будет работать, как 8-битовый?

Исправление ошибок в пакете X-Sharp

Т. Моран (T. Moran) из Саратоги (шт. Калифорния) решил переписать на языке Ada пакет программ трехмерной анимации X-Sharp, над которым мы работали последние четыре месяца, и обнаружил три ошибки в алгоритме. Две из них безобидны, а третья существен-

на, так как "сидит" в коде, выполняющем объединение матриц, и проявляется лишь в том случае, когда этот процесс включает в себя наряду с вращением матриц их преобразование. Но уж в этом случае крах наступает очень быстро: во время перевода матриц из виртуального пространства в видимое. Сам я не заметил ошибки, так как производил лишь вращение матриц, без их перевода. Однако в последнем дистрибутиве пакета X-Sharp я исправил все обнаруженные ошибки.

Сообщайте BIOS о новых параметрах экрана

Б. Линдли (B. Lindley) из Месы (шт. Аризона) предложил при программировании адаптера VGA для работы в нестандартных режимах, например в режиме 640x400 (16 цветов), о котором я писал в журнале "Dr. Dobbs' Journal", December 1991, сообщать BIOS о новых параметрах экрана по двум причинам.

Во-первых, резидентные утилиты часто используют переменные BIOS: у того же Б. Линдли есть резидентная программа печати содержимого экрана EGAD Screen Print, которая определяет число линий сканирования для вывода сообщения на экран как произведение переменных BIOS "число строк текста" и "высота знака".

Во-вторых, сама BIOS может не справиться с выводом текста на экран без должной информации: размеры текстового поля могут превысить размеры экрана или будет использован не тот знакогенератор (разумеется, в абсолютно нестандартном режиме, например режиме X, BIOS вообще не сможет отображать текст). Б. Линдли предлагает сразу после перехода адаптера VGA в режим 640x400 (16 цветов) вызывать код, приведенный на листинге 2, для сообщения BIOS о работе с экраном, вмещающим 25 строк при высоте символов 16 пикселей. Я нахожу эту идею превосходной: вреда не принесет, зато, возможно, избавит от обращения к группе технической поддержки.

Bitblt-компилирование

Д. Стаффорд (D. Stafford), специалист из японского отделения фирмы Borland, поделился необычной, но плодотворной идеей bitblt-компилирования. Bitblt (*bit-Block Transfer*) - это операции объединения участков одной или нескольких битовых карт-источников (*source bitmaps*) и битовой карты-приемника (*destination bitmap*) с помощью логических операций И, ИЛИ и замещения. Операции bitblt являются основой программирования окон в графическом режиме с использованием битовых карт.

Как правило, обычная двухоперандная операция bitblt осуществляется последовательным считыванием каждого байта битовой карты-источника при его обработке в случае необходимости (например, для получения прозрачных участков или преобразования монохромного изображения в цветное - цветового расширения) и объединением этого байта с битовой картой-приемником. Обеспечение прозрачности или цветового расширения может отнять уйму времени - ведь даже операция обычного замещения требует считывания всех байтов битовой карты-источника.

Пусть нужно получить на экране монитора несколько прозрачных цветных изображений монохромной пиктограммы размером 8x8 пикселей. Это означает, что все ненулевые биты пиктограммы изображаются на экране пикселями одного и того же цвета (цвета изображения), а нулевые биты оставляют соответствующие им пиксели без изменения. Все это можно сделать с помощью программного кода (см. листинг 3), осуществляющего задание цвета, проверку прозрачности и рисование (рис.1). Предложенный код не будет работать с хорошей производительностью из-за повторения процедур цветового расширения и проверки прозрачности при каждом рисовании пиктограммы.

Другой путь - bitblt-компилирование, при котором процедуры цветового расширения и проверки прозрачности выполняются лишь один раз (см. листинг 4). Функция `CompileReplaceXor()` преобразует данные пиктограммы не в другие данные, а в код, осуществляющий операцию bitblt вместе с этими процедурами. Другими словами, эта функция компилирует данные пиктограммы в код, выполняющий процедуры цветового расширения и проверки прозрачности и, кроме того, все то же, что делает стандартная операция bitblt (рис.2). Функция `ExecuteCompiled()` выполняет скомпилированный код с тем же результатом, что и эквивалентная операция bitblt. При этом ей остается только отобразить на экране соответствующие участки битовой карты, поэтому функция `Execute-Compiled()` работает быстрее кода, приведенного на листинге 3, например при использовании быстродействующего 16-битового адаптера VGA (см. листинг 5) - в три раза быстрее. Код, представленный на листинге 4, можно оптимизировать разными способами, например написанием функции bitblt-компилирования на языке ассемблера, помещением скомпилированного кода в сегмент кода, во

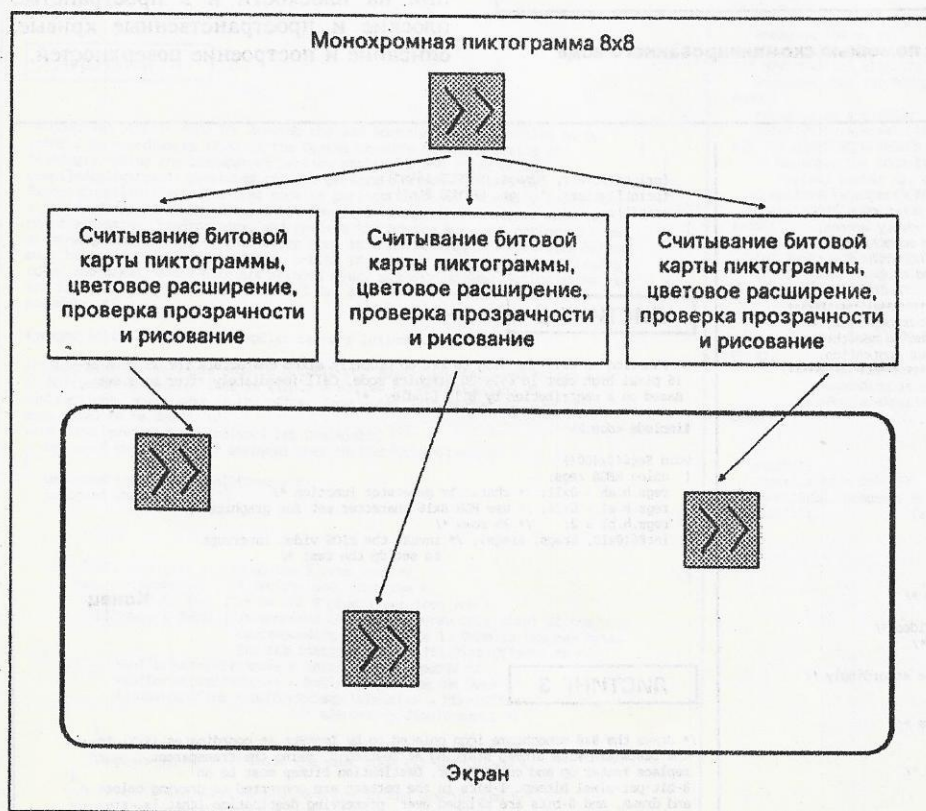


Рис.1. Троекратное изображение пиктограммы с помощью обычной операции bitblt

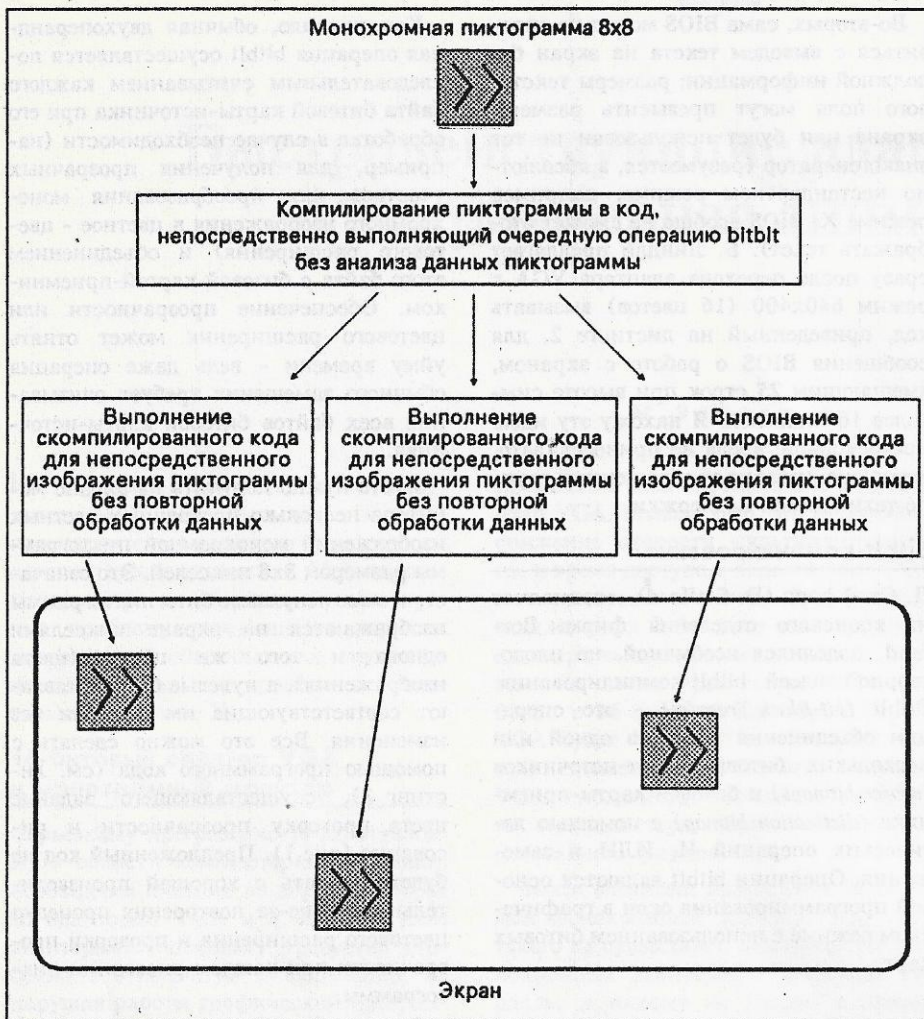


Рис.2. Троекратное изображение пиктограммы с помощью скомпилированного кода

избегание глобальных переходов управления, или использованием 16-битовых команд MOV там, где это возможно. Здесь же я просто показал возможности метода bitblt-компилирования, и трехкратное увеличение производительности при этом - достаточно веский аргумент в его пользу.

Как правило, выгоднее код преобразовывать в данные (например, в виде таблиц переходов и ссылок, функций состояния и т.п.). Однако, оказывается, бывают ситуации, когда выгоднее данные преобразовывать в код. Bitblt-компилирование тому пример. Идея странная, но результаты впечатляют.

Рекомендуемая литература

Тем, кто стремится всесторонне разобратся в математических основах компьютерной графики, я очень рекомендую книгу Д. Роджерса и Д. Адамса (Rodgers D.F., Adams J.A. Mathematical Elements for Computer Graphics, Second Edition. - McGraw-Hill, 1990, ISBN 0-07-053529-9), которая проще книги Фоли и ван Дама (Foley, van Dam. Computer Graphics). Она написана, как учебник (хотя и без упражнений для читателя). В ней на 500 страницах содержится поразительное количество информации в виде ясных объяснений и примеров. Первая глава посвящена аппаратному обеспечению, в следующих представлены: теория преобразований на плоскости и в пространстве, плоские и пространственные кривые, описание и построение поверхностей.

ЛИСТИНГ 1

```
/* Utility to force an MM4000-based VGA into 16- or 8-bit operation even if
a monochrome adapter is in the system. (Note that only 16-bit memory access,
not 16-bit I/O, is enabled; the I/O state is not altered.) The monochrome
adapter won't work properly while SETBUS 16 is in effect. Tested with
Portland C++ 3.0. Courtesy of Charles Marslett of STB. Commented and
reformatted by Michael Abrash. */
*****
* This utility isn't known to cause problems, but use it at your own risk, as
* the monochrome and VGA adapters will both respond to accesses to monochrome
* display memory while SETBUS 16 is in effect, resulting in bus contention.
*****
#include <dos.h>
#include <stdio.h>
#include <string.h>

void main(int argc, char *argv[])
{ int crtc, val;
  union REGS regset;

  if (inp(0x3CC) & 0x01) /* decide where to address the CRT */
    crtc = 0x3DM; /* Controller by reading the I/O */
  else /* Address Select bit of the */
    crtc = 0x3B4; /* Miscellaneous Output register */

  outportb(0x3BF, 0x03); /* key sequence to enable access to */
  outportb(crtc+4, 0xA0); /* MM4000-specific registers */
  outportb(crtc, 0x36); /* get the current setting of the Video */
  val = inp(crtc+1); /* System Configuration 1 register */

  /* Decide whether 16- or 8-bit access desired, and configure accordingly */
  if (argc == 2) {
    if (strcmp(argv[1], "16") == 0) {
      outportb(crtc+1, val | 0x40); /* 16-bit memory access */
      goto ModeSet;
    } else if (strcmp(argv[1], "8") == 0) {
      outportb(crtc+1, val & 0xBF); /* 8-bit memory access */
    }
  }

ModeSet:
  regset.x.ax = 0x0003;
  int86(0x10, &regset, &regset); /* do a text mode mode set */
  exit(0);
}
```

```
}
fprintf(stderr, "Usage: SETBUS 16\n");
fprintf(stderr, " or SETBUS 8\n");
exit(1);
}
```

Конец

ЛИСТИНГ 2

```
/* Function to tell the BIOS to set up properly sized characters for 25 rows of
16 pixel high text in 640x400 graphics mode. Call immediately after mode set.
Based on a contribution by Bill Lindley. */

#include <dos.h>

void Set640x400()
{ union REGS regs;
  regs.h.ah = 0x11; /* character generator function */
  regs.h.al = 0x24; /* use ROM 8x16 character set for graphics */
  regs.h.bl = 2; /* 25 rows */
  int86(0x10, &regs, &regs); /* invoke the BIOS video interrupt
to set up the text */
}
```

Конец

ЛИСТИНГ 3

```
/* Draws the 8x8 monochrome icon pointed to by IconPtr at coordinates (X,Y) in
the DestWidth-wide bitmap starting at DestSeg:0, using the transparent
replace raster op and color Color. Destination bitmap must be an
8-bit-per-pixel bitmap. 1-bits in the pattern are converted to drawing color
and drawn, and 0-bits are skipped over, preserving destination (that is, are
transparent). Tested with Borland C++ 3.0; when USE_C is 0, uses
BC++ dependent inline assembly. */
```



```

#define USE_C 0 /* set to 1 to compile C code, 0 to compile assembler */
#if USE_C
#pragma inline /* tell the compiler there's inline ASM code */
#else
#include <dos.h>
#endif

void DrawReplaceXpar(unsigned int X, unsigned int Y,
    unsigned int DestSeg, unsigned int DestWidth,
    unsigned int Color, unsigned char *IconPtr)
{
    if USE_C
    {
        unsigned char far *ScreenPtr, Temp;
        int i, j;
        /* Point to the first destination pixel */
        ScreenPtr = MK_FP(DestSeg, Y*DestWidth*X);
        for (i=0; i<8; i++) { /* do the 8 icon rows */
            Temp = *IconPtr++; /* get the next icon row */
            for (j=0; j<8; j++) { /* do the 8 pixels per icon row */
                if (Temp & 0x80) /* draw this pixel if the */
                {
                    *ScreenPtr = Color; /* corresponding icon bit is 1 */
                    ScreenPtr++; /* point to the next destination pixel */
                    Temp <<= 1; /* shift the next icon bit into place */
                }
                /* Point to the start of the next row */
                ScreenPtr += DestWidth - 8;
            }
        }
    }
    else
    {
        asm cld /* make LODSB increment SI */

        asm mov es, DestSeg /* point ES to the bitmap */
        asm mov ax, Y
        asm mov bx, DestWidth
        asm mul bx /* DestWidth*Y+X = offset of first */
        asm add ax, X /* dest pixel */
        asm mov di, ax /* point ES:DI to the first dest pixel */
        asm mov si, IconPtr /* point DS:SI to the first icon byte */
        asm mov dx, 8 /* we'll do 8 rows */
        asm mov ah, byte ptr Color /* color we'll draw with */
        asm sub bx, 8 /* offset from end of one dest row to start of next */

        RowLoop:
        asm lodsb /* get the next icon row */
        asm mov cx, 8 /* we'll do 8 pixels on each row */

        PixelLoop:
        asm shl al, 1 /* shift the next icon bit into CF */
        asm jnc NoDraw /* bit is 0, so don't draw */
        asm mov es: [di], ah /* bit is 1, so draw this pixel */

        NoDraw:
        asm inc di /* point to the next destination pixel */
        asm loop PixelLoop /* do the next pixel */
        asm add di, bx /* point to the start of the next dest row */
        asm dec dx /* count down rows */
        asm jnz RowLoop /* do the next row */
    }
}

```

Конец

ЛИСТИНГ 4

/* Compiled bitblit code for drawing the 8x8 monochrome icon pointed to by IconPtr at coordinates (X,Y) in the DestWidth-wide bitmap starting at DestSeg:0, using the transparent replace raster op and color Color. CompileReplaceXpar() generates code to perform desired bitblit, and ExecuteCompiled() executes that code to perform bitblit. Generally faster than a standard approach when drawing the same icon many times, because this way color expansion, transparency, and reading the source are only performed once, at expansion time, rather than every time an icon is drawn. Destination bitmap must be an 8-bit-per-pixel bitmap, 1-bits in pattern are converted to drawing color and drawn, and 0-bits are skipped over, preserving destination (that is, are transparent). Tested with Borland C++ 3.0; uses BC++ dependent inline assembly. */

#pragma inline /* tell the compiler there's inline ASM code */

/* Generates far-callable code to bitblit the specified icon, and stores code in BufferToCompileInto. Code is simply a series of MOV [DI+xxxx],AL instructions, where xxxx is the offset from upper left corner of icon of each pixel to be drawn. */

```

void CompileReplaceXpar(unsigned int DestWidth,
    unsigned char *IconPtr, unsigned char *BufferToCompileInto)
{
    unsigned int i, j, PixelOffset = 0;
    unsigned char Temp;

    for (i=0; i<8; i++) { /* do the 8 icon rows */
        Temp = *IconPtr++; /* get the next icon row */
        for (j=0; j<8; j++) { /* do the 8 pixels per icon row */
            if (Temp & 0x80) { /* generate the code to draw this pixel if the
                corresponding icon bit is 1. Code is the hex bytes
                for the instruction MOV [DI+PixelOffset],AL */
                *BufferToCompileInto++ = 0x88; /* MOV opcode */
                *BufferToCompileInto++ = 0x85; /* mod-reg-rm byte */
                *((unsigned int *)BufferToCompileInto)++ = PixelOffset;
                /* addressing displacement */
            }
            PixelOffset++; /* point to the next destination pixel */
            Temp <<= 1; /* shift the next icon bit into place */
        }
        /* Point to the start of the next row in the destination */
        PixelOffset += DestWidth - 8;
    }
    /* Put a RET at the end to return to the calling code, and done */
    *BufferToCompileInto = 0xCB; /* RETF instruction = 0xCB */
}

```

```

}
void ExecuteCompiled(unsigned int X, unsigned int Y,
    unsigned int DestSeg, unsigned int DestWidth, unsigned int Color,
    unsigned char far *BufferToExecute)
{
    asm push ds /* preserve the default data segment */
    asm mov ds, DestSeg /* point ES to the bitmap */
    asm mov ax, Y
    asm mul word ptr DestWidth /* DestWidth*Y+X = offset of */
    asm add ax, X /* first dest pixel */
    asm mov di, ax /* point ES:DI to the first dest pixel */
    asm mov al, Color /* color with which to draw */
    asm call dword ptr BufferToExecute
    /* perform a far call to execute the
    compiled bitblit code, and done */
    asm pop ds /* restore the default data segment */
}

```

Конец

ЛИСТИНГ 5

/* Sample program to tile screen with an 8x8 monochrome icon. Tested with Borland C++ 3.0. */

```

#include <dos.h>
#include <conio.h>

```

#define USE_COMPILED_BITBLITS 1 /* set to 1 and link to Listing 4 to use compiled bitblits, set to 0 and link to Listing 3 to use conventional bitblits */

```

#define SCREEN_WIDTH 320
#define SCREEN_HEIGHT 200
#define SCREEN_SEGMENT 0xa000

```

```

#if USE_COMPILED_BITBLITS
extern void CompileReplaceXpar(unsigned int, unsigned char *, unsigned char *);
extern void ExecuteCompiled(unsigned int, unsigned int, unsigned int,
    unsigned int, unsigned int, unsigned char far *);
#else
extern void DrawReplaceXpar(unsigned int, unsigned int,
    unsigned int, unsigned int, unsigned int, unsigned char *);
#endif

```

```

static unsigned char TestIcon[8] =
    {0x88, 0x44, 0x22, 0x11, 0x11, 0x22, 0x44, 0x88};
#if USE_COMPILED_BITBLITS
/* Storage for the compiled icon-drawing code; must be large enough for the
largest possible code size, because no error checking is performed */
static unsigned char CompiledBuffer[1000];
#endif

```

```

void main()
{
    unsigned int x, y;
    union REGS regset;
    #if USE_COMPILED_BITBLITS
    unsigned char far *CompiledBufferPtr;
    #endif
    regset.x.ax = 0x0013; /* AL = 0x13 selects 320x200 */
    int86(0x10, &regset, &regset); /* 256-color graphics mode */
    #if USE_COMPILED_BITBLITS
    /* Generate the code for drawing the icon with the transparent
    replace raster op, and store the code in CompiledBuffer */
    CompileReplaceXpar(SCREEN_WIDTH, TestIcon, CompiledBuffer);
    CompiledBufferPtr = MK_FP(DS, CompiledBuffer);
    #endif
    /* Tile TestIcon over the entire screen */
    for (y=0; y<SCREEN_HEIGHT; y++) {
        for (x=0; x<SCREEN_WIDTH; x++) {
            #if USE_COMPILED_BITBLITS
            /* Draw the icon by executing the code in CompiledBuffer */
            ExecuteCompiled(x, y, SCREEN_SEGMENT, SCREEN_WIDTH, 14,
                CompiledBufferPtr);
            #else
            /* Draw the icon with the transparent replace raster op, color
            expanding it and handling transparency on the fly */
            DrawReplaceXpar(x, y, SCREEN_SEGMENT, SCREEN_WIDTH, 14, TestIcon);
            #endif
        }
    }
    getch(); /* wait for a key press */
    regset.x.ax = 0x0003; /* AL = 3 selects 80x25 text mode */
    int86(0x10, &regset, &regset); /* return to text mode */
    exit(1); /* done */
}

```

Конец

Управление цветом в адаптере VGA

Н. Г. Краснокутский

Как известно, адаптер EGA фирмы IBM позволяет выводить одновременно 16 из 64 возможных цветов. Раньше это вполне устраивало многих пользователей. Однако с расширением графических возможностей программных систем требования к качеству цветного изображения возросли, что, в частности, и обусловило появление на компьютерном рынке адаптера VGA, разработанного той же фирмой. Помимо всего прочего конструкторы предусмотрели в нем возможность одновременного вывода 16 или 256 цветов из палитры (*palette*) с числом цветов 262144. Разумеется, управление цветом при этом усложнилось.

16-цветные режимы

Читатель, наверное, уже знаком с 16-регистрационной палитрой (в дальнейшем будем называть ее стандартной) адаптеров EGA и VGA. Напомню, как при этом формируется цвет.

Стандартной палитрой управляет контроллер атрибутов (*Attribute Controller - AC*), используя регистры 0 - 15, каждый из которых может определять один из 64 возможных цветов. В графических режимах регистр стандартной палитры адресуется значением пикселя, а в текстовых режимах - значениями цветов фона и символа, размещенными в байте-атрибуте. В формировании номера регистра стандартной палитры участвует регистр маскирования цвета (*Color Plane Enable Register - CPER*), входящий в состав AC и предназначенный для маскирования отдельных битов значения пикселя или значений цветов фона и символа.

На рис.1 представлена структура регистра стандартной палитры адаптера EGA. Каждая из трех составляющих искомого цвета может иметь до четырех оттенков, следовательно, можно задать 64 различных цвета, одновременно работая только с 16 из них.

В адаптере VGA число цветов, которые может задавать регистр стандартной палитры, увеличено с 64 до 262144. Такое усовершенствование реализовано с помощью специального цифро-аналогового преобразователя, который мы будем называть RGB-контроллером. В состав этого контроллера входит еще одна палитра (назовем ее RGB-палитрой), включающая в себя 256 регистров, каждый из которых может задать один из 262144 возможных цветов.

Регистр стандартной палитры адресуется так же, как и в адаптере EGA, но теперь он не определяет непосредственно цвет, а указывает на один из регистров RGB-палитры, который и задает искомый цвет. Естественно, у читателя может воз-

никнуть следующий вопрос: "Если в регистре стандартной палитры для указания номера регистра RGB-палитры используются только 6 младших битов, то как адресовать регистры с номерами, большими 63?" Ответ прост - в каждый момент времени в формировании цветов изображения участвует не вся RGB-палитра, а лишь некоторая ее часть, называемая активной страницей (*active page*).

Активная страница и число используемых страниц RGB-палитры (оно может быть равно 4, если в одной странице 64 регистра, или 16, если в одной странице 16 регистров) устанавливаются программно. В первом варианте разбивки RGB-палитры на страницы для адресации ее регистров используются 6 младших битов регистра стандартной палитры, во втором - 4 младших бита.

На рис.2 представлена схема управления цветом в 16-цветных режимах для адаптера VGA.

Блоки f_1 , f_2 и f_3 реализованы логическим элементом AND. Блок f_4 формирует цвет пикселя в зависимости от значения бита 3 MCR, задающего мерцание. По умолчанию для графических режимов этот бит устанавливается в 0 (мерцание выключено), а для текстовых - в 1 (мерцание включено). Вы можете сами устанавливать бит 3 MCR, используя для этого прерывание 10h со следующими значениями входных регистров микропроцессора:

AX - 1003h;

BL - <значение бита 3 MCR>.

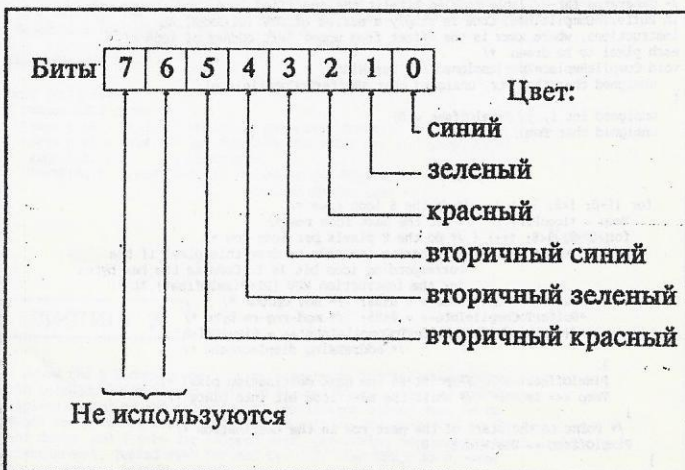


Рис.1. Регистр стандартной палитры адаптера EGA

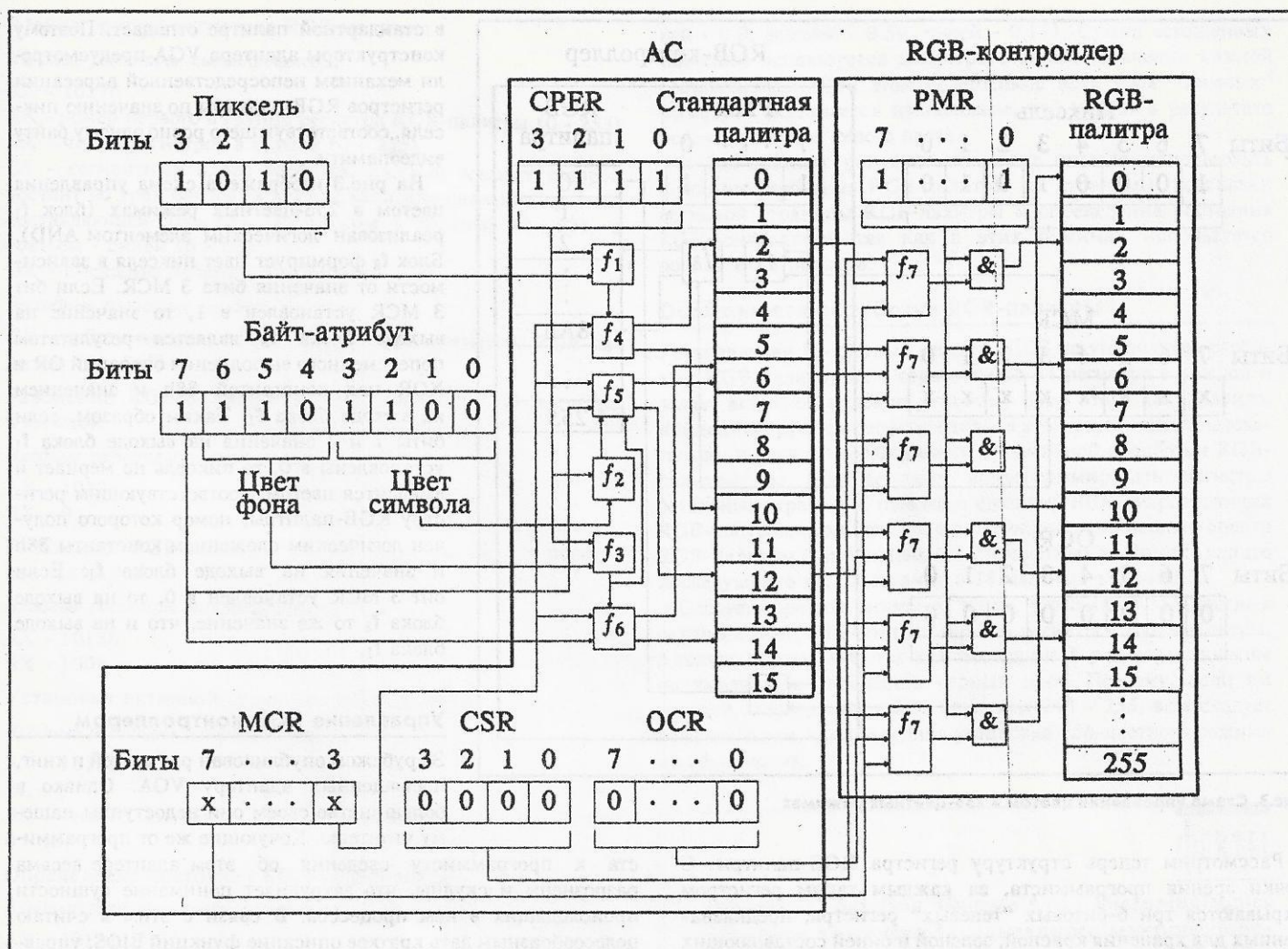


Рис.2. Схема управления цветом в 16-цветных режимах: OCR (Overscan Color Register) - регистр цвета бордюра; CSR (Color Select Register) - регистр выбора активной страницы RGB-палитры; MCR (Mode Control Register) - регистр управления режимом AC; PMR (Pixel Mask Register) - регистр маскирования значения пикселя в 256-цветных режимах; x - неопределенное значение

Если бит 3 MCR установлен в 1, то блок f_4 формирует выходные значения, попеременно выполняя логические операции OR и XOR над константой 8 и значением на выходе блока f_1 . Очевидно, что если бит 3 этого значения установлен в 0, то пиксель не мерцает и выводится цветом, соответствующим регистру стандартной палитры, номер которого получен логическим сложением константы 8 и значением на выходе блока f_1 , т.е. цвет немерцающего пикселя выбирается из второй половины стандартной палитры. Если бит 3 MCR установлен в 0, то на выходе блока f_4 то же значение, что и на выходе блока f_1 .

Блок f_5 формирует цвет символа в зависимости от значения бита 3 MCR и бита 7 байта-атрибута. Если бит 3 MCR и бит 7 байта-атрибута установлены в 1, то на выходе блока f_5 попеременно формируются два значения, первое из которых равно значению на выходе блока f_2 , а второе - значению на выходе блока f_3 . Если бит 3 MCR или бит 7 байта-атрибута установлен в 0, то на выходе блока f_5 формируется только одно значение, равное значению на выходе блока f_2 .

Блок f_6 формирует цвет фона в зависимости от значения бита 3 MCR. Если бит 3 MCR установлен в 1, то на выходе блока f_6 формируется значение, равное значению на выходе блока f_3 . Если бит 3 MCR установлен в 0, то на выходе блока f_6 формируется значение, бит 3 которого равен биту 7 байта-атрибута, а в битах 2 - 0 записано значение, появляющееся

на выходе блока f_3 , т.е. диапазон возможных цветов фона увеличивается в этом случае вдвое.

Блок f_7 функционирует следующим образом. Если бит 7 MCR установлен в 0 (RGB-палитра разбита на 4 страницы), то биты 5 - 0 значения на выходе этого блока формируются из битов 5 - 0 регистра стандартной палитры или OCR, а биты 7 - 6 - из битов 3 - 2 CSR (номер активной страницы RGB-палитры). Если бит 7 MCR установлен в 1 (RGB-палитра разбита на 16 страниц), то биты 3 - 0 значения на выходе блока формируются из битов 3-0 регистра стандартной палитры или OCR, а биты 7 - 4 - из битов 3 - 0 CSR (номер активной страницы RGB-палитры).

Странная "судьба" у PMR! Первоначально он задумывался как аналог CPER для RGB-палитры, т.е. с помощью PMR предполагалось маскировать выходные биты блока f_7 . Однако фирма IBM категорически запрещает программистам обращаться к этому регистру. Тем не менее я попытался проверить на адаптере VGA фирмы Paradise undocumented функции BIOS, управляющие PMR. Ничего страшного не произошло: дисплей и адаптер остались целы, функции оказались работоспособными, и я наконец-то узнал, маскирует ли PMR биты OCR в 256-цветном режиме, что интересовало меня больше всего (в доступных мне источниках я не смог найти ответа на этот вопрос).

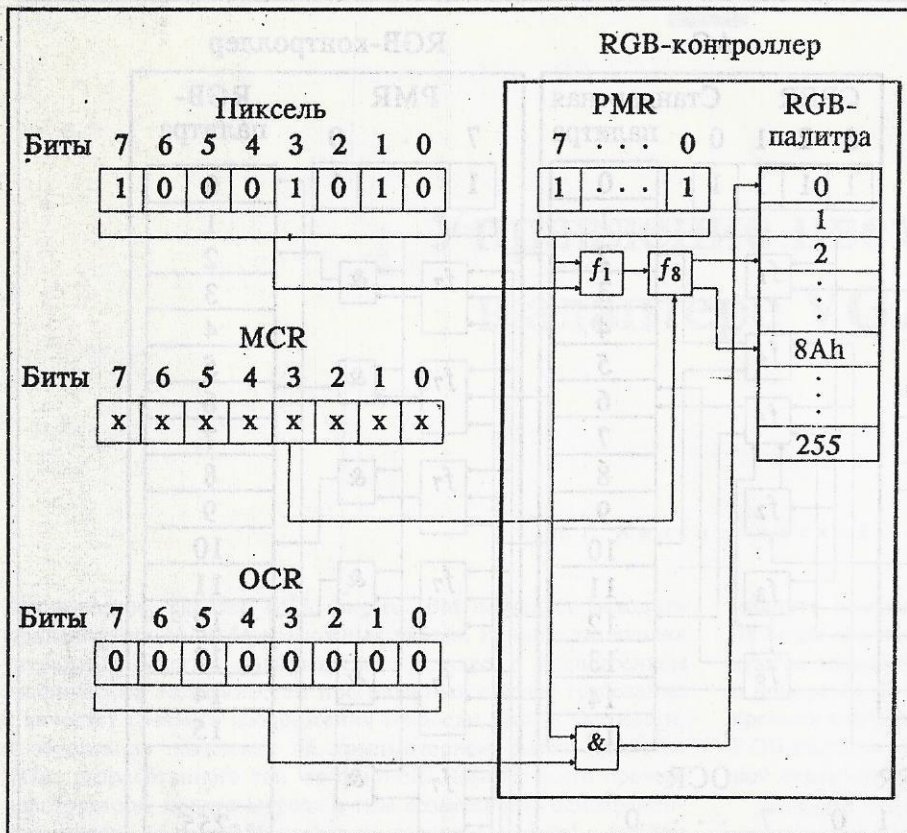


Рис.3. Схема управления цветом в 256-цветных режимах

Рассмотрим теперь структуру регистра RGB-палитры. С точки зрения программиста, за каждым таким регистром скрываются три 6-битовых "теневых" регистра, предназначенных для хранения красной, зеленой и синей составляющих цвета. Следовательно, число цветов, задаваемых одним регистром RGB-палитры, равно 262144. В сущности, "теневые" регистры представляют собой списковую структуру с дисциплиной обслуживания FIFO (First In - First Out), в соответствии с которой первой читается или записывается красная составляющая, второй - зеленая, третьей - синяя.

Из рис.2 видно, что бит 3 байта-атрибута участвует в формировании номера регистра стандартной палитры наравне с другими битами цвета символа. Поэтому если вы работаете одновременно с двумя текстовыми шрифтами, для выбора которых используется этот бит, то при единичном его значении соответствующий символ будет выводиться одним из цветов, определенных регистрами 8 - 15 стандартной палитры. Для использования в этом случае регистров 0 - 7 стандартной палитры вы можете либо переслать значения регистров 0 - 7 в регистры 8 - 15, либо обнулить бит 3 CPER, что не исключает использования бита 3 байта-атрибута для выбора одного из двух текстовых шрифтов.

Заметим, что регистр OCR адресует не регистр стандартной палитры, а непосредственно регистр активной страницы RGB-палитры. Эту особенность адаптера VGA следует иметь в виду при задании цвета бордюра.

256-цветные режимы

Очевидно, что 256-цветными могут быть только графические режимы, так как того числа битов, которое выделено в текстовых режимах для цветов фона и символа, недостаточно для адресации регистра RGB-палитры с номером, большим 15. Нетрудно понять и то, что в 256-цветных режимах надобность

в стандартной палитре отпадает. Поэтому конструкторы адаптера VGA предусмотрели механизм непосредственной адресации регистров RGB-палитры по значению пикселя, соответствующего ровно одному байту видеопамати.

На рис.3 изображена схема управления цветом в 256-цветных режимах (блок f_1 реализован логическим элементом AND). Блок f_8 формирует цвет пикселя в зависимости от значения бита 3 MCR. Если бит 3 MCR установлен в 1, то значение на выходе блока f_8 является результатом попеременного выполнения операций OR и XOR над константой 88h и значением на выходе блока f_1 . Таким образом, если биты 7 и 3 значения на выходе блока f_1 установлены в 0, то пиксель не мерцает и выводится цветом, соответствующим регистру RGB-палитры, номер которого получен логическим сложением константы 88h и значения на выходе блока f_1 . Если бит 3 MCR установлен в 0, то на выходе блока f_8 то же значение, что и на выходе блока f_1 .

Управление RGB-контроллером

За рубежом опубликован ряд статей и книг, посвященных адаптеру VGA. Однако в большинстве своем они недоступны нашему читателю. Кочующие же от программиста к программисту сведения об этом адаптере весьма разрознены и скудны, что затрудняет понимание сущности происходящих в нем процессов. В связи с этим я считаю целесообразным дать краткое описание функций BIOS, управляющих RGB-контроллером.

Запись в регистр RGB-палитры

Входные регистры микропроцессора

- AX - 1010h (номера функции и подфункции)
- BX - номер регистра RGB-палитры (0 - 255)
- DH - значение красной составляющей (0 - 63)
- CH - значение зеленой составляющей (0 - 63)
- CL - значение синей составляющей (0 - 63)

Запись в группу регистров RGB-палитры

Входные регистры микропроцессора

- AX - 1012h
- BX - номер первого в группе регистра RGB-палитры (0 - 255)
- CX - число регистров в группе (0 - 255)
- ES - сегментный адрес таблицы, каждый элемент которой состоит из трех байтов: в первом задается значение красной составляющей (0 - 63), во втором - зеленой (0 - 63), в третьем - синей (0 - 63)
- DX - смещение таблицы в сегменте

Чтение регистра RGB-палитры

Входные регистры микропроцессора

- AX - 1015h
 - BL - номер регистра RGB-палитры (0 - 255)
- Выходные регистры микропроцессора
- DH - значение красной составляющей (0 - 63)
 - CH - значение зеленой составляющей (0 - 63)
 - CL - значение синей составляющей (0 - 63)

Чтение группы регистров RGB-палитры

Входные регистры микропроцессора

AX - 1017h

BX - номер первого в группе регистра RGB-палитры (0 - 255)

CX - число регистров в группе (0 - 255)

ES - сегментный адрес таблицы, каждый элемент которой состоит из трех байтов: в первом записано значение красной составляющей (0 - 63), во втором - зеленой (0 - 63), в третьем - синей (0 - 63)

DX - смещение таблицы в сегменте

Выходные регистры микропроцессора

CX - действительное число прочитанных регистров RGB-палитры (0 - 255)

Разбиение RGB-палитры на 4 страницы по 64 регистра в каждой

Входные регистры микропроцессора

AX - 1013h

BX - 0

Разбиение RGB-палитры на 16 страниц по 16 регистров в каждой

Входные регистры микропроцессора

AX - 1013h

BX - 100h

Установка активной страницы RGB-палитры

Входные регистры микропроцессора

AX - 1013h

BL - 1

BH - номер активной страницы RGB-палитры (0 - 3 или 0 - 15)

Определение состояния RGB-контроллера

Входные регистры микропроцессора

AX - 101Ah

Выходные регистры микропроцессора

BL = 0 - в RGB-палитре 4 страницы по 64 регистра в каждой

BL = 1 - в RGB-палитре 16 страниц по 16 регистров в каждой

BH - номер активной страницы RGB-палитры (0 - 3 или 0 - 15)

Запись в PMR

Входные регистры микропроцессора

AX - 1018h

BL - значение регистра

Чтение PMR

Входные регистры микропроцессора

AX - 1019h

Выходные регистры микропроцессора

BL - значение PMR

Преобразование в серый цвет (*gray scale summing*)

Входные регистры микропроцессора

AX - 101Bh

BX - номер регистра RGB-палитры, с которого начинается преобразование (0 - 255)

CX - число регистров, подлежащих преобразованию (0 - 255)

С помощью последней функции выбранные цвета преобразуются в оттенки серого цвета. Такое преобразование, используемое, как правило, для монохромных дисплеев, выполняется по алгоритму, в соответствии с которым каждой составляющей цвета присваивается определенный вес (крас-

ной - 0,3, зеленой - 0,59, синей - 0,11). Сумма взвешенных значений используется для переустановки значения каждой составляющей. При этом содержимое всех трех "теневых" регистров оказывается одинаковым, что и дает в результате искомый оттенок серого цвета.

В 256-цветных режимах не имеет смысла использовать функции разбиения RGB-палитры на страницы, установки активной страницы RGB-палитры и определения состояния RGB-контроллера, так как в этих режимах используется полная RGB-палитра.

Особенности обработки RGB-палитры

Устанавливая 16-цветный режим, BIOS по умолчанию разбивает RGB-палитру на 4 страницы по 64 регистра в каждой и делает активной нулевую страницу. При этом, как правило, инициализируются регистры только этой страницы. Следовательно, в случае необходимости страничной обработки RGB-палитры вы должны сами запрограммировать регистры остальных страниц. С помощью функций BIOS, управляющих RGB-контроллером, довольно легко определить полный спектр RGB-палитры (применяемый в 256-цветных режимах) для его последующего использования в 16-цветных режимах.

Следует обратить внимание на то, что BIOS в 256-цветных режимах программирует не все 256 регистров RGB-палитры, а только первые 248, так как последние 8 регистров должны по умолчанию определять черный цвет. Поэтому, если вы решили запрограммировать регистры 248 - 255, вам следует помнить о том, что очередная установка 256-цветного режима их не изменит.

факты

□ Лондон, 12 октября. Фирма AT&T Microelectronics объявила о выпуске 32-разрядного микропроцессора Hobbit ATT92010 (тактовая частота 20 МГц, производительность 13,5 млн оп./с, потребляемая мощность 0,25 Вт). Его архитектура позволяет осуществлять обмен данными между ячейками памяти. Входящие в состав процессора независимые блоки для предварительной выборки, расшифровки и выполнения команд дают возможность свести практически к нулю время выполнения команд перехода.

□ Санта Клара (шт. Калифорния) 12 октября. Третий квартал 1992 г. стал самым успешным для фирмы Intel за всю ее историю. Доходы фирмы составили 1,43 млрд. дол., чистая прибыль - 241 млн. дол. Наиболее сильны позиции фирмы Intel в США и Западной Европе, но укрепляются они и в Японии. Успех вызван ростом продаж персональных компьютеров, созданных на базе процессоров Intel486.

□ На проходившей в ноябре 1992 г. в Лас-Вегасе выставке COMDEX'92 фирмы Intel и Xing Technology показали обработку видеофайлов в формате MPEG на процессоре i750. Для демонстрации использовались платы ActionMedia II, разработанные совместно фирмами Intel и IBM для воспроизведения оцифрованной видеoinформации. Эти платы получили звание лучшего экспоната осенней выставки COMDEX'91.

Контрольные суммы Флетчера

Д. Кодис

В настоящее время компьютеры находят широкое применение в различных системах передачи данных, например в системах электронной почты, факсимильной и телеконференц-связи, так что пересылка файлов на большие расстояния стала обычным делом. Борьба с помехами и ограничением полосы частот всегда была и останется важной проблемой при организации компьютерной связи, даже после замены устаревших линий телефонной связи на современные системы, например на цифровые сети с интеграцией служб (ISDN).

Самый распространенный метод борьбы с помехами в каналах передачи данных заключается в разбиении данных на отдельные сообщения и посылке с каждым из них информации (в виде нескольких дополнительных байтов) о соответствующей контрольной сумме, которая затем используется адресатом для проверки достоверности полученных данных. За прошедшие годы было создано множество методов подобной проверки.

В самых ранних методах проверка осуществлялась передачей контрольных битов четности. К каждому байту сообщения добавлялся бит, чтобы этот байт содержал четное число ненулевых битов. Нечетное число ненулевых битов в полученном байте указывало на ошибку в сообщении. Такая проверка не обеспечивала надежного выявления ошибок: например, байт, переданный с четным числом ошибок в битах, легко проходил эту простенькую проверку.

В другом методе, известном как "горизонтальная проверка на четность" (*horizontal parity*), несколько байтов данных группируются в блок. К концу его добавляется байт четности, значение которого должно быть таким, чтобы в каждом двоичном разряде блока передаваемых данных было четное число ненулевых битов. Значение байта четно-

сти вычисляется при выполнении простой логической операции "исключающее ИЛИ" над всеми байтами передаваемого блока данных. Однако и в этом случае пара ошибок в битах не будет обнаружена, если они возникнут в одном и том же двоичном разряде блока.

В настоящее время более надежным методом обнаружения ошибок в системах передачи данных является проверка с помощью циклического избыточного кода (*Cyclic Redundancy Check - CRC*). В большинстве алгоритмов CRC для каждого блока данных формируется 16-битовое контрольное слово. Оно определяется как остаток от деления большого числа, образуемого всеми битами передаваемого блока данных, на известное 17-битовое число-делитель. Полученное контрольное слово обеспечивает обнаружение всех ошибок в отдельных битах и пакетов ошибок длиной 16 бит и меньше, а также всех парных ошибок в битах, разделенных цепочкой данных длиной до 65536 бит (8192 байт). Это прекрасные показатели для метода, основанного на использовании столь малого количества дополнительных данных.

Реализация алгоритма CRC осуществляется последовательным выполнением операций сдвига, маскирования и "исключающее ИЛИ" над каждым битом передаваемых блоков данных. К сожалению, архитектура большинства компьютеров не позволяет применять команды, необходимые для эффективного осуществления подобных операций, поэтому программная реализация алгоритма CRC очень громоздка. Даже в хорошо написанной на языке ассемблера CRC-программе на каждый передаваемый бит приходится не меньше шести машинных команд, а на каждый передаваемый байт - свыше 50. При больших объемах передаваемых данных недостаточно быстродействующие процессоры могут просто не справиться с такой работой.

Метод Флетчера

В конце 70-х годов был разработан метод обнаружения ошибок, по эффективности почти не уступающий алгоритму CRC, но требующий значительно меньших затрат вычислительных мощностей. Этот метод, основанный на применении контрольных сумм, предложен Д. Флетчером (J. Fletcher) из Национальной лаборатории им. Лоуренса (Ливермор) Калифорнийского университета.

Алгоритм проверки с помощью этого метода, а также результаты анализа его быстродействия и сравнения с методом CRC были опубликованы Д. Флетчером в январском номере журнала "IEEE Transactions on Communications" за 1982 г. Вариант метода Флетчера, известный как "метод поразрядного дополнения", был применен в сетевых протоколах Международной организации по стандартизации (ISO). И все же метод Флетчера распространен менее других.

Как и CRC, метод Флетчера позволяет распознать в битах все одиночные ошибки, оставшиеся незамеченными очень малую их долю: 0,001538% (CRC - 0,001526%). Оба метода дают также возможность обнаруживать парные ошибки в битах, если такие биты расположены достаточно близко друг к другу: 2040 бит для метода Флетчера и 65535 бит для CRC. На практике это означает, что при проверке с помощью CRC будут найдены все парные ошибки в блоке данных длиной до 8191 байт, тогда как метод Флетчера гарантирует обнаружение таких ошибок в блоках данных не длиннее 255 байт.

Д. Кодис (John Kodis) - разработчик систем управления производственными процессами для сталелитейной промышленности, распределения базы данных для дальней телефонной связи, телеметрических систем для NASA и электронной навигационной системы для морских судов.

Сущность алгоритма

Подсчет контрольной суммы по алгоритму Флетчера осуществляется с помощью двух 8-битовых чисел. Вначале они равны 0. Затем к первому числу по мере обработки передаваемых байтов прибавляется их значение. Полученное число делится на 255, а остаток от деления сохраняется в качестве нового значения первой контрольной суммы (sum1). Вторая контрольная сумма (sum2) определяется почти так же. Вначале равная 0, она обновляется с каждым обработанным байтом, что достигается прибавлением текущего значения sum1. Полученная сумма тоже делится на 255, и остаток запоминается как новое значение sum2. В примере 1,а приведен псевдокод, реализующий такой алгоритм.

Передать обе контрольные суммы для проверки адресатом можно их простым добавлением в конец передаваемого блока данных. Тем самым будет обеспечено надежное обнаружение ошибок, однако при этом не реализуются многие из достоинств метода Флетчера. Предпочтительнее вместо контрольных сумм передавать два их новых значения, позволяющих получить на другом конце нулевую контрольную сумму для всего передаваемого блока данных. Первое значение определяется вычитанием из 255 остатка от деления на 255 результата сложения двух байтов, соответствующих контрольным суммам сообщения, второе значение - вычитанием из 255 остатка от деления на 255 суммы байта, соответствующего текущему значению первой контрольной суммы, и его предыдущего значения. Псевдокод этих вычислений приведен в примере 1,б.

Пример из реальной жизни

На практике я столкнулся с методом контрольных сумм Флетчера в 1989 го-

ду, когда занимался оптимизацией экспериментальной волоконно-оптической локальной сети с несколькими узлами обработки данных. Производительность этой сети не оправдала ожиданий пользователей.

Изучив ситуацию, я установил, что при загрузке сети на полную мощность центральный процессор тратит больше половины своего времени работы на выполнение функции (на языке C), вычисляющей первую контрольную сумму Флетчера для блоков данных длиной 2040 байт (пример 2,а). После компиляции на язык ассемблера для процессора Motorola 680x0 был получен псевдокод, приведенный в примере 2,б.

Это достаточно эффективный псевдокод: все переменные содержатся в регистрах, нет лишних операций загрузки и сохранения, так что простых путей оптимизации ее работы, казалось бы, не видно. Однако на выполнение 13 команд внутреннего цикла требуется в среднем 84 машинных цикла, с учетом того, что каждая из двух операций вычитания производилась в половине всех проходов. При использовании процессора 68010 (10 МГц) такие вычисления занимали 8,4 мкс на один байт, или примерно 17 мс при обработке блока из 2040 байт данных, передаваемых по сети.

Направивается такой путь оптимизации данного алгоритма: исключить последний оператор if и заменить его на операцию вычисления остатка, которая выполняется при выходе из цикла. (Это вполне возможно благодаря линейности оператора вычисления остатка: остаток от деления на 255 суммы чисел равен сумме остатков от деления на 255 тех же чисел.) Однако при этом сумма sum2 должна быть представлена как длинное целое (32-битовое) число. Переполнение в таком случае невозможно при длине передаваемого блока данных (len)

меньше $2^{32}/255$ байт, так как это значение лежит даже немного выше верхней границы адресного пространства процессора 68010 (2^{24} байт).

Далее оператор цикла for можно заменить на чуть более быстрый - while - и при каждой проверке условия выхода из цикла уменьшать переменную len на 1 (с помощью команды DBRA на языке ассемблера), поскольку сохранять начальное значение этой переменной не требуется. Полученный в результате псевдокод приведен в примере 3,а. При этом регистры используются следующим образом:

D0 содержит значение первой контрольной суммы (sum1);

D1 содержит значение второй контрольной суммы (sum2);

D2 содержит длину (в байтах) блока передаваемых данных (len);

A0 содержит указатель блока передаваемых данных (ptr).

Псевдокод, выполняющий внутренний цикл, приведен в примере 3,б. Все примеры не содержат ни кода загрузки регистров перед входом в основной цикл вычисления, ни кода вычисления остатка от sum2 и сохранения значений по окончании основного цикла. Такие изменения сократили внутренний цикл до восьми операций на байт, и длительность его выполнения составила 56 машинных циклов, или 5,6 мкс на один байт. Хотя и это уже неплохо, в запасе осталось еще несколько ухищрений.

Очевидно, что после прибавления значения байта к текущему значению суммы sum1 для определения остатка от деления на 255 нужно выполнить три операции. Их можно сократить до одной: `add.b (a0)+,d0`, применяя правила однобайтовой арифметики, поскольку значение суммы sum1 не должно выходить за пределы интервала 0 - 254. Как выяснилось, эта замена обеспечивает почти безошибочные вычисления: только в случае переполнения полученная контрольная сумма окажется на 1 меньше правильного результата.

К счастью, в наборе команд процессора 68010 предусмотрена команда расширенного сложения для выполнения арифметических вычислений с повышенной точностью: содержимое двух регистров складывается и результат увеличивается на 1, если при выполнении команды добавочный бит стал ненулевым. Этот бит будет установлен в 1 предыдущей командой сложения, если сумма превысит 255. Если сохранять в регистре 0 и прибавлять к нему с помощью операции расширенного сложения результат предшествующего суммирования, последний можно увеличивать на 1 при каждом переполнении.

а)

```
integer i, sum1, sum2;

sum1 = 0;
sum2 = 0;
for i from 1 to message_length do
    sum1 = ( sum1 + message[i] ) modulo 255;
    sum2 = ( sum2 + sum1 ) modulo 255;
end for
```

б)

```
check1 = 255 - (( sum1 + sum2 ) modulo 255);
message[message_length+1] = check1;
check2 = 255 - (( sum1 + check1 ) modulo 255);
message[message_length+2] = check2;
```

Пример 1. Псевдокод для выполнения первой (а) и второй (б) частей алгоритма Флетчера


```

а)
register unsigned char *ptr;
register short int i, len, sum1, sum2;

sum1 = sum2 = 0;
for (i=0; i<len; i++)
{
    sum1 += *ptr++;
    if (sum1 >= 255) sum1 -= 255;
    sum2 += sum1;
    if (sum2 >= 255) sum2 -= 255;
}

б)
        moveq    #0,d5
        move.w   d5,d4      ;sum1 = sum2 = 0;
        moveq    #0,d6
        bra.s    L_29      ;for(i=0; i<len; i++) {
L_28:    moveq    #0,d0
        move.b    (a3)+,d0
        add.w     d0,d4      ; sum1 += *ptr++;
        cmp.w     #255,d4
        blt.s     L_32
        sub.w     #255,d4    ; if(sum1 >= 255) sum1 -= 255;
L_32:    add.w     d4,d5      ; sum2 += sum1;
        cmp.w     #255,d5
        blt.s     L_36
        sub.w     #255,d5    ; if(sum2 >= 255) sum2 -= 255;
L_36:    addq.w    #1,d6
L_29:    cmp.w     d7,d6
        blt.s     L_28      ;}

```

Пример 2. Неоптимизированная функция на языке C (а) и псевдокод на языке ассемблера (б) для вычисления контрольной суммы

```

а)
register unsigned char *ptr;
register short int sum1, len;
register unsigned long int sum2;

sum1 = sum2 = 0;
while (len--)
{
    sum1 += *ptr++;
    if (sum1 >= 255) sum1 -= 255;
    sum2 += sum1;
}
sum2 %= 255;

б)
loop:    moveq    #0,d3
        move.b    (a3)+,d3
        add.w     d3,d0      ; sum1 += *ptr++;

        cmp.w     #255,d0
        blt.s     no_sub
        sub.w     #255,d0    ; if (sum1 >= 255) sum1 -= 255;
no_sub:  add.l     d0,d1      ; sum2 += sum1;
        dbra     d2,loop    ; while (len--);

```

Пример 3. Оптимизированная функция на языке C (а) и оптимизированный псевдокод на языке ассемблера (б) для вычисления контрольной суммы

Правда, в одном случае эта схема не сработает: если при сложении со значением суммы sum1, хранящимся в регистре D0, получится 255. Такое число не подчиняется алгоритму контрольных сумм, так как выходит за пределы интервала 0 - 254, и в то же время недостаточно велико для обеспечения

переполнения. Эта проблема решается вне данного цикла. Если такая ситуация возникает в конце передаваемого блока данных, то вместо требуемого равенства 0 сумма sum1 будет равна 255, что легко проверяется на выходе из цикла. Если же такая ситуация возникает не в конце цикла, то прибавление

к sum1 следующего ненулевого байта вызовет переполнение, которое исправит ошибку. Значение sum1, как вы уже знаете, используется для вычисления sum2. Поскольку его значимой частью является остаток от деления на 255, то прибавление ошибочного значения 255 равносильно прибавлению правильного нулевого значения.

Использование регистра D4 для хранения константы 0, необходимой для выполнения расширенного сложения, позволило сократить число операций во внутреннем цикле до 4 (пример 4). Такая оптимизация значительно улучшает исходную версию. На обработку каждого байта проверяемых данных приходится лишь 4 машинные команды. Каждая из трех операций сложения длится соответственно 8, 4 и 6 машинных циклов, а операция отрицательного приращения и условного перехода - 10 машинных циклов, итого 28 машинных циклов на байт. Однако время выполнения цикла можно значительно сократить путем его развертывания.

В окончательном варианте псевдокода я развернул внутренний цикл в 16 последовательных проходов следующим образом. Предварительно в адресный регистр был помещен адрес последовательности из трех команд для выполнения сложения, соответствующей первому проходу цикла. Определение адреса осуществлялось по остатку от деления значения счетчика байтов на 16. Если он был равен 0, цикл завершался, если - 1, устанавливался режим выполнения последней группы команд, и т.д. После этого для компенсации потерь от развертывания цикла счетчику байтов присваивалась 1/16 его текущего значения.

Теперь на каждый обрабатываемый байт стало приходится по три команды сложения, что составило 18 машинных циклов. К тому же на каждые 16 байтов выполнялось еще по одной операции отрицательного приращения и условного перехода, итого в среднем 10/16 машинного цикла на один байт.

Таким образом, псевдокод стал выполняться в четыре с лишним раза быстрее исходного (18 5/8 машинных циклов вместо исходных 84). Применение оптимизированной версии в сети почти удвоило ее пропускную способность. Хотя приведенный псевдокод вычисления контрольных сумм Флетчера написан на языке ассемблера для процессора Motorola 680x0 и является машинно-зависимым, он может получить широкое распространение, поскольку сходные команды имеются на всех известных мне современных компьютерах.

Register	Usage
D0	the first checksum value (sum1)
D1	the second checksum value (sum2)
D2	the length in bytes of the buffer (len)
D4	contains zero, for the addx.b instruction
A0	pointer to the buffer (ptr)


```

loop    add.b    (a0)+,d0    ; sum1 += *ptr++;
        addx.b   d4,d0      ; if (sum1 >= 256) sum1 -= 1;
        add.l    d0,d1      ; sum2 += sum1;
        dbra     d2,loop    ; while (len--);

```

Пример 4. Окончательный вариант псевдокода для вычисления контрольной суммы

Метод Флетчера против CRC

Интересно сравнить предложенный выше алгоритм с написанной на языке ассемблера, хорошо отлаженной CRC-программой.

Лучшие из проанализированных мною CRC-программ имеют внутренний цикл, в котором на каждый бит сообщения выполняются по две команды сложения и сдвига, условный переход и одна или две операции "исключе-

чающее ИЛИ". В среднем получается 6,5 операций на бит, или 52 операции на каждый байт сообщения - почти в 20 раз больше, чем нужно при использовании метода Флетчера.

Итак, метод контрольных сумм Флетчера, реализуемый с помощью простого алгоритма, который можно записать в виде нескольких строк, обеспечивает превосходное качество обнаружения ошибок в передаваемых блоках данных. По уровню надежности алгоритм Флетчера сравним с алгоритмом CRC, но требует меньших (в несколько раз) затрат вычислительных мощностей. Если вы хотите добиться надежной компьютерной связи в системах с аппаратными средствами средней и низкой производительности, применяйте метод Флетчера.

Уважаемые читатели!

Мы продолжаем публикацию списков статей из очередных выпусков журналов "Dr. Dobb's Journal", "LAN Technology" и "DBMS".

Вы можете заказать копии заинтересовавших Вас статей, а также их перевод на русский язык.

Стоимость копии одной статьи независимо от ее объема (с учетом почтовых расходов и НДС) 149 руб.

Для получения копий следует перечислить соответствующую сумму ТОО "НОРМА-СП" на р/с 467094 в ГИБ "Балчуг" кор/с 161250 ОПЕРУ ГУ ЦБ РФ г. Москвы, МФО 201779 и направить копию платежного поручения вместе с заявкой по адресу: 115230, Москва, М-230, Варшавское шоссе, д.42, ТОО "НОРМА-СП".

В заявке необходимо указать название статьи, фамилию автора, название журнала, его номер и год выпуска, а также свои имя, отчество, фамилию и адрес (с почтовым индексом).

Копии статей будут высланы в Ваш адрес в недельный срок после получения заявки и копии платежного поручения.

Для получения перевода статьи на русский язык следует направить заявку на ее перевод по указанному выше адресу.

Вам будет выслан счет на оплату расходов по переводу и его пересылке.

Телефон для справок: 111-32-72

Dr. Dobb's Journal September 1992

POSTMORTEM DEBUGGING by Matt Pietrek

Средство отладки, заключающейся в сохранении состояния компьютера в момент краха системы, позволяет эффективно находить ошибки, особенно для программ, работающих в среде Windows в защищенном режиме.

AN EXCEPTION HANDLER FOR WINDOWS 3 by Brett Salter

Отладчик WINX.386 позволяет отслеживать все события при работе Windows в расширенном (enhanced) режиме.

YOUR OWN PROTECTED-MODE DEBUGGER by Rick Knoblauch

Резидентный отладчик для защищенного режима работы процессоров 386/486, использующий отладочные регистры, позволяет размещать точки останова в любом участке выполняемого кода.

HIGH-RESOLUTION TIMING by Thomas Roden

Способ реализации таймера с высоким разрешением для программ, работающих в среде DOS.

AN IMPROVED LISP-STYLE LIBRARY FOR C by Douglas Chubb

Усовершенствованный вариант C-библиотеки, включающей в себя мощное средство для "сборки мусора", содержит конструкции, аналогичные используемым в языке Lisp.

THE UNIVERSAL DEBUGGER INTERFACE by Daniel Mann

Обсуждение проблемы создания машинно-независимых отладочных средств.

DEBUGGING REAL-TIME SYSTEMS

by Gurjit Singh, Moses Joseph and Dave Barnett

Принцип модульности и постепенного наращивания и отлаживания систем реального времени позволяет сделать их надежными и функциональными.

CRASH TRACEBACKS IN UNIX by Alan Dunham

Компилирование кода с отладочной информацией позволяет в случае краха программы произвести "обратную прокрутку" ее работы путем анализа зафиксированных состояний стека.

WINDOWS TOOLHELP by Mike Sax

Описание библиотеки DLL, предназначенной для анализа и модификации текущего состояния среды Windows, в том числе выгрузки из памяти находящихся там программ и DLL.

A VIDEO COMPATIBILITY INTERFACE FOR TURBO DEBUGGER

by Danny Thorpe

Описание интерфейса, используемого отладчиком Turbo Debugger для переключения видеорежимов при работе в среде Windows 3.

HELP AND THE INSTALLATION BLUES by Al Stevens

Описание справочника (help) в пакете D-Flat.

DOWNSTREAM by Jeff Duntmann

Структурное программирование: анализ использования класса "поток" в диалоге для открытия файлов.

POOH AND THE SPACE STATION by Michael Abrash

Способ быстрого определения цвета поверхности в процессе преобразования пространственных фигур при трехмерной анимации.

COMING TO GRIPS WITH THE INFORMATION AGE by William F. Jolitz

Анализ взаимодействия вычислительных средств и телекоммуникаций.

October 1992

SIZING UP APPLICATIONS FRAMEWORKS AND CLASS LIBRARIES

by Ray Valdes

Анализ и оценка различных средств, предназначенных для разработки графических пользовательских интерфейсов высокого уровня.

OBJECT-ORIENTED PROGRAM CONSTRUCTION by William G. Wong

Описание класса Plugs, служащего для стыковки объектов при объектно-ориентированном программировании.

SUPERDISTRIBUTION AND ELECTRONIC OBJECTS by Brad Cox

Обсуждение возможности перехода к нетрадиционному способу распространения программных средств: бесплатному приобретению программ и их компонентов и оплате по мере использования.

A TASTE OF DYLAN by David Betz

Новый объектно-ориентированный язык Dylan фирмы Apple Computer обеспечивает автоматическое распределение памяти, проверку типов данных и динамическую стыковку объектов на этапе выполнения.

DPMI MEETS C++ by Frederick Hewett

DPMI - средство, позволяющее DOS-программам работать в защищенном режиме. В статье описывается организация взаимодействия программ и DPMI с помощью библиотеки классов языка C++.

Продолжение см. на с. 62

Верификация файлов

Проверка файлов с помощью 32-разрядного циклического избыточного кода.

М. Нельсон

Верификация файлов - это проверка их целостности. Пользователю важно знать, не изменилось ли без его ведома содержимое файла, независимо от причины изменения: из-за сбоя оборудования, ошибки в программе или по злому умыслу.

Несколько неприятных событий в моей практике побудили меня заняться разработкой удобного и надежного способа проверки целостности файлов.

Выполняя на домашнем компьютере ряд непроверенных программ, я увидел, что по меньшей мере две из них вызвали случайные повреждения файловых систем. Одна из этих систем работала под управлением операционной системы (ОС) UNIX, другая - под управлением MS-DOS. После восстановления явно испорченных данных я уже не мог доверять и остальным частям файловых систем: может быть, другие файлы тоже получили повреждения, но не столь заметные?

Второй, более тревожный случай заставил меня усомниться в достоверности всех файлов, хранящихся в компьютерах вычислительной системы моей фирмы. Это произошло, когда один из наших поставщиков программных продуктов в разговоре между прочим упомянул, что компьютеры в фирме, где он работает, заражены вирусом "Stoned". (К счастью, тревога оказалось ложной.)

Наконец, недавно было решено разработать программу, позволяющую клиентам переписывать на свои компьютеры из нашей информационной вычислительной сети "заплаты" для обновления приобретенных у нас программ. Я написал небольшую программу ввода файла-"заплаты" в исходный файл. Однако для упрощения программы был необходим способ идентификации и верификации файла, подлежащего исправлению. Иначе, если бы перед вы-

полнением этой программы пользователь успел изменить в файле хоть несколько строк, она могла бы его просто испортить.

Все эти проблемы решаются в два этапа. На первом работает алгоритм CRC-32 (32-bit Cyclical Redundancy Check - проверка с помощью 32-разрядного циклического избыточного кода), позволяющий идентифицировать файлы подобно тому, как идентифицируют отпечатки пальцев. На втором этапе программа CRCMAN создает список контрольных значений CRC-32 для всех файлов в дереве каталогов, чтобы потом можно было их сравнивать с вновь вычисленными значениями кода CRC-32 для файлов в том же дереве каталогов.

Алгоритм CRC-32

В основе этого алгоритма лежит формула вычисления значения кода CRC-32, одобренная Международным консультативным комитетом МСЭ по телеграфной и телефонной связи (CCITT). Алгоритм CRC-32 используется в протоколах обмена данными, например HDLC и ZMODEM, для проверки целостности блоков данных, передаваемых в различных средах. Вычисление значения кода CRC-32 производится методом деления полиномов: "полинома-генератора" на "полином-сообщение". В передаваемом блоке данных (сообщении), независимо от его длины каждый бит рассматривается как коэффициент некоего длинного полинома. Например, байт F0H (в шестнадцатичном представлении) будет выглядеть следующим образом: $1 \cdot X^7 + 1 \cdot X^6 + 1 \cdot X^5 + 1 \cdot X^4 + 0 \cdot X^3 + 0 \cdot X^2 + 0 \cdot X^1 + 0 \cdot X^0$.

Члены полинома с нулевыми коэффициентами отбрасываются, и он принимает такой вид: $X^7 + X^6 + X^5 + X^4$.

Показатели степени членов полинома, соответствующего всему файлу, будут очень велики, но это не влияет на алгоритм вычисления значений кода CRC-32.

В алгоритме CRC-32 в качестве "полинома-генератора" используется следующее выражение: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{16} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$.

После деления этого полинома на "полином-сообщение" частное отбрасывается, а остаток используется как значение кода CRC-32 для данного файла.

Этот вычислительный процесс выполняется на персональных компьютерах достаточно быстро и эффективно. Для программы CRCMAN я выбрал версию алгоритма CRC-32 с использованием таблицы преобразования, что позволило ускорить вычисления благодаря некоторому увеличению необходимого объема памяти.

Способы вычисления значения кода CRC подробно обсуждаются во многих изданиях (см. ниже раздел "Литература").

Особенности алгоритма CRC-32

В программе CRCMAN каждому конкретному файлу соответствует 32-разрядное число, как бы "отпечаток пальца" данного файла. Отпечаток пальца человека строго индивидуален, однако, к сожалению, этого нельзя сказать о файлах. В мире существует больше 4 294 967 296 (2^{32}) различных файлов, поэтому заранее очевидно, что некоторые из них могут иметь одинаковые значения кода CRC-32. Однако алгоритм CRC-32 обладает очень удобными для верификации файлов особенностями:

- каждый бит сообщения вносит свой вклад в формирование значения кода CRC-32, так что изменение любого бита в сообщении изменяет и соответствующее данному файлу контрольное значение;

• даже небольшие изменения в сообщении обязательно порождают изменения значения кода CRC-32 (у двух файлов, различающихся лишь несколькими битами, несомненно, будут разные контрольные значения);

• гистограмма зависимости значений кода CRC-32 от входных сообщений имеет вид почти прямой линии, т.е. для данного сообщения вероятность получения определенного контрольного значения примерно одинакова во всем диапазоне возможных значений кода CRC-32: от 0 до FFFFFFFFH.

Именно эти особенности были учтены комитетом CCITT при выборе алгоритма CRC-32. Программа CRCMAN практически обязательно "выловит" случайное изменение в файле, поскольку вероятность такого нарушения или модификации файла без изменения значения кода CRC-32 ничтожно мала.

Еще одной особенностью кода CRC-32 является его невоспроизводимость, или необратимость (*noninvertability*). Для защиты от случайного повреждения файла это свойство кода CRC-32 не используется, однако, если наша цель - обнаружить зараженные вирусом файлы, оно становится просто необходимым.

Типичный вирус может изменить интерпретатор команд COMMAND.COM системы MS-DOS. Моей версии файла COMMAND.COM соответствует контрольное значение 02f8690cH. При изменении вирусом содержимого этого файла у него, несомненно, будет другое значение кода CRC-32. При этом разработчику программы-вируса нужно добавить новые байты к концу файла таким образом, чтобы первоначальное контрольное значение было восстановлено.

К сожалению, это вполне возможно. Разработчик вируса может сначала добавить ограниченное число байтов в конце файла COMMAND.COM, вычислить новое значение CRC-32 для измененного файла, а затем систематически перебирать все новые и новые дополнительные байты до тех пор, пока случайно не получит комбинацию байтов с правильным контрольным значением. При этом, используя эффективный алгоритм, значение кода CRC-32 можно вычислить за 1 мс, а для подбора комбинации байтов с необходимым значением кода CRC-32 потребуются несколько недель работы компьютера. Однако применение достаточно мощного процессора позволяет решить задачу за один вечер.

Итак, программу, вероятно, сможет одолеть особенно "умный" вирус, зато она непременно вызовет случайно нару-

шенные файлы. Дальнейшие усовершенствования программы улучшат ее возможности в борьбе с вирусами. Например, если наряду со значением кода CRC-32 запоминать также длину файла, уже это существенно затруднит задачу разработчика вируса, если не сделает ее невыполнимой.

```
363476a4 \CRC.TXT
b97a5169 \CRC.BAK
d6a5f5f5 \CRCMAN.C
02f8690c \TEST\COMMAND.COM
88f2e4d6 \CRCMAN.EXE
23123ele \TEMP.CRC
```

Пример 1. Содержимое файла TEST.CRC

```
Checking \CRC.TXT Error: Expected 76a414c6, got 86793634
Checking \CRC.BAK Error: Expected 516914c6, got 76a4b97a
Checking \CRCMAN.C OK
Checking \TEST\COMMAND.COM OK
Checking \CRCMAN.EXE OK
Checking \TEMP.CRC OK
```

Пример 2. Выходной файл программы CRCMAN

Работа программы CRCMAN

Выполняя две различные задачи, программа CRCMAN работает в двух режимах: в одном для создания списка имен файлов и соответствующих им значений кода CRC-32, а в другом для проверки файлов по этому списку. Режим работы определяется командной строкой вызова программы. Для работы в первом режиме программу CRCMAN следует запускать с помощью такой командной строки: CRCMAN—b directory crc-file-name. Программа CRCMAN вычислит значение кода CRC-32 для каждого файла в указанном в командной строке каталоге и во всех его подкаталогах и поместит результаты в файл, имя которого задано в качестве второго параметра в командной строке. Создаваемый файл - это обычный текстовый ASCII-файл, с которым можно работать, используя любой текстовый редактор. Этот файл содержит последовательность строк, в каждой из которых записано значение кода CRC-32 для данного файла, а за ним - имя этого файла.

Например, для проверки каталога с моей статьей на компьютере, работающем под управлением MS-DOS, я запустил программу CRCMAN с помощью следующей командной строки: CRCMAN—b C:\CRCFILES\TEST.CRC

При этом был создан файл TEST.CRC (пример1). Впоследствии я смогу проверить целостность файлов (с моей статьей), запустив программу CRCMAN для работы во втором режиме с помощью командной строки CRCMAN C:\CRCFILES\TEST.CRC. В этом режиме программа CRCMAN считывает каждую строку списка значений кода CRC-32, снова вычисляет контрольное значение CRC для соответствующего файла из списка и определяет, совпадает ли оно с сохраненным ранее значением. Работая над этой статьей, я изменил в текстовом файле несколько

строк, запустил программу CRCMAN и получил информацию, приведенную в примере 2. Программа правильно определила два измененных файла. Текущая реализация программы при обнаружении изменений выводит на экран сообщение об ошибке. Промышленная версия программы может выполнять сколь угодно сложные действия.

Программный код

Программа CRCMAN (см. листинг 1) в среде MS-DOS работает с любым из компиляторов языка C, в среде ОС XENIX - с компилятором языка C, соответствующим соглашению K&R (Б. Кернигана и Д. Ритчи), в среде ОС UNIX - с компилятором языка C фирмы Microsoft. Возможность работы этой программы в различных средах (*portability*) достигается без выполнения каких-либо дополнительных действий, за несколькими исключениями. Например, версия 6.0 компилятора языка C фирмы Microsoft выводит на экран предупреждающие сообщения уровня 4, поэтому компиляция данного кода должна выполняться с опцией /W3, а не с опцией /W4. Кроме того, если программа компилируется для работы под управлением ОС UNIX, то получить правильный программный код обработки каталогов можно, определив именованную константу OC UNIX. Некоторые места исходного кода выделены операторами #ifdef OC UNIX. Это вынуждает программиста, компилирующего данный исходный код в среде ОС UNIX или XENIX, указать именованную константу OC UNIX либо в тексте программы, либо в командной строке.

Марк Р. Нельсон (Mark R. Nelson) -
вице-президент фирмы
Greenleaf Software и автор книги
"The Data Compression Book"
(M&T Publishing).
Ego адрес: 16479 Dallas Parkway,
Suite 570, Dallas, TX 75248.

Программный код CRCMAN выполняется следующим образом. Прежде всего функция `main()` программы CRCMAN должна вызвать функцию `BuildCRCTable()`, которая инициализирует все значения в массиве `CRCTable[]`. Далее по ходу выполнения программы они используются всякий раз, когда вычисляются значения кода CRC-32.

Следующая после формирования таблицы значений кода CRC-32 задача функции `main()` - проверка выбранного пользователем режима работы программы CRCMAN, основанная на анализе значения параметров командной строки. Если значение параметра `args` функции `main()` равно 2, считается, что в командной строке был указан единственный параметр - имя файла, поэтому вызывается функция `CheckFiles()`. Если значение параметра `args` равно 4 и первый параметр равен `-b`, считается, что программа работает в режиме создания файла-списка значений кода CRC-32, и вызывается функция `BuildCRCFile()`. Если же ни одно из этих условий не выполнено, то программа завершается появлением на экране монитора указания по ее использованию.

Создание файла-списка значений кода CRC-32

Программа CRCMAN должна вычислять контрольные значения для одного или более файлов, работая в обоих режимах, но при создании списка значений кода CRC-32 появляется дополнительная работа - перемещение по дереву каталогов, которое в зависимости от среды (OC UNIX или MS-DOS) выполняется по-разному.

Функция `BuildCRCFile()` открывает выходной файл, куда будут занесены имена всех файлов и соответствующие им значения кода CRC-32, и вызывает функцию `ProcessAllFiles()` для выполнения всей работы. Этой функции передается два параметра командной строки: составное имя каталога и указатель структуры `FILE` для файла-списка значений кода CRC-32.

Функция `ProcessAllFiles()` одинаково выполняется в средах OC UNIX, XENIX или MS-DOS с использованием различных компиляторов языка C. К сожалению, имена функций и структур данных, необходимые для работы в различных средах, немного отличаются друг от друга. Программный код для этой функции показан в примере 3.

При его выполнении в различных средах используются разные функции: для открытия каталога и чтения очередного имени файла в среде OC UNIX - функции `opendir()` и `readdir()`, в среде

```
ProcessAllFiles(paht)
dir = OpenDirectory(paht)
while FilesLeftInDirectory(dir)
    filename = GetNextFile(dir)
    if filename is a directory then
        ProcessAllFiles(filename)
    else
        crc = CalculateCRC(filename)
        write filename and crc
    endif
end of while
end of ProcessAllFiles
```

Пример 3. Программный код для функции `ProcessAllFiles()`

MS-DOS - `findfirst()` и `findnext()`. К тому же для реализации этих системных вызовов в среде MS-DOS компиляторы фирмы Borland используют имена функций и структур данных, отличные от выбранных другими фирмами.

Из-за всех этих вариаций программный код представляет собой нагромождение операторов `#ifdef` и макроопределений. Однако, разобравшись в этом коде, вы убедитесь, что язык C совсем не так плох. Функция `ProcessAllFiles()`, в частности, хороша тем, что она дает отличный пример задачи, легче всего решаемой с помощью рекурсии.

Почти в конце этой функции вызывается функция `CalculateFileCRC()`, которая вычисляет значение кода CRC-32 для очередного файла. Затем результат вместе с именем этого файла записывается в файл-список значений кода CRC-32. Так, постепенно, по мере выполнения функции `ProcessAllFiles()` создается полный список значений кода CRC-32 для всего дерева каталогов.

Вычисление значения кода CRC-32 для файла

Такие вычисления довольно просты. Функция `CalculateFileCRC()` последовательно считывает блоки данных длиной 512 байт и обрабатывает их с помощью функции `Calculate-BufferCRC()`. В качестве параметра в ней используется значение кода CRC-32, вычисленное для данного файла от его начала до текущей позиции к моменту начала чтения очередного блока данных, а затем контрольное значение, соответствующее новой текущей позиции. Процесс повторяется вплоть до завершения обработки всего файла.

При этом данная функция предвзято вычисления заданием начального значения кода CRC-32 - "все единицы", или `0xFFFFFFFF`. По завершении всех вычислений разряды этого числа инвертируются с помощью логической операции "исключающее ИЛИ". Такие на-

чальные и конечные условия задаются для обеспечения большей защищенности кода от ошибок и используются протоколом обмена данными ZMODEM и программой архивации PKZIP или ARJ. Таким образом, значения кода CRC-32, полученные по завершении программы CRCMAN, совпадут с результатами аналогичных вычислений, сделанных с помощью программы PKZIP для того же набора файлов.

Проверка файлов

Второй режим выполнения программы CRCMAN - проверка файлов по их значениям кода CRC-32. При этом большую часть работы функция `CheckFiles()` выполняет, не перемещаясь по дереву каталогов, поскольку имена всех проверяемых файлов уже хранятся в файле-списке значений кода CRC-32. Эта функция последовательно, строка за строкой, считывает из списка имена файлов и соответствующие им значения кода CRC-32, вычисляет для текущего файла новое контрольное значение и сообщает о его несовпадении с хранимым значением.

Обычный текстовый формат файла-списка обеспечивает легкость сопровождения таких файлов. Допустим, список содержит значения кода CRC-32 для дерева каталогов, в котором хранится компилятор языка C++ фирмы Borland. При этом вам регулярно будут поступать сообщения от программы CRCMAN об изменениях файла задания конфигурации `TURBOC.CFG`. Зная о них, вы без особого труда можете отредактировать файл-список и удалить строку, относящуюся к файлу `TURBOC.CFG`.

Применение программы CRCMAN

Программа CRCMAN помогает быстро проверить целостность одного или всех файлов вашей вычислительной системы. Вызывая программу CRCMAN с параметром `-b` для каждого каталога, содержащего большое число загрузочных модулей, вы создаете ряд файлов-списков значений кода CRC-32. Затем, вызывая программу CRCMAN, вы можете периодически проверять файлы в каждом из этих списков. Программа выполняется настолько быстро, что при работе в среде MS-DOS можно даже включить командную строку для ее вызова в файл `AUTOEXEC.BAT` - производительность от этого практически не пострадает.

Программу CRCMAN легко приспособить для эффективного обнаружения вирусов. Как уже упоминалось выше,

наряду с контрольным значением неплохо запоминать длину соответствующего файла, и, возможно, даже время его создания. Хотя даже эти меры не дадут полной гарантии от нападения вируса, они, вероятно, помогут обнаружить подавляющее большинство случаев заражения.

Наконец, наилучший путь использования программы CRCMAN на вашем компьютере - непрерывное ее выполнение с низким приоритетом в параллельном режиме. Это довольно просто реализовать в среде ОС OS/2 или UNIX, и весьма проблематично, удастся ли это сделать в среде MS-DOS.

Литература

1. Campbell, Joe. C Programmer's Guide to Serial Communications. Indianapolis, Ind.: Howard W. Sams, 1988.

2. Ramabadran, Tenkasi V. and Sunil S. Gaitonde. A Tutorial on CRC Computations. IEEE Micro (August 1988).

ЛИСТИНГ

```

/* ***** START OF CRCMAN.C ***** */
* This program is used to build a list of CRC-32 values for all of files in a
* given directory tree. After building file, program can be run later to
* verify CRC values, giving assurance of integrity of files. To build CRC file
* command line is: CRCMAN -b root-dir crc-file-name
* To check list of files created, run with: CRCMAN crc-file-name
* Should work with most 16 and 32-bit compilers under MS-DOS and UNIX. **

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>

unsigned long CRCTable[ 256 ];

/* To build this program under UNIX, define UNIX either on the command line or
* by editing this file. To define it on the command line, program should be
* built like this: cc -o crcman -DUNIX crcman.c Code in this program assumes
* UNIX compiler is K&R variety; does away with real function prototyping. */
#ifdef UNIX

#include <varargs.h>
#ifdef M_XENIX
#include <sys/ndir.h>
#else
#include <sys/diront.h>
#endif /* M_XENIX */

#define SEPARATOR "/"
#define FILENAME_SIZE 81

void FatalError();
unsigned long CalculateFileCRC();
void ProcessAllFiles();
void BuildCRCFile();
void CheckFiles();
unsigned long CalculateBufferCRC();
void BuildCRCTable();

#else /* not UNIX, must be MSDOS */
/* Most MS-DOS compilers have converged on same names for structures and
* functions used when searching directories. Borland C implementations use a
* variant, requiring macro definitions. Functions work in an identical manner,
* so actual implementation of code is straightforward. Addition of MS-DOS
* definition helps convince Zortech compiler to use same structure and
* function names as everyone else. */
#define MSDOS 1
#include <stdarg.h>
#include <dos.h>

#define SEPARATOR "\\"
#define FILENAME_SIZE FILENAME_MAX

#ifdef __TURBOC__

#include <dir.h>
#define FILE_INFO struct fblk
#define FIND_FIRST( n, i ) findfirst( ( n ), ( i ), FA_DIRC )
#define FIND_NEXT( info ) findnext( ( info ) )
#define FILE_NAME( info ) ( ( info ).ff_name )

#else

#define FILE_INFO struct find_t
#define FIND_FIRST( n, i ) _dos_findfirst( ( n ), _A_SUBDIR, ( i ) )
#define FIND_NEXT( info ) _dos_findnext( ( info ) )
#define FILE_NAME( info ) ( ( info ).name )

#endif

#endif

void FatalError( char *fmt, ... );
unsigned long CalculateFileCRC( FILE *file );
void ProcessAllFiles( char *path, FILE *crc_file );
void BuildCRCFile( char *input_dir_name, char *crc_file_name );
void CheckFiles( char *crc_file_name );
unsigned long CalculateBufferCRC( unsigned int count, unsigned long crc,
void *buffer );

void BuildCRCTable( void );

#endif /* UNIX */

/* Main program checks for valid occurrences of two different types of command
* lines, and executes them if found. Otherwise, it prints out a simple
* usage statement and exits. */
int main( argc, argv )
int argc;
char *argv[];
{
    setbuf( stdout, NULL );
    BuildCRCTable();
    if ( argc == 2 )
        CheckFiles( argv[ 1 ] );
    else if ( argc == 4 && strcmp( argv[ 1 ], "-b" ) == 0 )
        BuildCRCFile( argv[ 2 ], argv[ 3 ] );
    else {

```

```

        printf( "Usage: CRCMAN [-b input_dir] crc-file\n" );
        printf( "\n" );
        printf( "Using the -b option checks all files under the input_dir\n" );
        printf( "and appends their data to the crc-file. Otherwise, the\n" );
        printf( "program checks the CRC data of all of the files in the\n" );
        printf( "crc-file and prints the results\n" );
        return( 1 );
    }
    return( 0 );
}

/* Instead of performing a straightforward calculation of the 32-bit CRC using
* a series of logical operations, program uses faster table lookup method. */
#define CRC32_POLYNOMIAL 0xEDB88320L

void BuildCRCTable()
{
    int i;
    int j;
    unsigned long crc;

    for ( i = 0; i < 255; i++ ) {
        crc = i;
        for ( j = 8; j > 0; j-- ) {
            if ( crc & 1 )
                crc = ( crc >> 1 ) ^ CRC32_POLYNOMIAL;
            else
                crc >>= 1;
        }
        CRCTable[ i ] = crc;
    }
}

/* Routine checks CRC values for a list of files. */
void CheckFiles( crc_file_name )
char *crc_file_name;
{
    FILE *crc_file;
    FILE *test_file;
    unsigned long log_crc;
    unsigned long crc;
    char log_name[ FILENAME_SIZE ];
    int result;

    crc_file = fopen( crc_file_name, "r" );
    if ( crc_file == NULL )
        FatalError( "Couldn't open the log file: %s\n", crc_file_name );
    for ( ; ; ) {
        result = fscanf( crc_file, "%lx %s", &log_crc, log_name );
        if ( result < 2 )
            break;
        test_file = fopen( log_name, "rb" );
        if ( test_file != NULL ) {
            printf( "Checking %s", log_name );
            crc = CalculateFileCRC( test_file );
            fclose( test_file );
            if ( crc != log_crc )
                printf( "Error: Expected %08lx, got %08lx\n",
                    log_name, log_crc, crc );
            else
                printf( "OK\n" );
        } else
            printf( "Could not open file %s\n", log_name );
    }
}

/* ProcessAllFiles() scans directory. Routine also makes sure that directory
* name passed on command line is stripped of trailing '/' or '\' character. */
void BuildCRCFile( input_dir_name, crc_file_name )
char *input_dir_name;
char *crc_file_name;
{
    char path[ FILENAME_SIZE ];
    FILE *crc_file;

    strcpy( path, input_dir_name );
    if ( path[ strlen( path ) - 1 ] == SEPARATOR[ 0 ] )
        path[ strlen( path ) - 1 ] = '\0';
    crc_file = fopen( crc_file_name, "w" );
    if ( crc_file == NULL )
        FatalError( "Can't open crc log file: %s\n", crc_file_name );
    ProcessAllFiles( path, crc_file );
}

/* This routine is responsible for actually performing the calculation of the
* 32-bit CRC for the entire file. We precondition the CRC value with all 1's,
* then invert every bit after the entire file has been done. This gives us a
* CRC value that corresponds with the values calculated by PKZIP and ARJ. */
unsigned long CalculateFileCRC( file )
FILE *file;
{
    unsigned long crc;
    int count;
    unsigned char buffer[ 512 ];
    int i;

    crc = 0xFFFFFFFFL;
    i = 0;
    for ( ; ; ) {
        count = fread( buffer, 1, 512, file );
        if ( ( i++ % 32 ) == 0 )
            putc( '.', stdout );
        if ( count == 0 )

```



```

        break;
        crc = CalculateBufferCRC( count, crc, buffer );
    }
    puts( "\n", stdout );
    return( crc ^ 0xFFFFFFFF );
}
/* This is the routine that is responsible for calculating all of CRC values
 * for files in a given directory. The CRC values and file names are written
 * out to the crc_file. */
void ProcessAllFiles( path, crc_file )
char *path;
FILE *crc_file;
{
#ifdef UNIX
    DIR *dirp;
#endif
#ifdef M_XENIX
    struct dirent *entry;
#else
    struct dirent *entry;
#endif
#define NAME entry->d_name
    FILE_INFO fileinfo;
    int done;
#define NAME FILE_NAME( fileinfo )
    char fullname[ FILENAME_SIZE ];
    struct stat buf;
    unsigned long crc;
    FILE *file;

    printf( "Searching %s\n", path );
    strcat( path, SEPARATOR );
#ifdef UNIX
    dirp = opendir( path );
    if ( dirp == NULL )
        FatalError( "Error opening directory %s\n", path );
    entry = readdir( dirp );
    while ( entry != 0 ) {
#else
        strcpy( fullname, path );
        strcat( fullname, "." );
        done = FIND_FIRST( fullname, &fileinfo );
        while ( done == 0 ) {
            strcpy( fullname, path );
            if ( strcmp( NAME, "." ) && strcmp( NAME, ".." ) ) {
                strcat( fullname, NAME );
                if ( stat( fullname, &buf ) == -1 )
                    FatalError( "Error reading stat from file %s\n", fullname );
                if ( buf.st_mode & S_IFDIR )
                    ProcessAllFiles( fullname, crc_file );
                else {
                    file = fopen( fullname, "rb" );
                    if ( file != NULL ) {
                        printf( "Scanning %s ", fullname );
                        crc = CalculateFileCRC( file );

```

```

        puts( "\n", stdout );
        fprintf( crc_file, "%08lx %s\n", crc, fullname );
        fclose( file );
    }
    else
        printf( "Could not open %s\n", fullname );
}
#endif
entry = readdir( dirp );
else
    done = FIND_NEXT( &fileinfo );
#endif
}
/* Routine calculates the CRC for a block of data using table lookup method.
 * It accepts an original value for the crc, and returns the updated value. */
unsigned long CalculateBufferCRC( count, crc, buffer )
unsigned int count;
unsigned long crc;
void *buffer;
{
    unsigned char *p;
    unsigned long temp1;
    unsigned long temp2;

    p = (unsigned char*) buffer;
    while ( count-- != 0 ) {
        temp1 = ( crc >> 8 ) ^ 0x00FFFFFF;
        temp2 = CRCTable[ ( int ) crc ^ *p++ ] & 0xff;
        crc = temp1 ^ temp2;
    }
    return( crc );
}
/* Fatal error handler prints a formatted error message and then exits. */
#ifdef UNIX
void FatalError( va_alist )
va_dcl
{
    char *fmt;
    va_list argptr;

    va_start( argptr );
    fmt = va_arg( argptr, char * );
    else
        void FatalError( char *fmt, ... )
        {
            va_list argptr;
            va_start( argptr, fmt );
            printf( "Fatal error: " );
            vprintf( fmt, argptr );
            va_end( argptr );
            exit( -1 );
        }
}

```

Внимание! Внимание! Внимание! Внимание! Внимание!

Если Вы занимаетесь версткой под Windows и применяете:

- ♦ принтеры не поддержива- ♦ хотите видеть на экране
- ющие язык PostScript
- ♦ шрифты с национальными символами
- реальное отображение
- Вашей верстки...

ПараГраф поставит Вам шрифты в формате PostScript вместе с программой Adobe Type Manager® 2.02

За более подробной информацией обращайтесь по адресу:

103051 Москва, Петровский бульвар 23, телефоны: (095) 200 25 66, (095) 924 17 81, телефакс: (095) 928 27 68
 ТОО "Поликом" 194223, г. С.-Петербург, ул. Пушкинская, д. 2, М-н "Техническая книга" телефон: (812) 164 56 33
 Adobe Type Manager, ATM лого является торговыми марками Adobe Systems Inc

PARAGRAPH

Как киевляне OS Pick рекламировали

Почта "Журнала д-ра Добба" обширна. В письмах, идущих в наш адрес, читатели иногда критикуют нас, иногда дают полезные советы, а зачастую и благодарят. Но как бы то ни было, обычно письма наших читателей носят конструктивный характер, поэтому мы были несколько удивлены следующим письмом, пришедшим как отклик на материал, опубликованный в первом номере журнала за 1992 год.

Уважаемый господин редактор!

На западном рынке программного обеспечения имеется много эффективных программных средств, которые у нас малоизвестны. К таким относится и имеющая действительно уникальную архитектуру OS Pick, давно ожидаемая статья о которой опубликована в вашем журнале N 1 за 1992 год. К сожалению, отсутствие у авторов статьи подробной информации не позволило им дать вполне объективный анализ и сделать положительные выводы о перспективности использования OS Pick. В связи с этим хотелось бы осветить некоторые весьма важные детали, которые помогут составить более полное представление об OS Pick (в ее последней версии - Advanced Pick).

В стандартном варианте OS Pick объединяет в себе эталонную операционную систему и СУБД. Главное ее свойство - простота, ориентация на конечного пользователя. Виртуализация памяти позволяет использовать минимальное количество системных ресурсов - при работе на ПК AT/286 до 10-ти пользователей достаточно 640 Кб оперативной памяти. Переменная длина записей и введение многозначных полей дает существенную экономию дискового пространства при хранении данных. Экономичность и эффективность OS (соотношение производительности к цене) делает ее привлекательной для пользователей, имеющих минимальный состав аппаратных средств. Использо-

вание специализированного ассемблера дает возможность делать программное обеспечение мобильным (совместимым) и в течение нескольких недель привязывать OS Pick к ЭВМ новой платформы (уже имеются реализации на ЭВМ более 70 платформ, в т.ч. IBM RS/600). Последняя версия системы является многозадачной, с возможностью перенаправления результатов другим пользователям, приостановки выполнения любой задачи и вложенного запуска новых. Поддерживаются также процедуры транзакций, выборочного сохранения на стриммере только измененных данных, диалоговой настройки на любой тип терминала. Версия OS Pick на host-ПК AT/386/486 позволяет работать одновременно 32 пользователям (OS MS/DOS может находиться тут же в другом разделе).

Встроенная в OS Pick СУБД "активного типа" управляемая словарями и позволяет производить гибкую модификацию структуры файлов и связей между ними. Программы алгоритмического языка Pick/BASIC могут вызываться прямо из атрибутов словаря БД. Записи файлов данных могут иметь неограниченную длину. Технология В-деревьев, двоичные индексы и процессор макросов ускоряет скорость работы и доступ к данным. Средства генерации приложений включают в себя двунаправленный генератор отчетов, генерацию меню и Update Processor. Полнофункциональный интегрированный редактор Update Processor использует словари для создания форм ввода-вывода, обновления связанных данных по заданным формулам и правилам (с использованием программ Pick/BASIC), осуществление проходов ("крейсирование") через связанные файлы и экранные формы. Функционально, как СУБД, Pick конкурирует с такими системами как Oracle, Ingres, FOCUS, ADABAS, dBASE.

В настоящее время рынок Pick-продуктов оценивается в 3 млрд. долларов, более 3 тысяч приложений, которые используют около 3 млн. пользователей. Десятки фирм предлагают широкий спектр развитых прикладных средств

для работы в среде OS Pick: текстовые процессоры, деловую графику, электронные таблицы, телекоммуникацию, генераторы приложений и т.п. Процесс разработки ПО в OS Pick становится все более привлекательным, что связано с развитием инструментальных средств и появлением языков четвертого поколения, CASE-систем в OS Pick, которая как никакая другая ОС способствует созданию современных технологий проектирования и разработки ПО.

Западные аналитики отмечают большую перспективность использования OS Pick как СУБД в различных операционных средах в связи с переходом фирмы Pick Systems к реализации принципов архитектуры открытых систем. Уже имеется реализация продукта Advanced Pick в среде OS Unix (шлюз обмена данными, компилятор языка C, доступность использования функций OS Unix и периферийных устройств), в стадии завершения - под OS MS/DOS и Windows. Ряд фирм предлагают ПО, позволяющее host-ЭВМ с OS Pick работать в JBC с OS NetWare фирмы Novell, пользуясь системой синонимов создавать распределенные базы данных на ЭВМ различных платформ.

Мир Pick объединяет клубы пользователей во многих странах. Существует организация Spectrum Manufacturers Association (г. Сан-Диего, США), которая объединяет разработчиков Pick-продуктов и определяет базовый уровень совместимости Pick-основанных систем. Издаются несколько ежемесячных журналов (в т.ч. Pick World) и газет, а также каталогов Pick-продуктов.

В странах СНГ локализованная система Pick эксплуатируется на ЭВМ фирм Bull, Hewlett-Packard, IBM, McDonnell Douglas Computer Systems, Sequoia.

Литература

(Приведен список из пяти наименований. - Прим. ред.)

С наилучшими пожеланиями в Вашей работе,

Б.В. Рычко, к. э. н. И.С. Кара
г. Киев

Смутило нас, что в письме ничего не говорится о нашей статье "по-существу" (что же там не так, кроме вывода о перспективах ОС Pick?), да еще этот список литературы, на которую авторы письма в тексте вообще не ссылаются... При повторном же прочтении письма мы сообразили: в огороде-то бузина, а значит в Киеве дядька. И еще поняли, что весь текст, кроме первого и двух последних абзацев представляет собой перевод рекламных материалов фирмы Pick Systems (критиковать перевод не будем, скажем лишь, что предпочитаем оригинальные тексты). Обсуждать рекламу тоже смысла нет, поскольку она всегда носит общий характер и зачастую содержит термины, понятные только ее авторам.

Стали обсуждать в редакции, к чему бы авторы такое письмо написали. Нашлись горячие головы, высказали резкое суждение: мол, "телега" это, и понимать ее следует так: "Вам пишут, не зная толком предмета, а вы сразу печатать".

Но такое предположение сразу отменили. С одной стороны, письмо написали солидные люди, а с другой - не могут же читатели "Журнала д-ра Добба" не знать, что нашим западным партнером является известная фирма M&T, одно из изданий которой (журнал "DBMS") целиком посвящено проблемам баз данных. Кроме того, мы сотрудничаем еще с рядом зарубежных и отечественных фирм, работающих на информационном рынке (включая крупнейшие научно-технические библиотеки страны), не говоря уж о представительствах фирм, специализирующихся на разработке компьютеров и программного обеспечения.

И, наконец, при нашем журнале действует Лаборатория тестирования, в которой работают вполне квалифицированные люди. Так что мы в состоянии запросить нужную информацию, технические или программные средства из любого источника и провести необходимое тестирование. Мы вполне могли бы очень существенно расширить список приведенной в письме литературы по ОС Pick. Чего-чего, а недостатка в информации у нас нет.

Журнал анализирует деятельность различных фирм и имеет возможность сравнить ОС Pick (и, в частности, входящую в нее СУБД) со многими други-

ми программными продуктами. Мы стараемся улавливать тенденции на мировом рынке компьютеров и программного обеспечения. Поверьте, это очень большой труд. А для статьи "Врастаем в Pick?" авторы вычленили суть системы - отсюда и выводы...

Кроме того, по "счастливой случайности" авторы статьи в свое время проделали практически всю содержательную работу по локализации клона ОС Pick для фирмы Bull (на нашем рынке она называется ULT-PICK), так что могут с полным правом судить об этой ОС, как, впрочем, и о качестве упомянутых в рекламе средств четвертого поколения, а также возможностях языка ассемблера ОС Pick, языка Pick/Basic и т.д.

Мы уже было решили просто камерно высказать свое недоумение авторам письма, воспользовавшись электронной почтой, как вдруг "процесс пошел".

Один из наших известнейших журналистов из русскоязычного издания "PC Magazine" посетовал, что ему "прозрачно намекнули" (цитата) о тех благах в виде рекламы, которые получит его журнал при условии выхода в нем статьи о достижениях Pick Systems. (Журналист, понятное дело, отказался, поскольку сначала решил ознакомиться с сутью ОС Pick. К слову сказать, большинство наших коллег себя и читателей уважают и профессиональной репутацией дорожат.)

Дальше - больше. В газете "ComputerWorld-Moscow" (№31/1992) появилась вполне объемная реклама "системы, в которую вы вырастаете, но которую вы никогда не перерастаете", а затем (см. №36-37/1992) было опубликовано ее продолжение (так и написано - "часть II"!) с пометкой в конце "продолжение следует" (!). Выхода продолжения мы дожидаться не стали, поскольку впервые в 1992 г. фирма Pick Systems дала рекламу и в журнал нашего американского партнера M&T (DBMS, Nov 1992, Vol. 5, N 12). Стало ясно, что без заказной статьи решили обойтись, а приступили к прямой рекламной кампании. (Хотя, справедливости ради, нам хотелось бы отметить, что в одном из компьютерных периодических изданий нам попала заметка, посвященная банковским системам, в которой в отрыве от контекста упомянуто и ОС Pick.)

Нам очень хотелось бы сказать, что "Журнал д-ра Добба" явился одним из

катализаторов рекламной кампании фирмы Pick Systems, но мы боимся показаться нескромными. Несомненно одно: "Журнал д-ра Добба" сегодня улавливает тенденции грядущего завтра!

Видимо, такое же "шестое чувство" одолело и авторов приведенного выше письма, и они по зову интуиции включились в рекламную кампанию фирмы Pick Systems.

Наш журнал всегда идет навстречу своим читателям! Поэтому мы опубликовали это письмо на его страницах.

Что же до ОС Pick, то не будем беспокоиться - у фирмы Pick Systems достаточный потенциал для защиты своих завоеваний. Мы с симпатией следим за ее борьбой на рынке. Смогли же они сообразить, что следует отказаться от предложений ряда "лидеров движения Pick" и создали в России независимую команду. Нам приходилось сталкиваться с входящими в нее классными профессионалами, настоящими фанатиками (в самом хорошем смысле этого слова) ОС Pick. Кстати, это один из немногих, известных нам примеров, когда отечественные программисты действительно работают по заказу иностранной фирмы непосредственно на Запад. Они очень тепло восприняли нашу статью и при ее обсуждении, в отличие от авторов письма, обошлись без пересказа содержания рекламных материалов. Мы, в свою очередь, желаем своим коллегам настоящей творческой удачи.

О широкой экспансии на Восток, по нашим оценкам, фирме Pick Systems думать не приходится (в России достаточно хорошо профессионально подготовленных здравомыслящих людей) - ей бы сохранить свои завоевания на Западе, а то, что около большой рыбы всегда кормятся мелкие рыбешки, так это в порядке вещей...

Немногочисленных поклонников ОС Pick может утешить и тот факт, что фирма-производитель вовремя первой сделала шаги навстречу производителям ОС Unix. О деловой хватке руководства фирмы Pick Systems мы уже упоминали и в статье. Цель же статьи была крайне проста: она начинала серию материалов под условным названием "Малоизвестные и нетрадиционные архитектуры компьютеров и программного обеспечения". К этой же серии относится и статья, посвященная компьютерам

фирмы Ascom, которая была опубликована в третьем номере нашего журнала за 1992 год.

“Журнал д-ра Добба” - издание для профессионалов, и пишут в него, по нашему мнению, вполне квалифицированные люди. Мы не думаем, что их можно заставить изменить свое мнение по какому-либо вопросу огульными обвинениями в неинформированности, подкрепленными малосодержательными рекламными текстами.

В одном автору письма, безусловно, правы: на Западе, а особенно в США вообще очень любят создавать различные общества и объединения - все перечисленные в письме организации и издания, посвященные ОС Pick, “имеют место быть”. Да и то сказать, если даже какой-нибудь “Клуб домохозяек” в окраинном американском Хрюкино-Дятловом Ущелье может позволить себе выпускать свой информационный листок, то вполне солидной фирме Pick Systems иметь свое издание, а пользователям ее продукции - соответствующие объединения, сам Бог велел. Борьба за удержание места на рынке - дело серьезное!

М.Л. Сальников
Главный редактор

Уважаемая редакция!

С большим интересом знакомлюсь с каждым номером вашего журнала. Считаю, что среди периодических изданий подобной тематики ваше - одно из лучших. По моему мнению, привлечение к сотрудничеству высококвалифицированных специалистов и умелая подборка статей из иностранных журналов - главные составляющие этого успеха. (Попутно позвольте высказать следующее замечание. Вы почему-то перестали указывать фамилии переводчиков, которых, полагаю, читатели должны знать.)

В третьем и четвертом номерах журнала за 1991 г. опубликованы статьи Майкла Эйбраша (Michael Abrash) “Режим X: 256-цветовая магия адаптера VGA” и “256-цветовая магия адаптера VGA” (в дальнейшем именуемые “первая статья” и “вторая статья” соответственно). Разрешите мне в порядке комментария к ним сделать следующие замечания.

Процедура Set320x240Mode, описанная в первой статье, нормально работает

без сброса и запуска блока синхронизации (sequencer) адаптера VGA, т.е. совсем не обязательно дважды устанавливать регистр 0 этого блока. Кстати, об этом же пишут Джордж Сатти (George Sutti) и Стив Блэр (Steve Blair) на стр. 205 своей книги “Advanced Programmer’s Guide to the EGA/VGA” (Brady Books, 1988, ISBN 0-13-729039-X).

Неясно, для чего нужна довольно громоздкая предварительная установка регистра 11h контроллера ЭЛТ - CRTС (CRT Controller) адаптера VGA. Действительно, старший бит этого регистра, будучи установленным в 1, запрещает изменение регистров 0-7 CRTС. Но коль скоро мы все равно впоследствии устанавливаем регистр 11h, то достаточно всего лишь предварительно его обнулить. Тем самым мы решаем проблему корректного выполнения процедуры Set320x240Mode после ее возможного прерывания. Если эта процедура будет прервана в промежутке между записью в индексный регистр и чтением или записью в регистр данных CRTС и при этом прерывающая программа изменит содержимое индексного регистра CRTС, то выполнение процедуры после обработки прерывания может привести к непредсказуемым последствиям. Поэтому перед обращением к индексному регистру необходимо запретить, а после чтения или записи в регистр данных разрешить прерывания. Это же советует делать Ричард Уилтон (Richard Wilton) в статье “VGA Video Modes” (Byte, 1988, vol. 13, N11). Конечно, по этому поводу можно спорить, считая, например, что прерывающая программа не должна изменять регистры данных VGA. Но неужели ей нельзя позволить хотя бы читать эти регистры? Поэтому представляется целесообразным включение в соответствующие места программ М. Эйбраша инструкций CLI и STI.

Пользуясь случаем, хотел бы порекомендовать читателям, привыкшим к работе с BIOS, установить в процедуре Set320x240Mode двухбайтовую переменную BIOS, хранящую длину страницы, по адресу 0:44Ch. В данном случае длина страницы определяется как $320 \cdot 240 / 4$, т.е. она равна 19200 байтам.

Сделав такую установку, можно без опаски переключать страницы посредством функции 5 прерывания 10h BIOS,

в то время как М. Эйбраш в очередной статье для переключения страниц скорее всего будет использовать непосредственное обращение к портам адаптера VGA.

Позвольте обратить ваше внимание на то, что в текстах первой и второй статей имеются следующие опечатки и неясные места.

В первой статье при ссылке на рис.1 (с.41, 2-я колонка) упоминается режим 13h, хотя на самом деле имеется в виду режим X.

Во второй статье начало второго абзаца в 1-й колонке на с.33 должно выглядеть следующим образом: “В режиме X быстрое заполнение шаблоном шириной 4 пикселя...”.

Там же в преамбуле листинга 2 текст “void FillPatrnedX” необходимо заменить на “void FillPatternX”.

В листингах 4-6 первой статьи и 2-4 второй статьи в комментариях к параметрам процедур, обозначающим координаты правого нижнего угла, необходимо поменять местами слова “row” и “column”.

На рис.3 второй статьи текст “Смещение 65532” размещен у конца временного буфера для шаблона, а в действительности число 65532 - смещение начала этого буфера, т.е. этот текст должен быть расположен у начала буфера.

Из этого же рисунка можно сделать ошибочное предположение о том, что страницы 0 и 1 отображаются одновременно при каждом втором переключении страниц.

Заодно отмечу опечатки в статье того же М. Эйбраша “Оптимизация кода для микропроцессоров семейства 80x86”, опубликованной во втором номере “Журнала д-ра Добба” за 1991 г. Спецификатор выравнивания в ассемблере определяется как <граница выравнивания> PTR, например, BYTE PTR. В статье же всюду вместо PTR используется PRT. Кроме того, во фрагменте программы, приведенном на с.48, команду

```
LEA EBX,BaseTable[EDX+EBX],0
```

следует заменить на

```
LEA EBX,BaseTable[EDX+EBX]
```

Желаю вам дальнейших творческих успехов.

Н.Г. Краснокутский
г. Киев

TIMED CALLBACKS IN C++ by *Christian Stapfer*

Организация систем контроля и управления в реальном времени, обеспечивающая выстраивание функций в очереди и их вызов после отсчета соответствующего числа тактов системных часов.

IMPLEMENTING NLM-BASED CLIENT/SERVER ARCHITECTURES

by *Michael Day*

Использование загружаемых модулей операционной системы NetWare в качестве программного обеспечения серверов сетей.

SAFE PROGRAMMING WITH MODULA-3 by *Sam Harbison*

Обзор возможностей объектно-ориентированного языка программирования Modula-3, по мощности, надежности и гибкости сравнимого с языками Ada и C++, а по простоте превосходящего их.

A SOURCE CODE PROFILER by *Keith W. Boone*

Анализ работы профайлера (*profiler*) - отладочного средства для построения профиля программы.

D-FLAT, THE HOME STRETCH by *Al Stevens*

Описание работы диалоговых окон, метода сжатия базы данных справочника и других компонентов пакета D-Flat.

PARTS ISN'T PARTS by *Jeff Duntmann*

Все более усложняющиеся средства разработки программ, например Turbo Vision, заставляют разработчиков менять стратегию программирования.

HOW TO SHEAR A SHEEP, AND OTHER TEXTURE-MAPPING NICETIES

by *Michael Abrash*

Математические аспекты изображения многоугольников при трехмерной анимации.

SLAYING THE DRAGON by *Andrew Schulman*

Анализ книг, посвященных компиляторам языка C.

November 1992

SIZING UP GUI TOOLKITS by *Ray Valdes*

Вторая статья из серии, посвященной сравнительному анализу различных средств для разработки графических интерфейсов пользователя.

PIE MENUS FOR WINDOWS by *Carl Rollo*

Построение меню, имеющего вид круга, поделенного на секторы, для программ, работающих в среде Windows.

DYNAMIC DIALOG BOXES AND C++ by *Robert Sardis*

Создание диалоговых окон, параметры которых определяются на этапе выполнения, для программ, работающих в среде Windows.

GARBAGE COLLECTION FOR C PROGRAMS

by *Giuliano Carlini and Susan Rendina*

Описание работы "сборщика мусора", избавляющего программиста от необходимости явного освобождения памяти.

DESIGNING C++ CLASSES by *Steven Sinofsky*

Анализ работы функций-членов класса, автоматически создаваемых компилятором C++ в случае, если программист явно не определяет их при описании класса: конструктора, деструктора, оператора присваивания и конструктора копий.

DESIGNING PORTABLE USER INTERFACES by *John L. Bradberry*

Обсуждение проблем, возникающих при переносе программ с развитым графическим интерфейсом пользователя из среды DOS в среду UNIX.

CONVERTING DITHERED IMAGES BACK TO GRAY SCALE by *Allen Stenger*

Проблемы восстановления градаций серого цвета в изображениях, подвергшихся цифровому преобразованию полутонов.

DESIGNING A REAL-TIME DEBUGGER by *David Potter*

Описание отладчика, работающего как в среде iRMX, так и в среде Windows.

TIME DILATION AND RELATIVISTIC DEBUGGING by *Edward N. Adams III*

Требования к методике отображения информации при отладке и построении профиля программы.

DEBUGGING MOTIF WIDGETS by *Kamran Husain*

Методы отладки элементов графического интерфейса Motif.

EXAMINING TURBO PASCAL FOR WINDOWS by *Michael Floyd*

Автор тестирует пакет Turbo Pascal for Windows фирмы Borland с помощью программной среды ExpertWin, созданной им для экспертной оценки программ, работающих в среде Windows.

DEVELOPING A PORTABLE C++ GUI CLASS LIBRARY by *Andreas Meyer*

Создание библиотеки классов для разработки программ, обладающих развитым графическим интерфейсом и работающих в операционных средах Microsoft Windows, OS/2 Presentation Manager, Apple Macintosh, Open Look и OSF/Motif.

USER-INTERFACE WORRIES by *Michael Swain*

Защита авторских прав и другие проблемы рынка программных продуктов.

TOOLS USA AND D-FLAT++ OFF AND RUNNING by *Al Stevens*

Описывается обработка событий и сообщений в программах, использующих библиотеку классов пакета D-Flat++.

IMPLEMENTATION LICENSE by *Jeff Duntmann*

Рассуждения о принципах проектирования программ и способах реализации проектов.

THE GOOD, THE BAD, AND THE RUN-SLICED by *Michael Abrash*

Описание быстрого послыного (*run-length slice*) алгоритма изображения прямых линий.

UNDOCUMENTED WINDOWS by *Ray Duncan*

Хвалебная рецензия на книгу "Undocumented Windows", являющуюся, по мнению автора статьи, одной из первых попыток проникнуть в сущность и внутреннее устройство системы Windows глубже, чем это делается разработчиками SDK и DDK.

EXTENDING TURBO VISION by *Scott Nichol*

Автор дополняет пакет Turbo Vision фирмы Borland, вводя новый метод порождения событий, связанных с работой таймера.

December 1992

SPATIAL DATA AND THE VORONOI TESSELATION

by *Hrvoje Lukatela and John Russell*

Лавина информации, связанная с исследованиями в области географии, требует особых методов представления данных и доступа к ним.

SOUND AS A DATA TYPE by *Aaron E. Walsh*

Работа со звуковыми данными в системе QuickTime фирмы Apple.

PERSISTENT OBJECTS IN C++ by *Al Stevens*

Использование в программах на языке C++ сохраняемых объектов, т.е. объектов, состояние которых сохраняется после окончания сеанса работы с программой.

PERSISTENCE IN A PROGRAMMING ENVIRONMENT by *Richard P. Gabriel*

Решение проблемы сохраняемых объектов путем создания объектно-ориентированных баз данных, содержащих информацию об используемых объектах, при программировании на языках C и C++.

SPLAY TREES by *Dean Clark*

Описание самонастраивающихся двоичных деревьев, каждая операция над которыми облегчает доступ к наиболее часто используемым элементам.

SIMULATION AND TESTBOARD FOR EMBEDDED SYSTEM DESIGN

by *Michael Kutter*

Методика параллельной разработки программных и аппаратных компонентов встроженных контроллеров.

SIMULATING HYPERCUBES IN UNIX, PART I

by *Jeffrey W. Hamilton and Eileen M. Ormsby*

Первая статья из серии, посвященной эмуляции работы многопроцессорного комплекса для параллельного программирования (*hypercube*) с помощью стандартных средств ОС UNIX.

INSIDE THE ISO-9660 FILESYSTEM FORMAT

by *William Frederick Jolitz and Lynne Greer Jolitz*

Описание формата записи на компакт-дисках.

BLOBS AND OBJECT-ORIENTED DATABASE ENIGNS by *Sam Felton*

Использование объектно-ориентированных баз данных для хранения больших массивов изображений и аудиоинформации.

A CURMUDGERY ON PROGRAMMING LANGUAGE TRENDS by *Scott Guthery*

Одна из редких попыток критически взглянуть на объектно-ориентированное программирование, в частности использование языка C++.

CUA++ by *Al Stevens*

Продолжение знакомства с библиотекой пакета D-Flat++: описание классов Desktop, Screen и др.

TOSS IT IN THE CART! by *Jeff Duntmann*

Рассмотрение понятия "ресурсы" в пакете Turbo Vision.

MOVING, FASTER LINES, AND PAGE FLIPPING by *Michael Abrash*

Оптимизированная ассемблерная версия описанного ранее послыного метода изображения прямых линий, а также манипуляции со страницами видеопамати.

WAKE UP AND SMELL THE WORKING SET by *Andrew Schulman*

Рецензия на книгу Харольда Стоуна "Архитектура высокопроизводительных компьютеров" (Harold Stone "High-performance Computer Architecture").

December 1992, Special Supplement

A CONVERSATION WITH BJARNE STROUSTRUP by *Al Stevens*
Создатель языка C++ рассуждает о путях его развития.

STANDARD C++: A STATUS REPORT by *Dan Saks*
Эволюция языка C++ на пути к общепринятому стандарту, перспектива разработки стандартной библиотеки языка C++.

A C++ BEAUTIFIER by *Tim Maher*
Автор описывает свою методику приведения текстов программ на языке C++ к наиболее удобочитаемому виду. Методика заключается в использовании сначала препроцессора для перевода программ на язык C, а затем постпроцессора для обратного перевода на язык C++.

TEMPLATES IN C++ by *Nicholas Wilt*
Применение шаблонов для функций и классов позволяет сделать программу на языке C++ более компактной и эффективной и сводит к минимуму дублирование участков кода программы.

TOWARDS A LESS OBJECT-ORIENTED VIEW OF C++ by *Harris Shiffman*
Критический анализ языка C++ и его соответствия концепции объектно-ориентированного программирования.

WHY C++ WILL REPLACE FORTRAN by *Thomas Keffer*
Возможности языка C++ в области числового программирования чрезвычайно велики. На это, как правило, не обращают внимания из-за того, что другие объектно-ориентированные языки не столь пригодны для математических целей. Автор статьи полагает, что язык C++ может стать серьезной альтернативой языку Fortran в сфере числового программирования.

USING MULTIPLE INHERITANCE IN C++ by *Tom Cargill*
Принятое в объектно-ориентированных языках понятие множественного наследования часто вызывает затруднения у изучающих язык C++. Статья посвящена детальному рассмотрению понятия множественного наследования и тех случаев, когда его применение может оказаться выгодным.

IMPLEMENTING CURVES IN C++ by *Stephen P. Johnson and Tom McReynolds*
Методы изображения различных кривых на плоскости с использованием иерархии классов, производных от класса "кривая" языка C++.

LAN Technology September 1992

WHEN IS ENOUGH ENOUGH? by *W. D. Riley*
Поиск компромисса между возможностями и потребностями при выборе аппаратных средств.

BACK TO CENTRALIZATION by *Mark Freund*
Централизация управления сетями в ряде случаев позволяет достичь больших гибкости и удобства для пользователей.

THE STEPS TO NETWORK KNOW-HOW by *Jack Butler*
Программа фирмы Novell по подготовке квалифицированных пользователей сетей.

THE BIG SWEEP by *Emmet A. Dulaney*
Обсуждение способов наиболее эффективного использования пространства на магнитных носителях.

A PATH THROUGH THE SERVER FOREST by *Mark Hall*
Анализ возможностей различных программных и аппаратных средств применительно к организации сетей.

WINDOWS DRESSING by *Ethan Willansky*
Сравнительный анализ сетевых возможностей пакетов NetWare фирмы Hewlett-Packard и Norton Desktop for Windows фирмы Symantec.

MasterDat: DISTRIBUTED NETWORK BACKUP by *Richard T. Beerman*
Пакет MasterDat фирмы GigaTrend позволяет быстро создавать резервные копии данных большого объема.

October 1992

BULLIES: BETTER THAN WE EXPECT by *W. D. Riley*
Обсуждение проблемы выбора операционной среды.

WHAT ABOUT BOB? by *Leo S. Spiegel*
Сопоставление возможностей различных пакетов при организации управления сетями.

CAST YOUR NETWORK OS BALLOT by *Jim Cavanagh*
Выбор операционной среды при организации локальных сетей.

SURVEYING OS/2 NETWORKING OPTIONS by *Jim Gilliland*
Усовершенствованный интерфейс, многозадачность и интеграционные возможности делают среду OS/2 конкурентоспособной в области организации сетей.

CONTROL FROM THE CONSOLE by *Terry Treece*
Построение мощных систем управления сетями на базе простых протоколов управления сетью (SNMP).

THE SERVER AS NETWORK HUB by *Alan Frank*
Использование на сервере дополнительной платы адаптера концентратора сети позволяет сэкономить деньги и расширить функциональность сервера.

LAN MANAGER IN A MULTI-VENDOR WORLD by *Chris M. Culbertson*
Преимущества и недостатки сети с разнородными элементами, построенной с применением пакетов LAN Manager 2.1 фирмы Microsoft и LAN Manager/X 1.3 фирмы Hewlett-Packard.

October 15, 1992

MAPPING THE WIRING MAZE by *Russel Sanders*
Рациональное проектирование кабельной сети JIBC способно предотвратить многие проблемы.

THE CRITICAL STEPS TO HUB SELECTION by *Ed Barret*
Принципы выбора концентратора для сети.

November 1992

SCI-FI MANAGEMENT TOOLS TODAY by *W. D. Riley*
Появившаяся на рынке новая версия операционной системы NetWare 4.0 во многом удовлетворяет требованиям к идеальному средству для управления сетями.

EXPANDING BRAIN BANDWIDTH by *Leo S. Spiegel*
Автор вводит термин "информационный двигатель" (Information engine) для изложения своей концепции идеальной системы передачи и обработки информации.

SCANNING OPTICAL OPTIONS by *Wayne Robertson*
Использование оптических дисков как для однократной, так и для многократной записи позволяет решить проблему дешевого хранения больших объемов информации в сетях.

DECISION'92: THE NETWORK CHOICE by *Jim Cavanagh*
Завершающая статья из серии публикаций, посвященной выбору операционной среды для организации сети.

SHARE AND SHARE ALIKE by *Grace M. Carr*
Использование пакета Lotus Notes для работы с распределенными ресурсами.

PUT X TERMINALS IN YOUR GAME PLAN by *Jim Walker and Gary Kelley*
Включение X-терминалов в сеть, работающую под управлением ОС UNIX.

LANSTREAM: BACKUP FOR NTE MANAGERS AND USERS by *William Wong*
Система LANStream фирмы Maynard существенно упрощает резервирование используемой в сетях информации.

December 1992

WRONG AGAIN by *W. D. Riley*
Работа с электронной почтой на примере пакета cc:Mail.

NETWORK OF DREAMS by *Leo S. Spiegel*
Анализ пакета SFT III фирмы Novell, обеспечивающего защиту сетей от ошибок.

MANAGING THE HUBS, MANAGING THE NET by *Alex Castaldo and Sean Q. Flynn*
Детальный анализ и сопоставление характеристик пяти программных пакетов управления сетями.

FRAME RELAY: TECHNOLOGY FOR OUR TIME by *Robert W. Harbison*
Технология передачи блоков (frame relay), используемая в глобальных вычислительных сетях, находит все более широкое применение при организации локальных вычислительных сетей.

BUILD A SAFETY NETWORK by *Bruce Scheier*
Практические советы по обеспечению надежности и защищенности локальных сетей.

SIMPLE, SAFE SECURITY SOLUTIONS by *Andres Llana Jr.*
Несколько простых основополагающих принципов защиты при организации сетей (дополнение к предыдущей статье).

SHIVA SERVES REMOTE APPLES by *Peter Shank*
Устройство Lan Rover/L фирмы Shiva позволяет осуществлять удаленный доступ к сети AppleTalk.

DBMS September 1992

THE IMPORTANCE OF TIME by Robin Bloor

Автор излагает свое понимание философии, лежащей в основе работы с базами данных, уделяя особое внимание временному фактору этой работы.

TRANSACTION PROCESSING VS. DECISION SUPPORT by Robert C. Bolt

Трехуровневая иерархия при построении системы типа "клиент-сервер" позволяет обеспечить оптимальное по производительности сочетание двух типов обращений к базам данных: выборки данных и внесения изменений в базу данных.

HANDS ON: BUSINESS OBJECTS by Sean Kelly

Обзор новых продуктов на рынке программных средств: пакета BusinessObjects for Windows фирмы BusinessObjects, пакета Ocelot2-the SQL! фирмы Ocelot Computer Services и созданной фирмой Ensemble интегрированной среды Command Center для разработки программ в среде СУБД Paradox 3.5.

BUILDING AN APPLICATION DEVELOPMENT WORKBENCH

by Jonathan Syles

Первая статья из серии, посвященной интегрированным средствам разработки прикладных программ для работы с базами данных.

NEXT'S TOOLS FOR DATABASE DEVELOPERS by Gary Lang

Обзор характеристик пакета NextStep 3.0 фирмы Next, представляющего собой объектно-ориентированную интегрированную среду для разработки прикладных программ, работающих с базами данных на компьютерах фирмы Next.

USING STORED PROCEDURES AND TRIGGERS by Herb Edelstein

Использование сохраняемых процедур (stored procedures) и триггеров (triggers) в СУБД с архитектурой типа "клиент-сервер", позволяет улучшить производительность работы с базами данных и обеспечить сохранность данных.

AN INTRODUCTION TO CONCURRENCY CONTROL by Joe Celko

Принципы обеспечения надежности и защиты данных в многопользовательских системах с распределенными ресурсами.

DECODING AND PRINTING PCX IMAGES by Stephen Nebel

Описание графического формата PCX и методов вывода на лазерный принтер файлов, имеющих этот формат.

FOXPRO TEXTMERGE APPLICATIONS by P. L. Olympia

Автор делится своим опытом в написании прикладных программ, использующих возможности пакета FoxPro по объединению фрагментов текстов.

October 1992

WHY DEVELOPMENT PROJECTS FAIL by Robin Bloor

Рассуждения о причинах неудач в разработках больших программных проектов.

SPREADSHEET WINGZ by Robert C. Bolt

Автор делится своими впечатлениями от знакомства с пакетом Wingz фирмы Informix. Пакет включает в себя среду для работы с электронными таблицами, мощный макроязык HyperScript и средство доступа к базам данных - Datalink.

HANDS ON: POWER BUILDER 2.0 by Sean Kelly

В обзоре новых продуктов представлены описания пакетов: PowerBuilder 2.0 фирмы Powersoft, FoxPro2 Oracle фирмы Bitop, Watcom SQL for DOS фирмы Watcom и CA-Clipper/dBASE IV Kit - программы, распространяемой фирмой Computer Associates.

OBJECT DATABASES AT WORK by Larry P. English

Изложение концепции объектных СУБД и аналитический обзор существующих продуктов этого класса.

ARE YOU READY FOR ODBC? by R. Lynn Kernighan

Создаваемая фирмой Microsoft среда для разработки программ Open Database Connectivity (открытая система для обеспечения совместимости баз данных) призвана обеспечить пользователю универсальный доступ к данным в различных форматах.

DB2 DOES CLIENT/SERVER by David Herndon

Рассказ о созданном в шт. Кентукки проекте информационной системы для работников социальных служб. Система строится на основе СУБД DB2 в рамках архитектуры типа "клиент-сервер".

TWELVE STEPS FOR DESIGNING AND IMPLEMENTING AN ADW

by Jonathan Syles

Очерченная статья из серии, посвященной интегрированным средствам разработки прикладных программ для работы с базами данных, содержит описание 12 стадий разработки и внедрения таких средств.

SQL SERVER AND NETWORK by David McClanahan

Особенности установки сервера SQL Server 4.2 фирмы Microsoft в сети NetWare фирмы Novell.

ONLINE COMPLEX PROCESSING by David McGovern

Статья посвящена диалоговым информационным системам с комплексной обработкой данных.

INSIDE THE SQLBASE 5.0 by Steve Roti

Обзор характеристик сервера баз данных SQLBase 5.0 фирмы Gupta Technologies.

EXPERT SYSTEMS AND DATABASES by P. L. Olympia

Расширение возможностей баз данных с помощью экспертных систем на примере использования пакета VP-Expert 3.0 фирмы WordTech.

November 1992

IN RESPONSE TO DR. CODD by Robin Bloor

Автор отвечает на критику в адрес своей статьи о перспективах развития реляционных баз данных, опубликованной в июльском номере журнала "DBMS" за 1992 г.

ORACLE'S TRUMP by Robert C. Bolt

Описание пакета Oracle Card 1.1 фирмы Oracle, предназначенного для разработки прикладных программ и функционирующего как в среде Windows, так и на компьютерах Macintosh.

PREPARING YOUR LAN FOR CLIENT/SERVER by David McClanahan

Требования, предъявляемые к программным и аппаратным компонентам информационной системы, имеющей архитектуру "клиент-сервер".

THE SECRET LIFE OF 4GLs by Susan Perschke

Многие средства разработки программ с использованием языков четвертого поколения, уступая в известности лидерам мирового рынка, часто превосходят их по гибкости и мощности.

THE PARADOX CONNECTION TO SQL SERVER by William G. Ryan

Описание программного модуля SQL Link фирмы Borland, позволяющего СУБД Paradox той же фирмы получать доступ к данным, хранящимся на удаленных серверах СУБД, таких, как SQL Server фирмы Microsoft, Oracle 6.0 фирмы Oracle и др.

USING INGRES EVENT ALERTERS by Martin Rennackkamp

Описана работа механизма оповещения о событиях - компонента реляционных СУБД, сообщающего программе об изменениях в базе данных и о других событиях.

A 32-BIT R:BASE by Ken Lebeiko

СУБД R:Base 4.0 фирмы Microrim использует набор 32-разрядных команд процессоров 386/486 и целиком загружается в расширенную память. Это, а также расширенный набор утилит и встроенных возможностей делает СУБД R:Base удобной, гибкой и мощной.

INFORMIX-ONLINE 5.0 by Steve Roti

Представлена СУБД Informix-Online 5.0 фирмы Informix.

SOURCE CODE SECURITY by P. L. Olympia

Автор описывает выпущенный фирмой Off-World Technology декомпилятор программ, написанных для СУБД FoxBase и FoxPro, и рассуждает о проблемах защиты интеллектуальной собственности программистов и моральных аспектах использования декомпиляторов и дизассемблеров.

December 1992

TACTICS FOR DOWNSIZING by Robin Bloor

Проблемы, возникающие при переносе программных средств на новые аппаратные платформы.

C JYACC JAM by Robert C. Bolt

Описание выпущенного фирмой JYACC пакета JAM - среды для разработки программ.

MICROSOFT MAKES ITS MOVE by Maurice Frank

Появившийся на рынке пакет Access - средство для работы с базами данных в среде Windows, и новые версии СУБД FoxPro 2.5 для работы в средах DOS и Windows выдвигают фирму Microsoft в ряд крупнейших производителей программной продукции в данной сфере.

LOCKING AND REFERENTIAL INTEGRITY IN ORACLE

by Claudio Fratarcangeli

Обсуждение проблемы сохранности данных при многопользовательском режиме выполнения операций.

THE "INTELLIGENT" INGRES 6.4 by Steve Roti

Наличие в СУБД Ingres 6.4 фирмы Ingres системы управления базами знаний и системы управления объектами дает разработчикам повод называть свой продукт "интеллектуальной базой данных".



Лаборатория тестирования проводит независимое тестирование отечественных и зарубежных разработок в области информатики и вычислительной техники.

На тестирование принимаются
как программные,
так и аппаратные средства.

Передача разработки на тестирование оформляется специальным договором, предусматривающим, в частности, обязательство Лаборатории не передавать изделие для коммерческих или иных целей третьим лицам.

При любом результате тестирования Вы получите исчерпывающий отчет специалистов Лаборатории, а при положительном результате - диплом "Журнала Д-ра Добба" и помощь в продвижении Вашего изделия на рынке.

Адрес: 125190, Москва, А-190, а/я 240

Телефон: (095) 124-98-49

Факс: (095) 124-70-21

Электронная почта: @elbi.msk.su

**Вы избавитесь от многих хлопот,
если поручите издательству**



**отредактировать,
набрать на компьютере,
сверстать (изготовить оригинал-макет)
и выпустить в свет
любую печатную продукцию:
книги, каталоги, брошюры,
буклеты, бюллетени,
рекламные объявления,
приглашения,
визитные карточки и пр.**

**Издательство "ТРИАДА"
также решит для Вашей организации
проблему разработки фирменного стиля.**

**Адрес: 125190, Москва, А-190, а/я 240
Телефон: (095) 124-98-49
Факс: (095) 124-70-21
Электронная почта: @elbi.msk.su**