ХАКЕР
СПЕЦ
04|65|АПРЕЛЬ 2006
ЕЖЕМЕСЯЧНЫЙ
ТЕМАТИЧЕСКИЙ
КОМПЬЮТЕРНЫЙ
ЖУРНАЛ

СОВРЕМЕННЫЕ ПЛАТФОРМЫ,
ТЕХНОЛОГИИ И СРЕДСТВА РАЗРАБОТКИ

ПЕРЕДОВОЕ ПРОГРАММИРОВАНИЕ

НЕСТАНДАРТНЫЙ C++
.NET: ЧТО ТАКОЕ ХОРОШО И ЧТО ТАКОЕ ПЛОХО
РАЗРАБОТКА СОВРЕМЕННЫХ СЕТЕВЫХ ПРИЛОЖЕНИЙ
ЗАВОЮЕТ ЛИ JAVA МИР
ECLIPSE — ХИТРАЯ СРЕДА РАЗРАБОТКИ
ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ J2ME
DELPHI — СОВЕТ ПРОФЕССИОНАЛА

(game)land hi-fun media
9771609 102006 04>
publishing for enthusiasts

Попробуйте подписаться в редакции, позвоните нам.

(это удобнее, чем принято думать :)


SYNC


Лучшие цифровые камеры


Хакер


Хакер Спец


Железо


Страна Игр


PC Игры


CyberSport


Мобильные компьютеры


Total DVD


DVD Эксперт


Total Football


Onboard


Mountain Bike Action


Хулиган


Свой бизнес

- Для подписчиков в Москве курьерская доставка **БЕСПЛАТНО** в день выхода журнала
- Дешевле, чем в розницу
- Гарантия доставки и замены в случае потери
- Специальные предложения для подписчиков
- Первый номер подписки высылается по звонку вместе с заполненной квитанцией для оплаты

8-495-780-88-29 (для Москвы)
8-800-200-3-999 (для России)
ВСЕ ЗВОНКИ БЕСПЛАТНЫЕ
Мы работаем с 9 до 18 по рабочим дням

(game)land

Однажды, сидя всем редакционным составом в древнем шотландском замке, мы глубоко размышляли. Размышляли о прошлом, настоящем и будущем. Вот как оно получилось! Мы были октябрятами, пионерами, пили газировку из автоматов, на уроках труда обрабатывали драчовым напильником всяческие детали. В общем, мы стремились в будущее — узреть свет в конце тоннеля!

Мы вели себя хорошо, слушались маму, чтобы не огорчить дедушку Ленина. Мы учились так, как он завещал. На уроках истории читали про дяденьку Стаханова, про его успехи на угольном фронте и, конечно, хотели стать такими же, как он, когда вырастем. Правда, уже тогда многие из нас мыслили себя тружениками интеллектуального труда, такими как, например, Капица или Склифосовский. И что же? Вот сейчас мы, более-менее выросшие дети СССР, сидим в обеденном зале этого древнего мрачного замка и думаем уже о будущем. Мы обдумываем план номера — о передовом программировании! Кажется, он уже перед тобой. Будущее настало!

Dr.Klouniz

передовое программирование

СПЕЦ

ЕЖЕМЕСЯЧНЫЙ
ТЕМАТИЧЕСКИЙ
КОМПЬЮТЕРНЫЙ
ЖУРНАЛ
04(65) АПРЕЛЬ 2006

www.xakep.ru

Мнение редакции не всегда совпадает с мнением авторов.
Все материалы этого номера представляют собой лишь информацию к размышлению.
Редакция не несет ответственности за незаконные действия, совершенные с ее использованием, и возможный причиненный ущерб.
За перепечатку наших материалов без спроса — преследуем.

РЕДАКЦИЯ
Главный редактор
Николай «AvaLANche» Черепанов (avalanche@real.xakep.ru)
Выпускающие редакторы
Александр «Dr.Klouniz» Лозовский (alexander@real.xakep.ru)
Андрей Каролик (andrusha@real.xakep.ru)
CD/OFFTOPIC
Иван «SkyWriter» Касатенко (sky@real.xakep.ru)
Литературный редактор
Валентина Иванова (valy@real.xakep.ru)
Арт-директор
Иван Васин (vasin@real.xakep.ru)
Дизайнер
Наталья Жукова (zhukova@real.xakep.ru)
Иллюстратор
Анна Журко
Цветокорректор
Александр Киселев
Фотографы
Андрей Мохов
Иван Скориков

РЕКЛАМА
Директор по рекламе ИД (game)land
Игорь Пискунов (igor@gameland.ru)
Руководитель отдела рекламы цифровой группы
Ольга Басова (olga@gameland.ru)
Менеджеры отдела
Ольга Емельянцева (olgaeml@gameland.ru)
Евгения Горячева (goryacheva@gameland.ru)
Оксана Алехина (alekhina@gameland.ru)
Менеджер по работе с сетевыми РА, корпоративные продажи
Максим Григорьев (grigoriev@gameland.ru)
Трафик-менеджер
Марья Алексеева (alekseeva@gameland.ru)
тел.: (495) 935.70.34
факс: (495) 780.88.24

РАСПРОСТРАНЕНИЕ
Директор отдела дистрибуции и маркетинга
Владимир Смирнов (vladimir@gameland.ru)
Оптовое распространение
Андрей Степанов (andrey@gameland.ru)
Подписка
Алексей Попов (popov@gameland.ru)
тел.: (495) 935.70.34
факс: (495) 780.88.24

PUBLISHING
Издатель
Сергей Покровский (pokrovsky@gameland.ru)
Учредитель
ООО «Гейм Лэнд»
Директор
Дмитрий Агарунов (dmitri@gameland.ru)
Финансовый директор
Елена Дианова (dianova@gameland.ru)

ГОРЯЧАЯ ЛИНИЯ ПО ПОДПИСКЕ
тел.: 8 (800) 200.3.999
Бесплатно для звонящих из России

ДЛЯ ПИСЕМ
101000, Москва,
Главпочтамт, а/я 652, Хакер Спец
spec@real.xakep.ru
http://www.xakep.ru

Отпечатано в типографии «ScanWeb», Финляндия
Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций
ПИ № 77-12014 от 4 марта 2002 г.
Тираж 42 000 экземпляров.
Цена договорная.

СВЕТЛОЕ БУДУЩЕЕ

ПЕРЕДОВЫЕ ПЛАТФОРМЫ

ОРУДИЯ ПРОЛЕТАРИАТА

ЭКСПЕРТ НОМЕРА

ДЕНИС БАТРАНКОВ

ЗАНИМАЕТСЯ ПРОГРАММИРОВАНИЕМ 20 ЛЕТ. НАЧАЛ СВОЮ РАБОТУ НА ПЕРВЫХ СОВЕТСКИХ КОМПЬЮТЕРАХ БК0010, ДВК, ПЕРЕЖИЛ ЭРУ ЕС1046, СМ ЭВМ И ДОЖИЛ ДО СОВРЕМЕННЫХ КОМПЬЮТЕРОВ ФИРМЫ INTEL И SUN. В НАСТОЯЩЕЕ ВРЕМЯ РАЗРАБАТЫВАЕТ КЛИЕНТ-СЕРВЕРНЫЕ ПРОГРАММЫ НА MSVC С ИСПОЛЬЗОВАНИЕМ COM/ATL. ПИШЕТ ДРАЙВЕРЫ ПОД WINDOWS. ЗА ВРЕМЯ СВОЕЙ ПРАКТИКИ ОСВОИЛ ТРИ АССЕМБЛЕРА (DEC-СМ ЭВМ, ЕС1046, IBM PC) И ОДИН ДИЗАССЕМБЛЕР IDA, ЯЗЫКИ ВЫСОКОГО УРОВНЯ: PL-1, DELPHI, C++, НЕ СЧИТАЯ ФОКАЛА, BASIC, PERL И CLIPPER. КРОМЕ ТОГО, ЯВЛЯЕТСЯ ЭКСПЕРТОМ ПО КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ. АДМИНИСТРИРУЕТ СЕРВЕРЫ SOLARIS, FREEBSD И WINDOWS. ПИШЕТ СТАТЬИ В ИНТЕРНЕТЕ. В КОМПАНИИ «ИНФОРМЗАЩИТА» ЧИТАЕТ ЛЕКЦИИ ПО БЕЗОПАСНОСТИ КОРПОРАТИВНЫХ СЕТЕЙ. ИМЕЕТ СЕРТИФИКАТ CCNA

SPECIAL DELIVERY

offtopic

HARD

SOFT

CREW

STORY

ТОВАРИЩ!
ХВАТИТ ПРОШЛЫЙ ВЕК РАСТЯГИВАТЬ!
ПРОГРАММИРУЙ НА ЯЗЫКАХ, В СРЕДАХ НОВЫХ!
ВСТАВЬ НАШ ДИСК В CD-ROM —
СБРОСЬ КОСНОСТИ ОКОВЫ!

CD:

ECLIPSE

Eclipse 3.1.2
Subclipse 0.9.108
PHPEclipse 1.1.7
CDT 3.0.0
Java Runtime Environment 1.5.0.06

SPECIAL DELIVERY

JVCS 2.40
JVCL 3.20
Group Policy Management Console

ИСХОДНИКИ К СТАТЬЯМ

«Бесплатная отправка SMS!»
«Сеть для стахановцев»
Trivial AIM Messenger
Лечение .NET

СОФТ ОТ NONAME

1Click DVD Ripper 2.03
7Canaries 1.0 Professional
AceISO 2005 v2.0
Advanced Uninstaller Pro 2006 7.5
Flash Decompiler 2.6
Flash Player Pro v.2.8
FreePromote 1.7
Handy Scheduler v.3.1
IP Address Changer 1.0
Light Alloy 3.5 Build 5953
PasswordsPro v2.0.2.1
SmartWhois 4.1.191
Steganos Internet Anonym 2006 8.0.1
Teleport Pro 1.40
The Bat! 3.71.01 (Release)
eMule Xtreme 0.47a v5.0

+
ФЕВРАЛЬСКИЙ НОМЕР СПЕЦА
ОБНОВЛЕНИЯ WINDOWS ЗА МЕСЯЦ

ДЕННИС РИТЧИ

Сделал решающий шаг в эволюции UNIX. Он добавил типы данных и новый синтаксис в язык Кена Томпсона «B». Так появился новый язык «C» — фундамент переносимости операционной системы UNIX. В 1973 году Деннис Ритчи и Кен Томпсон переписали ядро системы UNIX на языке C. Соответственно, они отошли от общепринятого стандарта, при котором операционные системы писались исключительно на Ассемблере. Позже язык C стал широко использоваться во многих приложениях и системах, разработанных для компьютеров разных размеров и марок — от умещающихся в руках до суперкомпьютеров. Стал очень популярен благодаря многим остроумным решениям, сделавшим запись программы на C очень компактной. Кроме того, C накладывает на программиста не слишком много ограничений и допускает множество «трюков».

взамен миллиона вычислений

1949

Джон Моучли создал систему под названием Short Code — по сути, первый, пусть даже примитивный язык программирования высокого уровня. Программист записывал решаемую задачу в виде математических формул, а затем, используя таблицу перевода, символ за символом преобразовывал эти формулы в двухлитерные коды. Специальная программа превращала коды в двоичный машинный код.

1954

Одним из первых удачных компиляторов стал язык Фортран, разработанный фирмой IBM. Название языка образовано от «FORmulae TRANslation» («преобразование формул»). Фортран предназначался для решения вычислительных задач в математике, физике, инженерных расчетах, экономике и т.п.

1958

Модификация языка Фортран была названа Фортран II, она содержала понятие подпрограммы и общих переменных для обеспечения связи между сегментами.

1960

Появился язык программирования Алгол (расшифровывается как «Алгоритмичный язык»). Точнее, Алгол 60, более «современный», чем просто Алгол. Из него в свою очередь родился Алгол W, разработанный как учебное пособие для студентов.

1962

Появляется язык Фортран IV, наиболее употребительный и в наше время. Примерно в те же годы комиссия при Американской ассоциации стандартов (ASA) выработала два стандарта: языки Фортран и базисный (основной) Фортран (Basic FORTRAN).

1962

Дж. Маккарти создал другой старейший язык программирования — Лисп (List Information Symbol Processing), предназначенный, скорее, для работы со строками символов, а не с числами. Лисп открыл для программистов новую область деятель ности — «искусственный интеллект». Лисп успешно применяется и сейчас — в экспертных системах, системах аналитических вычислений и т.п.

1964

Появился Бейсик — язык обучения программированию. «Бейсик» — это общепринятый акроним от «Beginner's All-purpose Symbolic Insruction Code» (BASIC) — «многоцелевой символический обучающий код для начинающих». Так как изучить Бейсик легко и работа с ним проста, обычно программы на нем создавали быстрее, чем на Фортране.

1966

Рефал — один из языков, разработанных в России (СССР). Этот язык прост и удобен для описания манипуляций над произвольными текстовыми объектами. Рефал является языком функционального типа, в отличие от обычных операторных языков типа Алгола и Фортрана.

1970

Николаус Вирт создает знаменитый язык Pascal, названный в честь изобретателя вычислительного устройства Блеза Паскаля. Pascal был разработан как обучающий язык, демонстрирующий принципы алгоритмизации. Он получился удачным в плане возможности дальнейших расширений, но долгое время не пользовался популярностью среди разработчиков.

1972

Язык C был создан Денисом Ритчи на основе существующего интерпретатора Би. Язык был разработан специально для операционной системы Unix, в «изготовлении» которой Ритчи принимал непосредственное участие. Несмотря на принадлежность C к классу высокоуровневых языков, он содержит развитые низкоуровневые средства, и поэтому используется для написания ядер операционных систем и драйверов. К примеру, ядро и модули операционной системы Linux написаны на C.

1983

Появляется пакет расширения для Pascal от фирмы Borland — Turbo Pascal.

1983

Язык программирования C++ был разработан на основе языка C Бьярном Страуструпом (Bjarne Stroustrup). На первых порах (1980 год) язык носил условное название «C с классами», а в 1983 году Рик Массити придумал название «C++», что выразило происхождение от языка C. Язык C++ является расширением (надмножеством) C, поэтому программы, написанные на C, могут обрабатываться компилятором C++. Более того, в программах на C++ можно использовать тексты на C и обращаться к библиотечным функциям языка C.

1984

Вышло уже семь версий Turbo Pascal'а, после чего на смену ему пришел Delphi — среда программирования для ОС Windows.

2000

В этом году стал известен новый язык программирования, родившийся в недрах компании Microsoft, — C# (читается как «C sharp» или «Си шарп»). Он стал частью новой технологии Microsoft, названной .NET (читается как «Dot Net»). В рамках этой прогрессивной технологии предусмотрена единая среда выполнения программ (Common Language Runtime, CLR), написанных на разных языках программирования

революционные шутки aol

ПРИНАРОДНАЯ ПРЕПАРАЦИЯ НОВОЙ ВЕРСИИ ПРОТОКОЛА TOC

ВОТ МЫ И ДОЖДАЛИСЬ. ДОЖДАЛИСЬ ТОГО, ЧЕГО СТОИЛО ОЖИДАТЬ ВОТ УЖЕ МНОГО ЛЕТ. В ЯНВАРЕ 2006 ГОДА AOL ОФИЦИАЛЬНО ПРЕКРАТИЛ РАБОТУ TOC V.1, ВМЕСТО КОТОРОГО СЕЙЧАС РАБОТАЕТ НОВЫЙ ПРОТОКОЛ — TOC V.2 («С НОВЫМ ГОДОМ, РЕБЯТА») | WOLF D.A. AKA PAYHASH FROM (AOLHACKERS.RU)

TOC — ЭТО ОБЛЕГЧЕННЫЙ И ПРОСТОЙ ДЛЯ ПОНИМАНИЯ ПРОТОКОЛ, ПОЭТОМУ ПРИ ЕГО ИСПОЛЬЗОВАНИИ ЛЮБАЯ ПРОГРАММА БУДЕТ ПРОСТОЙ И ЛЕГКОЙ ПО ВЕСУ. ДЛЯ ЗЛОБНЫХ ПРОГРАММ, КОТОРЫМИ, КОНЕЧНО, МЫ С ТОБОЙ НЕ ЗАНИМАЕМСЯ, ЭТО ОЗНАЧАЕТ, ЧТО БРУТФОРСЕР ИЛИ ТРОЯН БУДУТ ОТЛИЧАТЬСЯ ЭФФЕКТИВНОСТЬЮ И МАЛОЗАМЕТНОСТЬЮ

Как мы уже знаем, TOC — упрощенный открытый протокол (в отличие от закрытого OSCAR), рассчитанный на third party. В этой статье мы принародно вскроем особенности второй версии протокола, ужасно удобного злобным хакерам. Кстати, по окончании вскрытия стало еще меньше объяснений тому, что именно побудило AOL к такому шагу, поистине необъяснимому. Протокол модифицировался несильно, зато дополнился и стал тяжелее, чем TOC v1.

В один прекрасный момент масса сторонних клиентов, поддерживающих протокол TOC/AIM, перестали работать (например, AIM-плагин в Miranda IM). Сотни хакеров, уберкодеров (в том числе мы) ринулись расшифровывать дампы официального клиента AIM от AOL, работающего с TOC-протоколом, причем уже с версией v2.

Пожалуй, начнем с авторизации. В принципе, ее механизм не изменился, разве что увеличился в размерах пакет. По статье из сентябрьского Спеца за 2005 год (#58) мы знаем, что в протоколе TOV v.1 существует пакет toc_signon, который как раз авторизует нас на сервере AIM (TOC). Была также функция, которая собирала пакет toc_signon. В функцию мы передавали три аргумента: указатель на буфер (куда будем собирать пакет), идентификатор AIM-пользователя (screenname) и пароль от идентификатора (password). Наша функция выглядела вот так:

```
/*
    Функция конструирует пакет, с помощью
которого мы будем проходить аутентификацию.
*/
static char *encode_toc_signon(char *buf,
const char *screenname, const char *password)
{
 char *sflap; char *data;
 data=(char *)malloc(256 * sizeof(char *));
 memset(data, 0, 256);
 buf=sflap=flap_begin(buf, TYPE_DATA);

 sprintf(data,
 "toc_signon %s %d %s %s %s \"%s\",
  AUTH_HOST, AUTH_PORT, screenname,
  roast_password(password), LANGUAGE,
 REVISION);

 buf=writes(buf, data, strlen(data));
 buf=writeb(buf, 0x00);
 flap_end(buf, sflap);
 free(data);
 return buf;
}
```

Картинка №1. Авторизация на сервере AIM при помощи протокола TOC v2

Посмотрим на дамп-пакет №1 (см. картинку №1), построенный нашей функцией (дамп снят программой snort) и, перевернув страницу, распотрошим наш пакет по байтам (см. таблицу №1).

разобрались? Итак, мы расчленили и пощупали каждый байт в пакете toc_signon протокола TOC v.1. А теперь самое главное ;). В версии протокола TOC v.2 пакета с таким названием нет! Тогда для чего мы его разбирали? Дело в том, что в протоколе TOC v.2 существует такой же пакет, только с другим названием и с другим полем REVISION. Посмотрим на дамп №2 пакета toc_signon, который был описан выше, но теперь в измененном виде протокола TOC v.2.
Что видим тут? В принципе, изменения не такие уж существенные. Только название пакета toc_signon переделано на toc2_login. Идем дальше. Сервер, порт (может быть любым), идентификатор и пароль остались неизменными (алгоритм кодирования пароля остался прежним). Также не тронуто поле «Язык». Второе, что изменилось, — поле REVISION. Оно стало более громоздким и более содержательным. Так как официального документа по TOC v.2. мы еще не видели (на момент сдачи номера), можно только догадываться, что означает это поле. Посмотрим на строку ниже:

```
"TIC:\$Revision: 1.1 $" 160 US "" "" 3 0 30303 -kentucky -utf8 31584384.
```

Представим ее в виде, показанном на таблице №2.

Итак, какие же изменении претерпит функция toc_signon, которая работала при TOC v.1? Думаю, стоит оставить прежнее название функции: нас интересует не то, как ее обозвали, а то, что она делает. Итак, сама функция:

```
#define REVISION "\"TIC:\\$Revision: 1.1 $\" 160 US \"\" \"\" 3 0 30303 -kentucky -utf8 31584384"

static char *encode_toc_signon(char *buf,
        const char *screenname, const char *password)
{
 char *sflap; char *data;
 data=(char *)malloc(256 * sizeof(char *));
 memset(data, 0, 256);
 buf=sflap=flap_begin(buf, TYPE_DATA);

/*
Here is TOC v1.0 but Blocked by AOL
sprintf(data, "toc_signon %s %d %s %s %s \"%s\"",AUTH_HOST, AUTH_PORT, screenname, roast_password(password), LANGUAGE, REVISION);
*/

// The TOC v2.0
sprintf(data, "toc2_login %s %d %s %s %s %s", AUTH_HOST, AUTH_PORT, screenname, roast_password(password), LANGUAGE, REVISION);

 buf=writes(buf, data, strlen(data));
 buf=writeb(buf, 0x00);
 flap_end(buf, sflap);
 free(data);
 return buf;
}
```

Ничего себе! Вот как просто ;). Узнав о том, что AOL сменил версию протокола, мы сразу же ринулись расшифровывать дампы. Думали, в протоколе произошли кардинальные изменения. Однако нам открылась картина, которая почти не отличается от старой, — наше удивление было безмерно :). Изменения были формальны и поверхностны: не так был страшен черт, как его малевали.

Если авторизация в базе AOL-сервера пройдет успешно, к нам прилетит пакет SIGN_ON:TOC2.0 (см. ХС #58). Кстати, ты легко убедишься в этом, если самостоятельно сделаешь дамп этого пакета любимым снифером (под Windows рекомендую Iris).

все ясно? Итак, по процессу авторизации нарисовалась четкая и понятная картина. Перейдем к следующему этапу — получение и передача текстовых сообщений в протоколе TOC v2.

Начнем с отправки сообщений. В принципе, здесь не изменилось ничего. Остался тот же пакет toc_send_im, и никаких ошибок в отправке сообщений не наблюдалось (имеется в виду протокол TOC). Вместо toc_send_im рекомендуется ставить toc2_send_im — это единственное, что нужно отметить.

Посмотрим дампы отправки сообщений двух протоколов (см. дампы №3, №4 и картинку №2).

Однако и здесь кардинальных изменений нет, почти все то же самое. Однако, поскольку препарирование протокола вслепую основано на изучении и разборе дампов, разберем поближе дамп исходящего сообщения.

Функция, которая собирала пакет для отправки сообщения, теперь приняла вот такой вид:

```
static unsigned char *encode_toc_send_im(char *buf, const char *remscreenname, char *mess)
{
 char *sflap, *message;
 message=(char *)malloc(128, sizeof(char *));
 memset(message, 0, 128);
 buf=sflap=flap_begin(buf, TYPE_DATA);
//TOC v.1.0
```

шифрование пароля

КСТАТИ, НАСТАЛО ВРЕМЯ ВСПОМНИТЬ ФУНКЦИЮ (СМ. СПЕЦ #58) ДЛЯ ШИФРОВАНИЯ ПАРОЛЯ ДЛЯ ПРОТОКОЛА TOC V.1:

```
static unsigned char *roast_password(const char *pass)
{
static unsigned char rp[256];
static char *roast = ROAST; //Где ROAST это "Tic/Toc".
int pos = 2;
int x;
strcpy(rp, "0x");
for (x = 0; (x < 150) && pass[x]; x++)
pos += sprintf(&rp[pos], "%02x", pass[x] ^ roast[x % strlen(roast)]);
rp[pos] = '\0';
return rp;
}
```

В ЭТОЙ ФУНКЦИИ НЕТ НИЧЕГО ОСОБЕННОГО, ТАК КАК В НЕЙ ПРОИСХОДИТ ОБЫЧНОЕ ПОБАЙТОВОЕ XOR'РИРОВАНИЕ.

В ОДИН ПРЕКРАСНЫЙ МОМЕНТ МАССА СТОРОННИХ КЛИЕНТОВ, ПОДДЕРЖИВАЮЩИХ ПРОТОКОЛ TOC/AIM, ПЕРЕСТАЛИ РАБОТАТЬ (НАПРИМЕР, AIM-ПЛАГИН В MIRANDA IM)

дамп пакета №1 — авторизация на сервере TOC (протокол v1)

```
2A 02 00 02 00 50 74 6F 63 5F 73 69 67 6E 6F 6E  *....Ptoc_signon
20 6C 6F 67 69 6E 2E 6F 73 63 61 72 2E 61 6F 6C  login.oscar.aol
2E 63 6F 6D 20 35 31 39 30 20 3x 3x 3x 3x 3x 3x  .com 5190 xxxxxx
20 30 78 31 63 3x 64 3x 3x 3x 66 3x 3x 3x 3x 3x  0x1cxdxxxfxxxxx
64 30 65 20 65 6E 67 6C 69 73 68 20 22 4D 69 72  d0e english "Mir
61 6E 64 61 22 00                                anda".
```

Таблица №1.Побайтовая расшифровка Дампа №1

байты	пояснение
BYTE: [2A]	Символ «звездочка», который говорит клиенту, что это пакет AIM
BYTE: [02]	Канал сообщений (см. #58 Спец)
WORD:[00 02]	Номер пакета (SEQUENSID)
WORD:[00 50]	Длина пакета (в данном случае toc_signon 80 байт без учета FLAP-заголовка, который имеет размер 6 байт)
STRING:[10 bytes]	Название пакета toc_signon (packet name).
BYTE: [20]	Обычный пробел в HEX
STRING:[19 bytes]	Сервер авторизации, на который сервер TOC должен будет передать uin и пароль. В данном случае это **login.oscar.aol.com**
BYTE: [20]	Пробел ;)
DWORD: [35 31 39 30]	Порт сервера **login.oscar.aol.com** (прошу не путать с типом word 16 bit структуры [sock-addr_in].sin_port=htons(STRING); так как это сюда не относится)
BYTE: [20]	Пробел
STRING: [3x 3x 3x 3x 3x 3x]	Screenname/UIN. Идентификатор ICQ/AIM
BYTE:[20]	Пробел
STRING:[30 78 31 63 3x 64 3x 3x 3x 66 3x 3x 3x 3x 3x 64 30 65]	Зашифрованный пароль от твоего ICQ/AIM-идентификатора (см. на врезке)
BYTE: [20]	Пробел
STRING:[65 6E 67 6C 69 73 68]	Язык, в нашем случае English
BYTE: [20]	Пробел
BYTE: [22]	Двойные кавычки
STRING: [4D 69 72 61 6E 64 61]	Ревизия (REVISION) протокола в TOC v.1. могло быть любое значение, в нашем случае это название клиента Miranda
BYTE: [22]	Двойные кавычки

дамп пакета №2 — авторизация на сервере TOC (протокол v2)

```
2A 02 EE CA 00 8F 74 6F 63 32 5F 6C 6F 67 69 6E  *.....toc2_login
20 6C 6F 67 69 6E 2E 6F 73 63 61 72 2E 61 6F 6C  login.oscar.aol
2E 63 6F 6D 20 32 39 39 39 39 20 39 36 33 36 33  .com 29999 96363
32 20 30 78 31 63 35 64 35 35 37 66 36 36 32 37  2 0x1c5d557f6627
32 64 30 65 20 45 6E 67 6C 69 73 68 20 22 54 49  2d0e English "TI
43 3A 5C 24 52 65 76 69 73 69 6F 6E 3A 20 31 2E  C:\$Revision: 1.
31 20 24 22 20 31 36 30 20 55 53 20 22 22 20 22  1 $" 160 US "" "
22 20 33 20 30 20 33 30 33 30 33 20 2D 6B 65 6E  " 3 0 30303 -ken
74 75 63 6B 79 20 2D 75 74 66 38 20 33 31 35 38  tucky -utf8 3158
34 33 38 34 00                                   4384.
```

Таблица №2. Разбор строки REVISION "TIC:\$Revision: 1.1 $" 160 US "" "" 3 0 30303 -kentucky -utf8 31584384.

байты	пояснение
"TIC:\$Revision: 1.1 $"	Скорее всего, это версия и номер ревизии
160 US	Возможно, это географическое место локации
"" ""	Загадочные поля. Возможно, они просто зарезервированы
3 0 30303	Скорее всего, какой-нибудь zip-код, хотя пока не уточнено
-kentucky	Уточнение по географическому месту локации, какой-нибудь штат Кентукки
-utf8 31584384	Ну это и так понятно — используемая кодировка на клиенте

дамп пакета №3 — отправка текстового сообщения (протокол TOC v1)

TOC v.1.0:

```
2A 02 00 09 00 21 74 6F 63 5F 73 65 6E 64 5F 69  *....!toc_send_i
6D 20 3x 3x 3x 3x 3x 3x 20 22 48 65 6C 6C 6F 20  m xxxxxx "Hello
62 72 6F 21 0A 22 00                             bro!.".
```

Таблица №3. Побайтовая расшифровка Дампа №3

байты	пояснение
6 BYTES [2A 02 00 09 00 21]:	Эти поля ты уже знаешь!
STRING: [74 6F 63 5F 73 65 6E 64 5F 69 6D]	Имя пакета, в данном случае toc_send_im
BYTE: [20]	Пробел
STRING:[3x 3x 3x 3x 3x 3x]	AIM/ICQ-идентификатор получателя текстового сообщения
BYTE: [20]	Пробел
BYTE: [22]	Двойные кавычки
STRING:[48 65 6C 6C 6F 20 62 72 6F 21 0A]	Текстовое сообщение Hello bro!
BYTE: [22]	Двойные кавычки
BYTE: [00]	Завершающий пакет байт 0

дамп пакета №4 — отправка текстового сообщения (протокол TOC v2)

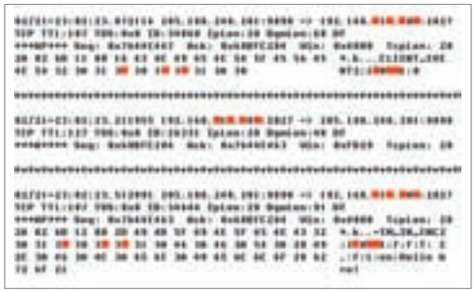
TOC v.2.0:

```
2A 02 00 09 00 21 74 6F 63 32 5F 73 65 6E 64 5F  *.P..!toc2_send_
69 6D 20 3x 3x 3x 3x 3x 3x 20 22 48 65 6C 6C 6F  im xxxxxx "Hello
20 62 72 6F 21 22 00                             bro!".
```

дамп пакета №5 — пакет входящего сообщения TOC v2

```
2A 02 88 82 00 2D 49 4D 5F 49 4E 5F 45 4E 43 32  *....-IM_IN_ENC2
3A 3x 3x 3x 3x 3x 3x 3A 46 3A 46 3A 54 3A 20 49  :100721:F:F:T: I
2C 3A 46 3A 4C 3A 65 6E 3A 68 65 6C 6C 6F 20 62  ,:F:L:en:hello b
72 6F 21                                         ro!
```

Картинка №2. Отправка текстового сообщения на сервер AIM при помощи протокола TOC v2



Картинка №3. Получение текстового сообщения с сервера AIM при помощи протокола TOC v2

```
 //sprintf(message, "toc_send_im %s \"%s\"",
remscreenname, mess);
//TOC v.2.0
 sprintf(message, "toc2_send_im %s \"%s\"",
remscreenname, mess);
 buf=writes(buf, message, strlen(message));
 buf=writeb(buf, 0x00);
 flap_end(buf, sflap);
 free(message);
 return buf;
}
```

Что такое входящее сообщение? Это частный случай исходящего сообщения :), только относительно сервера. Почему именно так? Допустим, мы сформировали пакет toc2_send_im, отправили его с исходящим сообщением на сервер TOC/AIM. Сервер TOC/AIM, получив такое сообщение, конвертирует его в пакет IM_IN_ENC2 — исходящего сообщения для сервера (он же — пакет входящего сообщения для клиента).

Задача нашей AIM/TOC/ICQ-программы должна заключаться в том, чтобы суметь разобрать пакет IM_IN_ENC2. Под занавес глянь, каким образом на дампе №5 (и картинке №3) выглядит пакет IM_IN_ENC2 входящего сообщения.

Картинка №4. AIM.DLL — внутренности известного интернет-пейджера Miranda IM

ОСОБАЯ БЛАГОДАРНОСТЬ ЛОЗОВСКОМУ АЛЕКСАНДРУ AKA DR.KLOUNIZ, ШАКИРОВУ АЛЕКСЕЮ AKA TYPUCT, K.SUN И VKE ЗА ИХ НЕОЦЕНИМУЮ ПОМОЩЬ…

шутки старых ICQ-хакеров

ПОМНИТСЯ, В НЕДАЛЕКОМ 2004 ГОДУ МЫ ВМЕСТЕ С ТУРИСТОМ ИЗ C4 TEAM РЕШИЛИ ПОДНЯТЬ 6D NUMERIC С НУЛЕВЫМ ПАРОЛЕМ, КОТОРЫЕ, ПО ЕГО СЛОВАМ В 1998 ГОДУ (ICQ ТОГДА БЫЛ У MIRABILIS) ПОЛЬЗОВАТЕЛИ РЕГАЛИ ВООБЩЕ БЕЗ ПАРОЛЯ, ТО ЕСТЬ ПРИ РЕГИСТРАЦИИ УНИКАЛЬНОГО НОМЕРА ПОЛЬЗОВАТЕЛЬ ОСТАВЛЯЛ ПУСТЫМ ПОЛЕ ПАРОЛЯ. ОБЪЯСНЯЮ ПОЧЕМУ. ПЕРВЫЕ ВЕРСИИ ICQ-ПРОТОКОЛА ДОПУСКАЛИ БЕСПАРОЛЬНЫЕ НОМЕРА И БЫЛИ ОЧЕНЬ ДЫРЯВЫМИ.

СООТВЕТСТВЕННО, КОГДА ШЛА ПЕРЕДАЧА ПРАВ НА ICQ, ДЕЛАЛСЯ ДАМП ОСНОВНОЙ БАЗЫ НОМЕРКОВ, И ТАКИЕ БЕСПАРОЛЬНЫЕ НОМЕРА ОСТАЛИСЬ (КОНЕЧНО, ЕСЛИ ПОЛЬЗОВАТЕЛЬ НЕ ДЕЛАЛ ПРИВЯЗКУ НОМЕРА К ПРИМАКУ) В НОВОЙ БАЗЕ AIM/ICQ. ТАК ВОТ, ОДНИМ ИЗ ОСНОВНЫХ ПРАВИЛ НЫНЕШНЕГО ПРОТОКОЛА OSCAR (AIM/ICQ) БЫЛО «ПОЛЕ С ПАРОЛЕМ НЕ МОЖЕТ БЫТЬ ПУСТЫМ!» К НАМ, ПРОСТЫМ ПАРНЯМ, ТАКИЕ ПРАВИЛА, КОНЕЧНО, НЕ ОТНОСИЛИСЬ ;).. ПОЧЕМУ БЫ И НЕ ПОПРОБОВАТЬ ПОДДЕЛАТЬ ПУСТОЕ ПОЛЕ С ПАРОЛЕМ В ПАКЕТЕ LOGIN И НУЛЕВОЕ ЗНАЧЕНИЕ ДЛИНЫ ПАРОЛЯ В ПОЛЕ TLV-КОНТЕЙНЕРА?

ICQ-номер с пустым паролем

В КАЧЕСТВЕ ЦЕЛИ МЫ ВЫБРАЛИ ОДИН ИЗ ТЕСТОВЫХ AOL-СЕРВЕРОВ, НА КОТОРОМ КРУТИЛАСЬ ПАРА СЕРВИСОВ AIM (ОНИ КАК РАЗ И ОТОБРАЖАЛИ WARN-СОСТОЯНИЕ КЛИЕНТА ICQ, НО ЭТО УЖЕ СОВСЕМ ДРУГАЯ ИСТОРИЯ ПРО WARN-БОТА И ДАВНО УШЕДШУЮ В ЛЕТУ КОМАНДУ TERMINAL ENVASION). ТАК ЖЕ, КАК И БОЛЬШИНСТВО ДРУГИХ СЕРВЕРОВ ОТ AOL, ЭТОТ СЕРВЕР РАБОТАЛ НА ПРИЕМ ВХОДЯЩИХ СООБЩЕНИЙ НА АВТОРИЗАЦИЮ AIM. НАМ ПРЕДСТОЯЛО ПОДДЕЛАТЬ AIM-СЕГМЕНТ С АВТОРИЗАЦИЕЙ, ГДЕ ПОЛЕ С ПАРОЛЕМ БЫЛО ПУСТЫМ, А ПОЛЕ С РАЗМЕРОМ ПАРОЛЯ РАВНЯЛОСЬ НУЛЮ. ЕСЛИ БЫ, КАК ОЖИДАЛОСЬ, СО СТОРОНЫ СЕРВЕРА ПРИШЕЛ ПОЛОЖИТЕЛЬНЫЙ ОТВЕТ С COOKIE, ЭТО ОЗНАЧАЛО БЫ UIN AND PASSWORD ALL READY SUCCESS. МОЖНО БЫЛО СМЕЛО ПОДНИМАТЬ ТАКИЕ ВОЛШЕБНЫЕ НОМЕРКИ.

Wolf D.A. aka Payhash

ПРАВДА, В РЕЗУЛЬТАТЕ ВСЕ ПОЛУЧИЛОСЬ НЕ ОЧЕНЬ КРУТО. МЫ ПОЛУЧИЛИ ОТРИЦАТЕЛЬНЫЙ РЕЗУЛЬТАТ С СЕРВЕРА И С ИРОНИЧЕСКОЙ УЛЫБКОЙ НА ЛИЦЕ ПРИНЯЛИСЬ СОВЕРШАТЬ БЕЗНАДЕЖНЫЕ ДЕЙСТВИЯ (ЗАДАНИЕ НЕФИКСИРОВАННОЙ ДЛИНЫ ПОЛЕЙ С ПАРОЛЕМ И ЕГО ДЛИНОЙ, МЕНЯЛАСЬ ИНТЕНСИВНОСТЬ ПОДКИДЫВАНИЯ ПАКЕТОВ С АВТОРИЗАЦИЕЙ. В ОБЩЕМ, ТАКИЕ ИГРЫ И ЭКСПЕРИМЕНТЫ ПРОДОЛЖАЛИСЬ В ТЕЧЕНИЕ ТРЕХ С ПОЛОВИНОЙ ЧАСОВ, ПОСЛЕ ЧЕГО СЕРВИС AIM ПАРУ ДНЕЙ ВООБЩЕ НЕ ПРИНИМАЛ НИКАКИХ КЛИЕНТОВ). ПОЛУЧИЛОСЬ ТАК, ЧТО В КАКОЙ-ТО КВАНТ ВРЕМЕНИ НА СЕРВИСЕ ПРОИЗОШЛО ПЕРЕПОЛНЕНИЕ ТЕСТОВОГО СЕРВЕРА, ДАЛЬШЕ ПО НЕПОНЯТНЫМ ПРИЧИНАМ СЛУЖБА AIM УЛЕТЕЛА В КЛОЗЕТ. ПО ПОНЯТНЫМ ПРИЧИНАМ, В ЦЕЛЯХ СОБСТВЕННОЙ ЖЕ БЕЗОПАСНОСТИ, Я НЕ СТАЛ ЗАНИМАТЬСЯ ПОИСКОМ УЯЗВИМОСТИ НА СЕРВИСЕ AIM (НА ПРЕДМЕТ ПЕРЕПОЛНЕНИЯ В БУФЕРЕ) — ШУТКИ С AOL НЕ ПРИВОДЯТ НИ К ЧЕМУ ХОРОШЕМУ.

МОРАЛЬ СЕЙ БАСНИ ТАКОВА ;): ICQ/AIM И ПО СЕЙ ДЕНЬ ОСТАЮТСЯ ВЕСЬМА ДЫРЯВЫМИ СЛУЖБАМИ В AOL. НОМЕРКИ С НУЛЕВЫМИ ПАРОЛЯМИ ТАК И ОСТАЮТСЯ В БАЗЕ AIM ;). МЫ С ТУРИСТОМ УЖЕ СЛИШКОМ ПЬЯНЫ И СТАРЫ, ЧТОБЫ ЗАНИМАТЬСЯ ПОДОБНЫМИ ВЕЩАМИ, ПОЭТОМУ ПРЕДОСТАВЛЯЕМ ЗЕЛЕНУЮ УЛИЦУ СВЕЖИМ УМАМ :).

сети для стахановцев

РАЗРАБОТКА СОВРЕМЕННЫХ СЕТЕВЫХ ПРИЛОЖЕНИЙ

ФОРМАТ XML, ПОЯВИВШИЙСЯ НА СТЫКЕ ТЫСЯЧЕЛЕТИЙ, ПОСЛУЖИЛ ТОЛЧКОМ ДЛЯ НОВОГО ЭТАПА ЭВОЛЮЦИИ В СФЕРЕ СЕТЕВЫХ ПРИЛОЖЕНИЙ. СЕГОДНЯ ИСПОЛЬЗОВАНИЕ XML — ЭТО НЕ ДАНЬ МОДЕ И НЕ БЕЗДУМНОЕ СЛЕДОВАНИЕ СОВРЕМЕННЫМ ТЕНДЕНЦИЯМ, А ТЩАТЕЛЬНО ПРОДУМАННЫЙ БРАК ПО РАСЧЕТУ |

ПАЛАГИН АНТОН АКА TONY (TONY@EYKONTECH.COM)

работа с документами xml в языках программирования реализуется с помощью специальных компонентов — парсеров XML. Основная задача парсера — это умение создавать документы и предоставлять клиенту возможность навигации по элементам дерева документа (узлам и атрибутам). Сегодня существует множество парсеров для любого языка программирования, следовательно, XML-документы стали отличным механизмом для реализации обмена формализованными данными, независимого от языка и платформы. Например, для C++ можно вспомнить с десяток различных парсеров.

Посмотрим поближе на три из них: msxml, xerces и libxml2. Первое — продукт творчества компании Microsoft, что ясно по названию, и хорошая мощная библиотека, предоставляющая DOM-интерфейс для работы с XML-документами, но реализованная, по большому счету, только для одной платформы. Методами дедукции или индукции можно легко установить, для какой именно.

Xerces — это один из проектов Apache, он предоставляет и DOM-, и SAX-интерфейсы. Существует для языков C++, Java и Perl на широком спектре платформ. Весьма надежен и стабилен, но испорчен двумя существенными недостатками: рыхлый клиентский код и медленный парсер.

Наконец, libxml2, как и Xerces, распространен на всех основных платформах и является частью проекта Gnome. Предоставляет очень удобный и простой в использовании интерфейс, клиентский код получается очень компактным и лаконичным. Простота и производительность этой библиотеки иногда перестает радовать из-за наличия мелких багов, правда, не мешающих разрабатывать с помощью libxml2 серьезные проекты.

xml настолько удобен как формат представления данных, что он пришел на смену бинарному обмену между распределенными приложениями. Разберемся почему. Все дело в сложности кода анализа данных и открытости стандарта. Парсить бинарные форматы данных на порядок сложнее, чем XML с помощью специальных библиотек, выполняющих за тебя всю рутинную работу. Расплатой становится избыточность данных, впрочем, современные каналы связи позволяют не обращать на это особое внимание. Упрощение кода для анализа данных позволяет создавать более сложные распределенные системы, что мы и наблюдаем в наши дни. В web-службах, в .NET Remoting и Indigo для обмена данными между клиентом и сервером используются документы XML, основанные на спецификации SOAP (Simple Object Access Protocol).

Последняя версия (1.2) протокола SOAP датируется 24 июня 2003 года. Этот протокол описывает обмен SOAP-сообщениями, содержащими произвольную информацию, между отправителем и получателем. Важно, что SOAP предоставляет не готовые решения, а инструменты, с помощью

mono

МОНО РАЗРАБАТЫВАЕТСЯ ДЛЯ СЛЕДУЮЩИХ ПЛАТФОРМ: WINDOWS, LINUX, MAC OS X, SOLARIS 8. ВОЗМОЖНО, К КОНЦУ 2006 ГОДА МЫ БУДЕМ ИМЕТЬ ДЕЙСТВИТЕЛЬНО МОЩНУЮ И В ТО ЖЕ ВРЕМЯ ПРОСТУЮ КРОСС-ПЛАТФОРМЕННУЮ ТЕХНОЛОГИЮ ДЛЯ СОЗДАНИЯ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ. ПРОЕКТ MONO СПОНСИРУЕТСЯ КОМПАНИЕЙ NOVELL. ТЫ ТОЖЕ МОЖЕШЬ ТАК ИЛИ ИНАЧЕ ВНЕСТИ СВОЮ ЛЕПТУ.

www.w3.org/2002/07/soap-translation/russian/part0.html — SOAP Версия 1.2 Часть 0: Учебник для начинающих
www.citforum.ru/internet/xml/xml_rpc — XML-RPC: вызов процедур посредством XML
http://ws.apache.org/xmlrpc — About Apache XML-RPC
http://xmlrpc-c.sourceforge.net —A lightweight RPC library based on XML and HTTP.

Проекты Apache, связанные с XML

которых разработчики строят собственные приложения. Облегченная версия SOAP (она называется XML-RPC) пришла на смену бинарному RPC-обмену (Remote Procedure Call), который использовался как в чистом виде, так и под оберткой DCOM и CORBA.

в далекие от нас времена разработчик, чтобы создать распределенное приложение, был вынужден возиться с изучением громадного талмуда и загадочных аббревиатур в нем. Теперь достаточно прочитать текст этой статьи — и ты уже подготовлен к созданию своего первого приложения с использованием технологии .NET Remoting. Итак, напишем простейший сервер, который выполняет простой тестовый метод для тестового объекта. Как минимум, для работы нам понадобится .NET Framework 1.1, а еще лучше — среда разработки MS Visual Studio 2003 или 2005.

```
using System;
using System.Runtime.Remoting;

namespace SimpleServer
{
  class Program
  {
    [STAThread]
    static void Main(string[] args)
    {
      //Единственное отличие от обычного
консольного приложения .NET
      RemotingConfiguration.Configure
("SimpleServer.exe.config");
      //Завершим работу сервера только
в случае нажатия любой кнопки
      Console.WriteLine("Press any key...");
      Console.ReadLine();
    }
  }
```

Этот сервер ничего не делает, только конфигурирует .NET Remoting указанным файлом настроек. Вот он:

```
<configuration>
  <system.runtime.remoting>
    <application name="Hello remoting">
      <service>
        <wellknown
        mode="SingleCall"
        type="TestObject.Test, TestObject"
         objectUri="Test.rem" />
      </service>
      <channels>
        <channel ref="http" port="8000" />
      </channels>
    </application>
  </system.runtime.remoting>
</configuration>
```

Формат конфигурационного файла детально и весьма душевно описан в MSDN, а мы сейчас остановимся только на ключевых моментах. Узел wellknown описывает, что именно будет делать сервер, когда к нему придет запрос от клиента. Значение атрибута mode может быть равно SingleCall, и в этом случае для каждого клиентского обращения на сервере будет создаваться новый объект. Атрибут mode может быть равен Singleton, и тогда на сервере создастся единственный экземпляр объекта, к которому будут обращаться все клиенты.

Атрибут type указывает имена объекта и вызываемого метода, а также имя сборки, в которой содержится объект. Обрати внимание на узел channel. Здесь в атрибуте ref указывается, каким образом будет происходить обмен: http (с помощью протокола http) или tcp (обмен напрямую через TCP/IP). А в атрибуте port указывается порт, по которому происходит обмен. Чтобы реализовать приложения, которые взаимодействуют по общедоступным каналам, будет разумно использовать пропускаемые межсетевыми экранами порты, например 8080.

.net remoting инициализирует код клиента аналогично серверному коду, после чего он инстанцирует объект обмена и вызывает тестовый метод.

```
using System;
using System.Runtime.Remoting;
using TestObject;

namespace Client
{
  class Client
  {
    [STAThread]
    static void Main(string[] args)
    {
      //Инициализируем .NET Remoting
      RemotingConfiguration.Configure
("SimpleClient.exe.config");
      //Инстанцируем тестовый объект
      Test test = new Test();
      //Вызываем тестовый объект и выводим
на экран то, что он вернул
      Console.WriteLine(test.hello());
    }
  }
}
```

Для того чтобы клиент «знал» о тестовом объекте, необходимо сослаться на сборку TestObject. Выбор вызываемого объекта осуществляется с помощью сценария конфигурации SimpleClient.exe.config. Вот он:

```
<configuration>
  <system.runtime.remoting>
    <application>
      <client>
        <wellknown
        type="TestObject.hello, TestObject"
        url="http://192.168.10.87:8000/Remo-
tingTest/Test.rem" />
      </client>
```

```
    </application>
  </system.runtime.remoting>
</configuration>
```

Ключевым для нас является узел wellknown. Здесь указывается, какой именно объект и его метод будут вызываться (атрибут type). Также важна ссылка на сервер в атрибуте url, здесь можно указать способ общения (http или tcp). Завершающим штрихом к этой картине служит код сборки, которая содержит реализацию тестового объекта. Для того чтобы мы могли обращаться к объекту удаленно по ссылке, необходимо пронаследовать его от класса MarshalByRefObject.

```
using System;

namespace TestObject
{
  public class Test: MarshalByRefObject
  {
    public string hello()
    {
      return "Hello world!!!";
    }
  }
}
```

откомпилируем код. В результате должно получиться два консольных приложения (клиент и сервер), а также сборка, хранящая тестовый объект. Запустим сервер, запустим клиент… Здравствуй, мир — в результате наш сервер обменяется с клиентов гигантской кучей пакетов. Если отбросить в сторону избыточные данные протокола TCP/IP, то останутся HTTP-запросы и SOAP-сообщения. Серверу с адресом 192.168.10.87 SOAP-клиент послал запрос на выполнение метода TestObject.hello, а в ответ получил SOAP-сообщение с результатами выполнения, то есть со строкой «Hello world!!!». Кстати, трафик на одно такое обращение составил 2500 байт для протокола SOAP и 1000 байт для бинарного протокола.

Как видно по таблице «Способы общения клиента и сервера», нам позволено произвольно выбирать то, в какой именно форме будет происходить обмен данными между приложениями: можно использовать TCP/IP и сообщения SOAP, а можно обмениваться бинарными данными с помощью HTTP. По скорости работы .NET Remoting слегка отстает от DCOM и CORBA, но весьма незначительно. Разница в работе с режимами TCP/IP и HTTP минимальна. Кроме самого первого обращения клиента к серверу, в этом случае обмен с использованием протокола HTTP идет в несколько раз медленнее.

В статье «Будущее уже сегодня» в этом же номере Спеца я писал о компонентных технологиях и упоминал web-службы как еще одно средство для построения распределенных систем. Итак, эта технология описывает создание программных систем, которые однозначно идентифицируются строкой URI и процесс взаимодействия с которыми описывается с помощью языка XML.

ДОСТУП К WEB-СЛУЖБАМ ВОЗМОЖЕН С ЛЮБОЙ ПЛАТФОРМЫ И ЛЮБОГО ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА

Если сравнивать возможности этой технологии и .NET Remoting, то выделится одно-единственное существенное различие — существенное ограничение типов данных, пригодных для использования при разработке web-служб. Собственно, ничего непонятного в этом нет: web-службы рассчитаны на широкий спектр платформ, клиентов и языков разработки, но за такой праздник приходится расплачиваться ограничением типов данных. Взамен клиенту предоставляется возможность обращаться к службе любым удобным ему способом. Главное, что он должен знать, — это формат сообщения SOAP для нужной службы. В .NET Remoting спектр клиентов ограничивается шириной платформы .NET, зато там больше возможностей удаленного взаимодействия.

Кроме технологий распределенного взаимодействия, основанных на принципах RPC (парадигма вызова методов), в природе есть и другие. Например, существует MO-подход (Message-Oriented). Его идея, по сути, моделирует человеческое общение: программы обмениваются сообщениями, отвечать на которые не обязательно. В отличие от парадигмы «клиент-сервер», эта парадигма выглядит как «отправитель-получатель». Участники такого обмена могут взаимодействовать напрямую или через специальные серверы. Message-Oriented-подходы предназначены для реализации асинхронного взаимодействия между независимыми приложениями. На сегодня из MO-технологий наиболее известны IBM MQSeries и M$MQ.

для новых платформ, рождение которых состоится только в будущем, Microsoft готовит следующую итерацию своих коммуникационных систем под не очень скромным названием — Indigo. Как планируется, эта система создаст инфраструктуру для распределенного взаимодействия, обеспечения безопасности и транзакций. В качестве хоста для серверной части приложений Indigo (по традиции их называют сервисами) могут использоваться: IIS 5.1, 6.0 и 7.0 (только для Vist), службы NT, автономные приложения, а также еще одна характерная для Vist технология — Windows Activation Services.

В основе Indigo лежат идеи, проработанные еще в COM+, MSMQ, .NET Remoting и ASP.NET Web-services. Базой для приложений Indigo стала популярная сейчас сервис-ориентированная архитектура: в ней единицей строительства является не объект (класс), а автономный сервис, несущий определенную функциональность и общающийся с внешним миром с помощью сообщений SOAP. В Indigo основное отличие от web-сервисов ASP.NET и .NET Remoting — это отказ от наследования реализации объектов обмена и применение атрибутов. Вот, посмотри для сравнения TestObject из примера приложения .NET Remoting в интерпретации Indigo:

```
using System;
using System.ServiceModel;

namespace TestObject
{
  //Атрибут указывает контракт сервиса
  [ServiceContract]
  public class Test: MarshalByRefObject
  {
    //Атрибут указывает контракт метода
    [OperationContract]
    public string hello()
    {
      return "Hello world!!!";
    }
  }
}
```

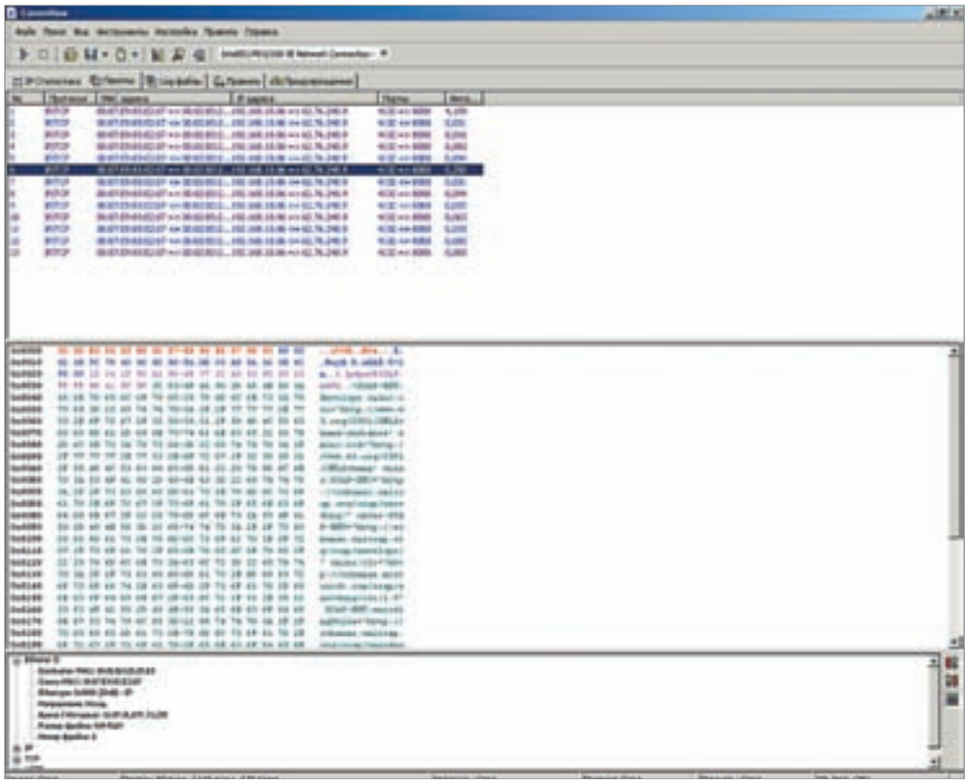
Чтобы создать клиент Indigo, можно воспользоваться web-ссылкой, а можно и встроенной утилитой — svcutil. Для указанного URL-сервиса svcutil сгенерирует прокси-классы (интерфейсы взаимо-

Способы общения клиента и сервера

Протокол обмена	SOAP	Binary
TCP/IP	X	X
HTTP	X	X

Indigo в браузере

www.gotdotnet.ru/LearnDotNet/XMLWebServices/30619.aspx
www.rsdn.ru/article/inet/indigo.xml
www.rsdn.ru/summary/2280.xml
www.mono-project.ru



TCP-пакеты, передаваемые при одном вызове

действия), описывающие работу клиента с сервисом. Предварительная версия Indigo доступна уже сейчас как компонент для WinFX SDK.

обзорам технологий Microsoft посвящена львиная доля статьи, но не думай, что только они заботится о будущем технологий распределенных систем. В качестве своей базы все описанные технологии используют открытый стандарт SOAP. Доступ к web-службам возможен с любой платформы и любого инструментального средства. Для их разработки совсем не обязательно пользоваться технологиями MS. Вот, пожалуйста — бери решения от IBM или Apache.

Повышенный интерес системных интеграторов к .NET и возможностям, предоставляемым этой платформой, связан с надеждами на проект mono. Следовательно, он связан и с возможностью использовать технологию с открытым кодом для разработки современных кросс-платформенных распределенных приложений. Напомню, Mono — это кросс-платформенная реализация .NET, C# и CLI, совместимая с .NET Framework 1.1. Последняя версия 1.1.13 датируется 11 января 2006 года, поддерживает ADO.NET, позволяет создавать приложения ASP.NET, используя в качестве хоста Apache. Разработка Mono стала возможной благодаря тому, что технологии .NET стандартизированы ECMA. В отношении будущей коммуникационной технологии Indigo (а также графического ядра Avalon) Microsoft придерживается закрытой стратегии и, скорее всего, будет заставлять разработчиков портов лицензировать эти технологии.

если особенности выбранных аппаратных или программных платформ не позволяют использовать ни одну из современных технологий построения распределенных приложений, можно использовать , отпочковавшийся от SOAP, — облегченный вариант сериализации данных о методах, вызываемых на стороне сервера, и возвращаемых ими данных. Приведу пример пакетов, которыми обмениваются клиент с сервером при вызове уже описанного метода hello() (использован XML-RPC):

```
<? xml version="1.0"?>
<methodCall>
  <methodName>hello</methodName>
    <params>
    </params>
</methodCall>
```

Получив этот пакет XML, сервер пропарсит его, найдет и вызовет метод hello, после чего вернет следующий пакет.

```
<? xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <string>Hello world!!!</string>
      </value>
    </param>
  </params>
</methodResponse>
```

В качестве типов данных можно использовать: строковый тип (тег string), целый тип (int), логический (bool), с плавающей точкой (double), временной штамп (dateTime.iso8601), двоичные данные (base64), структуры (struct) и массивы (array). Если в ходе работы вызываемого метода произошла ошибка, то можно вернуть XML-документ следующего содержания:

```
<? xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <string>Поздороваться невозможно
в связи с наступлением Апокалипсиса</string>
    </value>
  </fault>
</methodResponse>
```

Если функциональность, предоставляемая XML-RPC, кажется тебе недостаточной, всегда есть возможность разработать собственную спецификацию, а основой для нее станут изложенные мной в этой статье идеи

Сервер .NET Remoting

Объект взаимодействия .NET Remoting

Думаешь, что посмотреть сегодня вечером? Выбираем кино с TOTAL DVD!

Все о кино – читай о блокбастерах месяца, размышляй о лентах вместе со звездами, выбирай на какой сеанс пойти

- Все о DVD – самые лучшие релизы месяца, более 50 обзоров, море интервью
- ...и немного о технологиях будущего! Телевидение высокой четкости, плазмы и многое другое!

Total DVD – ультимативный журнал для киноманов!

Каждый журнал комплектуется DVD-приложением с великолепным полнометражным фильмом категории «А» (качество изображения и звука на диске соответствует лучшим мировым релизам), подборкой трейлеров и анонсов новых картин и роликами к DVD-релизам.

Ищешь себе технику для домашнего кинотеатра? «DVD Эксперт» – самый лучший гид по аудио-видео- новинкам!

Все о Hi-Fi, High End и Home Cinema!

- Пошаговые инструкции по составлению и инсталляции системы домашнего кино
- Лучшие системы и компоненты месяца – рай для новичков. Более 50 самых новых моделей в оценочных и сравнительных тестах
- Готовые системы, интервью, самые свежие новости индустрии

Всегда на лезвии прогресса!

Выбираем домашний кинотеатр с журналом «DVD Эксперт»! Сейчас это стильно, это модно, это доступно, это просто!

Каждый журнал комплектуется DVD-приложением с великолепным полнометражным фильмом категории «А» (качество изображения и звука на диске соответствует лучшим мировым релизам) и тестами для настройки системы хоум синема.

будущее уже сегодня

СОВРЕМЕННОЕ ПРОГРАММИРОВАНИЕ

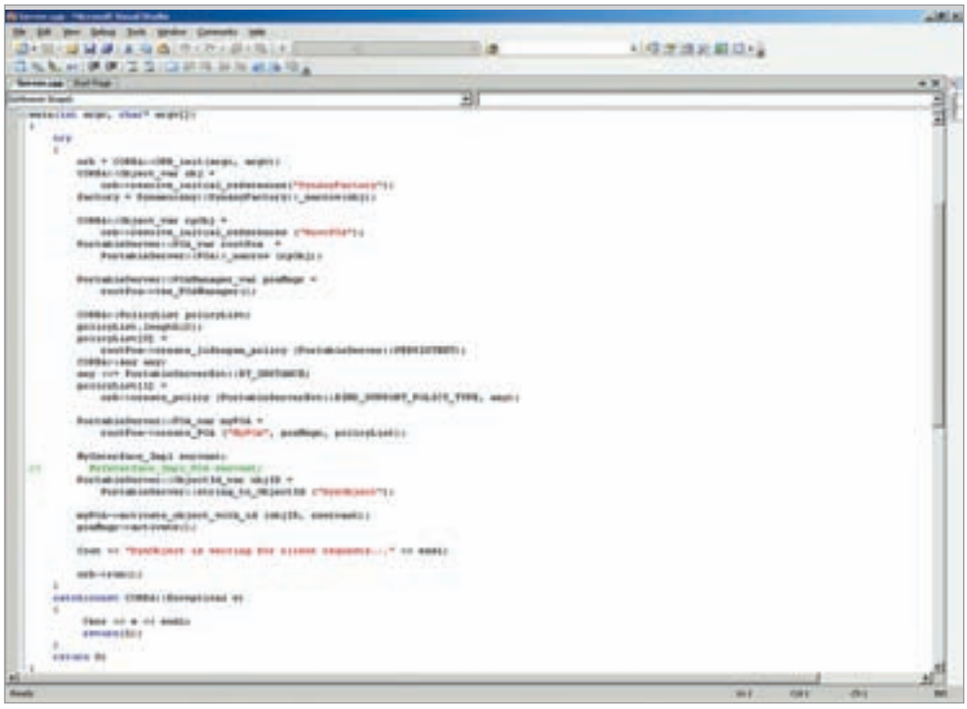
ИДЕТ РАННИЙ ЭТАП РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ, МЕЖДУ ЗАКАЗЧИКОМ И ИСПОЛНИТЕЛЕМ УТРЯСАЮТСЯ ПУНКТЫ ТЕХНИЧЕСКОГО ЗАДАНИЯ. КАК РАЗ В ЭТО ВРЕМЯ ОБНАРУЖИВАЕТСЯ ПРОБЛЕМА ВЫБОРА СРЕДСТВ ИНТЕГРАЦИИ КОМПОНЕНТОВ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ |

ПАЛАГИН АНТОН АКА TONY (TONY@EYKONTECH.COM)

требования к современному прикладному программному обеспечению основаны на возможностях глобальной интеграции информационных технологий, возникшей благодаря развитию мировой паутины. Если программа варится в собственном соку на одиноком «пентиуме», закопанном в землю, ей никто не заинтересуется. Сегодня человек, который возвращается на электричке из загородной поездки, может заказать на дом мороженое к ужину, а все благодаря мобильному телефону. Если ты современный водитель-дальнобойщик, за твоим вояжем наблюдают из центра управления поездками, и только попробуй слить бензин — твоя зарплата уменьшится из-за слитого горючего.

Как же выглядит современная информационная система? Банальная схема «клиент — сервер» здесь не подойдет. Ближе к правде будет схема «много клиентов — много серверов». Объем современных потоков данных давно перестал быть таким, чтобы было возможно перевести их в простые схемы. Так что любая деятельность предприятия дифференцируется на процессы, процессы разбиваются на подпроцессы, или сервисы, которые в свою очередь делятся на логические операции.

Если ты, к примеру, рядовой программист в аутсорсинговой конторе, то тебе хорошо знакомы процессы реализации программного кода, написания тестов, документирования, работы с багтреком и контроля версий. Менеджер твоего проекта оперирует процессами управления проектом. Заказчик программы знаком с процессом мониторинга состояния проекта, а отдел качества — с процессами контроля качества продукции. Каждый знает свою роль в ежедневной деятельности организма компании. Подобная дифференциация позволяет четко выделить частную функциональность в огромной кипе, которая образуется общей функциональностью информационной системы предприятия. Разработать такие системы на од-



Зубодробительная инициализация сервера в CORBA

ном языке программирования невозможно. Целевая система всегда получается гетерогенной с широким спектром программных и аппаратных платформ. Для интеграции компонентов системы используются компонентные технологии DCOM, CORBA, Enterprise JavaBeans и web-сервисы.

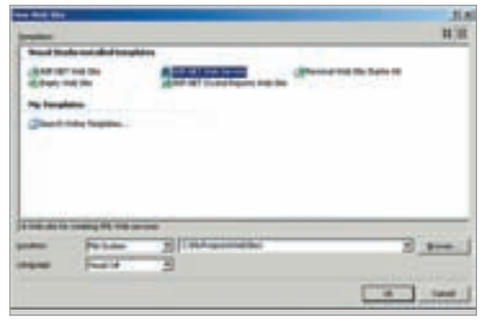
решения на базе технологий DCOM и CORBA в 90-х годах пользовались большой популярностью. Эти монстры программной инженерии пытались предоставить разработчику серебряную пулю — универсальный инструмент, решающий любые проблемы.

В чем заключается суть этих технологий? На серверной стороне реализуются методы объектов взаимодействия, абстрактные интерфейсы этих объектов описываются на специальном языке определения интерфейсов IDL. Клиент, который пытается обратиться к серверному объекту, должен компилироваться с учетом IDL-кода интерфейса. Взаимодействие по сети происходит с помощью специальной надстройки над протоколом TCP/IP — RPC (Remote Procedure Calling) в случае с DCOM или собственным аналогом RPC в случае CORBA. Клиент такой модели вызывает на своей стороне методы заглушки (stub), которая обращается к ядру DCOM/CORBA. Ядро запаковывает данные вызова в бинарные пакеты и отсылает их на сервер по TCP/IP. Сервер принимает эти пакеты, выбирает вызываемый метод в заглушке указанного объекта, выполняет его и возвращает результат выполнения клиенту, в результате у клиента возникает иллюзия того, что код на сервере выполнен. И все идет хорошо, пока клиентский и серверный код пишутся на одном языке и запускаются на одной платформе.

гигантский геморрой, однако, обнаруживается в случаях, когда хочется добиться кросс-платформенности и независимости от языка разработки. Все дело состоит в типах данных, которые варьируются от языка к языку, в особенностях платформ и даже кодировке текста, используемой по умолчанию. В итоге приходится ограничивать типы данных, передаваемых в методы. Например, для этого в технологии DCOM используется тип VARIANT, куда зашивается код типа данных, а сами данные лежат в union-поле структуры VARIANT. Код даже маленького проекта, если посмотреть на него вблизи, пугает своими очертаниями, а при взгляде издали становится похож на внебрачного сына Франкенштейна.

Кросс-платформенность, по версии Microsoft, заключалась в возможности обращаться из Windows 98 в Window NT, а независимость от языка ограничивалась поддержкой (кроме C++) Visual Basic, Delphi и Java (для COM+). Строго говоря, бинарное наследование кода в самой своей сути хранит невозможность использовать его повторно и невозможность применять объектно ориентированные подходы. Одно дело — когда бинарный компонент, к которому пользователь обращается по строгому бинарному абстрактному интерфейсу, работает в одной операционной системе и на одной аппаратуре. Тут можно не беспокоиться о работоспособности компонента. Другое дело — когда ты пытаешься использовать старый компонент на новом процессоре с новыми SIMD-инструкциями и в новой операционной системе. Вот тут-то никто и ничего не гарантирует.

технология corba оказалась более демократичной к платформам и языкам разработки, в отличие от dcom. Однако бинарная природа взаимодействия между компонентами также доставила кучу неприятностей разработчикам информационных систем. Также стоит упомянуть, что CORBA — это набор стандартов консорциума OMG, и крупнейшие системные интеграторы создают свои реализации этого стандарта, что, конечно, не способствует повсеместному распространению CORBA. Крупным недостатком обеих технологий являются «толстые» клиентские части систем, из-за которых разработчики вынуждены писать много лишнего кода. В то же время в определенных нишах они по-прежнему пользуются заслуженной популярностью, в основном в приложениях, требующих минимизации сетевого трафика, критичных ко времени отклика и ограниченных аппаратными ресурсами. Ниши для DCOM и CORBA — это встраиваемые и мобильные устройства, измерительные приборы (в них



Создаем web-сервис от MS

часто используется стандарт VXI-11 и RPC), то есть устройства с ограниченной производительностью и ресурсами.

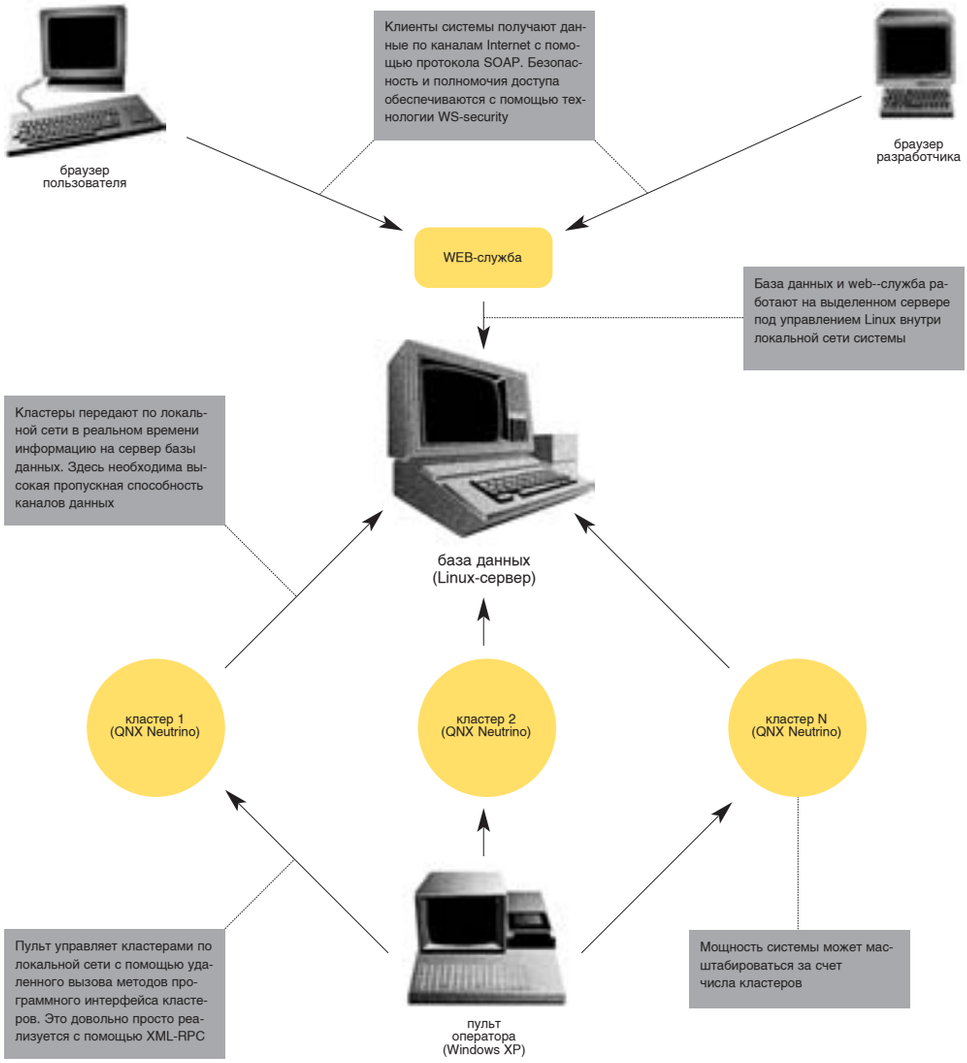
технология web-служб получила широкое распространение только в XXI веке, хотя и раньше возникали идеи сделать максимально тонкие клиенты с помощью браузера. После того как мир, содрогаясь, переполз в третье тысячелетие, на свет божий выполз новый компонентный гад. Что за гад? Это технология создания приложений, которые передают друг другу информацию по протоколу TCP/IP, уже распространившийся с ПК на мобильные платформы и встраиваемые устройства. Данные запаковываются в пакеты XML согласно спецификациям SOAP, которые затем передаются между компонентами. Отображение информации происходит с помощью браузера и динамических страниц HTML. Создать службу можно при помощи специального набора разработчика от IBM или Apache либо с помощью сред разработки MS Visual Studio 2003 и 2005.

Подобное решение позволяет избежать множества проблем, присущих старым компонентным технологиям. Языковые и платформенные различия автоматически решаются с помощью HTML и XML. Браузер играет роль универсального тонкого графического клиента. Проблемы блокировки ресурсов и транзакций решаются самим сервером и используемым сервером базы данных.

Однако в технологии web-служб есть и своя ложка дегтя: избыточный трафик — за счет пакетов SOAP и, конечно, динамических страниц HTML, на которых отображаются результаты, введенные пользователем. «Дегтя» прибавляет и слишком длительное время отклика, которое тратится на передачу данных по сети, парсинг XML-документов, генерацию HTML-страниц и, наконец, на рендеринг документов в окне браузера. Вот и выясняется, что технология web-служб неприменима в приложениях реального времени, мобильном секторе и секторе встраиваемых устройств. Однако эти технологии находят широкое применение в реализации инфраструктуры бизнес-процессов.

реальные задачи включают в себя интеграцию различных устройств и операционных систем, поэтому выбрать какую-то одну компонентную технологию невозможно. Обычно используется гибридная схема. Например, в аутсорсинговой компании была поставлена задача реализовать информационную систему. Для того чтобы реализовать эту задачу, разработали такие решения:

— С ПОМОЩЬЮ WEB-СЛУЖБ ПРЕДОСТАВЛЯЕТСЯ СТАТИСТИЧЕСКАЯ И АНАЛИТИЧЕСКАЯ ИНФОРМАЦИЯ ДЛЯ ПРОЦЕССОВ ЗАКАЗЧИКА, РУКОВОДСТВА И ОТДЕЛА КОНТРОЛЯ КАЧЕСТВА.



Система составления прогнозов погоды

мнение разработчика

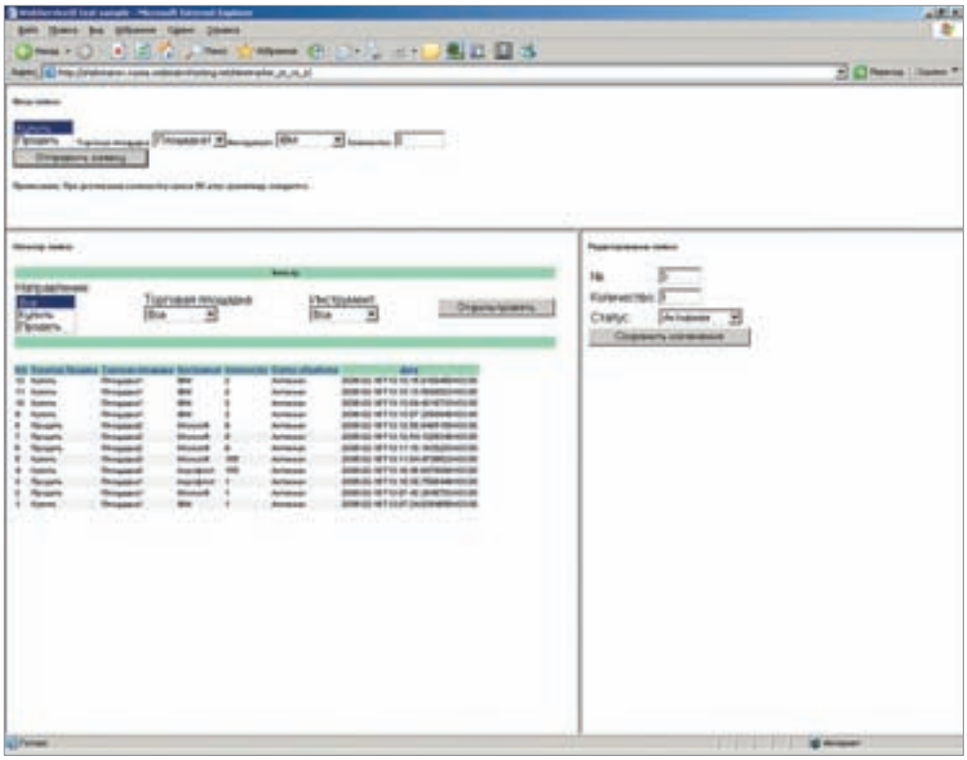
ВСЕ ЧАЩЕ ПРОГРАММИРОВАНИЕ СВОДИТСЯ К ПРОЕКТИРОВАНИЮ. НАЛИЧИЕ МНОЖЕСТВА БИБЛИОТЕК, ИЗ КОТОРЫХ МОЖНО СОБРАТЬ ПРАКТИЧЕСКИ ЛЮБОЙ СОФТ, — ЭТО ХОРОШО. В ТО ЖЕ ВРЕМЯ, К СОЖАЛЕНИЮ, СТАНОВИТСЯ ВСЕ БОЛЬШЕ «ПРОГРАММИСТОВ», КОТОРЫЕ НЕ УМЕЮТ НИЧЕГО, КРОМЕ КАК ИГРАТЬ В ПОДОБНЫЙ «КОНСТРУКТОР LEGO». В ПРИНЦИПЕ, Я НЕ ПРОТИВ ТАКОГО ПОДХОДА К РАЗРАБОТКЕ СОФТА, НО ВАЖНО, ЧТОБЫ ЛЮДИ, КОТОРЫЕ МОГУТ СЛЕПИТЬ ИЗ НИЧЕГО НОВЫЙ КИРПИЧИК, НЕ ВЫМЕРЛИ ОКОНЧАТЕЛЬНО.

Никита Бурцев — системный администратор

— С ПОМОЩЬЮ КОМПОНЕНТОВ НА БАЗЕ DCOM, CORBA И Т.Д. ОСУЩЕСТВЛЯЕТСЯ СОЗДАНИЕ БИЛДОВ, ДОКУМЕНТАЦИИ И КОНТРОЛЬ ВЕРСИЙ.

Иногда бывает целесообразно частично отказаться от формализации взаимодействия компонент предлагаемой технологии. Если же без технологии действительно не обойтись, применимость такой формализации лучше ограничить.

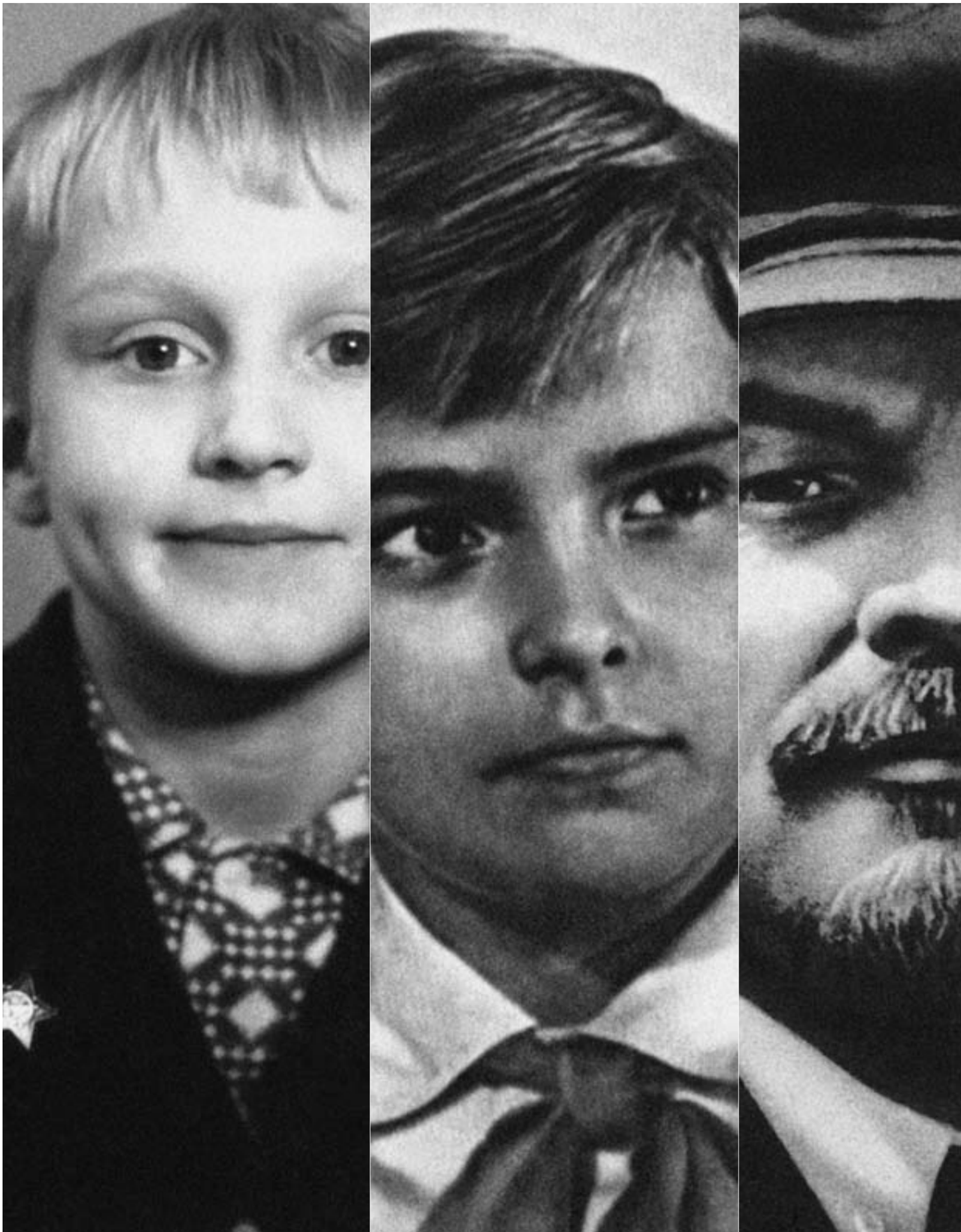
к примеру, необходимо разработать систему измерительных кластеров, управляемых с одного пульта. Допустим, кластеры анализируют данные о погоде и составляют прогноз по всем уголкам планеты. Кластеры работают под управлением QNX, оператор этой системы будет сидеть за пультом под управлением Windows XP. Компоненты, которые работают на пульте, можно формализовать с помощью технологии .NET Remoting. Также с помощью .NET будет реализовываться программа управления кластерами для оператора. Компоненты, работающие на кластере, организуются как обыкновенные библиотеки и несколько программ-демонов. Демоны управляются с пульта через специальный коммуникатор по протоколу XML-RPC, который при необходимости можно расширить до нужных рамок. В принципе, XML можно также использовать для связи между любыми компонентами, этот язык никак не ограничивает оперативность действий разработчиков. Прогнозы погоды, подготовленные кластерами, а также диагностическая и статистическая информация об их работе аккумулируется в отдельной базе данных, доступ в которую возможен через механизм web-служб. Доступ к этой информации по Сети ограничивается для обыкновенных пользователей (посетителей сайта прогноза погоды) и для заказчиков и разработчиков системы с помощью технологии WS-Security.



Web-сервис с использованием DHTML

компонентные технологии принципиально отличаются друг от друга быстродействием, распространенностью на целевых платформах и сложностью использования. По простоте использования web-сервисы стоят на первом месте. Код такого сервиса — десяток строк, все остальное за тебя делает среда разработки. Гораздо сложнее и массивнее выглядит код компонентов DCOM и CORBA. По быстродействию же на первом месте толпятся технологии, предоставляющие бинарное взаимодействие между компонентами.

P.S. В соответствующих разделах на сайте rsdn.ru можно найти исчерпывающую информацию по нашей теме и сравнение производительности всех описанных технологий. Если и этого мало — Яндекс, Гугл, Рамблер и иже с ними в помощь



ЭВОЛЮЦИЯ

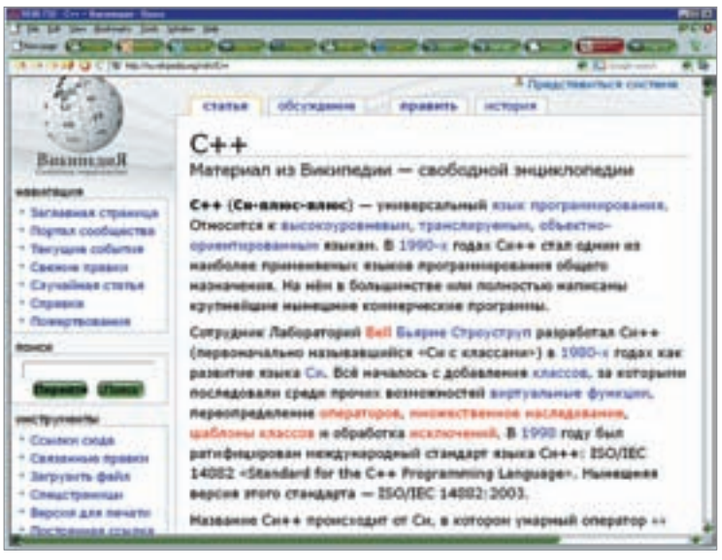
НЕСТАНДАРТНЫЙ C++

ЛЮБОЙ ДУРАК СОЗДАСТ АППАРАТ, КОТОРЫМ МОЖЕТ УПРАВЛЯТЬ ТОЛЬКО ГЕНИЙ. НО ТОЛЬКО ГЕНИЙ СОЗДАЕТ АППАРАТ, С КОТОРЫМ УПРАВИТСЯ ЛЮБОЙ ДУРАК... | **КРИС КАСПЕРСКИ АКА МЫЩЪХ**

священные войны давно разгораются вокруг языков программирования, но все как-то мимо писсуара и совсем не в тему. Обычный тезис: «Покажите мне пример, который нельзя реализовать на моем любимом XYZ, и тогда я съем свою тюбетейку». Еще бы козырек намазать маслом, чтобы было легче глотать!

Теоретически, если задачу можно решить на Машине Тьюринга, ее можно запрограммировать на любом существующем языке. Весь вопрос в том, за какое время, какой ценой и с какой эффективностью. Чистый язык сам по себе никому не интересен. Для полноценного программирования требуются средства разработки: трансляторы, линкеры, верификаторы, IDE, отладчики, библиотеки и т.п. И желательно, чтобы трансляторов было больше одного.

Основная разница между C++ и Basic'ом вовсе не в том, что трансляторы Basic'а генерируют тормозной код. Главное отличие C++ в том, что он никому не принадлежит. Есть открытый стандарт и десятки аттестованных компиляторов, которые генерируют стандартный промежуточный код, легко интегрируемый в любой проект, написанный, например, на Паскале или Ассемблере. Это не только упрощает перенос на другие платформы (LINUX, Palm OS), но и стабилизирует обстановку на рынке.



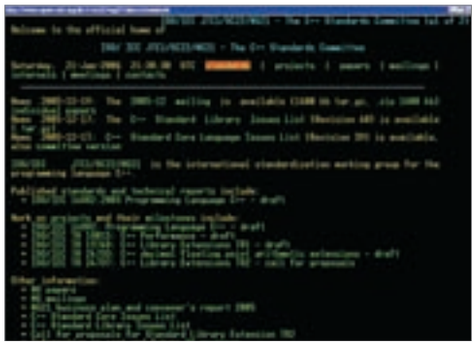
Opera написан на смеси оптимизированного C/C++ и заметно обгоняет «Лиса», реализующего графический интерфейс через переносимые библиотеки

Язык C++ нельзя просто взять и «свернуть», как Microsoft свернула Visual Basic, отправив в игнор совместимость с ранее написанным кодом. Десятки тысяч программистов оказались буквально выброшенными на улицу. Переделывать отлаженный код под .NET муторно, сложно, и нет никакой гарантии, что через несколько лет Microsoft еще раз не впадет в маразм. Переучиваться под что-то другое слишком поздно. Как показывает практика, Basic необратимо калечит образ мышления программиста, особенно Visual. Кстати, Basic тоже имеет свой стандарт, как европейский, так и американский. Однако возможности настолько скромны, что даже самая непривередливая девушка не возьмет его замуж.

выбирая язык, ты выбираешь судьбу. И чтобы эта судьба не зависела от воли левой пятки Microsoft или Borland, необходимо писать так, чтобы программа транслировалась любым независимым компилятором (или хотя бы несколькими). Но сказать намного проще, чем сделать! Чистые компиляторы сейчас не в моде. Молодое племя программистов с трудом отличает язык от IDE, прочно подсаживаясь на иглу «Мастеров» и прочих растительных заграничных штучек. Можно долго спорить, чего в Мастерах больше: пользы или вреда. Несомненно одно: человек, привыкший к Microsoft Visual C++ (вернее, к ее Microsoft C/C++ Optimizing Compiler, cl.exe - прим. AvaLANche), переходит на «правильные» трансляторы типа GCC с большим трудом, если переходит вообще, хотя, казалось бы, и то, и другое — компиляторы одного и того же языка C++.

чистый компилятор в стандартном языке неинтересен так же, как неинтересен «чистый» язык, поскольку возможности ввода-вывода (если можно так выразиться) очень ограничены, а стандартные библиотеки совершенно непригодны для создания программ с графическим интерфейсом. Механизмы взаимодействия с операционной системой отсутствуют как класс, и даже такую простую операцию, как выдвижение CD-ROM-каретки, невозможно осуществить стандартными средствами! Вот и приходится использовать готовые компоненты, заточенные под конкретный компилятор, и нестандартные языковые расширения, которые привязывают программиста к поставщику. К тому

www.open-std.org/jtc1/sc22/wg21 — главная страница комитета по стандартизации C++ (на английском языке).
http://david.tribble.com/text/cdiffs.htm — несовместимости между классическим C и C++ (на английском языке).
www.acceleratedcpp.com/authors/koenig/c++std/revisions.pdf — неофициальный список изменений (не diff) между старой и новой редакциями стандарта языка C++, составленный энтузиастами (на английском языке).



Главная страница комитета по стандартизации C++

же, хотя и существует множество открытых библиотек, которые написаны на стандартном C++ и позволяют создавать переносимые графические приложения, компилируемые любым компилятором, все-таки... Как же они тормозят! Взять хотя бы «Горящего Лиса» и сравнить его с Opera...

Решение проблемы в общем виде практически всегда проигрывает частному случаю. Это инженерный закон. Нестандартные языковые средства ускоряют процесс разработки в несколько раз, поэтому только глупые и невежественные откажутся от них там, где переносимость не требуется. Подавляющее большинство программистов руководствуется совсем не стандартом (о существовании которого многие из них даже и не догадываются), а документацией на конкретный компилятор или даже популярными книжками из серии «Microsoft Visual C++ для полных дебилов», что, в общем, правильно.

вопреки распространенному заблуждению, стандарт пишется не для программистов, а для разработчиков компиляторов. Изучать C по стандарту ни в коем случае нельзя, и не потому, что он написан заумным языком, в котором путаются даже профессионалы. Даже не из-за обилия фраз «неопределенно, зависит от конкретной реализации». Камень преткновения в том, что ни один из компиляторов не поддерживает стандарт на 100%! Реально можно использовать только базовые языковые средства, составляющие ядро C++, а все остальное — это сплошное «шаг влево, и компилятор разваливает программу без предупреждения».

Проходит много лет, прежде чем возможности, принятые новой версией стандарта, приобретают реальную поддержку среди компиляторов. Трансляция программы — очень сложная, можно сказать, магическая штука, намного более сложная, чем кажется людям со стороны. Некоторое представление о глубине проблемы дает статья «Редкая профессия» Евгения Зуева (**www.pcmag.ru/archive/9705s/05s979.asp**): как три российских программиста разрабатывали C++-компилятор и с какими граблями они воевали.

Пожалей компилятор! Не насилуй его навороченными конструкциями, почерпнутыми из новейшей редакции стандарта и до сих пор еще не отработанными. Так ты существенно ограничиваешь круг компиляторов, пережевывающих твою программу, и создаешь все условия для появления трудноуловимых ошибок, генерируемых самим компилятором. Правило номер один гласит: «Не греши на компилятор и прежде всего ищи ошибку у себя». Однако из этого вовсе не следует, что компилятор никогда не ошибается. Компиляторы содержат поистине гигантское количество ошибок, и bug lists обычно имеют очень и очень внушительные объемы.

процесс программирования на C++ нередко сравнивают с хождением по минному полю. Виновата чрезмерная сложность языка, который с возрастом становится все сложнее и сложнее. Поговаривают даже о скорой кончине C++. Симптомы загнивания и деградации налицо, пусть даже слухи о его смерти сильно преувеличены. C++ не решил тех проблем, ликвидации которых ожидали. Программирование не стало ни проще, ни эффективнее, количество ошибок ничуть не уменьшилось, удачные примеры повторного использования кода (о котором трубят все поклонники C++) можно пересчитать по пальцам одной руки… Притом совокупная себестоимость программирования существенно возросла — достаточно взглянуть на зарплату C++-программистов, цену на средства разработки и затраты на процесс обучения и освоения языка.

Программисты, одинакового хорошо владеющие двумя языками (C и C++), неоднократно замечали, что для 99% проектов 99% возможностей C++ просто не нужны! Взять хотя бы классический пример. Начинающие программисты убеждены, что форма записи «a = b + c» лучше, элегантнее и выразительнее, чем «a = add(b,c)». Однако это всего лишь заблуждение (по молодости и не такое случается). Первая запись скрывает логику программы, делая алгоритм неочевидным и заставляя программиста постоянно вспоминать, был или не был перегружен оператор сложения, какие побочные эффекты он имеет, как реализован и т.д. Стоп! Тут кто-то неожиданного говорит: «Необходимо программировать так, чтобы не было побочных эффектов». И как же можно запрограммировать?! Даже если нужно «сложить» всего две строки, дело уже не обходится без побочных эффектов, приходится выделять память, что реализуется разными путями, которые должны быть описаны в документации на перегруженный оператор сложения. Следовательно, удобство чисто внешнее, плюс в ряде случаев намного полезнее функция, которая не выделяет память, а берет ее из первой строки, что делает реализацию оператора «+» либо невозможной, либо нелогичной.

Это совсем не призыв к отказу от плюсов, а призыв к осмотрительности, осмысленности и осторожности. Программирование — инженерная дисциплина, а всякий инженер должен руководствоваться принципом целесообразности. Вот только один пример из личной жизни. Мы с другом пишем программу.

— А давай здесь используем вот такую возможность.

— А зачем?

Главная страница ECMA-372 — стандарта языка C++/CLI

— Она сократит программу на пять строк и сделает ее более «наглядной».
— А ты уверен, что другие компиляторы ее поддерживают? По мне, так лучше написать пять лишних строк сейчас и никогда потом не возвращаться к этому коду, чем править твой «элегантный» код при всяком переносе на другой компилятор, мучительно вспоминая, как он работает.

Замечено, что код от программистов, не знакомых со стандартом, часто бывает более переносимым, чем код от тех, кто излазил его вдоль и поперек, горя желанием применить полученные знания на практике. Тем не менее без карты минного поля далеко не уйти, а без знания стандарта — ничего не запрограммировать. Знание пунктов стандарта и номеров статей уголовного кодекса придает программисту шарм профессиональной солидности, значительно упрощает трудоустройство, особенно на руководящие должности. Так что близкое знакомство со стандартом обещает быть отнюдь не бесполезным.

стандартизацией языка C++ занимается множество различных «инициативных» групп, «костяк» которых составляют ISO, IEC, JTC1, SC22 и WG21. Вместе они образуют единый комитет, который так и называется — «ISO/IEC/JTC1/SC22/WG21 The C++ Standards Committee». Его формальная глава — ISO, псевдообщественная организация, которая продвигает коммерческие решения своих создателей в качестве международных стандартов. Главная страница комитета — **www.open-std.org/jtc1/sc22/wg21**. Здесь можно подписаться на рассылку, узнать новости, поживиться различными сопроводительными материалами, но текста самого стандарта нет — он распространяется только на платной основе в печатном виде, причем бумага делается отнюдь не из конопли, а из деревьев. Бесплатно можно заточить только черновую версию, так называемый draft: **www.open-std.org/jtc1/sc22/wg21/docs/papers/2001/n1316/body.pdf.** Для «простых смертных» программистов, не озабоченных сертификацией своего компилятора, она вполне пригодна.

ясь ECMA-372, необходимо быть готовым к любых неожиданностям и несовпадениям с ISO/IEC C++ (поэтому дальше по тексту упоминается только ISO/IEC, а ECMA используется как заначка).

Согласно ISO/IEC, стандарт, которым описывается новый C, получил номер 14882, за которым идет год, в котором был принят этот стандарт. В настоящее время самой ходовой версией является стандарт от 1998 года, обозначаемый как ISO/IEC 14882:1998 (далее по тексту «старый стандарт»). Последняя редакция принята в 2003 году (новый стандарт), она реально поддерживается только несколькими компиляторами, и то криво. Следующий стандарт выйдет примерно в 2007-2010 году и будет содержать кучу нововведений.

Списка изменений комитет не ведет и тем самым вынуждает нас сравнивать различные версии стандарта самостоятельно, вычитывая и сверяя порядка 750-ти листов на английском. Правда, в Сети можно найти неофициальный перечень изменений, подготовленный третьими лицами: **www.acceleratedcpp.com/authors/koenig/c++std/revisions.pdf**. Здесь обнаруживается целых 300 страниц изменений, большую часть которых составляет чисто «редакторская» правка, устраняющая разночтение в формулировках. Ковыряться в этой навозной куче — неинтересное и неблагодарное занятие. По существу, что нового появилось в стандарте? Берешь какой-нибудь компилятор, наиболее полно поддерживающий новый стандарт (например EDG C++ Front End), и читаешь What's new (можно слить с **www.edg.com/cpp_ftrs.html**).

изменения в основном касаются шаблонов, причем многие из них носят «отвоевательный» характер. Большой победой стало утверждение экспортируемых (export) шаблонов. Формально эта возможность присутствовала еще в старом стандарте, однако мало кем поддерживалась из-за невостребованности и технических сложностей реализации. Примеры, приведенные в учебниках по C++, не транслировались ни GCC, ни Microsoft C/C++ Comiler. Долгое время их переваривал только уже упомянутый компилятор EDG, на котором, кстати, основан популярный Borland C++ Builder и малоизвестный Comeau C++.

Секретарь комитета по стандартизации Эрб Саттер (Herb Sutter) выступил с предложением убрать экспортируемые шаблоны из нового стандарта (смотри дискуссию под лозунгом «Why We Can't Afford Export» — **http://std.dkuug.dk/jtc1/sc22/wg21/docs/papers/2003/n1459.html**). Однако предложение не прошло: восемь участников проголосовали за удаление export'а, а 28 были за то, чтобы оставить его. В итоге export оставили, что заставило разработчиков компиляторов сильно нервничать. Поддержка экспортируемых шаблонов требует значительных переделок не только компилятора, но и линкера. Многие куски кода вообще придется переписывать заново… Не поддерживать export нельзя — перестанут уважать. Полнота поддержки стандарта стала вполне весомым критерием при выборе компилятора.

Другим немаловажным достижением можно считать проработку шаблонов с частичной специализацией (partial specialisation), которые в старом стандарте описывались весьма туманно и никем, кроме EDG, не поддерживались, а зря. Как известно, шаблоны представляют собой механизм абстрактной работы с данными, «переваривающий» целочисленные переменные наряду с векторными

кто занимается стандартизацией

КОМИТЕТЫ, ЗАНИМАЮЩИЕСЯ СТАНДАРТИЗАЦИЕЙ C++, ПРЯЧУТСЯ ЗА МАЛОПОНЯТНЫМИ АББРЕВИАТУРАМИ. ЕСЛИ РАСШИФРОВАТЬ ИХ, СТАНЕТ ПОНЯТНЕЕ. РАЗБЕРЕМСЯ, КТО ЕСТЬ КТО.

ISO (INTERNATIONAL STANDARDS ORGANIZATION) — МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ СТАНДАРТИЗАЦИИ, ЧЛЕНОМ КОТОРОЙ МОЖЕТ СТАТЬ ТОЛЬКО ПРЕДСТАВИТЕЛЬ НАЦИОНАЛЬНОГО КОМИТЕТА ПО СТАНДАРТИЗАЦИИ.

JTC1 (JOINT TECHNICAL COMMITTEE) — СОВМЕСТНЫЙ ТЕХНИЧЕСКИЙ КОНСУЛЬТАЦИОННЫЙ СОВЕТ ПО ИНФОРМАЦИОННЫМ ТЕХНОЛОГИЯМ.

SC22 (SUBCOMMITTEE FOR PROGRAMMING LANGUAGES) — ПОДКОМИТЕТ ПО ПРАВАМ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ, ИХ ОКРУЖЕНИЮ И СИСТЕМНЫМ ИНТЕРФЕЙСАМ.

WG21 (WORKING GROUP FOR C++) — РАБОЧАЯ ГРУППА ПО СТАНДАРТИЗАЦИИ C++.

ANSI (AMERICAN NATIONAL STANDARDS INSTITUTE) — АМЕРИКАНСКИЙ НАЦИОНАЛЬНЫЙ ИНСТИТУТ СТАНДАРТОВ.

NCITS (NATIONAL COMMITTEE FOR INFORMATION TECHNOLOGY STANDARDS) — НАЦИОНАЛЬНЫЙ КОМИТЕТ ПО СТАНДАРТИЗАЦИИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ (БЫВШИЙ X3, ЧТО РАСШИФРОВЫВАЕТСЯ СОВСЕМ НЕ КАК ТЫ ПОДУМАЛ :), А КАК АККРЕДИТОВАННЫЙ КОМИТЕТ СТАНДАРТИЗАЦИИ).

J16 (TECHNICAL COMMITTEE FOR PROGRAMMING LANGUAGE C++) — ТЕХНИЧЕСКИЙ КОМИТЕТ ПО ЯЗЫКУ ПРОГРАММИРОВАНИЯ C++.

FERMILAB — ЛАБОРАТОРИЯ ФЕРМИ, ЧЛЕН J16.

НА WWW.DOSWIN32.COM ЕСТЬ БЕСПЛАТНЫЙ ПРОФЕССИОНАЛЬНЫЙ ЛИНКЕР ULINK ОТ ЮРИЯ ХАРОНА

Как вариант, можно воспользоваться европейским ECMA-стандартом на язык C++/CLI, где практически слово в слово копируется полная версия стандарта на C++ (поэтому стандарт нарвался на множество упреков и нападок). Стандарт за номером 372 (**www.ecma-international.org/publications/standards/Ecma-372.htm**) — наш. В отличие от буржуазной ISO, Европа раздает полные версии стандартов с одной хапки, не требуя за это ни денег, ни даже традиционной регистрации. Однако, пользу-

www.edg.com/cpp_ftrs.html — перечень новых возможностей языка c++, поддерживаемых компилятором edg (на английском языке).
www.ecma-international.org/publications/standards/ecma-372.htm — полная версия стандарта языка c++/cli. распространяется на бесплатной основе (без традиционной регистрации) и практически слово в слово копирует стандарт на c++ — спасибо microsoft (на английском языке).
http://lab.msdn.microsoft.com/productfeedback — центр управления багами в visual c++ и других продуктах фирмы microsoft.
http://std.dkuug.dk/jtc1/sc22/wg21/docs/papers/2003/n1459.html — текст одной из дискуссий комитета по стандартизации, то есть обсуждение экспортируемых шаблонов и другие вопросы (на английском языке).

и любыми другими типами. Удобно, но непроизводительно, поэтому для достижения максимума производительности необходимо создать специализированный шаблон (specialized template), обрабатывающий «свой» тип данных. Таким образом, у нас будет уже два шаблона: общий (general/genetic template), обрабатывающий все типы данных, и специализированный, обрабатывающий какой-то один конкретный класс с высшей эффективностью. Вот она — специализация, которая худо-бедно поддерживается большинством компиляторов еще со времен старого стандарта.

Теперь возьмем шаблон класса с несколькими параметрами. Полная специализация требует специфицировать либо все параметры шаблона, либо ни одного, что есть зло. При частичной же специализации можно специфицировать любое подмножество параметров, а остальные обрабатывать в общем виде. Дела обстоят так только по стандарту, в реальной же жизни большинство компиляторов либо совсем не поддерживают частичную специализацию, выдавая ошибку трансляции, либо молчаливо игнорируют ее, постепенно добивая программиста возле напрочь убитой программы, которая работает совсем не так, как задумывалось. Подробнее можно почитать в статье «Partial template specialisation» (**www.absoluteastronomy.com/reference/partial_template_specialisation**).

В целом, ситуация с шаблонами выглядит как откат к старым парадигмам и знаменует приближающийся провал. Большинство задач, решаемых шаблонами, легко решаются в рамках классического процедурного программирования и ассемблерных макросов. Если бы не убогость «сишного» препроцессора, шаблоны могли бы и не возникнуть — в них просто не было бы потребности. Специализированные шаблоны — это возврат к прежним способам обработки данных (своему типу — свой метод), но на «качественно новом уровне», который можно проиллюстрировать так. У нас было сверло отдельно по дереву и отдельно по металлу. Мы решили, что два сверла, дублирующие друг друга, — это не только невыгодно экономически, но и совсем не элегантно. Вот и изобрели универсальное сверло, берущее и дерево, и металл. Быстро выяснилось, что сверло режет погано, так как каждый материал имеет свои особенности обработки. Вместо того чтобы честно признать свое поражение, мы усовершенствовали сверлильный механизм — пусть самостоятельно распознает тип материала и автоматически изменяет геометрию профиля сверла. Конструкторы крякнули и послали всю эту раджу в небытие, даже не пытаясь ее реализовать... Ничего не напоминает?

ВПОЛНЕ ВОЗМОЖНО, ЧТО ТОНКОСТИ C++ ИСЧЕЗНУТ ПРЕЖДЕ, ЧЕМ УСПЕЮТ ПРИГОДИТЬСЯ

возможности метапрограммирования в новом стандарте также усилены. Опять-таки, мы столкнулись с возвратом к древнему самомодифицирующемуся коду, только под новым углом. В отличие от функционального программирования, метапрограммирование ориентировано на создание программ, манипулирующих другими программами или самими собой, что на C++ реализуется опять же посредством шаблонов. В частности, пример метареализации факториала выглядит так:

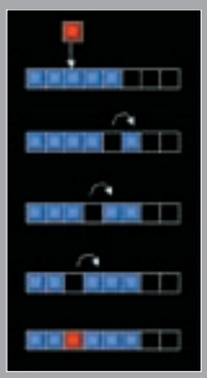
```
template struct Factorial; template <> struct
 Factorial<1>;
```

Красиво, конечно. Но, увы — неэффективно!

шаблоны представляют одну из тех областей языка, агрессивного использования которых по возможности следует избегать, поскольку качество реализации компиляторов оставляет желать лучшего. Баги в основном сосредоточены именно здесь. С другой стороны, шаблоны значительно упрощают программирование, ускоряя процесс разработки программы в несколько раз, а ошибки трансляторов исправляются по мере обнаружения. Нельзя просто сидеть и ждать. Разработчиков компиляторов нужно именно пинать, чтобы они довели поддержку шаблонов до ума.

В стороне от шаблонов идет возня по стандартизации механизма обработки структурных исключений и «декорации» (decoration) имен (также

основные направления развития



Пример, иллюстрирующий расширенную семантику операций присвоения данных на векторах

ЯЗЫК C++ НЕ СТОИТ НА МЕСТЕ И РАЗВИВАЕТСЯ НЕУКЛОННО. СВЕЖИЕ РЕДАКЦИИ СТАНДАРТА УЖЕ НЕ ЗА ГОРАМИ. КАКИХ ИЗМЕНЕНИЙ НАМ ЖДАТЬ? WALTER E. BROWN И MARC F. PATERNO, СОТРУДНИКИ ПОДРАЗДЕЛЕНИЯ ЛАБОРАТОРИИ УСКОРИТЕЛЬНОЙ ТЕХНИКИ ИМЕНИ ФЕРМИ (FERMI NATIONAL ACCELERATOR LABORATORY — БАЗИРУЕТСЯ В АМЕРИКАНСКОМ ГОРОДЕ БАТАВИЯ ШТАТА ИЛЛИНОЙС) И ПО СОВМЕСТИТЕЛЬСТВУ ЧЛЕНЫ КОМИТЕТА ПО СТАНДАРТИЗАЦИИ, УЖЕ АНОНСИРОВАЛИ ОСНОВНЫЕ НАПРАВЛЕНИЯ. НИЖЕ ПЕРЕВОД В СОКРАЩЕННОМ ВИДЕ.

ядро языка

- ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ;
- РАСШИРЕННАЯ СЕМАНТИКА ОПЕРАЦИЙ ПРИСВОЕНИЯ ДАННЫХ В ВЕКТОРНЫХ ТИПАХ;
- СОКРАЩЕНИЕ ВРЕМЕНИ КОМПИЛЯЦИИ;
- НОВЫЕ КОНЦЕПЦИИ И ПАРАДИГМЫ;
- СТАТИЧЕСКИЕ ASSERT'Ы;
- АВТОМАТИЧЕСКИЕ И DECLTYPE-ТИПЫ ДАННЫХ;
- FORWARDING-КОНСТРУКТОРЫ;
- ЛОКАЛЬНЫЕ КЛАССЫ КАК ПАРАМЕТРЫ ШАБЛОНОВ;
- ЛИТЕРАЛЫ, ОПРЕДЕЛЯЕМЫЕ ПОЛЬЗОВАТЕЛЕМ;
- NULL-POINTER-КОНСТАНТЫ;
- ALIAS'Ы ШАБЛОНОВ.

стандартная библиотека

- УЛУЧШЕННЫЙ ДАТЧИК СЛУЧАЙНЫХ ЧИСЕЛ;
- ПОДДЕРЖКА СПЕЦИАЛЬНЫХ МАТЕМАТИЧЕСКИХ ФУНКЦИЙ;
- «УМНЫЕ» УКАЗАТЕЛИ;
- РАСШИРЕННАЯ СВЯЗКА ФУНКЦИЙ;
- РАЗУПОРЯДОЧЕННЫЕ И ХЭШ-КОНТЕЙНЕРЫ;
- РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ;
- ПОЛИМОРФНЫЕ ОБЕРТКИ ВОКРУГ БИБЛИОТЕЧНЫХ ФУНКЦИЙ;
- ПОДДЕРЖКА ТИПОВ TUPLE И TRAITS;
- «АДАПТЕРЫ» ДЛЯ УКАЗАТЕЛЕЙ-ЧЛЕНОВ.

Живописный остров Whidbey, именем которого названа очередная версия компилятора Microsoft Visual C++

называемое «мангляжом»). До тех пор пока это не будет сделано, объектные файлы, сгенерированные различными компиляторами, останутся несовместимыми между собой и будут препятствовать созданию «смешанных» проектов. Впрочем, уже сейчас существуют линкеры, поддерживающие несколько компиляторов, например, в Microsoft Visual C++ и Borland Builder.

остальные «инновации» в новом стандарте носят сугубо «косметический» характер, к которому относится появление типа long long или возможность записи «list<vector<string>>» вместо «list<vector<string> >» (всегдя пользовался первым вариантом — cl не возражал - прим. AvaLANche).

Компилятор GCC начиная с версии 4.0.2 также поддерживает новый стандарт, однако не в полной мере. В архиве с исходными кодами находится директория gcc/testsuite/g++.dg/tc1 — здесь тестовые примеры и текущий статус. Проваленные тесты отмечаются ключевым словом «xfail» в комментариях, что означает «данный тест еще не реализован». Ко всем остальным прилагаются «дефектные рапорты» (defect report), по одному рапорту на файл.

Компилятор в Microsoft Visual C++ 8/2005, известной под кодовым именем Whidbey (в США в штате Вашингтон есть такой остров), также поддерживает новый стандарт, но… в очень незначительной мере. Основные усилия группы разработчиков направлены в сторону выдвижения C++/CLI и на устранение ранее обнаруженных ошибок предыдущих версий. А ошибок там… В общем, Microsoft наделала просто тьму ошибок. Впрочем, дела других производителей обстоят ненамного лучше, и чтобы написать портабельную программу, компилируемую более чем одним компилятором, необходимо ограничиться лишь базовыми языковыми функциями, да и то с кучей предосторожностей.

На сайте Mozill'ы лежит руководство по созданию переносимого кода, перечисляющее основные «разногласия» приплюснутых компиляторов. Оно так и называется — «C++ portability guide» (**www.mozilla.org/hacking/portable-cpp.html**). Полчаса увлекательного чтения в комплекте с отборным матом и истерикой гарантированы. Правда, не всему написанному можно верить. Несмотря на то, что последняя доступная на данный момент версия 0.8 датируется 2001 годом, ситуация вовсе не так плачевна и многие из упомянутых ошибок давно исправлены. Тем не менее при переносе программы на другие платформы далеко не всегда удается найти свежий компилятор, поэтому осторожность и осмотрительность не помешают.

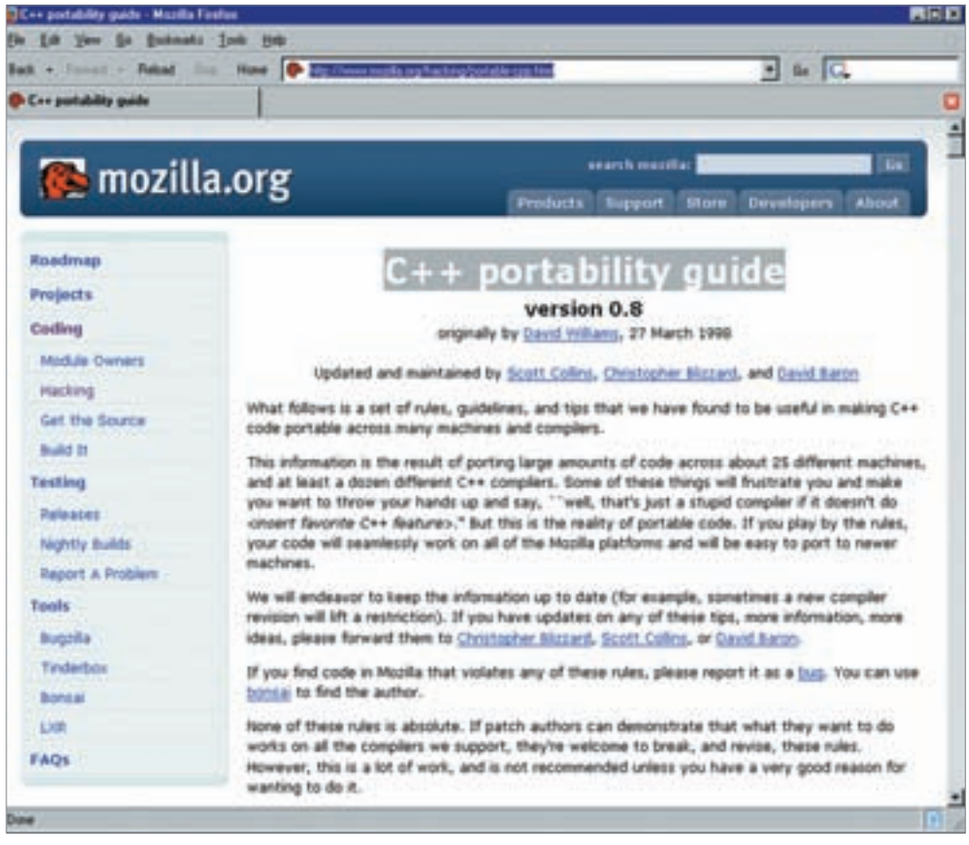
C++ эволюционирует. Хотим мы этого или нет, он развивается от плохого к еще более плохому. Впрочем, на этот счет имеются различные мнения. Некоторые хотят видеть язык предельно простым, каким был и остается классический C. Другим требуется навороченный монстр, которого сможет осилить до степени совершенства только эксперт. Какой из этих путей «правильный»? Обратимся к естественным языкам типа русского и английского.

Язык аристократов — это сплошное нагромождение условностей и противоестественных сложностей. В нем преобладают длинные слова, сложные грамматические правила и т.д. Язык трущоб обычно бывает намного более выразительным и в то же время незамысловатым, а просочившиеся в него аристократические слова со временем теряют все лишнее и усекаются, сокращаясь по длине в несколько раз. Естественно, аристократам это не нравится: главный признак образованности, с их точки зрения, заключен в языке, точнее, в умении «владеть» им. Но что стоит за этой «образованностью», кроме знания дутых конструкций?

Или вот музыка. Сначала было бум-бум, потом — во времена Баха и Моцарта — целая симфония чувств. Эволюция? А вот и нет! С приходом попа все вернулась к прежнему бум-бум. Народ устал от сложной музыки, захотелось простых мотивов, которые тоже не стоят на месте, а с каждым годом усложняются прямо на наших глазах. И в 21 век мы въезжаем с оркестровым пением и группами типа Sirenia и Penumbra. Надолго ли?

Точно так же в программировании. Первые машинные языки были очень простыми, но они все усложнялись и усложнялись до тех пор, пока не появился C, который был воспринят многими как «студенческая подделка», «варварский откат назад» и все в том же духе...

Вполне логично ожидать, что на смену C придет язык с предельно простым синтаксисом, который можно будет выучить буквально за ночь! Не спеши тратить время на углубленное изучение тонкостей C++, возможно, они исчезнут прежде, чем успеют пригодиться...



Руководство по написанию портабельного кода на приплюснутом C от Mozilla

мнение профессионалов

«И КТО ТОГДА БУДЕТ ДЕЛАТЬ "ГОТОВЫЕ КИРПИЧИКИ"?»

СПЕЦ: ХОРОШО ЛИ ТО, ЧТО ПРОГРАММИРОВАНИЕ ПОСТЕПЕННО СВОДИТСЯ К ПРОЕКТИРОВАНИЮ? КОГДА ЕСТЬ МНОЖЕСТВО ГОТОВЫХ КИРПИЧИКОВ И ИЗ НИХ ЛЕПИТСЯ НУЖНОЕ...

АНАТОЛИЙ СКОБЛОВ: Во-первых, программирование к «только проектированию» не сводится, и в обозримом времени ничто не предвещает этого. Во-вторых, конечно, хорошо, что из разработки уходит совсем уж банальная рутина. В-третьих, что плохо, народ отучается думать, от C++ пошли вопросы «А как бы сделать список, нет ли заготовочки?» (если вдруг «кирпичиков» под рукой не оказалось). И что же будет дальше?

АНТОН ПАЛАГИН: Ну, кирпичи-то еще надо уметь выбирать и класть :). Для адептов хардкорного программинга всегда остается возможность не использовать middleware, а создавать его. Впрочем, для этого необходимо самому хорошо владеть другим middleware. Как ты думаешь, сможет ли человек, никогда в жизни не видевший компьютера, собрать его с нуля?

INGREM: Нет, это не хорошо. Я понимаю, чем в общем обусловлена всеобщая тяга к «проектированию» (скорость разработки, переносимость и т.п.), но все равно не поддерживаю. Если все так и дальше пойдет, программисты совсем разучатся думать и вымрут как вид — останутся одни «проекторы». Кто тогда будет делать «готовые кирпичики»?

АЛЕКСЕЙ ЛУКАЦКИЙ: А оно не сводится ;). Это нормальный процесс унификации постоянно повторяющихся задач и часто используемых процедур и объектов. Просто современные системы проектирования мало чем отличаются от процедур и функции прошлого. Теперь к ним добавились и новые параметры (например графические элементы), а суть осталась та же. Часто повторяемые вещи реализуются готовыми «кирпичиками», а высвободившееся время можно посвятить творчеству и реализации уникальных задач.

ЗАРАЗА: Что в этом плохого? Первый шаг к использованию готовых кирпичиков был сделан тогда, когда появились модульные языки, дальше просто идет развитие в том же направлении. А разработка серьезного ПО в большом коллективе требует очень четкого разделения между проектированием и кодингом — к этому все и идет.

КРИС КАСПЕРСКИ: Когда есть множество готовых кирпичиков и из них лепится нужное... Программирование всегда было проектированием, а вот бездумной «лепкой» оно стало только недавно. Программирование (изначально) — это инженерная дисциплина с такими понятиями, как «целесообразность» и «учет рисков». Сейчас мы имеем ситуацию: если программа запускается и не падает, то можно биться от счастья в истерике. Взять хотя бы Windows или, например, MS Office. Micorosft сокрушается, что 99% пользователей используют только 1% продукта, а пользователи сокрушаются, что этот «ворд» постоянно падает. Какой вывод? Если бы программисты думали головой, то систему можно было сделать на 99% проще, а значит, надежнее...

ВЛАДИМИР ЯКОВЛЕВ

CEO КОМПАНИИ MEDIAMOBILE В СОСТАВЕ GFI. СФЕРА ДЕЯТЕЛЬНОСТИ — РАЗРАБОТКА МОБИЛЬНЫХ ИГР. ОБЩЕЕ ЧИСЛО ПРОЕКТОВ ИСЧИСЛЕНИЮ НЕ ПОДДАЕТСЯ :). ПОСЛЕДНИЙ ПРОЕКТ — СОЗДАНИЕ МОБИЛЬНЫЙ ИГРЫ «СТАЛКЕР»

ОЛЕГ КУРЦЕВ

ВЕДУЩИЙ И ИИ-ПРОГРАММИСТ КОМПАНИИ GFI UA. ТЕКУЩИЙ ПРОЕКТ — RTS «КОМБАТ». ДО ГЕЙМДЕВА УЧАСТВОВАЛ В РАЗРАБОТКЕ СИСТЕМ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ. КАНДИДАТ ТЕХНИЧЕСКИХ НАУК

НИКИТА DRAGOMIR БУРЦЕВ

РАБОТАЕТ СИСТЕМНЫМ АДМИНИСТРАТОРОМ. ДО ЭТОГО БЫЛ ИНЖЕНЕРОМ В УЧЕБНОМ ЦЕНТРЕ «СПЕЦИАЛИСТ» И ТЕСТИРОВАЛ АСУ СБЕРБАНКА РФ. В СВОБОДНОЕ ВРЕМЯ ЛЮБИТ ПОГУЛЯТЬ ПО ОКРЕСТНОСТЯМ СВОЕГО ДОМА С ФОТОАППАРАТОМ В РУКАХ

НИКОЛАЙ GORL АНДРЕЕВ

РЕДАКТОР ЖУРНАЛА «ХАКЕР», РУБРИКА «КОДИНГ». ЭКС-РЕДАКТОР «ХАКЕР СПЕЦ». РУТКИТЫ, ВИРУСЫ, ЧЕРВЯКИ И ПРОЧАЯ ЖИВУЧЕСТЬ, НАПИСАНИЕ КОТОРОЙ ВПОЛНЕ МОЖЕТ ПОДПАСТЬ ПОД СТАТЬЮ 273 УК РФ, — ПРЕДМЕТ АНАЛИЗА И УВЛЕЧЕНИЯ. УСТРАИВАЕТ ДОМА ХАНИПОТЫ, ИСКЛЮЧИТЕЛЬНО ЧТОБЫ ПОМУЧИТЬ ОЧЕРЕДНОГО БЕЗОБИДНОГО ЧЕРВЯЧКА. ПРОСТО ТАК

INGREM

СИСТЕМНЫЙ ПРОГРАММИСТ, АСПИРАНТ ПО СПЕЦИАЛЬНОСТИ «АЛГЕБРА» (НАПРАВЛЕНИЕ МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЙ — АЛГЕБРАИЧЕСКИЕ МНОГООБРАЗИЯ, ОБЛАСТЬ ПРИМЕНЕНИЯ — КРИПТОГРАФИЯ). ХОББИ — ТЕОРЕТИЧЕСКАЯ ВИРУСОЛОГИЯ, В ТОМ ЧИСЛЕ МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ПОЛИМОРФНЫХ АЛГОРИТМОВ В ВИРУСАХ

АЛЕКСЕЙ ЛУКАЦКИЙ

SECURITY BUSINESS DEVELOPMENT MANAGER, CISCO SYSTEMS. ЗА ПОСЛЕДНИЕ 13 ЛЕТ УСПЕЛ ПОБЫТЬ И ПРОГРАММИСТОМ (ПИСАЛ НА АССЕМБЛЕРЕ СИСТЕМУ ШИФРОВАНИЯ ДЛЯ ОДНОГО ИЗ РОССИЙСКИХ «ЯЩИКОВ»), И АДМИНИСТРАТОРОМ, И АНАЛИТИКОМ-АУДИТОРОМ. РАБОТАЛ И В КОМПАНИЯХ, ИСПОЛЬЗУЮЩИХ СРЕДСТВА ЗАЩИТЫ, И В КОМПАНИЯХ, КОТОРЫЕ РАЗРАБАТЫВАЮТ ИХ

КРИС КАСПЕРСКИ

МЕЛКИЙ СЕРЫЙ ГРЫЗУН (НЕ МАНИПУЛЯТОР). КОДОКОПАТЕЛЬ И ЖЕЛЕЗОКОВЫРЯТЕЛЬ. ПАЯЛЬНИК, ОСЦИЛЛОГРАФ, ДИЗАССЕМБЛЕР И ОТЛАДЧИК ВСЕГДА ПРИ НЕМ

АЛЕКСАНДР ПОЛУЭКТОВ

АВТОР ПРОЕКТА INSIDEPRO (**www.insidepro.com**). ХОББИ — ПРОГРАММИРОВАНИЕ И КРИПТОГРАФИЯ

ЗАРАЗА

РЕДАКТОР САЙТА **www.security.nnov.ru**. СФЕРА ДЕЯТЕЛЬНОСТИ — ТЕЛЕКОММУНИКАЦИИ. ХОББИ — ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И УЯЗВИМОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ВЛАДИМИР ЯКОВЛЕВ: Мои школьные да и вузовские преподаватели приходили в смятение от широкого внедрения в процесс обучения примитивных калькуляторов. Говорили, что из-за них обучающиеся стремительно тупеют. Но прогресс есть прогресс, а обучающиеся не отупели. От прогресса никуда не денешься. Программирование, сведенное к проектированию, позволяет обеспечить максимально быстрое появление продукта на рынке. Человеческая цивилизация относится к технологическому типу. С этим положением согласны не только философы, но и теологи. А это означает, что человек является, в своей мере, создателем, причем он всегда стремится создать больше за меньший период и лучший по качеству продукт. Можно говорить об эффективности такого программирования, но ведь вычислительная техника развивается колоссальными темпами! Освободившийся же интеллектуальный резерв уже не направляется на написание оператора «for», а служит совсем другим целям — созданию привлекательного для потребителя продукта.

ОЛЕГ КУРЦЕВ: Наверняка хорошо. Современное программное обеспечение достаточно сложное. Людей, работающих над одним проектом, может быть много. Просто со времен программистов-одиночек прижилось мнение о том, что проект, созданный одним человеком полностью, выглядит красиво и целостно, а если участников много — это уже не то. Если программирование сводится к проектированию и удается выделить «кирпичики», красиво разбить задачу — это действительно то, что нужно. И роль проектирования получается первичной.

СПЕЦ: БУДУЩЕЕ ЗА «ПАРАЛЛЕЛЬНЫМ» ПРОГРАММИРОВАНИЕМ?

АНАТОЛИЙ СКОБЛОВ: За ним настоящее. Там, где оно нужно.

АНТОН ПАЛАГИН: Распараллеливание вычислений и создание распределенных систем — это единственное направление, в котором пока возможен прогресс. Мы подходим к пределам миниатюризации электроники, поэтому увеличивать плотность транзисторов на сантиметр поверхности становится все труднее. Достаточно сравнить производительность одноядерных процессоров 2004 года и 2005 года — разница минимальна.

НИКИТА БУРЦЕВ: Если учесть то, что среди выпускаемых нынче процессоров доля многоядерников неуклонно растет, то и будущее, соответственно, за теми приложениями, которые будут эффективно использовать подобную архитектуру.

ЗАРАЗА: Есть задачи, которые хорошо параллелятся. Есть задачи, которые совсем не параллелятся. Хороший разработчик должен уметь писать многопоточные, распараллеленные программы. В Windows, например,

АНАТОЛИЙ СКОБЛОВ

ПОСЛЕДНИЕ 17 ЛЕТ — СИСТЕМНЫЙ ПРОГРАММИСТ, АНАЛИТИК. РАБОТАЕТ ДОМА НА СЕБЯ ИЛИ НА ЗАКАЗЧИКОВ. ИЗ ИЗВЕСТНОГО — ЯДРО OUTPOST PERSONAL FIREWALL, МОДЕМ RUSSIAN COURIER. СФЕРА ПРОФЕССИОНАЛЬНЫХ ИНТЕРЕСОВ — БЕЗОПАСНОСТЬ, ТЕЛЕФОНИЯ, ИНТЕРНЕТ И Т.Д.

АНТОН ПАЛАГИН

МЕНЕДЖЕР КОМПАНИИ EYKON (**www.eykontech.com**), СОТРУДНИК КАФЕДРЫ ИТФИ ФИЗИЧЕСКОГО ФАКУЛЬТЕТА ННГУ. СФЕРЫ ДЕЯТЕЛЬНОСТИ: СИСТЕМНАЯ ИНТЕГРАЦИЯ, АВТОМАТИЗАЦИЯ, СИСТЕМЫ ПОВЫШЕННОЙ НАДЕЖНОСТИ. ХОББИ: ГЕЙМДЕВ, ФУТБОЛ, «X-СПЕЦ»

ДМИТРИЙ СОШНИКОВ

КАНДИДАТ ФИЗ.-МАТ. НАУК, ДОЦЕНТ КАФЕДРЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ПРОГРАММИРОВАНИЯ МАИ, РУКОВОДИТЕЛЬ ГРУППЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА MAILABS, АРХИТЕКТОР КОМПАНИИ PARTNERS INTERNATIONAL, LLC

любое оконное приложение местами работает параллельно. Так что это, скорее, настоящее, а не будущее :).
КРИС КАСПЕРСКИ: Поживем — увидим. Быть может, через несколько лет никакого программирования в классическом понимании этого слова вообще не будет.
ОЛЕГ КУРЦЕВ: Не все алгоритмы можно распараллелить. Некоторые распараллеливаются легко (например нейронные сети), некоторые — с трудом. Если писать параллельные программы станет так же просто, как и обычные, появятся соответствующие языки, подходы и т.д.

СПЕЦ: СУЩЕСТВУЮТ ЛИ РЕАЛЬНЫЕ АЛЬТЕРНАТИВЫ ООП? ЧТО ПРИДЕТ НА СМЕНУ?

АНАТОЛИЙ СКОБЛОВ: Сейчас для мейнстрима — нет.
АНТОН ПАЛАГИН: Если под ООП понимать ОО проектирование и анализ, то альтернатив, пожалуй, нет, и как-то на смену пока не выходит ничего. А вот на помощь приходят сервис-ориентированный анализ и проектирование (SOAD), а также моделирование бизнес-процессов (BPM). Если же иметь в виду ОО-программирование… Сколько бонусов приносит ОО-подход, столько же граблей он услужливо подставляет, как, впрочем, и любая другая технология. Важным умением разработчика является умение выбрать инструмент, оптимальный для реализации конкретной задачи.
АЛЕКСАНДР ПОЛУЭКТОВ: Альтернатив ООП нет и не предвидится. Эволюция ООП будет продолжаться — это несомненно. Если принять, что «объект» в программировании — «данные + методы работы с этими данными», то эволюция ООП будет идти в следующих направлениях: увеличение надежности хранения данных, несмотря на уже имеющуюся инкапсуляцию, дополнительная защита данных, возможно, с использованием шифрования, увеличение количества методов для работы с данными, причем обязательно появятся методы для обмена данными между самими объектами (например, на основе XML-интерфейсов), повышение «интеллектуальности» новых и уже существующих методов и т.д.
ЗАРАЗА: ООП активно развивается, не стоит на месте. Так что вопрос, скорее, не в том, что придет на смену, а в том, как в результате это назовут :).
КРИС КАСПЕРСКИ: ООП — это только модный термин. Его «следы» можно найти практически в любом языке, если только хорошо поискать. К тому же, если взять, к примеру, C, то он неуклонно движется в сторону метапрограммирования, что, по сути, является новой парадигмой для ООП и очень близко по духу к самомодифицирующемуся коду ;).
ДМИТРИЙ СОШНИКОВ: Сложность программных систем растет так быстро, что уже сейчас ООП (а также ООД и ООА) удовлетворяют потребностям проектировщиков и разработчиков не полностью. В то время как ООП прочно зарекомендовало себя на некотором уровне абстракции, приходится прибегать к дополнительным методам борьбы со сложностью: компонентному программированию (именно компонентное программирование в модели COM позволило реализовать систему Windows в том виде, в котором мы ее видим), распределенным системам, сервис-ориентированным архитектурам.
ВЛАДИМИР ЯКОВЛЕВ: Очень трудно делать прогнозы в такой интенсивно развивающейся области. Обычно такого рода предсказаниями балуются популярные издания. Если просмотреть подшивки таких изданий за последние лет 20, то можно убедиться, что практически ни один из прогнозов не сбылся. Если верить этим изданиям, то мы должны были уже давно расшифровать язык дельфинов, побывать на Марсе в 2000 году, победить СПИД и т.п. Были прогнозы и по поводу вычислительной техники. Например, когда-то утверждали, что достижение частоты процессора 1 ГГц невозможно, поскольку это уже практически радиочастоты. Но техника успела перешагнуть этот рубеж. Как-то представитель компании Ford сказал примерно следующее: «Если бы автомобильная промышленность развивалась такими же темпами, как вычислительная техника, то автомобили ездили бы уже с околосветовой скоростью!» Так что в настоящее время видимых альтернатив ООП нет, а если и есть, то неизвестно, что будет лет этак через пять. Может, нейрокомпьютеры? Это штука, которая, в принципе, меняет все понятия программирования. Другими словами, если ты связан с ЭВМ, то будь готов к любому повороту событий :).
ОЛЕГ КУРЦЕВ: ООП меня сейчас полностью устраивает, поэтому искать альтернативу для себя не вижу смысла. Геймдев — знаете, штука затягивающая

НИКЛАУС ВИРТ

Хотел получить инструмент для обучения программированию студентов. Вирт был недоволен не только новым Алголом, но и всеми используемыми основными языками программирования, свойства и конструкции которых часто было невозможно объяснить логически и убедительно. Вскоре Никлаус и его сотрудники подготовили первую версию языка и нарекли его Pascal.

Pascal — прямой потомок Алгола. При описании синтаксиса Pascal'а Вирт использовал БНФ (Backus Normal Form — BNF), добавив в нотацию скобки { и }, означающие повторение конструкции, заключенной внутри них. Pascal (в оригинальной авторской версии) не содержит средств раздельной компиляции: модулей, многообразных числовых типов, строк переменной длины и многого из того, что добавлено в известных многим реализациях. Успех Pascal'а превзошел все ожидания. Одной из причин популярности этого языка стало то, что он способствовал развитию зарождавшегося тогда движения за структурное программирование.

интернациональное программирование

.NET: ЧТО ТАКОЕ ХОРОШО И ЧТО ТАКОЕ ПЛОХО

КАК ИЗВЕСТНО, СЛОЖИЛАСЬ (И ДОСТИГЛА СВОЕГО АПОГЕЯ) СИТУАЦИЯ, В КОТОРОЙ НА РЫНКЕ ВЫСОКИХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ ПОЯВИЛИСЬ ПРОЦЕССОРЫ, НЕСОВМЕСТИМЫЕ С X86 (ИЛИ СОВМЕСТИМЫЕ, НО ЦЕНОЙ БОЛЬШОЙ ПОТЕРИ ПРОИЗВОДИТЕЛЬНОСТИ), НАПРИМЕР — ПРОЦЕССОРЫ СТАНДАРТА IA64 |NIM[INT3 TEAM]{NIM@INT3.RU}

Появились и такие процессоры, производительность которых была бы выше, если бы компиляторы учитывали их особенности (например процессоры ia32 (64bit) или AMD (3DNOW и т.п.). В борьбе за драгоценные секунды разработчики стали делать несколько копий исполняемых файлов, откомпилированных под разное железо.

В аналогичную ситуацию попали программеры, если программа должна выполняться на разных операционных системах или на всевозможных мобильных устройствах. В результате увеличивался объем дистрибутива: в одном дистрибутиве были файлы для разного железа, и таких дистрибутивов было несколько для разных операционных систем. Соответственно, разработка затягивалась во времени и дорожала, что, естественно, влияло на стоимость программы.

Для решения подобных проблем предложили отличную идею компиляции программы в два этапа. На первом этапе исходный код программы переводили в некий байт-код. Второй этап компиляции (когда создается машинный код) должен происходить непосредственно на той платформе, на которой и будет работать данная программа. Преимущества метода ясны и понятны: программа компилируется непосредственно в машинный код с учетом всех особенностей железа и операционной системы, благодаря чему достигается высшая производительность программы, размер дистрибутива сокращается, а нервы разработчиков становятся мягкими и пушистыми.

Подобная идея была воплощена в платформе .NET (как известно, от компании Microsoft). Программа, написанная для .NET, может работать везде, где установлена эта платформа. .NET-программа, работающая в Windows, с таким же успехом пойдет в любой операционной системе семейства Unix или на мобильном телефоне. Однако здесь есть одно «но». Управляемый код будет выполняться только под определенной операционной системой, если он вызывает внутренние API, специфические для платформы, или библиотеку классов, специфическую для платформы.

изначально Microsoft выпустила три языка программирования для платформы .NET: C#, Visual Basic и managed C++ (входят в поставку Visual Studio .Net). Visual Basic .Net (VB.NET) является продолжением VB6, но от первого остался один только синтаксис. Когда я начал знакомиться с VB.NET, мне показалось, что резкий скачок языка VB подобен скачку при переходе с QBasic на VB1. Сейчас VB активно развивается и уже представляет собой девятую версию.

В то же время C++ после интеграции Visual Studio .Net претерпел изменения совсем не к лучшему. Ужасный синтаксис стал еще ужасней, хотя managed C++ имеет одно (хотя бы одно) большое преимущество — единственный язык платформы .NET, который поддерживает ассемблерные вставки в методах с пометкой unsafe. В остальном — одни недостатки :).

C#, на мой взгляд, намного лучше. Он вобрал в себя всю простоту VB и профессионализм C++. Позднее было написано множество языков для платформы .NET (по некоторым данным, до сорока).

написанная для платформы .NET программа, компилируется в некий псевдокод (MSIL), который

включает в себя всю необходимую информацию о коде. Это обстоятельство позволяет платформе целиком и полностью управлять программой (вот почему такие программы еще называют управляемыми).

Разработчикам здесь предоставлены средства для повышения безопасности кода. Например, теперь они могут запретить доступ своей программы к реестру или, обратившись к атрибутам доступа, — доступ к файлам. Эти атрибуты применимы и к программе целиком, и к ее отдельным частям. Контроль подобных ограничений происходит на уровне платформы .NET.

Второй этап компиляции (в машинный код) происходит при запуске программы. Компилируется только тот код, который должен выполняться в данный момент, — реализуется за счет заглушек, которые создаются для каждого метода при старте программы. Когда один из таких методов будет вызван, заглушка вызовет компилятор, а он в свою очередь скомпилирует тело метода. Соответственно, приходится потратить некоторое время, зато только один раз, так как после компиляции компилятор изменяет заглушку так, что выполнение передается в расположение машинного кода. Последующие вызовы метода, откомпилированного по требованию, передаются прямо в созданный ранее машинный код, что уменьшает время, необходимое на выполнение кода. Эта технология, только что описанная мной, носит имя JIT (Just In Time) — «компиляция по требованию».

Платформа .NET открывает отличную возможность разрабатывать программы на разных .NET-языках программирования, так как предоставляет систему общих типов, определенных средой выполнения и следующих ее правилам при определении новых типов, а также при создании, использовании, сохранении и привязке к типам.

предназначение сборок состоит в том, чтобы упростить развертывание приложений и решить вопросы, связанные с отслеживанием версий (они иногда возникают, если пользуешься приложениями, основанными на компонентах). Многие пользователи и разработчики знакомы с вопросами отслеживания версий и развертывания, которые возникли в современных системах, основанных на компонентах. (Речь идет о так называемом DLLHALL.) Вот, к примеру, пользователь установил одну программу, вдруг пара других начинают работать неправильно или вообще перестают исполнять свои обязанности — безумное счастье. В результате многие разработчики тратят бесчисленные часы в попытках сохранить в целостности все необходимые данные системного реестра для активации COM-класса.

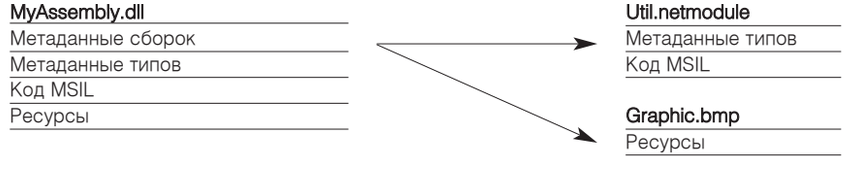
Многие проблемы разработки были решены сборками .NET Framework. Поскольку сборки являются самодокументирующимися компонентами, независимыми от данных системного реестра, они никак не влияют на установку приложений. Кроме того, сборки упрощают удаление и копирование приложений. По сути, для установки любого .NET-приложения ты просто копируешь папку с программой и создаешь ярлык на рабочем столе :). Физический смысл сборки заключается в том, что одна сборка — это один исполняемый файл (либо exe, либо dll). Также существует системный кеш сборок, он располагается по адресу %WINDIR%\assembly\. В нем хранятся сборки, распространять которые вместе с программой не имеет смысла, — эти сборки входят в стандартную поставку дистрибутива .NET Framework.

В общем случае статическая сборка может состоять из четырех элементов:

1. МАНИФЕСТ СБОРКИ, СОДЕРЖАЩИЙ ЕЕ МЕТАДАННЫЕ;
2. МЕТАДАННЫЕ ТИПОВ;
3. КОД, РЕАЛИЗУЮЩИЙ ТИПЫ, НА ПРОМЕЖУТОЧНОМ ЯЗЫКЕ MSIL;
4. НАБОР РЕСУРСОВ.

Обязательным является лишь манифест сборки, однако остальные типы ресурсов необходимы чтобы обеспечить нужную функциональность сборки. Существует несколько способов сгруппировать эти элементы в сборку. Все они могут быть сгруппированы в единый физический файл так, как показано на левой части схемы.

Кроме того, элементы сборки могут содержаться в нескольких файлах: 1) в модулях откомпилированного кода (.netmodule), 2) в ресурсах (например в файлах .bmp или .jpg), 3) в других файлах, необходимых приложению. Сборка из нескольких файлов создается тогда, когда необходимо собрать модули, написанные на различных языках, и оптимизировать загрузку приложения, выделяя редко ис-



Все три файла, принадлежащие сборке, описаны в манифесте сборки, который содержится в файле MyAssembly.dll

проблемы в приложениях Win32

СУЩЕСТВУЮТ ДВЕ ПРОБЛЕМЫ В ПРИЛОЖЕНИЯХ WIN32 ПРИ ОТСЛЕЖИВАНИИ ВЕРСИЙ:

1. ПРАВИЛА ОТСЛЕЖИВАНИЯ ВЕРСИЙ НЕ МОГУТ БЫТЬ ОПРЕДЕЛЕНЫ МЕЖДУ ЧАСТЯМИ ПРИЛОЖЕНИЯ — ОНИ ЗАДАЮТСЯ ОПЕРАЦИОННОЙ СИСТЕМОЙ. СУЩЕСТВУЮЩИЙ ПОДХОД ОСНОВЫВАЕТСЯ НА ОБРАТНОЙ СОВМЕСТИМОСТИ, ГАРАНТИРОВАТЬ КОТОРУЮ ЧАСТО БЫВАЕТ ТРУДНО. ОПРЕДЕЛЕНИЯ ИНТЕРФЕЙСОВ ДОЛЖНЫ БЫТЬ СТАТИЧЕСКИМИ И ДОЛЖНЫ ПУБЛИКОВАТЬСЯ ОДНОКРАТНО, А ЕДИНАЯ ЧАСТЬ КОДА ДОЛЖНА ОБЕСПЕЧИВАТЬ ОБРАТНУЮ СОВМЕСТИМОСТЬ С ПРЕДЫДУЩИМИ ВЕРСИЯМИ. СЛЕДОВАТЕЛЬНО, КОД ОБЫЧНО СОЗДАЕТСЯ ТАКИМ ОБРАЗОМ, ЧТО В ЛЮБОЙ ЗАДАННЫЙ МОМЕНТ ВРЕМЕНИ НА КОМПЬЮТЕРЕ МОЖЕТ ПРИСУТСТВОВАТЬ И ВЫПОЛНЯТЬСЯ ТОЛЬКО ОДНА ВЕРСИЯ КОДА.

2. НЕ СУЩЕСТВУЕТ ПУТИ ПОДДЕРЖКИ ЦЕЛОСТНОСТИ МЕЖДУ НАБОРАМИ КОМПОНЕНТОВ, СОБРАННЫХ ВМЕСТЕ, И НАБОРОМ, ПРИСУТСТВУЮЩИМ НА КОМПЬЮТЕРЕ В МОМЕНТ ВЫПОЛНЕНИЯ.

В СУММЕ ЭТИ ДВА МОМЕНТА ОТСЛЕЖИВАНИЯ ВЕРСИЙ ПОРОЖДАЮТ КОНФЛИКТЫ DLL: УСТАНОВКА ОДНОГО ПРИЛОЖЕНИЯ МОЖЕТ НЕПРЕДУМЫШЛЕННО НАРУШИТЬ РАБОТУ ДРУГОГО СУЩЕСТВУЮЩЕГО ПРИЛОЖЕНИЯ, ПОСКОЛЬКУ ОПРЕДЕЛЕННЫЙ УСТАНАВЛИВАЕМЫЙ ПРОГРАММНЫЙ КОМПОНЕНТ ИЛИ DLL НЕ ПОЛНОСТЬЮ СОВМЕСТИМ С ПРЕДЫДУЩЕЙ ВЕРСИЕЙ. ЕСЛИ СКЛАДЫВАЕТСЯ ПОДОБНАЯ СИТУАЦИЯ, В СИСТЕМЕ НЕ НАХОДИТСЯ СРЕДСТВ ПОДДЕРЖКИ, НАПРАВЛЕННЫХ НА ДИАГНОСТИКУ И ИСПРАВЛЕНИЕ ОШИБКИ.

пользуемые пользовательские типы в модуль, который будет загружаться только при необходимости.

Посмотри на картинку. Разработчик гипотетического приложения выделил в отдельный модуль некоторый вспомогательный код, а ресурс большого объема (.bmp-рисунок) оставил в первоначальном файле. Если используется .NET Framework, загрузка файла выполняется только при ссылке. Процесс загрузки кода оптимизируется, если код, ссылки на который используются редко, будет размещен в отдельном файле.

Для файловой системы они являются тремя различными файлами. Отмечу, что файл Util.netmodule был откомпилирован как модуль, так как он не содержит данных о сборке. Когда создавалась сборка, ее манифест был добавлен к файлу MyAssembly.dll, что указывает на связь сборки с файлами Util.netmodule и Graphic.bmp. Когда проектируешь исходный код, реши для себя точно то, каким способом разделить функции приложения между одним или несколькими файлами. Аналогичные решения принимай при проектировании кода .NET Framework: каким способом разделить функции между одной или несколькими сборками.

домены приложений Операционные системы и исполняющие среды обычно содержат определенные средства изоляции приложений друг от друга. Изолировать их необходимо для гарантирования того, что код, выполняемый одним приложением, в процессе своей работы не сможет повлиять на работу других приложений, не связанных с ним. Домены приложений предоставляют платформе .NET безопасные и гибкие блоки обработки, которые могут использоваться для изоляции отдельных приложений. Домены приложений обычно создаются хостами исполняющей среды, которые отвечают за загрузку общеязыковой среды выполнения перед запуском приложения.

Исторически сложилось так, что для разделения приложений, выполняющихся на одном компьютере, используются границы процессов. Каждое приложение загружается в отдельный процесс, который отделяет его от других приложений, выполняющихся на том же самом компьютере. Приложения оказываются отделенными друг от друга, поскольку адреса в памяти привязаны к приложению (бессмысленно использовать в процессе указатель, передаваемый ему из другого процесса).

Кроме того, прямые вызовы между процессами невозможны. Вместо них должен использоваться прокси, который позволяет реализовать межпроцессные вызовы. Перед запуском управляемый код должен пройти процесс проверки, в течение которого определяется, может ли код пытаться обратиться к неверным адресам памяти или делать еще что-то, что привело бы к нарушению правильной работы процесса, в котором выполняется этот код. Код, прошедший проверку, называется строго типизированным. Возможность проверки кода на строгую типизацию позволяет платформе .NET обеспечивать такой же высокий уровень изоляции процессов друг от друга, как при использовании границ процессов, но при значительно более низких затратах по производительности. В одном процессе можно запустить несколько доменов приложений с таким же уровнем изоляции, какой обеспечивают отдельные процессы, но без дополнительных издержек на межпроцессные вызовы или переключение между процессами. Возможность выполнять несколько приложений в одном процессе значительно повышает масштабируемость серверов. Изоляция приложений также играет важную роль в обеспечении безопасности. Например, можно запустить элементы управления нескольких web-приложений в одном процессе обозревателя так, что эти элементы не смогут получить доступ к данным и ресурсам друг друга. Изоляция приложений при помощи доменов приложений хороша тем, что:

1 СБОИ В ОДНОМ ПРИЛОЖЕНИИ НЕ ЗАТРОНУТ ДРУГИЕ ПРИЛОЖЕНИЯ. ТАК КАК СТРОГО ТИПИЗИРОВАННЫЙ КОД НЕ МОЖЕТ ВЫЗВАТЬ СБОИ В ПАМЯТИ, ИСПОЛЬЗОВАНИЕ ДОМЕНОВ ПРИЛОЖЕНИЙ ГАРАНТИРУЕТ, ЧТО КОД, ВЫПОЛНЯЮЩИЙСЯ В ОДНОМ ДОМЕНЕ, НЕ ПОВЛИЯЕТ НА ДРУГИЕ ПРИЛОЖЕНИЯ ПРОЦЕССА.

2 МОЖНО ПРЕКРАТИТЬ ВЫПОЛНЕНИЕ ОТДЕЛЬНЫХ ПРИЛОЖЕНИЙ, НЕ ОСТАНАВЛИВАЯ ПРОЦЕСС ЦЕЛИКОМ. ИСПОЛЬЗОВАНИЕ ДОМЕНОВ ПРИЛОЖЕНИЙ ПОЗВОЛЯЕТ ВЫГРУЖАТЬ КОД, КОТОРЫЙ ИСПОЛЬЗУЕТСЯ ОТДЕЛЬНЫМ ПРИЛОЖЕНИЕМ. ВЫГРУЖАТЬ ОТДЕЛЬНЫЕ СБОРКИ ИЛИ ТИПЫ НЕВОЗМОЖНО — ДОМЕН ВЫГРУЖАЕТСЯ ТОЛЬКО ЦЕЛИКОМ.

3 КОД, ИСПОЛЬЗУЕМЫЙ ОДНИМ ПРИЛОЖЕНИЕМ, НЕ МОЖЕТ ИМЕТЬ НЕПОСРЕДСТВЕННОГО ДОСТУПА К КОДУ ИЛИ РЕСУРСАМ ДРУГОГО ПРИЛОЖЕНИЯ. ОБЩЕЯЗЫКОВАЯ СРЕДА ВЫПОЛНЕНИЯ РЕАЛИЗУЕТ ЭТО РАЗДЕЛЕНИЕ, ПРЕДОТВРАЩАЯ ПРЯМЫЕ ВЫЗОВЫ МЕЖДУ ОБЪЕКТАМИ В РАЗЛИЧНЫХ ДОМЕНАХ ПРИЛОЖЕНИЙ. ОБЪЕКТЫ, ПЕРЕДАВАЕМЫЕ ОТ ДОМЕНА К ДОМЕНУ, КОПИРУЮТСЯ ИЛИ ВЗАИМОДЕЙСТВУЮТ ЧЕРЕЗ ПРОКСИ. ЕСЛИ ОБЪЕКТ КОПИРУЕТСЯ, ВЫЗОВ ЭТОГО ОБЪЕКТА СТАНОВИТСЯ ЛОКАЛЬНЫМ. ТАКИМ ОБРАЗОМ, ВЫЗЫВАЮЩАЯ ПРОГРАММА И ОБЪЕКТ, С КОТОРЫМ ОНА ВЗАИМОДЕЙСТВУЕТ, НАХОДЯТСЯ В ОДНОМ ДОМЕНЕ ПРИЛОЖЕНИЯ. ЕСЛИ ДОСТУП К ОБЪЕКТУ ОСУЩЕСТВЛЯЕТСЯ ЧЕРЕЗ ПРОКСИ, ИДЕТ УДАЛЕННЫЙ ВЫЗОВ ОБЪЕКТА. В ЭТОМ СЛУЧАЕ ВЫЗЫВАЮЩАЯ ПРОГРАММА И ОБЪЕКТ, С КОТОРЫМ ОНА ВЗАИМОДЕЙСТВУЕТ, НАХОДЯТСЯ В РАЗНЫХ ДОМЕНАХ ПРИЛОЖЕНИЙ. МЕЖДОМЕННЫЕ ВЫЗОВЫ ИСПОЛЬЗУЮТ ТУ ЖЕ ИНФРАСТРУКТУРУ УДАЛЕННЫХ ВЫЗОВОВ, ЧТО И ВЫЗОВЫ МЕЖДУ ДВУМЯ РАЗНЫМИ ПРОЦЕССАМИ ИЛИ ДВУМЯ РАЗНЫМИ МАШИНАМИ. ПО СУЩЕСТВУ, ДЛЯ ПОДДЕРЖКИ ПРАВИЛЬНОЙ JIT-КОМПИЛЯЦИИ ВЫЗОВА МЕТОДА МЕТАДАННЫЕ ИСПОЛЬЗУЕМОГО ОБЪЕКТА ДОЛЖНЫ БЫТЬ ДОСТУПНЫ ДЛЯ ОБОИХ ДОМЕНОВ ПРИЛОЖЕНИЙ. ЕСЛИ ВЫЗЫВАЮЩИЙ ДОМЕН НЕ ИМЕЕТ ДОСТУПА К МЕТАДАННЫМ ВЫЗЫВАЕМОГО ОБЪЕКТА, КОМПИЛЯЦИЯ МОЖЕТ ПРИВЕСТИ К СБОЮ С ГЕНЕРАЦИЕЙ ИСКЛЮЧЕНИЯ ТИПА SYSTEM.IO.FILENOTFOUND. МЕХАНИЗМ ОПРЕДЕЛЕНИЯ СПОСОБОВ МЕЖДОМЕННОГО ДОСТУПА ДЛЯ ОБЪЕКТА ЗАВИСИТ ОТ ОБЪЕКТА. ЕСЛИ ХОЧЕШЬ УЗНАТЬ ЧТО-ТО ДОПОЛНИТЕЛЬНО, СМОТРИ ОПИСАНИЕ КЛАССА MARSHALBYREFOBJECT.

ПРОСТРАНСТВО ИМЕН (NAMESPACE) — ЛОГИЧЕСКАЯ ГРУППИРОВКА КЛАССОВ В ЕДИНУЮ ГРУППУ КЛАССОВ, РЕШАЮЩИХ СХОЖИЕ ПО НАЗНАЧЕНИЮ ЗАДАЧИ

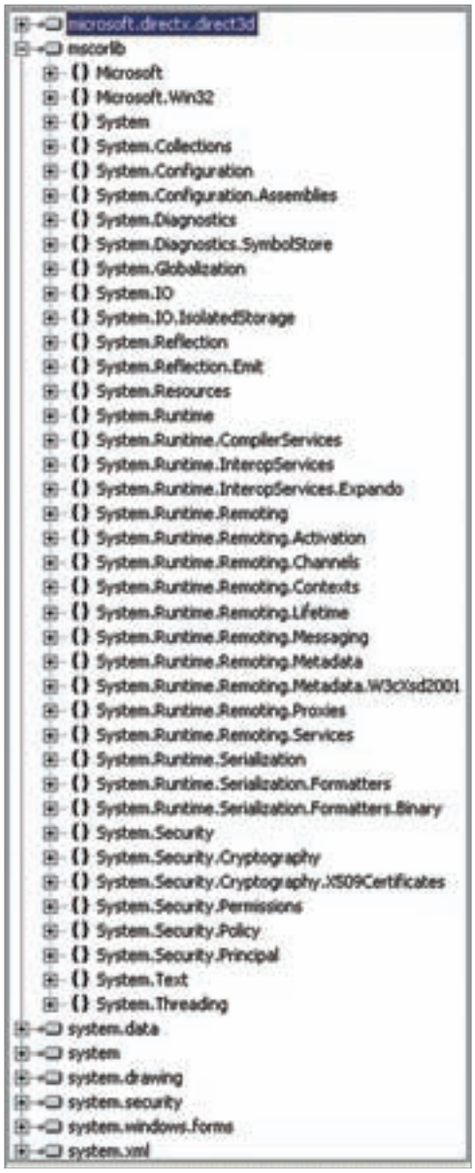
библиотека классов включает в себя средства для решения практически любой задачи. Кратко пробежимся по библиотеке.

Пространство имен System.IO содержит классы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также классы для работы с файлами и папками.

Чтобы продемонстрировать всю простоту написания кода на C#, приведу пример метода, который сохраняет переданный в виде параметра байтовый массив в файл:

```
void SaveByteArray (byte[] buffer)
{
  SaveFileDialog sfd = new SaveFileDialog();
  if (sfd.ShowDialog() == DialogResult.OK)
  {
    System.IO.FileStream fs = sfd.OpenFile();
    fs.Write(buffer, 0, buffer.Length);
  }
}
```

Пространство имен System.Collections содержит интерфейсы и классы, которые определяют раз-



Namespaces

личные коллекции объектов, такие как списки, очереди, двоичные массивы, хэш-таблицы и словари. Пространство имен System.Diagnostics предоставляет классы, позволяющие работать с системными процессами, журналами событий и счетчиками производительности. Пространство имен System.Reflection содержит классы и интерфейсы, обеспечивающие управляемое представление загруженных типов, методов и полей с возможностью динамического создания объектов и вызова методов. Пространство имен System.Security.Cryptography предоставляет службы криптографии, содержащие безопасную кодировку и декодировку данных, а также многие другие операции, например хэширование, генерирование случайных чисел и проверку подлинности сообщений. Пространство имен System.Security.Permissions определяет классы, которые управляют доступом к действиям и ресурсам на основании политики. Пространство имен System.Threading содержит классы и интерфейсы, которые позволяют программировать в многопоточном режиме.

Пространство имен System.Windows.Forms содержит классы для создания приложений Windows, которые получают преимущество при применении возможностей пользовательского интерфейса, доступных в операционной системе Microsoft Windows. Также это пространство имен содержит основные элементы управления, например Button, ListBox, Combobox, CheckBox, Textbox, etc. Пространство имен System.Resources предоставляет разработчику классы и интерфейсы для создания, сохранения и управления различными зависящими от культуры ресурсами, используемыми в приложении.

Из всех классов в пространстве имен System.Resources один из самых важных — класс ResourceManager. Он предоставляет пользователю доступ к ресурсам и управление ресурсами, содержащимися в главной сборке, так же, как и ресурсами сопутствующих сборок. Методы ResourceManager.GetObject и ResourceManager.GetString используются для получения объектов и строк, принятых в данной культуре. Пространство имен System.Data состоит в основном из классов, входящих в архитектуру ADO.NET (она позволяет выполнять сборку компонентов, эффективно работающих с данными из различных источников). В случае отключенного сценария (например Internet) ADO.NET предоставляет средства, позволяющие запрашивать, обновлять и согласовывать данные в многоуровневых системах. Архитектура ADO.NET также использована в приложениях клиента (таких как Windows Forms или страницы HTML), созданных ASP.NET.

System.Web осуществляет поддержку классов и интерфейсов, предназначенных для обеспечения взаимодействия между обозревателем и сервером. Пространство имен включает класс HttpRequest, предоставляющий развернутые сведения о текущем запросе HTTP, класс HttpResponse, управляющий выводом данных HTTP для клиента, и класс HttpServerUtility, с помощью которого можно получить доступ к программам и процедурам сервера. System.Web также включает классы для операций с файлами cookie, для передачи файлов, исключения сведений и управления выходными данными кеширования. И это далеко не полный список!

взломоустойчивость кода .NET-программ довольно высокая. Разработчики больше не обязаны делать проверки на переполнение буфера. Недавно один мой знакомый спросил меня, как в .NET-программах подсчитывается длина (Length) строки (так же, как в программах C++) по 00-байту. Мой ответ был прост: длину строки определяет свойство Length, а не строка определяет его ;). Друг не обрадовался ответу :) — данное обстоятельство делает невозможным переполнение буфера, и в .NET под любую строку будет создан буфер, соответствующий любому размеру строки вне зависимости от ее содержания. Даже если кто-то найдет способ внедрить шелл-код, то потенциальных хакеров определенно будут ждать сложности. В .NET-программах запрещено выполнение кода в хипах (heaps) и в стеке, что обессмысливает саму возможность переполнения. Кроме того, есть дополнительные средства защиты (я уже упомянул их в начале статьи): атрибуты привилегий кода, с помощью которых программе запрещают выполнять потенциально опасные действия, например можно запретить доступ программы к реестру поставив атрибут: [System.Security.Permissions.RegistryPermission(System.Security.Permissions.SecurityAction.Deny)]. Если гипотетический шелл-код попробует обратиться к реестру в контексте данной программы, то возникнет исключение. Более того, запретив обращение к реестру во всей программе, можно определить атрибут для класса или метода, которым все же будет разрешен доступ к реестру, и то только к определенным ключам :), [RegistryPermissionAccess.Write Or RegistryPermissionAccess.Read, "HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\FloatingPointProcessor\0")]

в заключение расскажу о том, что платформа .NET захватывает расположение разработчиков все больше. По непроверенным данным, то, что последует за Windows Vista, будет целиком написано на .NET. Уже сейчас в этом направлении ведутся экспериментальные разработки. Чтобы ознакомиться с ними, посмотри **http://research.microsoft.com/os/singularity**. Программирование в .NET исключает множество ошибок благодаря хорошо продуманному ООП, а принудительно-рекомендательный характер платформы делает код более понятным и, соответственно, в перспективе облегчает его поддержку. Я уже забыл о тех временах, когда после добавления незначительной детали в программе ты получал целую серию багов, для отладки которых тратил неделю на отладку программы. И часто хотелось переписать все с нуля.

В общем, потратив немного времени на изучение .NET, ты вряд ли разочаруешься

Visual Studio .Net GUI — разработаем это так, как принято в XXI веке :)

мнение профессионалов

«ВСЕ УЖЕ НАПИСАНО ДО НАС»

СПЕЦ: ДЛЯ КАКОЙ ПЛАТФОРМЫ ПРОГРАММИРОВАНИЕ СЕЙЧАС НАИБОЛЕЕ ВОСТРЕБОВАНО?

АНАТОЛИЙ СКОБЛОВ: Судя по предложениям работы, в первую очередь — Java во всех вариантах. Во вторую очередь — .NET. Потом — все остальное.

АНТОН ПАЛАГИН: Скорее, востребован гетероморфный код, способный компилироваться и работать на довольно широком спектре платформ. Связано это со стремлением к оптимизации расходов на владение кодом. Ни одна современная информационная система не нацелена на конкретную платформу, ее серверная часть часто создается под конкретную ОС, а клиенты создаются для множества платформ.

АЛЕКСАНДР ПОЛУЭКТОВ: Конечно же, для Windows! Все-таки объем ПО (включая игры) для нее гораздо больше, чем для Linux. И, несмотря на то, что объемы ПО для Linux растут с каждым годом, программирование под Windows приносит бОльший доход как компаниям, так и частным разработчикам.

НИКИТА БУРЦЕВ: А какая платформа сейчас наиболее распространена? Windows. Меня, как представителя лагеря open source, враждующего с Microsoft, это не может не расстраивать, но объективная реальность сейчас такова. Практически везде стоит Windows, и большую часть софта пишут именно под нее. Хотя… В последнее время все чаще разрабатывают различного рода web-сервисы, которые можно в некотором приближении назвать платформонезависимыми, по крайней мере на уровне клиента. Думаю, дальше приоритет еще больше сместится в эту область.

ЗАРАЗА: Сейчас программирование для любой платформы востребовано, тем более что для хорошего программиста переход на новую платформу не должен составить большого труда. А проектировать программу всегда надо так, чтобы код легко переносился на любую платформу.

КРИС КАСПЕРСКИ: «Потребность» есть борьба двух противоположностей: чем популярнее платформа, тем труднее написать под нее что-то новое. Платформа номер один (в России) — это Windows, под нее Microsoft толкает семейство .NET-языков, без знания которых устроиться в контору средней паршивости скоро будет невозможно. Если, конечно, не знать Delphi, позиции которого по-прежнему очень сильны. Однако что можно нового написать? Ничего! Все уже написано задолго до нас. А даже если написать, то как это продавать? (Учтем наш менталитет.) Короче говоря, на софтверном рынке Windows сейчас глубокий застой, а вот на сегменте сетевого программирования — большое оживление, Perl стагнирует, на смену ему приходит PHP и другие скриптовые языки типа парсера. Ну, естественно, ASP и Java.

Однако потребность в программистах сокращается с каждым днем. Все уже запрограммировано, и теперь остается только «лабать, бухать и дизайнить». Мобильные приложения — это та область, в которую можно прийти с улицы. Буквально через месяц выбрасываешь на рынок продукт, которым будут пользоваться миллионы! В основном спросом пользуются видеоигры, которые на Java не напишешь (производительности не хватит), а топовые проекты делаются при помощи бинарного транслятора Brew и оптимизированного ассемблера. Кстати (раз уж затронули игры), интерес к игровым приставкам за последнее время значительно возрос, а программистов, способных писать под них что-то толковое, не так уж и много... Linux тоже на гребне славы и популярности. В нем просто нет множества повседневных программ! Еще писать и писать...

ДМИТРИЙ СОШНИКОВ: Безусловно, эта платформа — Microsoft .NET. Наблюдается реальный дефицит профессионалов в этой области, поскольку платформа сравнительно «молодая» и нужно время, чтобы «переучиться». Microsoft сейчас не только лидирует на рынке персональных операционных систем, но и ведет активное наступление на серверных платформах, ток что в ближайшее время доля web-приложений на основе .NET будет только увеличиваться. Не стоит забывать также о выходе Windows Vista в конце этого года, в ней принципы программирования будут несколько отличаться, поэтому действительно хорошему специалисту не мешает «смотреть в будущее» уже сейчас.

СПЕЦ: КАКИЕ ПЕРСПЕКТИВЫ ОТКРЫВАЕТ ПЕРЕД ПРОГРАММИСТОМ ПЛАТФОРМА .NET?

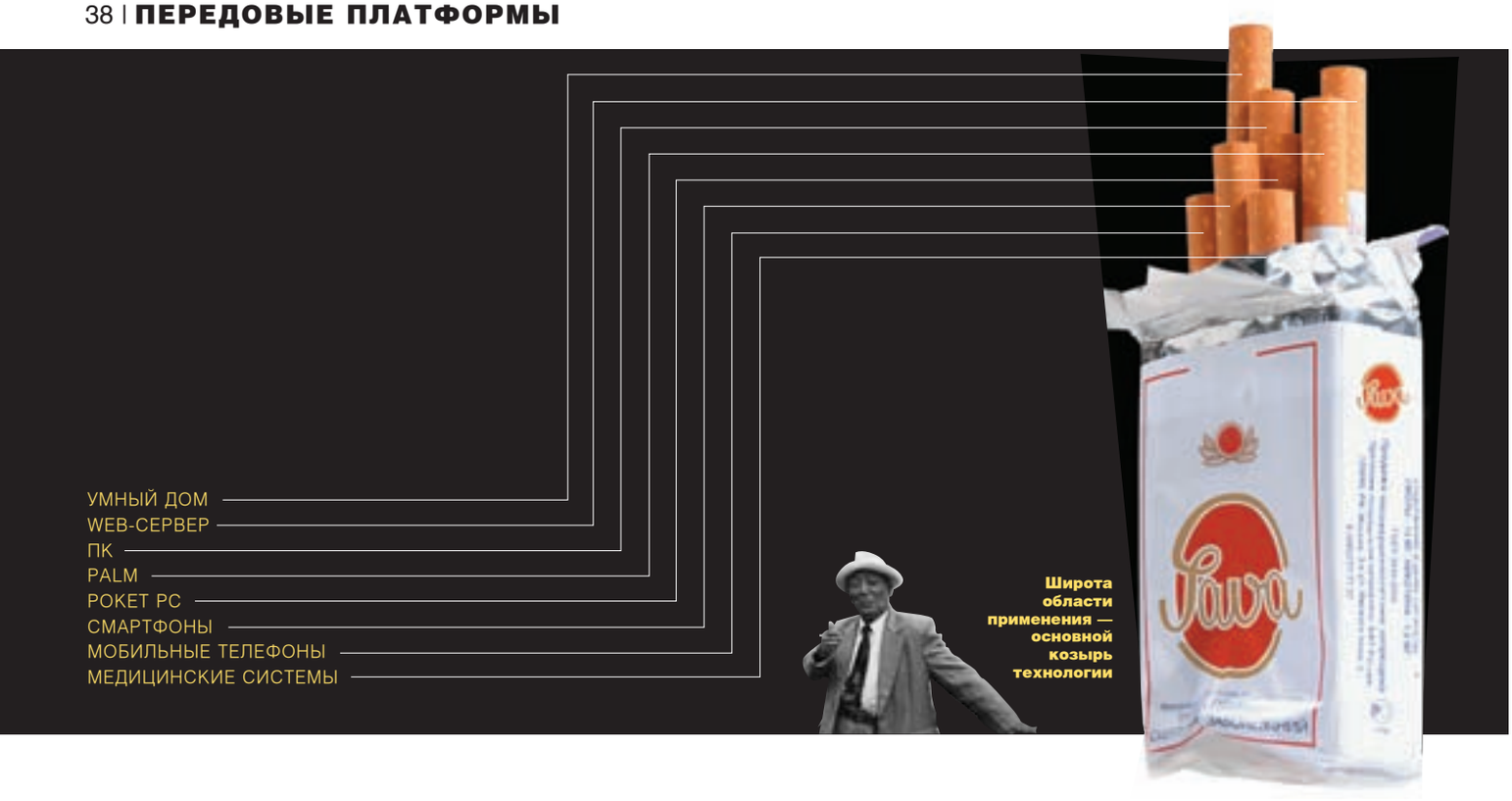
АНАТОЛИЙ СКОБЛОВ: Для большинства — много однообразной нетворческой работы в рамках «конвейерного» процесса разработки в компаниях, специализирующихся на мейнстрим-разработках. Как и раньше. Для маленьких компаний, одиночек и хакеров — никаких новинок и перспектив, только еще один язык программирования, полезный инструмент для отдельных задач, не более.

АНТОН ПАЛАГИН: Упрощение разработки прикладного ПО и распределенных систем. Также большие надежды связаны с mono — открытой реализацией .NET под *nix. Эта технология позволит упростить написание кросс-платформенного кода.

КРИС КАСПЕРСКИ: Лучше сказать, какие проблемы она открывает перед ним ;). Используя .NET, ты подсаживаешься на иглу Microsoft, попадаешь в зависимость от ее операционных систем, которые становятся уже практически неуправляемыми и валятся в пропасть. Я, например, в ближайшие несколько лет собираюсь полностью перейти на BSD, и потому .NET мне совершенно неинтересна.

НИКОЛАЙ АНДРЕЕВ: Если рассуждать с точки зрения программиста как создателя саморазмножающихся механизмов (читай «вирусов»), то перспективы очень даже симпатичные. Платформа .NET — в этом плане просто удивительная вещь. Ты только представь: полностью кросс-платформенный вирус. Вернее, не кросс-платформенный, а .NET-платформенный. Framework'и уже понаделаны к целой куче JC и аппаратных платформ. Под MacOS уже есть билд, под Linux — есть, уже не говоря о мобильных девайсах и т.п. При должном качестве реализации framework'ов программу, скомпилированную в .NET-код, можно будет запустить хоть на кофеварке. Через годик-два, после повсеместного введения .NET, вирусописатели, наверное, будут молиться на Microsoft за такую классную штуку. Телефон под .NET будет заражаться по MMS, потом в Сеть попадут все устройства в радиусе действия Bluetooth (к тому времени уже версии 2.0, то есть в радиусе 100 метров), КПК, ноутбуки и т.д. И хорошо если такой вирус напишут мирным, просто чтобы он собирал статистику, а не совершал какие-нибудь злобные действия, не предусмотренные защитой платформы .NET (29A — **http://29.vx.netlux.org**). А вообще .NET — это намного больше, чем кросс-платформенность для вирусов. Я говорю о том, что волнует меня больше всего. Стоит только на ASP.NET взглянуть! Теперь сайты не будут делать ни на чем другом. Особенно синтаксис C# (невероятно удобного языка) в HTML-вставках порадовал. Супер!

ДМИТРИЙ СОШНИКОВ: Очень важен тот факт, что платформа .NET предоставляет сборку мусора. Как известно, программирование на языках со сборкой мусора и без указателей, в классическом их понимании, позволяет повысить производительность разработки ПО на 30%. Также следует отметить, что .NET обеспечивает прозрачную интероперабельность между языками: можно в одном проекте использовать и традиционные языки, и более экзотические и удобные (например, функциональный язык SML или тот же Prolog). Добавим сюда поддержку промышленных стандартов (web-сервисов, XML и др.), защищенность управляемого кода, удобные средства разработки — и получим платформу, идеальную для решения подавляющего большинства современных задач!



дружественная ява

ЗАВОЮЕТ ЛИ JAVA МИР

ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ И ТЕЛЕВИЗОРЫ, МОБИЛЬНЫЕ ТЕЛЕФОНЫ И ПРОМЫШЛЕННЫЕ СЕРВЕРЫ, ГЕОСТАЦИОНАРНЫЕ СПУТНИКИ И СМАРТ-КАРТЫ... РАЗНООБРАЗИЕ УСТРОЙСТВ ОГРОМНО. ЕДИНСТВЕННОЕ, ЧТО ОБЪЕДИНЯЕТ ИХ, — ТО, ЧТО ОНИ ОТНОСЯТСЯ К СФЕРЕ ПРИМЕНЕНИЯ ПЛАТФОРМЫ JAVA |ЕВГЕНИЙ AKA SATURN (SATURN@LINKIN-PARK.RU; ICQ 587692)

Почему же язык Java стал завоевывать мир с такой потрясающей скоростью? Если десять лет назад со словом «Java» ассоциировалось в первую очередь слово «остров», то сейчас с «Java» ассоциируется, конечно, платформа от Sun Microsystems. Что же произошло за последние десять лет в сфере ИТ и что может измениться в будущем, если эта технология продолжит развиваться так же бурно?

Случилось то, что люди описывают емкой фразой «Мир изменился». Сетевые технологии распространились повсеместно. Сегодня мы не можем представить себе компьютер без Сети так же, как и жизнь без компьютера. Интернет добрался до мобильных телефонов, телевизоров и улиц города. Чтобы понять, почему Java распространилась настолько широко, переместимся в 1994 год.

Компьютеры уже достаточно мощные, но еще нет интернета в его современном виде, привычном для нас. Есть несколько операционных систем, и каждая из них представляет собой замкнутую систему в смысле прикладного программирования. Есть широкий выбор бытовой электроники, но каждая модель построена на собственной архитектуре и элементной базе. В Sun Microsystems возникает проект создания программного обеспечения для различных бытовых приборов. Начав реализацию на C++, компания столкнулась с проблемой несовместимости архитектур. В рамках традиционных языков программирования приходилось компилировать программы отдельно под каждую архитектуру, причем если производитель выпускал новую модель с небольшими изменениями по сравнению с предшествующими продуктами, приходилось переписывать приложение заново. Кроме того, поддержка сотен различных интерфейсов — очень трудоемкое и дорогое занятие.

ОДНО ИЗ ГЛАВНЫХ ПРЕИМУЩЕСТВ JAVA ПЕРЕД ДРУГИМИ ТЕХНОЛОГИЯМИ — СИСТЕМОНЕЗАВИСИМОСТЬ

Чтобы преодолеть подобные трудности, специалисты Sun пошли по пути создания нового языка программирования. В него были заложены принципы платформонезависимости, многопоточности, объектной ориентации. «Написано однажды — работает везде», — вот девиз Java.

Как известно, программа, написанная на одном из языков высокого уровня (так называемый исходный модуль), не может быть выполнена сразу. Сначала производится процесс компиляции, то есть перевод в последовательность машинных команд — объектный модуль, и часто конечный результат получают даже не на этом этапе. Далее необходимо скомпоновать полученный кусок машинных кодов с библиотеками функций, используемых в модуле, и разрешить перекрестные

этапы рождения языка

ДЕКАБРЬ 1990 ГОДА — НАЧИНАЕТСЯ РАБОТА НАД ПРОЕКТОМ GREEN В КОМПАНИИ SUN
ИЮНЬ 1991 ГОДА — РАЗРАБАТЫВАЕТСЯ ИНТЕРПРЕТАТОР OAK, ЧЕРЕЗ НЕСКОЛЬКО ЛЕТ ОН ПЕРЕИМЕНОВАН В JAVA
МАРТ 1992 ГОДА — К ПРОЕКТУ GREEN ПРИСОЕДИНЯЕТСЯ ДЖОНАТАН ПЕЙН, КОТОРЫЙ ПОЗДНЕЕ УЧАСТВУЕТ В НАПИСАНИИ HOTJAVA
ИЮНЬ 1994 ГОДА — НАЧАТ ПРОЕКТ LIVEOAK, НАЦЕЛЕННЫЙ НА ИСПОЛЬЗОВАНИЕ OAK В КРУПНОМ ПРОЕКТЕ НЕБОЛЬШОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ
ИЮЛЬ 1994 ГОДА — ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОЕКТА LIVEOAK ПЕРЕОРИЕНТИРОВАНА НА ИНТЕРНЕТ
СЕНТЯБРЬ 1994 ГОДА — ПОЯВЛЯЕТСЯ WEBRUNNER — БРАУЗЕР ТИПА MOSAIC, ПОЗДНЕЕ ПЕРЕИМЕНОВАННЫЙ В HOTJAVA
СЕНТЯБРЬ 1994 ГОДА — ВПЕРВЫЕ ПРОДЕМОНСТРИРОВАН ПРОТОТИП HOTJAVA
ОСЕНЬ 1994 ГОДА — РЕАЛИЗОВАН КОМПИЛЯТОР JAVA НА ЯЗЫКЕ JAVA (РАНЕЕ ОН БЫЛ РЕАЛИЗОВАН НА ЯЗЫКЕ C)
МАЙ 1995 ГОДА — КОМПАНИЯ SUN ОФИЦИАЛЬНО ПРЕДСТАВИЛА JAVA И HOTJAVA НА ВЫСТАВКЕ SUNWORLD '95

ссылки между секциями. В результате мы получаем загрузочный модуль — готовую к выполнению программу. Поскольку Java является языком высокого уровня, ему не чуждо ничто из названного мной. Однако для реализации принципа многоплатформенности в языке применяется элемент, отсутствующий в «классических» языках, — виртуальная машина.

виртуальная машина Java (JVM) — это краеугольный камень Java, это компонент, отвечающий за межплатформенную доставку, малый размер компилированного кода и возможность Java защитить пользователей от вредоносных программ. Это абстрактная вычислительная машина, она имеет набор инструкций и использует различные области памяти. Итак, JVM выполняет функцию «промежуточного слоя» между пользовательской программой и аппаратной платформой.

Исходный модуль, написанный на Java, компилируется в команды виртуальной машины. Она полностью стековая, так что не требуется сложной адресации ячеек памяти и множества регистров. В результате команды JVM получаются короткими, их средняя длина составляет 1,8 байта (большинство команд имеют длину 1 байт, поэтому их называют байт-кодами). Еще одна особенность Java — то, что стандартные функции, вызываемые в программе, не включаются в байт-коды. Вместо этого используется процедура, называемая динамической компоновкой, — подключение функций лишь на этапе выполнения. Так уменьшается объем откомпилированной программы.

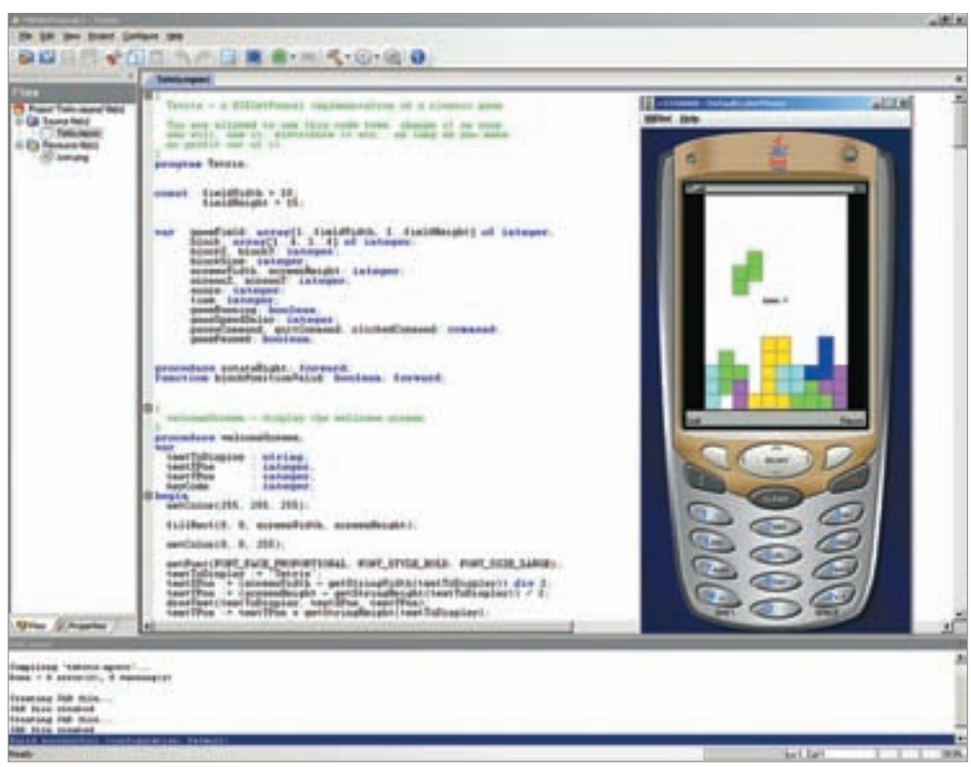
На первом этапе программа, написанная на языке Java, переводится компилятором в байт-коды. Эта компиляция не зависит от архитектуры компьютера, типа процессора и операционной системы. Байт-коды записываются в один или несколько файлов, и над ними (именно в таком виде) могут производиться стандартные операции (запись на внешний носитель, передача по Сети, копирование). Далее можно выполнять байт-коды на любой системе, реализующей Java Virtual Machine. Получается, что виртуальная машина «берет на себя» все особенности той или иной архитектуры, что позволяет запускать одно и то же приложение на персональном компьютере, КПК и мобильном телефоне.

к минусам архитектуры можно отнести медленное выполнение программ, причину которого нужно искать в интерпретации байт-кодов и динамической компоновки. В общем случае для выполнения Java-программ требуется вычислитель более мощный (здесь речь идет не о компьютере, а о любой вычислительной машине), чем для аналогичных задач на C/C++. Но Sun не стоит на месте и постоянно совершенствует интерпретаторы, стремясь повысить скорость. Существуют JIT-компиляторы (Just-in-Time), которые запоминают уже интерпретированные участки кода в машинных командах процессора и выполняют эти участки при повторном обращении, например, в циклах. Результат — значительное увеличение скорости повторяющихся вычислений.

реализации виртуальных машин Java созданы практически для всех платформ. Для самых распространенных существуют реализации от разных фирм-производителей. Все больше операционных систем и СУБД включают в свое ядро реализацию JVM. Практически во все браузеры встроена виртуальная машина Java для выполнения апплетов.

Несмотря на революционную архитектуру, появление Java (тогда еще Oak) не сопровождалось фурором в среде программистов. Однако ситуация изменилась после появления World Wide Web. В тот момент главный идеолог языка Патрик Нотон предложил использовать Java в среде WWW, был создан Java-браузер — WebRunner (позже переименован в HotJava). Шло время, Сеть проникала все глубже в человеческую жизнедеятельность, понадобилось подключать целый «зоопарк» устройств. Стали появляться различные Java-платформы. На этом моменте заканчивается история и начинается «сегодня», то есть наша реальность — существование нескольких платформ для различных применений. В 1999 году Sun анонсировала платформенные версии Java 2: Enterprise (J2EE), Standard (J2SE), Micro (J2ME).

J2ME расшифровывается как Java 2 Micro Edition. Эта платформа получилась в результате упрощения J2SE (платформа для создания приложений для ПК) и добавления специфических функций, важных для мобильных устройств. Основная область применения J2ME — игры. Однако с помощью Java-приложений также читают электронные книги, просматривают интерактивные карты, новости и финансовые сводки...



Пример реализации игры «Тетрис» на мобильном телефоне

совместное использование

ЯЗЫК JAVA МНОГИМ ОБЯЗАН C/C++, БЫЛИ ПОЗАИМСТВОВАНЫ СИНТАКСИС И БАЗОВАЯ СЕМАНТИКА. ОДНАКО СВЯЗЬ МЕЖДУ НИМИ НЕ ОГРАНИЧИВАЕТСЯ ТОЛЬКО ЭТИМ. ИСПОЛЬЗУЯ JNI (JAVA NATIVE INTERFACE), МОЖНО ВЫЗЫВАТЬ C/C++ ФУНКЦИИ ИЗ JAVA-ПРОГРАММЫ. ИЛИ, НАОБОРОТ, ИЗ ПРОГРАММЫ, НАПИСАННОЙ НА C/C++, МОЖНО СОЗДАВАТЬ JAVA-ОБЪЕКТЫ ЛИБО ВЫЗЫВАТЬ JAVA-МЕТОДЫ. НО ИСПОЛЬЗОВАНИЕ JNI В БОЛЬШИНСТВЕ СЛУЧАЕВ ВЕДЕТ К ПОТЕРЕ МНОГОПЛАТФОРМЕННОСТИ JAVA-КОДА. ХОТЯ, С ДРУГОЙ СТОРОНЫ, РАСШИРЯЕТСЯ СФЕРА ПРИМЕНЕНИЯ САМОГО ЯЗЫКА JAVA В ПРИЛОЖЕНИЯХ, ДЛЯ КОТОРЫХ ЭТО УСЛОВИЕ НЕ ЯВЛЯЕТСЯ НЕОБХОДИМЫМ. ТОГДА ОПРАВДАНО СОЧЕТАНИЕ СОВРЕМЕННОГО ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА JAVA С СУЩЕСТВУЮЩИМ СИСТЕМНО-ЗАВИСИМЫМ КОДОМ НА C/C++, ЧТО НЕОБХОДИМО ДЛЯ ПЕРЕХОДА К ИСПОЛЬЗОВАНИЮ JAVA ПРИ РАЗРАБОТКЕ КОМПОНЕНТОВ СЕРВЕРА.

ТАКИМ ОБРАЗОМ, JNI ЯВЛЯЕТСЯ ЕСТЕСТВЕННЫМ ДОПОЛНЕНИЕМ JAVA-ТЕХНОЛОГИИ, ПОЗВОЛЯЯ ИСПОЛЬЗОВАТЬ ЕЕ КАК ДЛЯ СОЗДАНИЯ ПЕРЕНОСИМЫХ (КЛИЕНТСКИХ) ПРИЛОЖЕНИЙ, ТАК И ДЛЯ СОЗДАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ (СЕРВЕРНЫХ) СИСТЕМ, ИСПОЛЬЗУЮЩИХ ВСЮ СПЕЦИФИКУ КОНКРЕТНОЙ ПЛАТФОРМЫ И АППАРАТУРЫ, СОХРАНЯЯ В ТО ЖЕ ВРЕМЯ ГЛАВНОЕ ДОСТОИНСТВО JAVA — СОВРЕМЕННЫЙ ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД.

Кроме того, существование мидлетов (Java-приложений для мобильных устройств) намного повышает ценность Java в сфере мобильных устройств. J2ME хороша именно своей популярностью. Однако существует и серьезный недостаток: заявленная концепция кроссплатформенности в действительности реализована не полностью. Дело в том, что аппаратные части телефонов существенно различаются, поэтому каждый производитель оптимизирует J2ME для более быстрой работы на устройствах именно собственного производства. Иногда такая «оптимизация» делает невозможной работу мидлетов на устройствах сторонних фирм. Кроме того, не забудем, что изначально платформа планировалась совсем не как игровая, но несмотря на это заметную долю Java-приложений сейчас составляют игры. Результат — медленное выполнение приложений, для борьбы с которым фирмы-производители пытаются создать оригинальную платформу, ориентированную на игры. Самой заметной из результатов этих попыток является Mophun от компании Synergenix. Mophun-приложения работают на порядок быстрее своих аналогов для Java, но как обратную сторону медали мы получаем низкую популярность платформы. Среди «независимых» платформ можно назвать и ExEn от компании InFusio и BREW от Qualcomm.

Для мобильных устройств существует множество разнообразных Java-игр

Можно сказать, что применение J2ME в мобильных устройствах буквально вселяет в них душу. Мобильные телефоны позволяют выполнять задачи, которые раньше были уделом КПК и персональных компьютеров. Следовательно, если мы промолчим о некоторых недостатках (главный из них — частичная несовместимость), можно сказать, что J2ME значительно расширяет круг возможностей мобильных устройств.

технология Java долгое время не рассматривалась в качестве серьезной платформы для корпоративных приложений. Ее уделом считались многоплатформенные клиентские апплеты, в основном для интернета. Однако со временем пришло понимание того, что Java может стать оптимальным выбором при разработке кроссплатформенного корпоративного программного обеспечения. Так появилась Java 2 Platform Enterprise Edition (известна также как J2EE) — набор взаимосвязанных спецификаций и технологий, предназначенных для разработки, развертывания и управления многозвенными приложениями, ориентированными на серверную архитектуру. Опять же, одним из главных преимуществ по сравнению с другими технологиями здесь является системонезависимость. Богатый набор готовых компонентов позволяет решать самые разнообразные задачи — от создания GUI (например графические интерфейсы СУБД Oracle) до разработки распределенных приложений в гетерогенных сетях. Кроме того, значительное внимание уделено проблемам взаимодействия приложений в сети. Возможности Java в корпоративном секторе настолько широки, что компания Sun Microsystems создала на основе этой технологии сервер приложений — Sun Java System Application Server, который, по оценкам аналитиков, успешно конкурирует с аналогичными решениями мировых лидеров в этой области (Microsoft, Oracle, SAP).

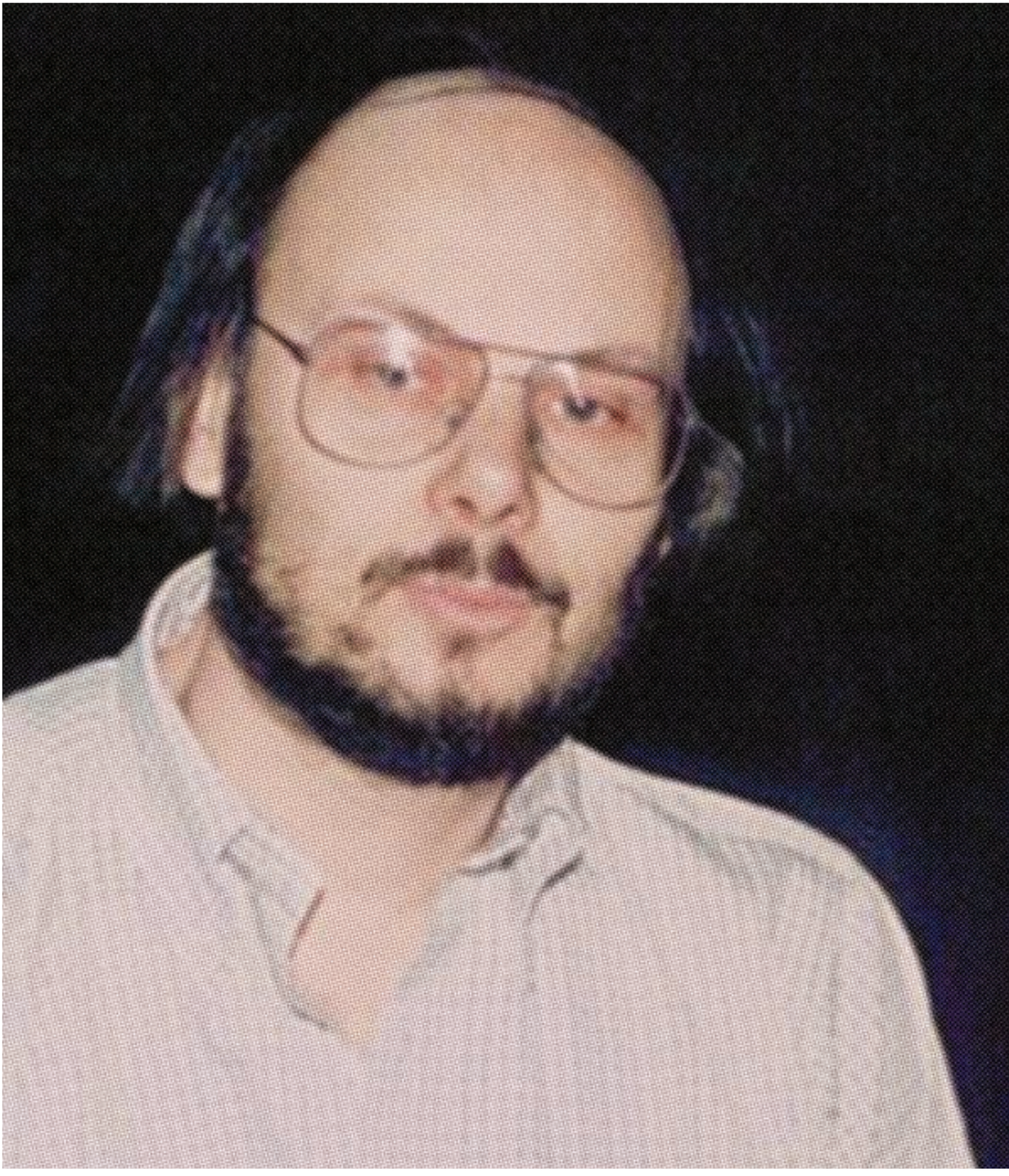
завоюет ли Java мир? За более чем десять лет своего развития Java превратилась в одну из самых популярных программных платформ. Java-технологии проникли в самые разнообразные области — от телефонов до космических кораблей. По количеству разработчиков, работающих с этой технологией, Java приблизилась к безусловному лидеру всех времен и народов — Visual Basic. Кроссплатформенность, многопоточность, универсальность — эти особенности языка обеспечивают огромные возможности развития в будущем. Однако технология имеет и несколько существенных недостатков. Главный из них — медленное выполнение программ и слишком высокая требовательность к аппаратным ресурсам. Если Sun Microsystems ликвидирует эти слабые моменты, Java получит все шансы завоевать мир сетевых вычислений

Иерархия элементов в Java Standard Edition начиная с уровня ОС

	Solaris	Linux	Windows	Other
Development tools & API's	java Compiler	java Debuger	javadoc	jPDA
Deployment technologyes	java Web Start		java Plug-in	
User interface toolkits	Swing		AWT	
	Sound	Input Methods	java 2D	Accessibility
Integration API's	RMI	jDBC	jNDI	COBRA
Core API's	XML	Logging	Beans	Locale Support
	Perferences	Collections	jNI	Security
	Lang	Until	New I/O	Networking
Java Virtual Machine	java Hotspot Client Compiler		java Hotspot Server Compiler	
	java Hotspot VM Runtime			

БЬЯРН СТРАУСТРУП

Придумал C++. Правда, название C++ родилось в голове Рика Масситти. Название указывает на эволюционную природу перехода от C. «++» — операция приращения в C. Чуть более короткое имя C+ является синтаксической ошибкой :). Знатоки семантики C находят, что C++ хуже, чем ++C. От названия «D» отказались, поскольку язык является расширением C. Изначально C++ был разработан для того, чтобы автору и его друзьям не приходилось программировать на Ассемблере, C или других современных языках высокого уровня. Его основное предназначение — упростить написание хороших программ и сделать его более приятным для программиста. Плана разработки C++ на бумаге никогда не было. Проект, документация и реализация двигались одновременно. Разумеется, внешний интерфейс C++ был написан на C++.



ВНИМАНИЕ!!!
НАПОМИНАЕМ, ЧТО ИНФОРМАЦИЯ В СТАТЬЕ ПРЕДОСТАВЛЕНА ТОЛЬКО ДЛЯ ДЕМОНСТРАЦИИ УЯЗВИМОСТИ ОПИСАННЫХ СИСТЕМ. НИ АВТОР СТАТЬИ, НИ РЕДАКЦИЯ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ЗА ПРИМЕНЕНИЕ ДАННОЙ ИНФОРМАЦИИ НА ПРАКТИКЕ!

штурм зимнего .NET'а

РЕВЕРСИНГ .NET FRAMEWORK-ПРИЛОЖЕНИЙ И КОМПОНЕНТОВ

ЧЕМ ЖЕ ОТЛИЧАЕТСЯ РЕВЕРСИНГ .NET FRAMEWORK-ПРИЛОЖЕНИЙ И КОМПОНЕНТОВ (ДАЛЕЕ ПРОСТО «.NET») ОТ ОБЫЧНЫХ ПРОГРАММ, НАПРИМЕР, СКОМПИЛИРОВАННЫХ КОМПИЛЯТОРОМ C\C++ (ДАЛЕЕ «NATIVE»)? ЧТОБЫ ОТВЕТИТЬ НА ЭТОТ ВОПРОС, НЕМНОГО ПОКОПАЕМСЯ В ТЕОРИИ |

NIM[INT3 TEAM](NIM@INT3.RU)

Программы, написанные для платформы .NET, компилируются в первую очередь не в native, а в некий псевдокод, названный MSIL (MicroSoft Intermediate Language). Чтобы преуспеть во взломе .NET, крэкер должен быть знаком с этим языком — здесь, пожалуй, кроется основная сложность.

Кстати, хотя я употребил термин «псевдокод», я совсем не пытался сказать, что IL выполняет некий интерпретатор (как в программах VB6, откомпилированных с ключом P-CODE). IL-код программы компилируется перед запуском (точнее, компилируется метод, помеченный атрибутом .entrypoint), остальной код — перед его использованием. Эта технология называется JIT (just in time) — «компиляция во время выполнения», но, несмотря на название, возможно осуществить ее и во время установки программы или даже вручную с помощью утилиты ngen (входит в дистрибутив .NET). IL немного похож на ассемблер тем, что каждая команда имеет не более двух операндов и присутствует работа со стеком. Стек .NET представляет собой типизированную реализацию стека в ассемблерном понимании, то есть в стек можно записывать и считывать объекты определенного типа (ясное дело, нельзя записать в стек dword, а считать из него — word :)). Если захочешь изучить IL подробнее, начни с чтения доки от создателя (**http://msdn.microsoft.com/library/rus/default.asp?url=/library/RUS/cpref/html/frlrfSystemReflectionEmitOpCodesMembersTopic.asp**), а мы пока двинемся дальше.

метаданные в .NET имеют ключевое значение, в них содержится вся информация о классах, методах, свойствах и полях, включая их оригинальное название, то есть то название, которое для них определил разработчик. Именно такое положение вещей делает .NET особо уязвимым декомпиляции.

На сегодня написано немало декомпиляторов, которые в точности показывают исходный код программы. Среди них, наверное, выделю только Reflector, который помогает быстро разобраться в логике программы на одном из .NET-языков. Для полного восстановления исходного кода используется плагин Reflector.FileDisassembler. Могу заверить, что выделить места для модификации с помощью Reflector'a и затем сделать патч в IL намного проще, чем добиваться компилируемости декомпилированных исходников. Декомпиляция в любом случае происходит неидеально, и возникает куча нюансов, которым необходимо уделить время.

strong key (далее «SN») — это криптостойкая (RSA1024) цифровая подпись .NET-программ, которую разработчик может внедрить в свою программу. SN служит для удостоверения факта того, что программа не была модифицирована кем-либо. Если программа была подписана, то в дизассемблерном листинге, среди прочих атрибутов сборки, мы увидим следующее (см. врезку справа).

В редких случаях после модификации требуется подписать сборку (Solution — еще одно название .NET-программ) своей SN. На этот случай существует утилита sn.exe, входящая в дистрибутив .NET framework. Для генерации ключа: sn -k 1.key, для подписи: sn -i MyApp.exe 1.key.

прежде чем говорить о взломе, нужно немного сказать и о светлой стороне силы :). Потому что больше она пока не заслужила :). Итак, современная protect .NET-индустрия добилась выпуска двух технологий обфускации и упаковки/криптования.

Принцип обфускации .NET заключается в простом переименовании названий классов, методов, свойств и полей на названия типа «_1_1», «_1_2» и т.д., и таким образом якобы затрудняется анализ программы. На самом же деле нередко последствия бывают прямо противоположные ;). Например, разработчики дали классам страшные названия, содержащие слова Protected или Security, что заставило нас просматривать код особо бдительно. Соответственно, мы потратили кучу времени и вдруг сделали ошеломляющее открытие: когда разработчик использовал пространство имен System.Security.Permissions или System.Security.Policy, он решал очень мирные задачи, и его действия никак не были связаны с защитой кода.

Несколько горе-упаковщиков, созданных также для защиты .NET, не тянут даже на средний уровень безопасности. Они бережно пакуют и криптуют .NET-программу, не подозревая о том, что нам достаточно лишь подождать распаковки ;). Ни о каких современных техниках защиты с применением драйверов не может быть и речи — это связано с архитектурными особенностями платформы .NET. Оригинальный (некриптованный/упакованный) модуль может потребоваться платформе в любой момент, например, если нужно определить некий атрибут кода или если программисты, решая свои прикладные задачи, используют пространство имен System.Reflection. Я уже не говорю о сериализации и ремоутинге, где рефлекция используется повсеместно. В заключение можно лишь сказать, что Microsoft сделала отличную современную платформу для разработки, которую, к сожалению, пока невозможно защитить ни в статике, ни в динамике :).

Тот самый сайт

рассмотрим технику взлома во всех ее тонкостях. Первый шаг — это дизассемблирование: ildasm MyApp.exe /out:MyApp.h. На выходе появится файл MyApp.h, в котором содержится IL-листинг программы. Почему *.h? Такие файлы понимает Visual Studio, она подсвечивает синтаксис и позволяет закомментировать сразу несколько строк кода. Удобство налицо. Комментарии в IL тоже представляют собой // и /* */. Забегая вперед, скажу, что после изучения и модификации кода необходимо сделать ассемблирование: ilasm MyApp.h.

Особо отмечу, что после дизассемблирования в каталоге, где находится MyApp.h, создаются и файлы ресурсов, поэтому при ассемблировании не нужно перечислять их в командной строке ilasm — уже перечислено в файле MyApp.h в виде атрибутов: .mresource public MyApp.Form1.resources. Если в файле присутствует SN, нужно убрать его из листинга, что делается простым удалением атрибутов .AssemblyKeyFileAttribute и .publickey (их пример см. выше).

Точка входа в программу — это метод, помеченный атрибутом .entrypoint, именно с него стоит начинать исследование. Также в каждом .NET-классе есть точка входа, помеченная атрибутом .ctor или .cctor (эти методы называют конструкторы). В 99,9% случаев код, интересный для реверсинга, находится именно в этих методах. Конструкторы вызываются в момент создания объекта из класса. Пример: newobj instance void MyApp.Form1::.ctor().

В отличие от .ctor, .cctor — это статический конструктор, который инициализирует статические (помеченные атрибутом static) поля класса. Для отладки ставим команду break в теле метода .entrypoint, остальные брейки можно расставить непосредственно в отладчике. Затем компилируем с генерацией отладочных файлов *.pdb вот таким образом: ilasm MyApp.h /debug. Если у тебя установлена Visual Studio, после запуска программы появится окошко, предлагающее на выбор два отладчика: Microsoft CLR Debugger и стандартный отладчик Studio. Стандартный хорош тем, что под-

пример
подписанная программа

```
custom instance void [mscorlib]System.Reflection.AssemblyKeyFileAttribute::.ctor(string) =
                     ( 01 00 7B 43 3A 5C 44 6F 63 75 6D 65 6E 74 73 20     // ..{C:\Documents
                       61 6E 64 20 53 65 74 74 69 6E 67 73 5C D0 90 D0     // and Settings\...
                       B4 D0 BC D0 B8 D0 BD D0 B8 D1 81 D1 82 D1 80 D0     // .......\..... .
                       B0 D1 82 D0 BE D1 80 5C D0 9C D0 BE D0 B8 20 D0     //.......\..... .
                       B4 D0 BE D0 BA D1 83 D0 BC D0 B5 D0 BD D1 82 D1     // .......\..... .
                       8B 5C 56 69 73 75 61 6C 20 53 74 75 64 69 6F 20     // .\Visual Studio
                       50 72 6F 6A 65 63 74 73 5C 4D 79 41 70 70 5C 62     // Projects\MyApp\b
                       69 6E 5C 44 65 62 75 67 5C 31 2E 6B 65 79 00 00 )   // in\Debug\1.key..
publickey =          (00 24 00 00 04 80 00 00 94 00 00 00 06 02 00 00     // .$..............
                       00 24 00 00 52 53 41 31 00 04 00 00 01 00 01 00     // .$..RSA1........
                       2D C3 3A 4A E0 FA EF 05 D5 FC 1C 9D 08 7D 67 5A     // -.:J.........}gZ
                       B0 48 EB 1A D4 D6 E4 E2 B7 93 11 A6 D8 68 5F 1B     // .H...........h_.
                       7E D7 E3 3C 2C 25 86 1F 34 26 F6 86 26 59 2D 8E     // ~..<,%..4&..&Y-.
                       F7 0B B3 DC 74 C1 3D 8E 00 79 06 68 06 82 C3 F2     // ....t.=..y.h....
                       E9 91 CB F1 F3 4E 87 CD 4A CB 55 CE 57 DE DF 4E     // .....N..J.U.W..N
                       93 64 42 5E A6 86 54 43 D8 25 D0 AD BF 49 F6 9B     // .dB^..TC.%...I..
                       53 9D 3B A7 7A C4 0F 5D A4 53 8E 7F 9A EB A5 E9     // S.;.z..].S......
                       92 09 B0 F5 C5 9B E5 33 CF C7 6E A5 5A 20 C5 C4 )   // .......3..n.Z ..
```

свечивает синтаксис и позволяет редактировать текст во время отладки, однако плохо, что он воспринимает атрибуты антиотладки. Методы, классы или сборка целиком могут быть помечены атрибутом: <System.Diagnostics.DebuggerStepThrough()> или другими подобными, и благодаря этой пометке отладчик не будет отлаживать программу/часть программы.

наш главный пример посвящен обфускатору Decompiler.NET $550 с гордой надписью в документации: «Decompiler.NET protects itself». Наверное, автор этой записи был или очень храбрый, или очень глупый ;). Почему же я выбрал именно этот обфускатор? Объяснение очень простое: мне очень не понравился дизайн сайта (**www.junglecreatures.com**), хотя и его название мне понравилось не больше. Тем не менее, по моему требованию на главную роль в главном примере статьи был приглашен именно обфускатор Dotfuscator Professional Edition for .NET $1495. Во-первых, потому что его цена великовата, во-вторых, потому что вот уже целый месяц я не могу получить его trial-версию. Чтобы добыть trial, ты заполняешь специальную форму, после чего появляется надпись: «Note: Acceptance of registration is not guaranteed». Вот оно что! Посмотреть trial могут только избранные. На самом деле причина, конечно, другая: они не уверены в своем обфускаторе :).

вернемся к decompiler.NET. Этот обфускатор по совместительству является дизассемблером, декомпилятором, оптимизатором, рефактором и Show MSIL :). Автор защиты — один из тех, кто верит, что если устроить настоящее MSIL-шоу, никто ни в чем не разберется. Однако первое, что ты видишь, когда открыл файл в Reflector'e, — это два необфусцированных name space'а (jungle.Deploy.NET.Launcher и jungle.Deploy.NET.Launcher.Licensing). Хм, само по себе очень подозрительно и привлекает внимание, как оазис в пустыне. Вот почему я не стал поддаваться на провокации/заниматься почитанием традиций и начал исследование с метода, помеченного атрибутом .entrypoint. В нем не было ничего интересного. Через определенное количество вызовов функций (.entrypoint->ab.a3::ah1(string[])->ab.a3::ah0(string[])) мы попадаем в ag0. В данном методе есть вызов метода string a4.a8::aba(int32). После просмотра кода стало ясно, что эта функция извлекает из коллекции зашифрованную строку под номером int32, переданном в параметре, и расшифровывает ее. Соответственно, я сделал функцию (DumpCryptedStrings), которая перебирает все строки в коллекции, расшифровывает их и печатает в отладчике в окошке Output. Эту функцию ты сможешь посмотреть в файле Decompiler.NET.h на диске к журналу. Ее вызов закомментирован в .entrypoint, а файл содержит более подробные комментарии относительно анализа кода.

Далее мы попадаем в функцию ab.az::ag2(string, bool, string, object[]). Первое, что привлекает внимание здесь, — это создание дополнительных доменов приложения System.AppDomain::CreateDomain (читаем матчасть). Ага! Код обфусцирован. Кроме того, мы имеем дело еще и с пакером/криптером: динамическое создание доменов предполагает динамическую загрузку сборок. А какие еще сборки необходимо загрузить динамически, если не распакованные? Схема простая: в определенный момент ресурс программы расшифровывается, потом распаковывается и динамически загружается в память. Наша задача — выяснить, когда наступает такой момент, и сделать дамп. Кроме того, в этой функции наблюдается вызов: L_00e9: callvirt instance bool jungle.Deploy.NET.Launcher.IConfiguration::get_CheckLicenseSignature(). Данный вызов нужно закомментировать, а далее — исправить L_00ee: brfalse.s L_0126 на L_00ee: brtrue.s L_0126.

Нам необходимо добиться, чтобы код выполнился внутри этого условия. Сразу после условия идет вызов ab.'at'::get_afy(). В такой функции нет ничего интересного, поэтому мы начинаем интересоваться серией переходов (ab.'at'::get_afy()->ab.au::af6(bool, bool)->ab.au::af8(class jungle.Deploy.NET.Launcher.Licensing.License, bool, bool)) — внимательно посмотри эту функцию в Reflector'e. Не знаю, что увидел ты, но я первым делом рассмотрел там: byte[] buffer2 = reader2.ReadBytes(num1); :). Вот и первая хитрая функция, извлекающая из ресурсов стандартную лицензию (ее я тоже запишу на диск). В этом загрузчике присутствуют всего три такие функции: для извлечения лицензии, конфигурации и шифрованной программы. Им соответствуют ресурсы: jungle.Deploy.NET.Launcher.License.resources, jungle.Deploy.NET.Launcher.Configuration.resources, jungle.Deploy.NET.Launcher.Archive.resources. Им же соответствуют функции для расшифровки: ab.au::af8(class jungle.Deploy.NET.Launcher.Licensing.License, bool, bool)), ab.au::af5(bool __0, bool __1), ab.ax::agz(Hashtable __0).

Остальные функции я нашел банальным поиском текста «MemoryStream stream3» — вряд ли автор стал бы писать разный алгоритм для этих трех функций. Copy-paste нам в помощь, товарищи :).

Как я узнал, что функции имеются в количестве всего трех штук? Дело в том, что эти три ресурса имеют нестандартный формат и редактор ресурсов .NET ругается на них. Когда знаешь, что ищешь, обязательно найдешь ;). Кроме того, из криптоколлекции строк в этих функциях извлекаются названия ресурсов, с которыми они работают, — так лишний раз подтвердилось, что я определил их правильно.

Вернемся к необфусцированному классу Licensing. Как я и предполагал, этот класс — просто мусор, и его патчинг не дал положительных результатов. Класс Licensing применяется в загрузчике, но использование идет вхолостую, поэтому перед распаковкой оригинального exe я попытался найти главную форму в загрузчике, чтобы пропатчить в ней trial'ные ограничения (в тот момент я не был полностью уверен, что имею дело с загрузчиком, а не с самой программой).

ИТОГ: ВСЕГО ЗА НЕСКОЛЬКО ЧАСОВ ПРОГРАММА СТОИМОСТЬЮ $550 БЫЛА ЗНАЧИТЕЛЬНО УЛУЧШЕНА ;)

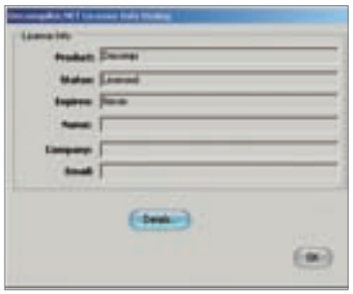
о формате PE

МАНЬЯКАМ — ЛЮБИТЕЛЯМ КОВЫРЯТЬ ВСЕ В HEX-РЕДАКТОРАХ ПОСВЯЩАЕТСЯ

РАЗБОР .NET-ПРОГРАММ НЕОБХОДИМО НАЧИНАТЬ СО СТРУКТУРЫ CLIHEADER, RVA КОТОРОЙ ЗАПИСАН В DATADIRECTORY(ENTRY_COM_DES-CRUIPTOR). НАПОМНЮ, ЧТО DATADIRECTORY(ENTRY_COM_DESCRUIPTOR) ИДЕТ СРАЗУ ПОСЛЕ DATADIRECTORY(ENTRY_IAT). СТРУКТУРЫ C#:

```
struct DataDirectory
{
 uint Rva;
 int Size;
}
struct CliHeader
{
 uint Size;
 ushort MajorRuntimeVersion;
 ushort MinorRuntimeVersion;
 DataDirectory File;
 uint Flags;
 uint EntryPointToken;
 DataDirectory Resources;
 DataDirectory StrongNameSignature;
 DataDirectory CodeManagerTable;
 DataDirectory VTableFixups;
 DataDirectory ExportAddressTableJumps;
 DataDirectory ManagedNativeHeader;
}
```

CLIHEADER.FILE.RVA СОДЕРЖИТ АДРЕС METADATASTARTOFFSET. ДАЛЬНЕЙШЕЕ ИЗУЧЕНИЕ PE ТЫ МОЖЕШЬ НАЧАТЬ С ЧТЕНИЯ ДОКУМЕНТОВ.



Долой ограничения

License Warning

После распаковки выяснилось, что в ....Archive.resources лежат четыре файла: оригинальный exe и три dll. Для распаковки я создал метод DecryptResurceLauncherArchive, который доступен в том же месте, что и DumpCryptedStrings.

Вот распаковка закончена. Единственное оставшееся действие — пропатчить функцию (в уже распакованном файле, конечно ;)) jungle.Decompiler.NET.Licensing::.ctor(string __0, class jungle.Decompiler.NET.Licensing.License __1). Пропатчить нужно всего в шести местах по меткам: IL_0043, IL_004a, IL_0051, IL_00f4, IL_00f6, IL_0102, заменив ldc.i4.0 на ldc.i4.1. Я только что перечислил флаги класса License: flsLicenseKeyValid, flsSignatureValid и flsLicenseKeyValidComputed. Все готово. Теперь обфускатор работает как нужно.

конечно, нужно раскрыть хитрости, использованные чтобы затруднить анализ кода. Я уже упоминал, что в методе ab.az::ag2 используется динамическое создание доменов. Так вот автор определил класс ProxyControl, посредством которого происходили вызовы методов оригинального exe из загрузчика и наоборот. Эта технология называется Remouting, и она позволила замусорить CallStack, причем при работе в отладчике я не мог понять, откуда вызываются методы, — адреса возврата уходили куда-то в неуправляемый код. Больше всего я удивился в тот момент, когда ковырялся в загрузчике и вдруг неожиданно выскочила форма (см. на рисунке), которая была определена в самом загрузчике ab.ap.cs. Лечится это просто: удаляются функции создания динамических доменов System.AppDomain::CreateDomain и ликвидируются сопутствующие им динамические создания объектов (C#): this.ag5 = (av) domain1.CreateInstanceAndUnwrap(type1.Assembly.GetName().Name, type1.FullName);. Для замены лучше написать просто this.ag5 = new type1.

На крайний случай можно написать так: System.AppDomain::CreateInstance(type1.Assembly.GetName().Name, type1.FullName), — и тогда объект будет создан в текущем домене плюс исчезнут все заморочки.

Вторая примененная хитрость заключается в хитром вызове функций, компрометирующие названия которых привлекают внимание. Предварительно создается поле с типом Type, класс Type поддерживает любые динамические махинации, например вызов метода или чтение поля класса. Это можно наблюдать в a4.a8::.cctor, создается Type a8.ai4 = Type.GetType("System.Security.Cryptography.TripleDESCryptoServiceProvider"); потом создается объект, описывающий конкретный метод: MethodBase a8.ai0 = a8.ai3.GetGetMethod(true); Этот метод вызывается вот так: a8.ai0.Invoke(). Такая технология называется рефлекцией (пространство имен — System.Reflection), но она имеет большой недостаток — очень непроизводительный способ вызова методов, который, соответственно, годится только на этапе инициализации, что, в принципе, устраивает автора защиты.

Следующая хитрость — это метод хранения лицензии. Классы, помеченные атрибутом [Serializable], могут быть сохранены в потоке (stream) или восстановлены из него. Притом сохраняются значения полей класса и информация о сборке. Есть маленький нюанс, благодаря которому этот метод используется в защите: при изменении версии сборки объект, сохраненный ранее, уже не может быть восстановлен. Получается, что если совершил маленькую оплошность, ты получаешь уникальный шанс потратить немало времени на ее выявление. Автор использует данную технологию по отношению к некоторым классам, в том числе к классу Licensing.

защита decompiler.net была сделана человеком — специалистом в этой области. Из всех кодов, которые довелось увидеть мне, этот был одним из лучших, причем он показал целых три часа стойкости ;). Что же можно сказать о защите обфускатором, которой пользуется обычный программист? Эта защита вряд ли проживет больше 15-20 минут. Вот так я подтвердил свои слова (см. начало статьи) об уязвимости .NET-программ

ЖУРНАЛ О КОМПЬЮТЕРНОМ ЖЕЛЕЗЕ

Тесты
- Ультракомпактные камеры
- Источники бесперебойного питания
- Бюджетные аудиосистемы 5.1
- Двухпроцессорные видеокарты
- Блоки питания
- Бюджетные видеокарты
- Versus-тест: U.S. Robotics USB Internet Phone vs Hawking HNT1 Net-Talk USB Internet Phone
- В сборе: OLOI Proxima Prestige
- Тест софта: программы для работы с локальной сетью

Инфо
- Мелочи железа
- Фишки IT
- Over-сцена
- Моддинг-сцена
- 3D-сцена
- Эволюция модемов
- Технология видеокарт семейства Radeon X1000
- Линейка: внешние накопители Maxtor
- Звездные железяки: приводы Iomega ZIP
- Конструктор: сервер для домашней сети
- FAQ

Практика
- Разгон материнских плат на базе чипсетов Intel i9XX
- Учим как собрать современный компьютер
- Ремонт
- Моддинг: проект SAMOVAR

Теперь 160 страниц!

ЖУРНАЛ КОМПЛЕКТУЕТСЯ ДИСКОМ С ЛУЧШИМ СОФТОМ

круглые отличники

ИНСТРУМЕНТЫ РАЗРАБОТКИ

СНАЧАЛА МЫ РЕШИЛИ СДЕЛАТЬ ОБЗОР СРЕДСТВ РАЗРАБОТКИ, УСТРОИВ ПАРАЛЛЕЛЬНО ОПРОС НА ФОРУМЕ. ОДНАКО ПОСЕТИТЕЛИ ПРОЯВИЛИ НАСТОЛЬКО ВЫСОКУЮ АКТИВНОСТЬ, ЧТО МЫ ПРИШЛИ К МЫСЛИ О МАТЕРИАЛЕ, ПОСТРОЕННОМ ПОЛНОСТЬЮ НА ИХ ОПЫТЕ. ВСЕ-ТАКИ ЖИВЫЕ И НАГЛЯДНЫЕ ПРИМЕРЫ ЛУЧШЕ ЛЮБОГО ОБЩЕГО ОБЗОРА. ИТАК, БЫЛ ВОПРОС: «КАКИЕ СРЕДСТВА РАЗРАБОТКИ ТЫ ИСПОЛЬЗУЕШЬ ПОД UNIX ИЛИ WINDOWS?» ВЫСКАЗЫВАЛИСЬ НЕ ТОЛЬКО СТРОГО ПО ЭТОЙ ТЕМЕ. ДУМАЮ, МНОГИЕ ОПУСЫ БУДУТ ДЛЯ ТЕБЯ ПОЛЕЗНЫМИ

АНДРЕЙ КАРОЛИК (ANDRUSHA@REAL.XAKEP.RU)

GO!
на форум
forum.xakep.ru/-view.asp?topicID=69710

f0x1er: Под винду с самого начала использую Visual C++. Сначала шестая версия, потом перешел на 7.1, так как прежняя была уже неактуальна. Пробовал много других IDE, в частности DevC++, Code Blocks. Правда, так как приучен к разным наворотам Visual C++ (такие как IntelliSence и умная система автоформатирования кода), переходить на другие среды не захотел. В DevC++ также был кривой отладчик, который запускался через раз, в Code Blocks была непонятная система watch'ей. Подумываю переходить на восьмой Visual C++. Уважаю его за хороший редактор кода и компилятор, поддерживающий большинство современных нововведений и технологий. Вообще, Visual C++ (да и вся Visual Studio) — один из немногих удачных продуктов Microsoft. Под Linux начал кодить недавно. По напутствиям разных гуру, пытался использовать vi + консоль + компилятор, но пересел на Anjuta IDE (я полный чайник в Linux). Теперь, поднабравшись опыта, собираюсь переходить на Emacs+autoconf+automake+make+gcc.

NightmareZ: Мое знакомство с программированием началось с очень оригинальных советских компьютеров... Что за система там была, я даже представить себе не могу, но точно не DOS и не *nix. В качестве среды разработки был текстовый редактор и интерпретатор Basic'a. Сохранять наработки было нельзя, да и некуда :). В общем, с ужасом вспоминаю тот период, хотя именно тогда я заразился программированием и, видимо, на всю жизнь. Через полгода приобрел неплохой по тем временам компьютер, побежал на рынок и купил первый попавшийся диск с подборкой софта по программированию. Так как в школе приходилось писать на Basic'е, его и поставил. Visual Basic 6.

Жаль, что Microsoft переориентировала VB под .NET. Прошло два года с момента моего знакомства с компами и кодингом, я подсел на Delphi. Сейчас считаю эту IDE наиболее удобной из всех существующих для разработки приложений с развитым GUI-интерфейсом. Имхо, Microsoft со своей VS и дотНетом нервно курят в сторонке. На Delphi же познакомился с 3D-графикой (OpenGL) и написал немало софта (**www.systemhalt.nm.ru**). Промедление в нашем мире смерти подобно, поэтому на Delphi я не остановился. Visual Studio .NET, по-моему — лучшая среда (но Delphi все же лучше ;)). Есть опыт разработки на VB.NET, C++/CLI, C#. Вообще от всего .NET'а я в восторге... Слишком уж много приятных фишек в этой технологии. Совсем чуть-чуть писал на Java. Пробовал Common Lisp, Prolog — не зацепило. Последние две недели пишу форум для своей локалки на PHP. Тут уже никакой среды не надо, достаточно редактора с подсветкой синтаксиса. Использую UltraEdit-32, и не потому что в нем есть какие-то качества, которых нет в других редакторах, а потому что просто он первым мне на глаза попался.

oxid: Я только под Windows. Для C++, C# и ASP.NET: VC++6, VS7. Для PHP, eLisp, Erlang, XSLT, XML, XHTML: Emacs и разные mode. Для Scheme: drScheme.

son_of_war: Под Windows работаю в VB, Delphi, C++ Builder, под Linux больше всего понравилось работать в PHP, приспособленном под системный интерпретатор. Front end'ы, если надо, собираю на C++ под QT. Кстати, по разработке front end'ов можно глянуть на **www.opennet.ru/docs/RUS/qt3_prog/index.html** — там есть замечательная книжка по разработке графического интерфейса с библиотекой QT. Есть и в HTML, и PDF — написана разработчиками самой QT. Все просто и ясно. Расстраивает, конечно, что для Linux нет IDE, таких как MS Visual Studio, Delphi. Borland'овскую Kylix пробовал, но она работает только со старыми библиотеками, то есть на новых дистрибутивах запустить ее весьма и весьма проблематично. QT designer от Trolltech, на мой взгляд, пока не дотягивает до них.

Alexiy: Начал программировать на Basic'е под Spectrum :). Изучил пять его диалектов... Потом на 386 TP 6.0. Дальше TP 7.0, 7.1. Немного освоил C/C++. Знаю на уровне «могу написать почти все и прочитать все». Потом освоил D5, перешел на D7. До сих пор на Delphi 7 и программирую. Вполне хватает. Хотя более поздние версии имеют свои плюсы, у них же есть и куча минусов. Особенно раздражает интерфейс в стиле MSVS. Зачем — непонятно, в плане интерфейса Delphi всегда был лучше. Далее изучил PHP. Долго не знал, на чем писать, писал в FAR'е. Потом пробовал PHP Editor. Потом случайно наткнулся в Сети на ZDE, скачал третью версию, порядка 30 метров на модеме. И я был прав: это офигенная IDE! Лучше ее (не только для PHP) я еще не видел. Ощущение такое, что она обладает искусственным интеллектом :). Сейчас изучаю Java. Пользуюсь пока FAR'ом. Пытался копаться в Eclipse — не понравилось. Вроде похоже на ZDE, но не то. Хотя говорят, что Eclipse — штука хорошая. Надо привыкать...

Bi0s-0x269: Оостановился на следующем подходе: весь функционал программы пишу на чистом C в Visual Studio и компилю в dll, а графический интерфейс «рисую» на Delphi или Builder. Ну а если надо по-быстрому написать халтурку, то пользуюсь Delphi — просто и быстро. Больше всего не люблю семейство .NET и Java, потому что такие языки порождают необразованных программистов, не разбирающихся в основах: тут класс, там класс — и все готово :). Думаю, программировать на .NET/Java можно только освоив основу, Assembler и C.

yan_kos: На сегодня под винду лучше Visual Studio 8.0 ничего нет, тем более если пользоваться Studio вместе с Visual Assist X. Хотя есть и два минуса: самая нестабильная IDE под вынь (бывает такое, что проект не собирается, но достаточно перегрузить программу — и соберется с первой же попытки, это

очень заметно в V S7.0, в 8.0 не замечал) и слабый редактор ресурсов. Под Linux лучше, чем KDEni (точное название забыл), нет. Под Mac OS — xCode рулез!

«basic-программист»

НА BASIC ДО СИХ ПОР УЧАТ АЗАМ ПРОГРАММИРОВАНИЯ, ТАК КАК ЭТО ЧУТЬ ЛИ НЕ САМЫЙ ПРОСТОЙ И НАГЛЯДНЫЙ ЯЗЫК, ПОЗВОЛЯЮЩИЙ ОСВОИТЬ ОСНОВНЫЕ ПРИЕМЫ, ЦИКЛЫ, МАССИВЫ И Т.П. МОЖНО СКАЗАТЬ, ЧТО С НЕГО «НАЧИНАЛИСЬ» МНОГИЕ ПРОГРАММИСТЫ. ИНТЕРПРЕТАТОР ЯЗЫКА BASIC ДЛЯ ПЕРВОГО МИНИКОМПЬЮТЕРА MITS ALTAIR РАЗРАБОТАЛ БИЛЛ ГЕЙТС СО СВОИМ ДРУГОМ ПОЛОМ АЛЛЕНОМ BASIC В КОНЦЕ 1975 ГОДА В ГАРВАРДСКОМ УНИВЕРСИТЕТЕ, СТУДЕНТАМИ КОТОРОГО ОНИ В ТО ВРЕМЯ ЯВЛЯЛИСЬ.

В КОНЦЕ 80-Х ГОДОВ БЫЛО УЖЕ ОКОЛО ДЕСЯТКА BASIC'ОВ ОТ РАЗЛИЧНЫХ РАЗРАБОТЧИКОВ. НО ОСНОВНАЯ БОРЬБА ШЛА МЕЖДУ QUICKBASIC ОТ MICROSOFT И TURBOBASIC ОТ BORLAND. РЕЗУЛЬТАТОМ СТАЛО НЕЯВНОЕ МИРОВОЕ СОГЛАШЕНИЕ, MICROSOFT ОТКАЗАЛАСЬ ОТ ДАЛЬНЕЙШЕЙ ПОДДЕРЖКИ PASCAL, А BORLAND — ОТ BASIC. И ДО СИХ ПОР, ПЕРЕЧИСЛЯЯ СВОИ ТИТУЛЫ, БИЛЛ ГЕЙТС ДОВОЛЬНО ЧАСТО ДОБАВЛЯЕТ «BASIC-ПРОГРАММИСТ».

красное затмение

ECLIPSE – ХИТРАЯ СРЕДА РАЗРАБОТКИ

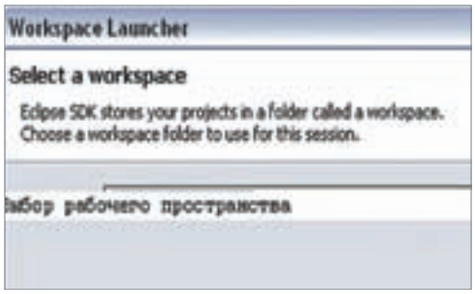
КАЖДЫЙ ПРОГРАММИСТ НАХВАЛИВАЕТ СРЕДУ РАЗРАБОТКИ, КОТОРУЮ ИСПОЛЬЗУЕТ САМ. Я ПОСТАРАЮСЬ БЕСПРИСТРАСТНО РАССКАЗАТЬ ОБ УНИВЕРСАЛЬНОЙ СРЕДЕ ДЛЯ РАЗРАБОТКИ ПРОГРАММ (И НЕ ТОЛЬКО), КОТОРАЯ СЕЙЧАС НАБИРАЕТ ПОПУЛЯРНОСТЬ. ЕЕ НАЗВАНИЕ ПЕРЕВОДИТСЯ КАК «ЗАТМЕНИЕ» — ECLIPSE. ПРОСЬБА НЕ ПУТАТЬ СО ЖВАЧКОЙ ;). ЧАСТО ЕЕ ЛЮБЯ НАЗЫВАЮТ «КЛИПСОЙ» |БОРИС ВОЛЬФСОН (BORISVOLFSON@GMAIL.COM)

how to install? Как ни странно, прежде чем начать работать с программой, нужно установить ее. Сразу замечу, что eclipse написана на Java, поэтому для ее работы требуется Java Runtime Environment (java-машина). Дистрибутив можно взять на сайте фирмы Sun: **http://java.sun.com**. Далее идем на официальный сайт eclipse (**www.eclipse.org**) и качаем архив eclipse-SDK-3.1.2-win32.zip из раздела Download (весит чуть больше 100 МБ). Можно поискать дистрибутив eclipse и на CD-дисках.

Теперь приступим к самой установке, хотя «установка» — громко сказано. Достаточно просто распаковать zip-архив в нужную папку, предположим C:\Program Files\Eclipse. Все — среда готова к использованию :). Для сохранения настроек eclipse использует файловую систему, а не реестр, поэтому для разных языков программирования можно установить eclipse в разные папки, но ничто не мешает поставить плагины для нескольких языков и на один экземпляр среды.

Java — родной язык для среды eclipse, в которой, соответственно, встроены все необходимые средства для использования этого языка программирования.

позвольте представить… На первый взгляд может показаться, что это очередная, ничем не выдающаяся IDE, однако спешу огорчить: eclipse — вообще не IDE :), а целая платформа для различных приложений. В стандартную поставку входят дополнительные плагины для поддержки языка Java (JDT)



Выбор рабочего пространства

и для разработки плагинов для eclipse (PDE — Plug-in Development Environment). Для работы с другими языками нужно поставить (распаковать архив) специальные плагины, ссылки и названия которых можно посмотреть на врезке. Также можно просто скачать и готовый дистрибутив с настроенными плагинами для предпочитаемого языка программирования. В этой статье для простоты мы будем использовать язык Java. Таким образом, в eclipse можно работать практически на любом доступном языке программирования. Да простят меня любители языка Java, но есть плагин (Improve C#), который позволяет писать программы на C# — это первый очевидный плюс. Второе преимущество тоже относится к расширяемости: гигантское количество утилит (особенно для Java) теперь поставляются и в виде плагинов для eclipse, например Ant, JavaDoc, JUnit, JDepend, Check Style, Subversion. Значит, нам не придется отказываться от своей системы контроля версий, от своей программы проверки качества кода и т.п. В-третьих, eclipse является кроссплатформенной средой, то есть существуют версии для различных операционных систем (этого не может позволить себе, скажем, Visual Studio). Подведем итог: в eclipse можно писать на чем угодно, можно использовать совместно с eclipse что угодно и в eclipse можно работать где угодно. :)

препарируем eclipse

Познакомимся с основными понятиями eclipse. Если ты уже пробовал включать эту среду разработки, то тебе уже немного знакомо понятие «рабочее пространство» (workspace). При первом запуске eclipse (и не только при первом, если не поставить галочку) появляется диалоговое окно с просьбой выбрать рабочее пространство.

Физически, рабочее пространство — это просто папка на жестком диске, где сохраняются все настройки и проекты по умолчанию. Это можно использовать для настройки eclipse для разных языков или для разработки разных программ. Следующим важным понятием является «проект» (project) — это обычный контейнер для других ресурсов, которыми могут быть папки, файлы, ссылки и т.д.

Внутренние окна обычно делят на редакторы и виды (вспомогательные окна). Одним из самых удобных механизмов в eclipse являются «перспективы» (perspectives) — настройки внешнего вида среды разработки. Перспективы можно переключать кнопками в левом верхнем углу либо из меню Window→ Open Perspective… Довольно просто понять эту концепцию на простом примере. Ты пишешь код. Тебе нужен, прежде всего, редактор кода. Возможно окно Package Explorer для просмотра проекта и окно для просмотра ошибок. Однако при отладке необходимы окна для просмотра значения переменных, точек прерываний, активных потоков (в многонитевой программе) и т.д.

практикум

Как можно познакомиться с той или иной средой программирования? По-моему, ответ один: написать в ней программу! Поначалу хотелось написать какую-нибудь простенькую систему управления космическим аппаратом с ионной пушкой на борту, но редакция посоветовала отложить эту тему для номера про взлом :(. С другой стороны, писать очередную программу в духе «Hello, world», опять же, не было желания. Я выбрал нечто среднее, чтобы суметь как следует пощупать eclipse. Напишем программу для вычисления корней квадратного уравнения с графическим интерфейсом. Даже сквозь страницу журнала вижу скуку на лицах («Опять высшая математика»:)). Но на этой простой программе мы будем использовать многое из функционала eclipse: от визуального редактора окон до модных нынче средств рефакторинга и тестирования. Начнем с создания проекта, для этого достаточно выбрать пункт меню File→ New→ Project… и выбрать тип проекта Java Project. Дальше нужно справиться со стандартным мастером создания проектов, для чего введем имя проекта, в нашем случае — QuadraticRoots, что теоретически должно означать квадратные корни :). Теперь можно заняться архитектурой нашей программы, которая будет состоять из четырех пакетов (модулей). Создать их можно при помощи того же меню File→ New, но есть и более коммунистический путь — воспользоваться контекстным меню проекта, заодно посмотрев, что там еще лежит. Пакетов у нас будет ровно четыре: main — пакет для главной программы, math — пакет для класса, который будет вычислять корни уравнения, tests — пакет для модульных тестов и ui (User Interface) — пакет для хранения интерфейса пользователя. Языком программирования мы выберем родной Java, но код будет несложным и вполне понятным программистам и на C/C++, C# и даже на PHP5 :).

Чтобы показать всю мощь eclipse, начнем, как ни странно, с написания тестов. Добавим (с помощью того же контекстного меню) набор тестов — JUnit Test Case в пакет tests. Набор тестов представляет собой потомок класса TestCase, который содержит все методы, необходимые для тестирования. Сам тест — это публичный метод с именем, которое начинается с приставки «test». Проверка правильности осуществляется при помощи методов assert, их довольно много. Например, метод assertEquals проверяет равенство своих аргументов, и если они равны, то тест считается пройденным. Наборы тестов обычно тестируют методы одного класса. Мы будем иметь дело с одним классом — QuadricRoots, который с помощью метода findRoots займется поиском корней квадратного уравнения. Методу будут передаваться три коэффициента квадратного уравнения, а возвращать он будет массив действительных чисел.

сначала – тесты

Начнем с самого простого теста, когда уравнение не имеет корней и, как следствие, наш метод возвращает null. Тесты обычно пишутся начиная с вызова assert, затем уже создаются и настраиваются объекты. Посмотрим, чем нам в этом поможет eclipse.

```
public class QuadricRootsTests extends TestCase {
  public void testFindRoots()
  {
    assertEquals(null, quadricRoots.FindRoots
(1, 1, 100));
  }
}
```

Когда ты напишешь этот метод, eclipse автоматически и совершенно прозрачно скомпилирует программу, что возможно благодаря инкрементальной

плагины для eclipse

ПЛАГИНЫ МОЖНО СКАЧАТЬ С МНОГИХ САЙТОВ, НО НАИБОЛЬШЕЕ ИХ КОЛИЧЕСТВО ЛЕЖИТ НА САЙТЕ ECLIPSE.ORG И SOURCEFORGE.NET (В СТРОКЕ ПОИСКА НУЖНО ВВЕСТИ «ECLIPSE»).

JDT (JAVA DEVELOPMENT TOOLS) — ВСТРОЕННЫЙ НАБОР ПЛАГИНОВ ДЛЯ ЯЗЫКА JAVA. ОБЕСПЕЧИВАЕТ НАИБОЛЕЕ ПОЛНУЮ ПОДДЕРЖКУ ЭТОГО ЯЗЫКА.

CDT (C/C++ IDE) — НАБОР ПЛАГИНОВ ДЛЯ ЯЗЫКОВ C/C++. C/C++ ФАКТИЧЕСКИ ЯВЛЯЮТСЯ ВТОРЫМИ ОФИЦИАЛЬНЫМИ ЯЗЫКАМИ ДЛЯ ECLIPSE, КОТОРЫЕ МОЖНО СКАЧАТЬ С ОФИЦИАЛЬНОГО САЙТА **eclipse.org**.

TRUSTUDIO — СРЕДА ДЛЯ ЯЗЫКОВ PHP И PYTHON, ТАКЖЕ ПОДДЕРЖИВАЕТСЯ HTML. ИМЕЕТСЯ КАК БЕСПЛАТНАЯ ВЕРСИЯ, ТАК И КОММЕРЧЕСКАЯ.

PHPECLIPSE — СРЕДА ПРОГРАММИРОВАНИЯ НА PHP СО СВОБОДНОРАСПРОСТРАНЯЕМЫМИ ИСХОДНЫМИ ТЕКСТАМИ.

RDT (RUBY DEVELOPMENT TOOL) — НАБОР ПЛАГИНОВ ДЛЯ ЯЗЫКА RUBY.

PYDEV — НАБОР ПЛАГИНОВ ДЛЯ ЯЗЫКА PYTHON.

SUBCLIPSE — ИНТЕГРАЦИЯ ECLIPSE И СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ ИСХОДНОГО КОДА SUBVERSION.

интервью с разработчиком

Павел Петроченко — инженер-программист крупной IT-компании

СПЕЦ: ПРЕДСТАВЬТЕСЬ, ПОЖАЛУЙСТА.
П.П: ПАВЕЛ ПЕТРОЧЕНКО. СЕЙЧАС РАБОТАЮ В ОДНОЙ КРУПНОЙ IT-КОМПАНИИ ИНЖЕНЕРОМ-ПРОГРАММИСТОМ.

СПЕЦ: ДАВНО ЛИ ВЫ РАБОТАЕТЕ СО СРЕДОЙ ECLIPSE?
П.П: ДВА ГОДА

СПЕЦ: ЧТО ВЫ РАЗРАБАТЫВАЛИ ДЛЯ ECLIPSE?
П.П: TRUSTUDIO IDE — СРЕДА РАЗРАБОТКИ ПОД PHP И PYTHON НА БАЗЕ ECLIPSE PLATFORM И SMART TESTING TOOLKIT — СРЕДСТВО АВТОМАТИЗАЦИИ QA-ПРОЦЕССА (QUALITY ASSURANCE — ПРОЦЕСС ПРОВЕРКИ КАЧЕСТВА ПРОГРАММЫ) НА БАЗЕ ECLIPSE И TPTP.

СПЕЦ: ПОЧЕМУ ДЛЯ СВОИХ РАЗРАБОТОК ВЫ ВЫБРАЛИ ПЛАТФОРМУ ECLIPSE?
П.П: ВО-ПЕРВЫХ, ВЫСОКОЕ КАЧЕСТВО ПРОГРАММНЫХ БИБЛИОТЕК ECLIPSE. ВО-ВТОРЫХ, ГРАМОТНО КРАСИВО РЕАЛИЗОВАННЫЙ INVERSION OF CONTROL (ШАБЛОН ДЛЯ УМЕНЬШЕНИЯ ЗАВИСИМОСТИ МЕЖДУ КОМПОНЕНТАМИ). В-ТРЕТЬИХ, ХОРОШАЯ АРХИТЕКТУРА (НЕ БЛЕСК, КОНЕЧНО, НО УСТОЙЧИВАЯ «ПЯТЕРКА С МИНУСОМ», ЧТО ДЛЯ ПРОЕКТА ТАКОГО РАЗМЕРА САМО ПО СЕБЕ КРУТО. ДЛЯ СРАВНЕНИЯ, NETBEANS, ПО МОЕЙ ОЦЕНКЕ — «СЛАБАЯ ЧЕТВЕРКА»). В-ЧЕТВЕРТЫХ, РАСШИРЯЕМОСТЬ, МОДУЛЬНОСТЬ КАК БАЗОВЫЙ ПРИНЦИП ПОСТРОЕНИЯ СИСТЕМЫ. В-ПЯТЫХ, ПРОГНОЗИРУЕМО РАСТУЩАЯ ПОПУЛЯРНОСТЬ ПЛАТФОРМЫ (ЧТО, КОНЕЧНО, НА САМОМ ДЕЛЕ ПЕЧАЛЬНО, НО, К СОЖАЛЕНИЮ, SUNOB'СКИЕ NETBEANS ПРИНЦИПИАЛЬНО ПРОИГРЫВАЮТ НА КОНЦЕПТУАЛЬНОМ УРОВНЕ).

СПЕЦ: КАКИЕ ВОЗМОЖНОСТИ ECLIPSE ВАМ НРАВЯТСЯ БОЛЬШЕ ВСЕГО?
П.П: КАК ПРОГРАММИСТУ — МОДУЛЬНОСТЬ, КАК ПОЛЬЗОВАТЕЛЮ — QUICK FIX, CLASS, CALL HIERARCHY HIERARCHY, CHANGE METHOD SIGNATURE, ПОИСКИ ВСЯКИЕ УМНЫЕ, QUICK OUTLINE И Т.Д. КОНЦЕПТУАЛЬНО: ОТКРЫТЫЙ КОД, ОЧЕНЬ ПРАВИЛЬНАЯ МОДУЛЬНОСТЬ, ХОРОШАЯ ЛИЦЕНЗИЯ, СТИМУЛИРУЮЩАЯ НАПИСАНИЕ ПЛАГИНОВ ПОД ECLIPSE, МНОЖЕСТВО TUTORIALS, ГРАМОТНАЯ ДОКУМЕНТАЦИЯ ПРОГРАММИСТА, PDE.

СПЕЦ: КАКИЕ ДОПОЛНИТЕЛЬНЫЕ ПЛАГИНЫ ДЛЯ ECLIPSE ВЫ ИСПОЛЬЗУЕТЕ?
П.П: WST (WEB STANDARD TOOLS), JST (J2EE STANDARD TOOLS), TPTP (TEST AND PERFORMANCE TOOLS PLATFORM), GEF (GRAPHICAL EDITING FRAMEWORK), ARGOUML, JEM, PMD (ПЛАГИН ДЛЯ ПРОВЕРКИ КАЧЕСТВА КОДА)...

СПЕЦ: КАКИЕ САЙТЫ ПОСОВЕТУЕТЕ ДЛЯ ПОЛУЧЕНИЯ ИНФОРМАЦИИ ПО ECLIPSE?
П.П: WWW.ECLIPSE.ORG :-).

Создание набора тестов

компиляции, во время которой компилируются только измененные файлы. В результате мы получаем несколько преимуществ от скорости до возможностей продвинутого рефакторинга и поиска. Но и это еще не все — eclipse поддерживает модную функцию подсветки ошибок на лету. Дело в том, что объект quadricRoots еще не объявлен, но не стоит сразу бросаться писать что-то, за нас это сделает eclipse :) Достаточно навести курсор на подчеркнутое слово и нажать <Ctrl>+<1> (запомни это сочетание клавиш, оно еще не раз пригодится). Тут появляется окошко Quick Fix с возможными вариантами исправления, из которых нужно выбрать Create local variable, в результате будет создана локальная переменная с заданным именем (совсем люди обленились — прим. Dr.).

Теперь объявим тип переменной b с помощью того же Quick Fix, создадим класс QuadricRoots и проинициализируем переменную:

```
QuadricRoots quadricRoots = new QuadricRoots();
```

Для инициализации достаточно написать new и выбрать нужный класс. Первый тест готов:

```
public class QuadricRootsTests extends TestCase {
  public void testFindRoots()
  {
    QuadricRoots quadricRoots = new QuadricRoots();
    assertEquals(null, quadricRoots.FindRoots(1, 1, 100));
  }
}
```

Осталось создать метод FindRoots в классе QuadricRoots при помощи все того же <Ctrl>+<1>, а дальше только переименовать параметры и сменить их тип. Создадим простую заглушку, чтобы тест сработал (это называется fake) — просто вернем null. Если же следовать классике/канонам, необходимо запустить тест без заглушки. Если он не сработает, значит, тест правильный :).

```
public class QuadricRoots {

  public QuadricRoots() {
  }

  public double[] FindRoots(double a, double b, double c) {
  }
    return null;
  }
}
```

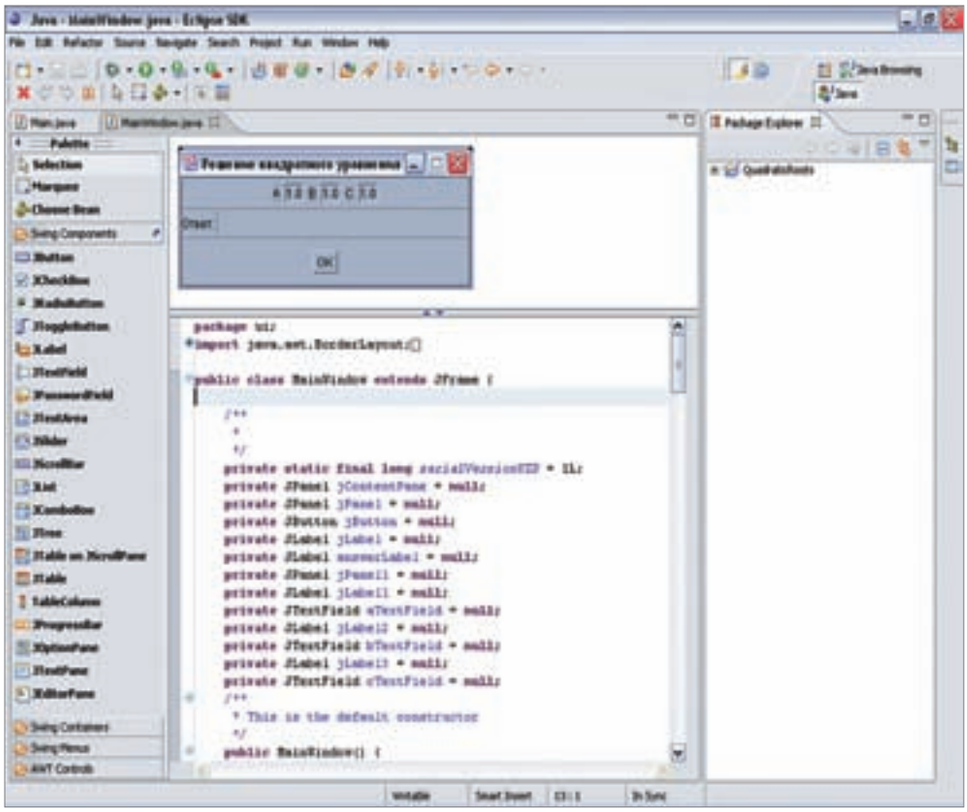
Теперь его необходимо запустить, для чего двигаемся в меню Run→ Run As→ JUnit Test либо нажимаем аккорд <Alt>+<Shift>+<T>, затем <X>. В результате запустится система JUnit, которая встроена в виде плагина в eclipse. Появится знаменитая «зеленая полоса» JUnit, которая означает, что все тесты прошли успешно.

Пришло время второго теста: уравнение имеет один корень. Напомню, что дискриминант для этого должен быть равен нулю. Сначала пишем тест:

```
  public void testOneRoot()
  {
    QuadricRoots quadricRoots = new QuadricRoots();
    double[] answer = {-1.0};
    double[] result = {-1.0};

    assertEquals(answer[0], quadricRoots.FindRoots(1, 2, 1)[0], 0);
  }
```

Я завел массив answer для хранения правильного ответа. Массив result остался в результате экспериментов и нигде не используется, но eclipse тут же

ИНКРЕМЕНТАЛЬНАЯ КОМПИЛЯЦИЯ ОТКРЫВАЕТ МНОЖЕСТВО ПРЕИМУЩЕСТВ: СКОРОСТЬ, ВОЗМОЖНОСТЬ ПРОДВИНУТОГО РЕФАКТОРИНГА И ПОИСКА



SWT может настраиваться для разных интерфейсов

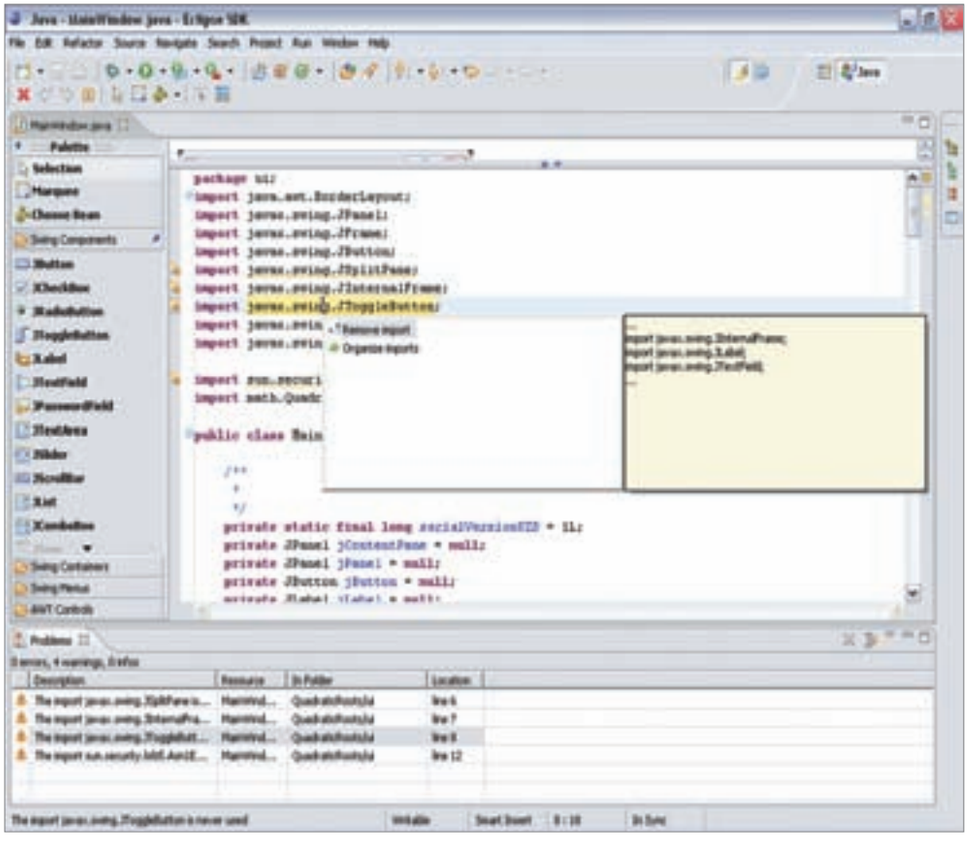
подчеркивает желтым все неиспользуемые переменные, которые можно удалить при помощи Quick Fix.

Кроме того, я сменил имя предыдущего теста на testNoRoots, что можно сделать при помощи меню Refactor-Rename (но в данном случае не дает никакого выигрыша). Запускаем тесты — полоска краснеет. Щелкаем по неудавшемуся тесту — и среда ставит курсор на нужную строчку. Как и ожидалось, не прошел тест с одним корнем. Пора реализовать эту функциональность или просто написать сразу второй тест — такой подход называют триангуляцией. Кстати, перед написанием третьего теста обрати внимание на появившееся дублирование кода в тестах и устрани его: строчка QuadricRoots quadricRoots = new QuadricRoots(), находится сразу в двух методах. Необходимо создать поле quadricRoots, для чего воспользуемся рефакторингом Convert Local Variable To Field. Притом это должно быть проделано в методе setUp, для того чтобы поля инициализировались перед каждым тестом. Заодно напишем и третий тест:

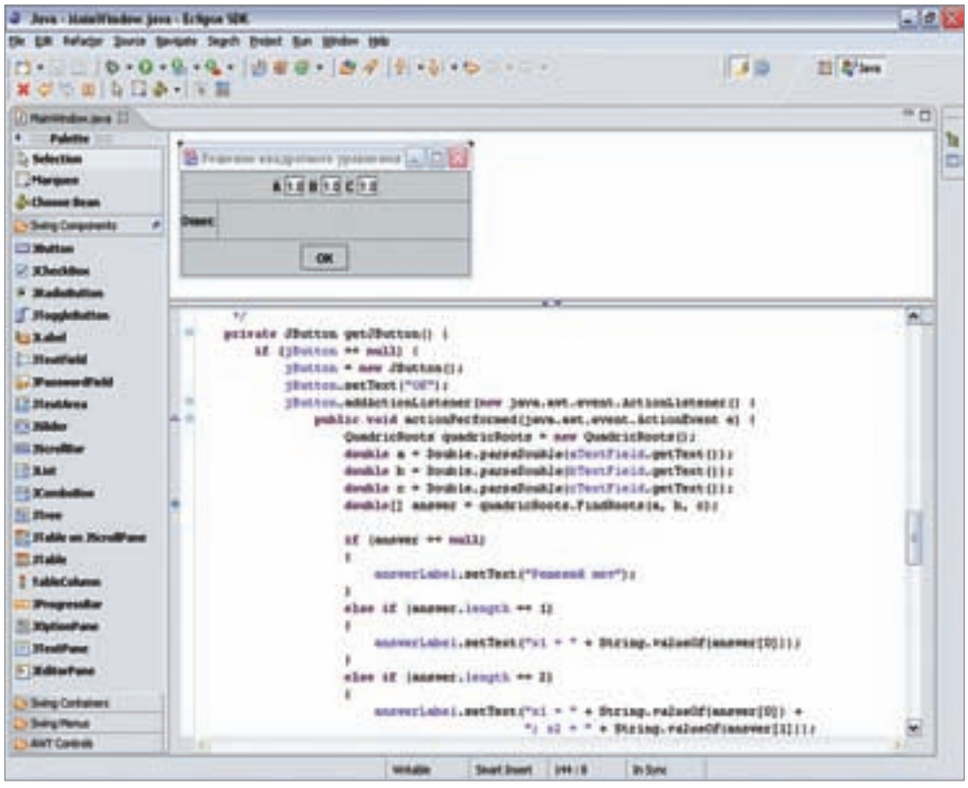
```
public class QuadricRootsTests extends TestCase {

  private QuadricRoots quadricRoots;

  protected void setUp() throws Exception {
    super.setUp();
    quadricRoots = new QuadricRoots();
  }

  public void testNoRoots()
  {
    assertEquals(null, quadricRoots.Find
Roots(1, 1, 100));
  }

  public void testOneRoot()
  {
    double[] answer = {-1.0};
    assertEquals(answer[0], quadricRoots.
FindRoots(1, 2, 1)[0], 0);
  }

  public void testTwoRoots()
  {
    double[] answer = {-4.0, 1.0};
    double[] result = quadricRoots.FindRoots
(1, 3, -4);
    assertEquals(answer[0], result[0], 0);
    assertEquals(answer[1], result[1], 0);
  }
}
```

Теперь просто реализуем метод FindRoots:

```
  public double[] FindRoots(double a, double
b, double c) {
    double[] result = null;

    double d = b * b - 4 * a * c;
```



Еще один вид Quick Fix — удаление ненужных ссылок на пакеты



Редактирование кода и вида окна происходит одновременно

```
    if (d == 0) {
      result = new double[1];
      result[0] = (-b) / (2 * a);
    }
    else if (d > 0)
    {
      result = new double[2];
      result[0] = (-b - Math.sqrt(d)) / (2 * a);
      result[1] = (-b + Math.sqrt(d)) / (2 * a);
    }
    return result;
  }
```

gui Дело осталось за графическим интерфейсом. Будем использовать SWT, который применяет элементы интерфейса операционной системы и поэтому работает быстро. Кстати, на SWT построена и сама eclipse.

Приступим! File→ New→ Visual Class, выбираем стиль JFrame (и не забудь положить класс в пакет ui). Слева появилась палитра инструментов, в которой должны быть выбраны нужные компоненты, возьмем три метки (JLabel) и три поля ввода (JTextField) для коэффициентов, две метки для вывода ответа: одну для надписи «Ответ», во вторую будем класть результат и кнопку для запуска вычисления.

Вот компоненты помещены на окно, теперь нужно настроить их свойства в виде Properties, он аналогичен окошкам для настроек в Delphi и Visual Studio. Затем вешаем на кнопку событие: выбираем из контекстного меню кнопки Events→ actionPerformed. В редакторе пишем код (точнее, его часть), который берет значения, введенные пользователем, преобразует их в тип double, вызывает метод FindRoots, затем заносит в метку результат.

```
  private JButton getJButton() {
    if (jButton == null) {
      jButton = new JButton();
      jButton.setText("OK");
      jButton.addActionListener
(new java.awt.event.ActionListener() {
        public void actionPerformed
(java.awt.event.ActionEvent e) {
          QuadricRoots quadricRoots =
new QuadricRoots();
          double a = Double.parseDouble
(aTextField.getText());
          double b = Double.parseDouble
(bTextField.getText());
          double c = Double.parseDouble
(cTextField.getText());
          double[] answer = quadricRoots.
FindRoots(a, b, c);
          if (answer == null)
          {
            answerLabel.setText("Решений нет");
          }
          else if (answer.length == 1)
          {
            answerLabel.setText("x1 = " +
String.valueOf(answer[0]));
          }
          else if (answer.length == 2)
          {
            answerLabel.setText("x1 = " +
String.valueOf(answer[0]) +
"; x2 = " + String.valueOf(answer[1]));
          }
        }
      });
    }
    return jButton;
  }
```

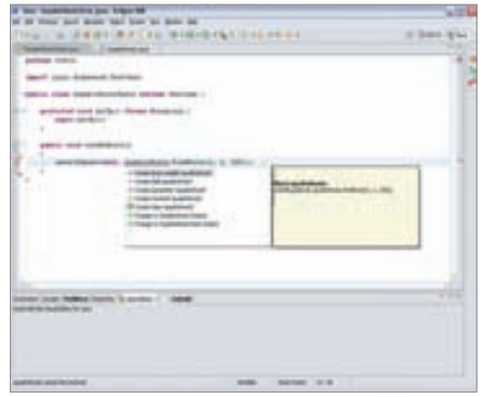
Осталось написать программу, которая будет показывать окно. Идем протоптанной дорожкой File→ New→ Class, помещаем класс в пакет main. Ставим галочку «public static void main(String[] args)», заполняем этот метод:

```
public static void main(String[] args) {

  MainWindow mainWindow =
  new MainWindow();
  mainWindow.show();
}
```

Для запуска программы жмем <Ctrl>+<F11> (или <F11>, если необходимо отладить программу). Программу, конечно, можно еще и доработать: разгрузить форму от логики, отрефакторить класс QuadricRoots, но мы остановимся на том, что есть.

все мы дровосеки Представь себе двух дровосеков. Один из них рубит деревья старым тупым топором, а второй использует для этого дела бензопилу. Каким бы мастером своего дела ни был первый дровосек, по количеству спиленных деревьев он никогда не догонит второго. Абсолютно аналогичная ситуация в программировании: очень много зависит от используемых инструментов. Посмотри на свой «топор». Может, он тоже стал старым и тупым, может, пора сменить его на нечто новое?

Eclipse — действительно хорошая и умная среда для разработки программ. Многое не удалось уместить в тесных статейных рамках: систему контроля версий, мощные средства рефакторинга, работу с исходным кодом и т.д. Стоит еще упомянуть о том, что в eclipse можно (и нужно) создавать апплеты и приложения для мобильных телефонов, поддерживающих Java



Quick Fix в действии: создание локальной переменной

команда молодости нашей

ИНСТРУМЕНТЫ РАЗРАБОТКИ

МЫ ВЗЯЛИСЬ ДЕЛАТЬ ПЕРЕДОВОЙ НОМЕР :). НАВЕРНОЕ, У ТЕБЯ СЛОЖИЛОСЬ ТАКОЕ ВПЕЧАТЛЕНИЕ, ЧТО МЫ ВЕСЬМА ПРОГРЕССИВНЫЕ ДЕЯТЕЛИ, ФАНАТЕЕМ ОТ ПЕРЕДОВЫХ ДЕВАЙСОВ И ВООБЩЕ ВЫРУБАЕМ КАМЕННЫЙ УГОЛЬ ПО 14 НОРМ НА ОДНУ СМЕНУ НЕИСПРАВНЫМ ОТБОЙНЫМ МОЛОТКОМ, ПОПУТНО ОТЛИВАЯ ПО 45 САЛАТОВЫХ УНИТАЗОВ НА КЕРАМИКО-ЛИТЕЙНОМ ЗАВОДЕ ИМ. ТОВ. ПОПУГАЕВА. К СОЖАЛЕНИЮ, НЕ СОВСЕМ ТАК. КОНЕЧНО, ПОРОЙ ЗА ОБЕДЕННЫМ СТОЛОМ МЫ И ТВОРИМ БОЛЬШИЕ ДЕЛА, НО ВОТ ИЗ ГАДЖЕТОВ, КАК ОКАЗАЛОСЬ, МЫ ПРЕДПОЧИТАЕМ СТАРЫЕ И ПРОВЕРЕННЫЕ МОДЕЛИ :) | **DR.KLOUNIZ**

команда телефонистов

DR.KLOUNIZ VS. AVALANCHE

ГАДЖЕТ: ТЕЛЕФОН

Вот это телефон! Всем телефонам телефон. Его вид наводит всех на мысль, что он изобретен до рождения А.Г. Бэла, но это не совсем так. Телефон относительно современный, только раздолбан до невозможности. Он встречался и с бетонным полом, и с бригадой индийских душителей, и с румынскими поджигателями. Видел Рим, Крым и попову грушу. Все это он перенес, пережил и продолжает звонить. Правда, уже сто лет Аваланч толкует что-то про покупку коммуникатора, но…

Мобила Доктора, кстати, намного старше и, в отличие от аваланчерской, находится в абсолютно рабочем состоянии. Клуниз пользуется своим телефоном, звонит по нему (кстати, кнопки главредского телефона бьют током, а докторский абсолютно безопасен, может быть рекомендован к использованию в детских учебных учреждениях) и совершенно не обламывается. Новый телефон ему совсем не нужен — вполне достаточно и этого.

команда социалистов

ВАНЯ ВАСИН VS. НАТАЛЬЯ ЖУКОВА

ГАДЖЕТ: КОММУНИСТИЧЕСКИЙ ФОТОАППАРАТ «ЗОРКИЙ»

Нам не удалось выяснить, кому именно из двух деятелей принадлежит фотоаппарат, так как в данном вопросе не наблюдается единства. Они перетягивают старинный кожаный кофр каждый в свою сторону и утверждают совершенно противоположные вещи :(. Скорее всего, прибор принадлежит-таки Наташе Жуковой, но впоследствии злобный Ваня отнял вещь и стал пользоваться несравненным фотографическим качеством наших коммунистических предков в одиночку. Скажу тебе честно, что ни один современный аппарат не сделает более качественное фото изваяния Иосифа Виссарионовича в контровом свете в конце туннеля. В общем, дело о разделе затягивается, поэтому выношу справедливое решение: железный, качественный, советский, фотографический аппарат будет распилен надвое и поделен между мной и Аваланчем.

вне конкурса

АНДРЕЙ КАРОЛИК

ГАДЖЕТ: СТУДЕНЧЕСКИЙ БИЛЕТ

Если по правде, Андрюша уже не студент. Прямо скажем, он давно не студент (и имеет два высших образования). Конечно же, возникают вопросы. Во-первых, зачем ему просроченный (еще при Сталине) студенческий? Почему он прислал его в качестве гаджета? Разве студенческий билет является гаджетом? Эти вопросы мы адресовали непосредственно Андрюше, но ответа не получили. Вместо ответа он просто шамкал беззубым ртом и пытался стукнуть нас по ногам своей суковатой палкой с металлическим навершием. Затем он бросил на пол свою потертую фетровую шляпу, смачно плюнул в монитор и пошел пить чай на кухню. В общем, фото студенческого прилагается. Вне конкурса

мнение профессионалов

«ДЛЯ СИСТЕМНЫХ ЗАДАЧ ХОРОШ СТАРЫЙ ДОБРЫЙ C/C++»

СПЕЦ: ЕСЛИ ТЫ ЗАНИМАЕШЬСЯ РАЗРАБОТКОЙ, ТО ПРЕИМУЩЕСТВЕННО НА ЧЕМ?

АНАТОЛИЙ СКОБЛОВ: Основной язык — C, реже C++. В последнее время — сразу в нескольких проектах, на втором месте вообще стоит своя поделка — вот это реально упрощает разработку.

АНТОН ПАЛАГИН: Никто не будет раскручивать гайку с помощью лопаты, для этого существуют гаечные ключи. Так же и в программировании. Для системных задач хорош старый добрый C/C++, альтернативы которому в этом контексте не существует. А для других — C#, Python или PHP. Третьи проще реализуются на связке HTML-XML.

АЛЕКСАНДР ПОЛУЭКТОВ: До последнего времени хватало возможностей Microsoft VC++ 6.0, но недавно «по техническим причинам» пришлось пересесть на Microsoft VC++ .NET, о чем сейчас ничуть не жалею. Прикрутив к нему последний Intel C++ Compiler, получил убойный симбиоз, позволяющий решать абсолютно любые задачи в области написания прикладного и системного ПО под Windows от DirectX-приложений и до крошечных модулей, целиком состоящих из ассемблерных вставок. С учетом того, что этот «гибрид» позволяет заточить код под любую целевую платформу, в области средств профессиональной разработки ПО для Windows на данный момент я других альтернатив не вижу.

INGREM: Выбор зависит от того, что я разрабатываю. Как системщик, я очень люблю ассемблер за его гибкость и понятность. Хотя наиболее универсальным языком считаю C++.

НИКИТА БУРЦЕВ: Я периодически пишу различного рода скрипты на Perl, чтобы облегчить свою админскую жизнь :). Под настроение могу сайтик на PHP написать. И более-менее сносно могу разобраться в исходниках того или иного софта перед его компиляцией. Но вот лезть в серьезный проект — увольте. Мои интересы немного в другой плоскости.

АЛЕКСЕЙ ЛУКАЦКИЙ: Давно не занимаюсь, но иногда (случается редко) требуется решить какую-ни-

будь локальную задачу, софта под которую я не нашел. Для этих целей применяю Delphi. Почему именно Delphi? Во-первых, я начинал свой путь программиста с Pascal'я, поэтому переход на Delphi был закономерным. Во-вторых, я не программист и мне нравится набор готовых «кирпичиков», из которых я могу слепить, как минимум, графический интерфейс для своих творений.

ЗАРАЗА: C, VBA/ASP. Но редко :).

КРИС КАСПЕРСКИ: Пишу мелкие утилиты для себя, в основном на C. И совсем немного — на ассемблере. Новыми технологиями не интересуюсь по той простой причине, что не могу работать инструментом, который совершенствуется в моей руке. Оставим это дело молодым. Мне нравится C, с ним легко и удобно. «Преимущества», якобы открываемые .NET'ом, не компенсируют время, потраченное на его изучение. К тому же, когда он будет уже освоен, выйдет что-то новое и т.д.

НИКОЛАЙ АНДРЕЕВ: На C++. Забавно, но в качестве среды я использую достаточно старенькую Visual Studio 7.0, как раз .NET. Правда, кроме шарпа, в ней меня не привлекает ничто. Однако я до сих пор жалею, что Microsoft'овские C-компилеры не умеют использовать прекомпилед MASM'овые макросы во вставках, а Borland C++ Builder умеет. Эх… А из ассемблеров, конечно, я выбрал FASM.

ДМИТРИЙ СОШНИКОВ: Сейчас использую инструменты от Microsoft. Речь, прежде всего, идет о Visual Studio 2005: покрывают все нужды не только программистов, но и архитекторов, тестировщиков, руководителей проектов. Есть еще, конечно, альтернативные средства разработки от Borland, но, думаю, их преимущества очевидны только при разработке родных Win32-приложений. Для платформы .NET все-таки лучше довериться создателям этой платформы, тем более что, насколько мне известно, последние версии Borland все еще не поддерживают .NET Framework 2.0.

СПЕЦ: КАКОЙ ЯЗЫК МОЖНО СЧИТАТЬ ЛУЧШИМ?

АНТОН ПАЛАГИН: Русский и английский :). На них проще излагать свои мысли с вкраплениями комментариев на UML и C/C++, нежели наоборот — писать на C/C++ и вставлять туда комментарии на русском или английском. Если ты хочешь написать какой-то код, опиши его сначала на естественном для тебя языке, который ты учишь с момента рождения, и только потом переходи к языкам, неестественным для органического существа.

АЛЕКСАНДР ПОЛУЭКТОВ: Язык программирования, считающийся лучшим для одних целей, будет совершенно неудобным для других целей. Так что для прикладного ПО на Windows/Linux-платформах лучший язык — это C/C++. Лучший для web-приложений — PHP. Лучший для создания софта, критичного по времени выполнения или по объему занимаемого кода, — ассемблер.

НИКИТА БУРЦЕВ: Что лучше: Intel или AMD? Nvidia или ATI? BMW или Audi? Какой дистрибутив лучше? Отличный вопрос для развязывания «священной войны». Можно, конечно, попытаться ответить на него, но получится так же, как и в случае с выбором дистрибутива. Лучше выбрать то, что позволяет решить поставленные задачи с минимальными затратами на разработку, в том числе и на обучение разработчиков.

АЛЕКСЕЙ ЛУКАЦКИЙ: Еще в школе меня учили, что главное — не знание многих языков, а умение создать алгоритм для своей задачи. Уже потом под конкретную задачу можно выбрать нужный язык программирования. Когда я учился в институте, мне не приходило в голову писать систему криптографической защиты на Pascal (хотя это можно сделать) — скорость будет не та, поэтому я использовал ассемблер. А вот создавать графический интерфейс на ассемблере я не буду :). Зачем? Я могу решить эту задачу за пару часов на Delphi. Каждой задаче — свой язык программирования. «Лучшего» языка просто не существует.

ЗАРАЗА: Родной :). То есть тот, который знаешь лучше других.

КРИС КАСПЕРСКИ: Хусаили. Потому что его никто не знает и, что бы ты ни сказал на нем, никто не назовет тебя дураком ;). Языков много хороших и разных. Мне нравится C, а в сторону остальных я даже не смотрю...

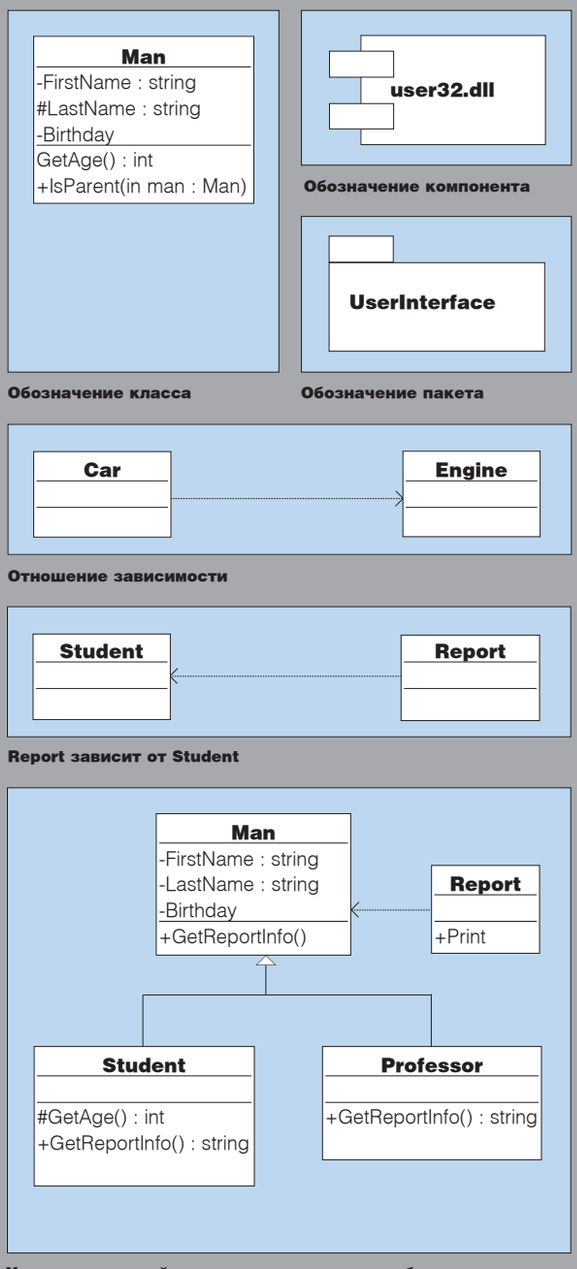
ВЛАДИМИР ЯКОВЛЕВ: «Все правоверные программисты пишут на C» — весьма сомнительно. Каждый язык предназначен для реализации определенных задач и достижения определенных целей. Притом назначение языка может постепенно изменяться. Известно, что Pascal создавался исключительно как обучающий язык, но в настоящее время он вышел далеко за эти рамки, став объектно-ориентированным. Fortran, с одной стороны — язык устаревший, тем не менее он интенсивно используется и в настоящее время. Достаточно сказать, что практически все мощнейшие вычислительные прочностные программы написаны на этом языке, например программа Nastran, разработанная в NASA. Дело в том, что именно этот язык обеспечивает максимально простую работу с матрицами. Наверное, никому не придет в голову использовать скриптовые языки для решения подобных задач. В то же время скриптовые языки широко применяются в Сети. Единственное, что можно сказать по этому поводу, — дать совет начинающему программисту: «Начни с языка C». Этот язык обеспечит максимально простой переход на любой другой.

ОЛЕГ КУРЦЕВ: Среди специализированных, универсальных, поддерживающих разные подходы и т.д. можно находить красивые, простые, удобные, еще какие-нибудь, а найти лучший не получается

программирование на доске почета

UML — УНИВЕРСАЛЬНЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ

UML (UNIVERSAL MODELING LANGUAGE — УНИВЕРСАЛЬНЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ) ПОЗВОЛЯЕТ СОЗДАВАТЬ В ГРАФИЧЕСКОМ ВИДЕ ИЕРАРХИИ КЛАССОВ, ПОКАЗЫВАТЬ ВЗАИМОДЕЙСТВИЕ ОБЪЕКТОВ, СОЗДАВАТЬ ТРЕБОВАНИЯ К ПРОГРАММЕ И МНОГОЕ ДРУГОЕ. В ЭТОЙ СТАТЬЕ Я РАССКАЖУ О НАИБОЛЕЕ ВОСТРЕБОВАННЫХ ФУНКЦИЯХ ЭТОГО ЯЗЫКА, ЧТО ПОЗВОЛИТ ТЕБЕ МАКСИМАЛЬНО ЭФФЕКТИВНО ИСПОЛЬЗОВАТЬ ЕГО В НАПИСАНИИ ПРОГРАММ | БОРИС ВОЛЬФСОН (BORISVOLFSON@GMAIL.COM; HTTP://PROGRAMPORTAL.NET.RU)



ты никогда не задумывался, когда появились первые программисты? После того как были созданы первые компьютеры? Нет, намного раньше… Вспомни наших предков. Вот они, завернутые в шкуры животных, рисуют на стенах пещер сцены охоты. С небольшой натяжкой первобытных людей можно назвать программистами :). Они создавали модели процесса охоты на своем диалекте языка моделирования. Я бы сказал, древний вид диаграммы прецедентов :). Мой несерьезный пример позволяет понять, какова цель моделирования. Во-первых, с помощью наскальных рисунков можно визуализировать процесс охоты и наглядно описать то, как он происходит. Во-вторых, по этим рисункам можно определить, как будет происходить процесс охоты и как распределяются действия каждого участника. В-третьих, при помощи рисунков можно организовать общение между программистами-охотниками. В программировании рисунки, схемы и диаграммы выполняют те же функции.

До того как господа Буч, Рамбо и Джекобсон занялись стандартизацией языка моделирования, каждый программист самостоятельно выбирал, как обозначить на схемах ту или иную сущность, что, разумеется, приводило к путанице и несогласованности. Однако три названных человека решили, что классы все-таки стоит обозначать в виде прямоугольников, а не в виде «облаков» (бывшая нотация одной известной фирмы). И стало так :).

uml — это язык, который является промышленным стандартом для создания диаграмм в области программирования, хотя его можно применять и в других процессах моделирования. UML позволяет графически изобразить тот или иной аспект поведения программы. Зубры программирования, наверное, вспомнят вымершие блок-схемы, с помощью которых в давние времена изображали алгоритмы. Модели, которые создаются на UML, позже могут быть переведены на тот или иной язык программирования. Инструменты UML стали стандартной частью многих сред программирования, хотя можно воспользоваться инструментами, не привязанными жестко к конкретной среде или языку.

краткий словарь UML (из официального руководства)

АГРЕГИРОВАНИЕ — СПЕЦИАЛЬНЫЙ ВИД АССОЦИАЦИИ, ОПИСЫВАЮЩИЙ ОТНОШЕНИЕ МЕЖДУ АГРЕГАТОМ (ЦЕЛЫМ) И КОМПОНЕНТОМ (ЧАСТЬЮ).

АКТЕР — МНОЖЕСТВО ЛОГИЧЕСКИ СВЯЗАННЫХ РОЛЕЙ, ИСПОЛНЯЕМЫХ ПРИ ВЗАИМОДЕЙСТВИИ С ПРЕЦЕДЕНТАМИ.

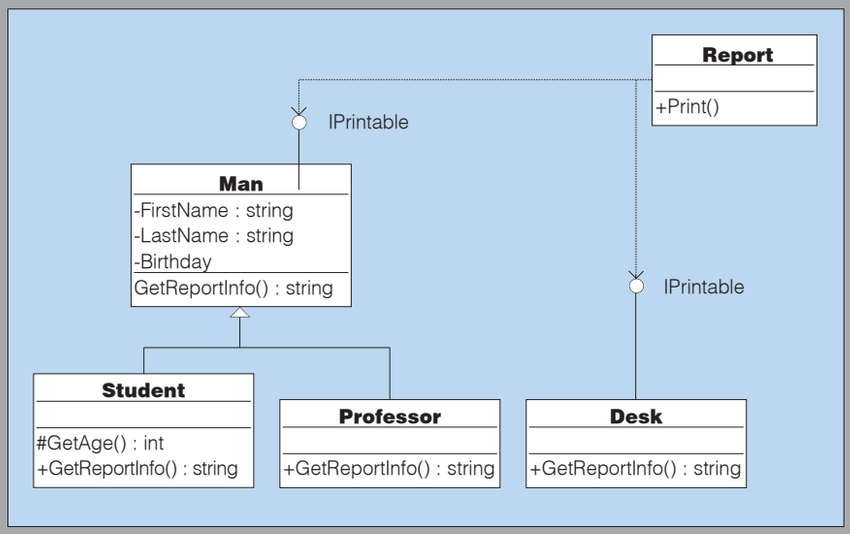
АССОЦИАЦИЯ — СТРУКТУРНОЕ ОТНОШЕНИЕ, ОПИСЫВАЮЩЕЕ НАБОР СВЯЗЕЙ, В КОТОРОМ КАЖДАЯ ИЗ НИХ ПРЕДСТАВЛЯЕТ СОБОЙ СОЕДИНЕНИЕ МЕЖДУ ОБЪЕКТАМИ; СЕМАНТИЧЕСКОЕ ОТНОШЕНИЕ МЕЖДУ ДВУМЯ ИЛИ БОЛЕЕ КЛАССИФИКАТОРАМИ, В КОТОРОМ УЧАСТВУЮТ СОЕДИНЕНИЯ МЕЖДУ ИХ ЭКЗЕМПЛЯРАМИ.

ДИАГРАММА — ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ МНОЖЕСТВА ЭЛЕМЕНТОВ. ОБЫЧНО ИЗОБРАЖАЕТСЯ В ВИДЕ ГРАФА С ВЕРШИНАМИ (СУЩНОСТЯМИ) И РЕБРАМИ (ОТНОШЕНИЯМИ).

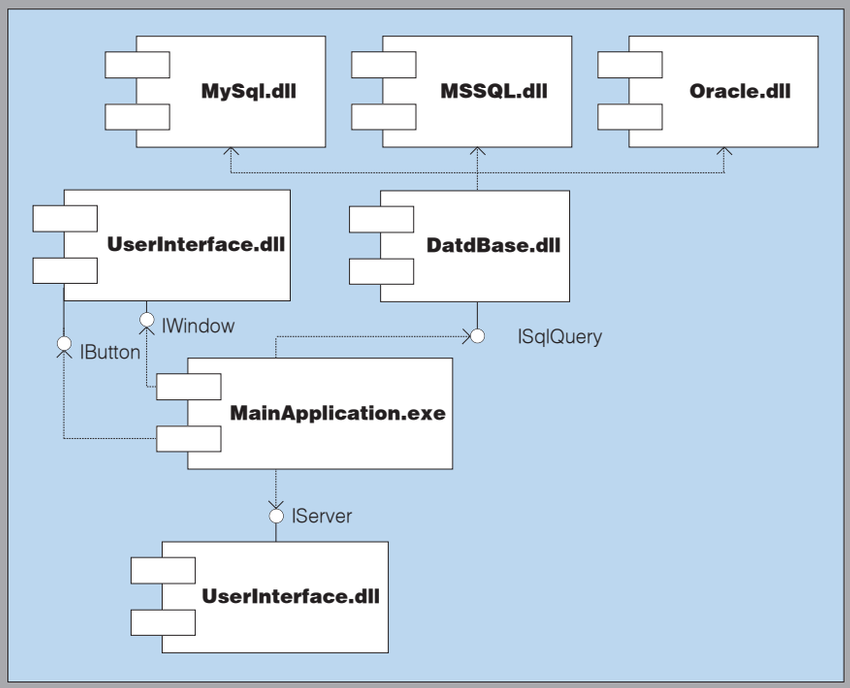
ЗАВИСИМОСТЬ — СЕМАНТИЧЕСКОЕ ОТНОШЕНИЕ МЕЖДУ ДВУМЯ СУЩНОСТЯМИ, ПРИ КОТОРОЙ ИЗМЕНЕНИЕ ОДНОЙ (НЕЗАВИСИМОЙ) СУЩНОСТИ МОЖЕТ ПОВЛИЯТЬ НА СЕМАНТИКУ ДРУГОЙ (ЗАВИСИМОЙ).

ИНТЕРФЕЙС — МНОЖЕСТВО ОПЕРАЦИЙ, КОТОРЫЕ СОСТАВЛЯЮТ СПЕЦИФИКАЦИЮ УСЛУГ, ПРЕДОСТАВЛЯЕМЫХ КЛАССОМ ИЛИ КОМПОНЕНТОМ.

КЛАСС — ОПИСАНИЕ МНОЖЕСТВА ОБЪЕКТОВ, ОБЛАДАЮЩИХ ОБЩИМИ АТРИБУТАМИ, ОПЕРАЦИЯМИ, ОТНОШЕНИЯМИ И СЕМАНТИКОЙ.



Люди и столы. Все можно напечатать



Разделяй на компоненты и властвуй!

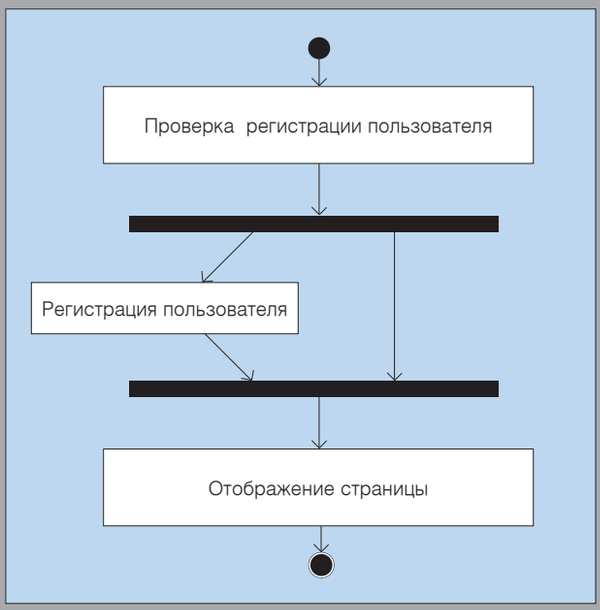


Диаграмма состояний

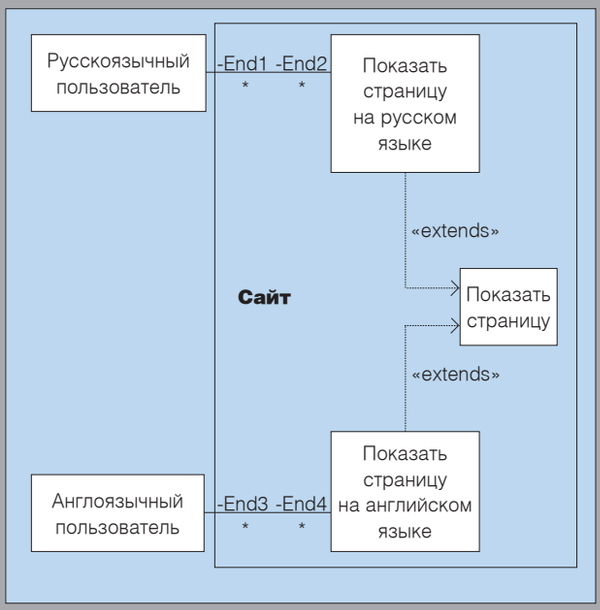
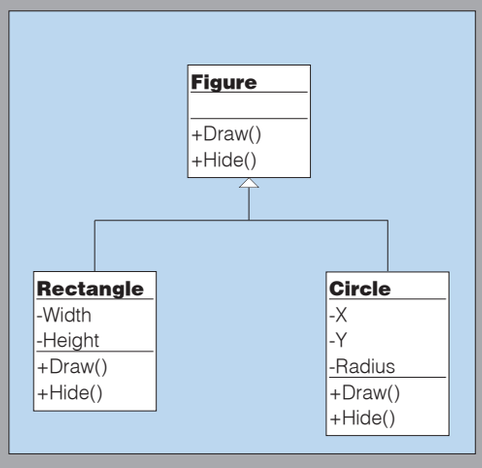


Диаграмма прецедентов для многоязычного сайта

КОМПОЗИЦИЯ — ФОРМА АГРЕГИРОВАНИЯ, В КОТОРОЙ ЦЕЛОЕ ВЛАДЕЕТ СВОИМИ ЧАСТЯМИ, ИМЕЮЩИМИ ОДИНАКОВОЕ ВРЕМЯ ЖИЗНИ.
КОМПОНЕНТ — ФИЗИЧЕСКАЯ ЗАМЕНЯЕМАЯ ЧАСТЬ СИСТЕМЫ, РЕАЛИЗУЮЩАЯ СПЕЦИФИКАЦИЮ ИНТЕРФЕЙСОВ.
МОДЕЛЬ — УПРОЩЕНИЕ РЕАЛЬНОСТИ. СОЗДАЕТСЯ ДЛЯ ЛУЧШЕГО ПОНИМАНИЯ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ.
НАСЛЕДОВАНИЕ — МЕХАНИЗМ, С ПОМОЩЬЮ КОТОРОГО БОЛЕЕ СПЕЦИАЛИЗИРОВАННЫЕ ЭЛЕМЕНТЫ ЗАИМСТВУЮТ СТРУКТУРУ И ПОВЕДЕНИЕ РОДИТЕЛЬСКИХ ЭЛЕМЕНТОВ.

ОБОБЩЕНИЕ — ОТНОШЕНИЕ СПЕЦИАЛИЗАЦИИ/ОБОБЩЕНИЯ, В КОТОРОМ ОБЪЕКТЫ СПЕЦИАЛИЗИРОВАННОГО ЭЛЕМЕНТА (ПОТОМКА) МОГУТ БЫТЬ ПОДСТАВЛЕНЫ ВМЕСТО ОБЪЕКТОВ ОБОБЩЕННОГО ЭЛЕМЕНТА (РОДИТЕЛЯ, ИЛИ ПРЕДКА).
ПАКЕТ — УНИВЕРСАЛЬНЫЙ МЕХАНИЗМ ОРГАНИЗАЦИИ ЭЛЕМЕНТОВ В ГРУППЫ.
ПРЕЦЕДЕНТ — ОПИСАНИЕ МНОЖЕСТВА ПОСЛЕДОВАТЕЛЬНЫХ СОБЫТИЙ (ВКЛЮЧАЯ ВАРИАНТЫ), КОТОРЫЕ ВЫПОЛНЯЮТСЯ СИСТЕМОЙ И ПРИВОДЯТ К РЕЗУЛЬТАТУ, НАБЛЮДАЕМОМУ АКТЕРОМ.



Отношение обобщения

Руководство по UML, написанное самими создателями языка, гласит, что 80% проектов можно реализовать используя 20% этого языка моделирования. Вот и будем рассматривать то, что используется наиболее часто.

Основными понятиями UML являются сущности, диаграммы и отношения. Сущности, основные элементы UML, могут вступать в отношения, и они группируются на диаграммах. Моделирование на языке UML является объектно-ориентированным, поэтому основной сущностью в UML является класс.

На диаграммах класс обозначается прямоугольником, в котором записывается его имя. Если диаграмма подробная, то также указываются атрибуты (поля) класса и его методы. В нашем случае класс Man (человек) имеет три поля и два метода. Поля и методы описываются в стиле «Паскаль»: сначала идет имя поля или метода, потом его тип или тип возвращаемого значения. Для методов в скобках указываются параметры, если они есть. В начале описания может стоять модификатор доступа «+», который обозначает, что член класса является публичным, то есть доступным для использования всем. Диезом «#» обозначают члены класса, доступные только самому классу и его потомкам. «–» служит для обозначения закрытых членов, доступных только самому классу.

Более крупной сущностью (по сравнению с классом) является компонент, который обычно представляет собой физическую часть системы, например DLL (динамически подключаемую библиотеку).

Аналогичную функцию выполняют пакеты: они также служат контейнерами для других сущностей, но являются более универсальным механизмом и могут служить просто для логического разделения сущностей. Например, чтобы нагляднее представить для себя структуру программы, ты объединил в пакет элементы графического интерфейса пользователя, но в реальности они будут находиться в разных компонентах. Пакеты очень удобны при проектировании «сверху вниз» или при нисходящем проектировании: при таком подходе ты разбиваешь будущую программу на большие части (пакеты), а затем детализируешь их. На диаграммах пакет обозначается в виде папки.

С основными сущностями разобрались, теперь переходим к отношениям.

Отношение — это связь между сущностями определенного вида. В языке UML существует три вида основных отношений. Пожалуй, самым общим их видом является отношение зависимости (Dependency), которое указывает, что изменение одной сущности повлияет на другую сущность. В нашем случае изменения в классе Engine (движок) могут повлиять на класс Car (машина).

Вторым важным отношением является ассоциация (Association), которая показывает, что объекты одной сущности связаны с объектами другой сущности. Следует отметить, что отношение ассоциации является довольно общим понятием и существует его частный вид — композиция. На диаграммах она обозначается в виде стрелки с ромбиком (схема «отношение композиции»).

На очереди третье отношение — обобщение (Generalization). Это отношение отражает понятие объектно-ориентированного программирования «наследование». При таком отношении родительский класс обобщает дочерний, наследующий все его открытые и защищенные поля и методы.

В данном случае мы имеем класс Figure, который является абстрактным (его имя написано курсивом), то есть создать экземпляры этого класса невозможно. Класс «Фигура» имеет два метода: Draw (для рисования) и Hide (для сокрытия). В классах-потомках эти методы могут быть переопределены для рисования конкретных фигур: Rectangle будет прорисовывать прямоугольники, а Circle — круги.

Виды отношения между сущностями могут меняться в зависимости от стадии проектирования. Например, сначала ты можешь установить, что класс Car связан с классом Engine, и обозначишь это зависимостью. Позже, при дальнейшем уточнении системы, это отношение сменится на композицию, когда станет ясно, что двигатель будет частью машины.

посмотрим на диаграммы, которые используются чаще всего. Чаще всего при создании объектно-ориентированных систем используется диаграмма классов. Довольно часто этот вид диаграмм путают с самим языком UML, хотя он гораздо шире. На таких диаграммах с той или иной степенью детализации изображаются классы, интерфейсы (иногда кооперации) и отношения между ними, то есть фактически строится объектно-ориентированная модель. Частое использование этих диаграмм также обусловлено тем, что большинство инструментов UML умеют создавать код по диаграммам классов. Другими словами, работа будет происходить в два этапа: на первом происходит создание диа грамм и кода по ним, на втором — реализация методов классов. Самые продвинутые инструменты UML умеют создавать диаграммы классов по готовому коду, такой процесс называется обратным проектированием, или реинженерингом. Его применяют в тех случаях, когда необходимо разобраться в написанном исходном коде программы: рассмотреть диаграмму с парой десятков классов проще, чем разбираться в тысячах строчек кода. На простом примере посмотрим, как строятся такие диаграммы.

Допустим, от нас потребовали написать программу, которая печатает некие отчеты о студентах. Мы спроектируем (хотя это громко сказано) ядро программы. Обычно проектирование начинается с составления словаря предметной области, в котором существуют два понятия: студент и отчет. Студент — это человек с именем и фамилией, датой рождения, факультетом, специальностью, группой и т.д. Отчет — это некая информация, представленная в бумажной форме, о студенте или студентах. Данные две сущности являются прямыми кандидатами в классы, причем класс Report (отчет) будет зависеть от класса Student — студент (см.картинку).

Если в отчетах, кроме фамилии и имени, необходимо печатать, скажем, и возраст студента на данный момент, то он должен вычисляться самим классом Student по текущему дню рождения студента. Для печати класс отчетов будет использовать метод Print, а сама распечатка будет происходить через подсистему печати. Данные о студентах будем брать из базы данных при помощи специальных классов. Теперь диаграммы выглядят так, как показано на схеме «Уточненная диаграмма классов».

Дальше можно уточнять подсистемы и довести программу до логического конца. Однако в мире есть три неизбежные вещи: смерть, налоги и изменение требований к программе. Наш заказчик говорит, что теперь необходимо поддерживать печать отчетов не только о студентах, но и о преподавателях. Вот и введем новый класс Professor и выясним, в каких отношениях с другими классами он состоит. Очевидно, что от него зависит класс Report, а с классом Student у него находится что-то общее. Действительно, и студенты, и преподаватели суть люди :). Создаем общий абстрактный класс с методом GetPrintInfo, который будет возвращать информацию для печати: по студентам — размер стипендии (или возраст, или номер группы), по преподавателям — размер зарплаты (схема «У преподавателей и студентов есть что-то общее» :)).

инструментарий

ARGOUML — СРЕДСТВО ПРОЕКТИРОВАНИЯ НА UML ДЛЯ ЯЗЫКА JAVA.
BORLAND TOGETHER — СИСТЕМА ДЛЯ РАБОТЫ С UML, ОТ СОЗДАТЕЛЕЙ DELPHI.
ECLIPSE — ПОДДЕРЖИВАЕТ ЯЗЫК UML ПРИ ПОМОЩИ ПЛАГИНОВ ECLIPSE MODELING FRAMEWORK (EMF) И UML 2.0.
MICROSOFT VISIO — ПРЕКРАСНЫЙ ПРОДУКТ ОТ НЕБЕЗЫЗВЕСТНОЙ ФИРМЫ (ВЫБОР АВТОРА, КАК ГОВОРИТСЯ :)).
MODEL MAKER — ПОСТАВЛЯЕТСЯ В ВИДЕ ПЛАГИНА ДЛЯ DELPHI ИЛИ VISUAL STUDIO .NET, ПОДДЕРЖИВАЕТ НЕСТАНДАРТНЫЕ ВИДЫ ДИАГРАММ, РЕФАКТОРИНГ И ШАБЛОНЫ ПРОЕКТИРОВАНИЯ.
RATIONAL ROSE И RATIONAL SOFTWARE ARCHITECT — САМЫЕ ИЗВЕСТНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ, ИХ ПОДДЕРЖИВАЕТ РАЦИОНАЛЬНЫЙ УНИФИЦИРОВАННЫЙ ПРОЦЕСС ПО СОЗДАНИЮ ПРОГРАММ.

И вот завершающий штрих проектирования — печать отчетов о столах. Создаем новый класс Desk, он будет представлять собой сущность «стол». Чтобы ввести поддержку печати отчетов о столах, создадим интерфейс IPrintable с методом GetPrintInfo. Класс, о котором можно создать отчет, должен поддерживать этот интерфейс — мы будем использовать не наследование классов, а интерфейсы.

Класс Report теперь использует интерфейсы, а не классы, поэтому, если потребуется напечатать отчет о каком-нибудь новом классе, то будет достаточно реализовать в нем метод GetReportInfo. Дальнейшее развитие программы в твоих руках :)…

когда-нибудь составлял требования к программе? Складывается такое ощущение, что заказчик прилетел к тебе с другой планеты и разговаривает на неопознанном языке :). В лучшем случае программист сам себе заказчик и пишет программу, которую будет использовать в своих же целях, но и тут нужна четкость и определенность — нужно же узнать, что собираешься писать. Диаграммы прецедентов графически изображают варианты использования нашей программы. При помощи диаграмм выразим требования к сайту о поддержке двух языков: русского и английского, для чего нужно выявить актеров (действующих сущностей — это пользователи сайта, они будут изображаться в виде человечков (вспомни начало статьи о пещерных программистах :)) и варианты использования — отображение страниц на разных языках («Диаграмма прецедентов для многоязычного сайта»).

Обрати внимание на связи между прецедентами: они точно такие же, как и на диаграмме классов.

на начальном этапе проектирования было бы очень неплохо разбить программу на части (обычно это отдельные библиотеки), для чего используют диаграмму компонентов. Опять же, рассмотрим пример приложения, которое работает с сетью и с базами данных MySQL, MSSQL и Oracle. Это приложение можно разбить на компоненты так, как показано на схеме «Разделяй на компоненты и властвуй».

Организацию взаимодействия библиотек можно осуществлять через интерфейсы, что тоже отражено в диаграмме.

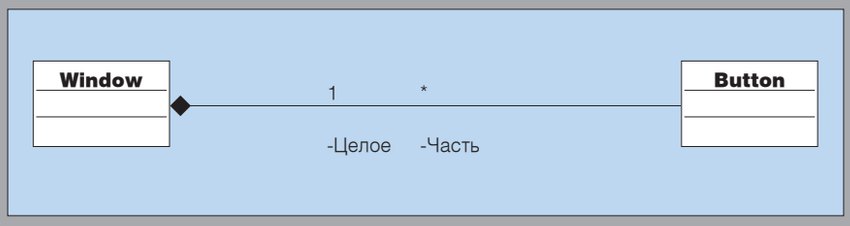
диаграмма развертывания Также ее называют диаграммой размещения, обычно она применяется при создании распределенных систем. Спроектируем систему удаленного управления (кому-то может показаться, что это Троян) :). Программа будет состоять из двух исполняемых файлов: server.exe — для запуска на управляемом компьютере, client.exe — для запуска на управляющем компьютере. Управляемых компьютеров может быть несколько, отразим это на диаграмме развертывания.

На этой диаграмме компьютеры (в рамках UML они называются узлами) обозначаются в виде кубов, а с обозначением компонентов ты уже хорошо знаком.

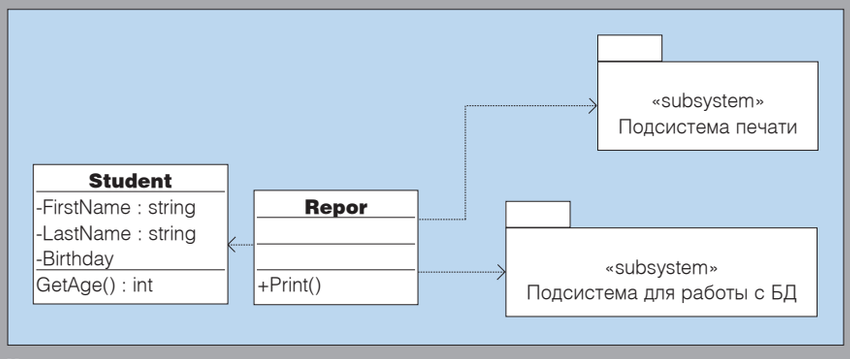
диаграмма состояний При построении диаграммы состояний рассматривается некий автомат, который имеет начальное, конечное и промежуточные состояния. В зависимости от событий, происходит переход из одного состояния в другое. Начальное состояние обозначается кружком, а конечное — кружком в окружности. Остальные состояния представляют собой скругленные прямоугольники с текстом. Переходы из состояния в состояние указываются при помощи стрелки.

Итак, на наш сайт пришел пользователь. Необходимо реализовать поддержку регистрации посетителей на сайте. Сценарий такой (его можно записать и в виде диаграммы прецедентов): если пользователь зарегистрирован, то просто показываем ему страницу, а если иначе, то предлагаем зарегистрироваться (обращаем внимание на «диаграмму состояний»).

вот и добрались до конца! Жаль, что удалось рассказать не обо всем, о чем хотелось. Некоторые виды диаграмм не были даже упомянуты, кое-что из статьи нам пришлось вообще удалить из-за того, что не влезало в размер :), но в целом мы выполнили свою главную задачу — теперь ты знаком с основами UML и при желании сможешь узнать более специализированные вещи (например, неплохо бы освоить официальное руководство по языку). Могу посоветовать только одно — поставить программу для работы с UML и попробовать его в деле. Не пожалеешь — это совершенно точно!



Отношение композиции



Уточненная диаграмма классов

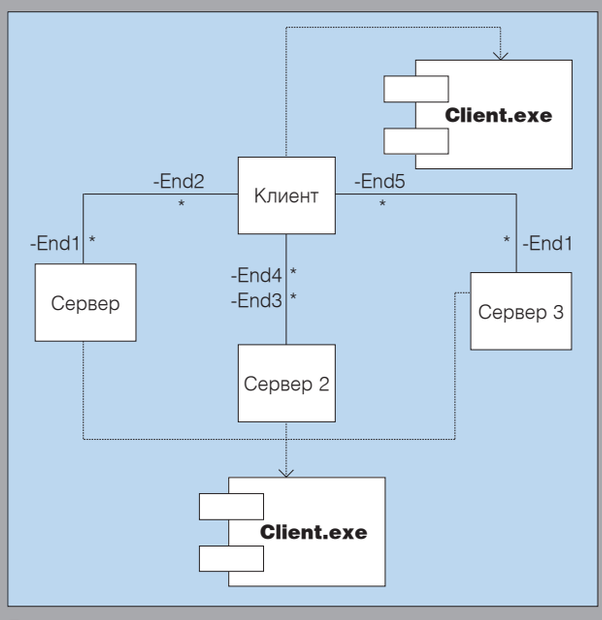


Диаграмма развертывания

каждому по потребностям

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ J2ME

УЖЕ НЕСКОЛЬКИХ ЛЕТ В УРАЛЬСКОМ РЕГИОНЕ РАБОТАЕТ ОПЕРАТОР СОТОВОЙ СВЯЗИ «МОТИВ» (ЕКАТЕРИНБУРГСКАЯ СОТОВАЯ СВЯЗЬ). НАС ИНТЕРЕСУЕТ НЕ САМ ОПЕРАТОР, А ЕГО САЙТ И НАГЛЯДНОЕ ИСПОЛЬЗОВАНИЕ НАВЫКОВ ПРОГРАММИРОВАНИЯ НА JAVA | NONAME

Операторы сотовой связи, как правило, оснащают свои сайты услугой бесплатной отсылки SMS внутри сети. Чтобы не давать волю спам-роботам, на странице отправки встраивается защита в виде «картинки с цифрами». Пользователям предлагается вручную ввести цифры (и/или буквы), нарисованные на картинке, — и так нужно удостоверить то, что SMS отправляет человек, а не робот. Как выяснилось, на сайте «МОТИВа» подобной проверки нет (**www.ycc.ru/sendsms**). Мое удивление было огромно! Для спамеров открываются большие перспективы: если использовать бот-сети, то можно запросто отправить всем абонентам «МОТИВа», к примеру, SMS с рекламой.

Однако эта «уязвимость» может послужить и более мирным целям. К примеру, написанию мидлета, который выполнял бы те же функции, как если бы ты отправлял SMS с сайта.

плюсы мидлета

- ЗАТРАТЫ ТРАФИКА МИНИМАЛЬНЫ.
- ОБЕСПЕЧИВАЕТСЯ МОБИЛЬНОСТЬ ЧЕЛОВЕКА, КОТОРЫЙ ОСУЩЕСТВЛЯЕТ ОТПРАВКУ.
- ОФИЦИАЛЬНЫЕ РАСЦЕНКИ (В СЕТИ «МОТИВА») НА ПЕРЕСЫЛКУ SMS ВЫШЕ, ЧЕМ ПРИ ИСПОЛЬЗОВАНИИ ЭТОГО МИДЛЕТА.

минусы мидлета

- SMS ОТПРАВЛЯЮТСЯ ТОЛЬКО ВНУТРИ СЕТИ (АБОНЕНТЫ ЛЮБОГО ОПЕРАТОРА, КРОМЕ «МОТИВА», НЕ СМОГУТ ДАЖЕ ПОЛУЧАТЬ SMS ОТ ТЕБЯ — СМОГУТ ТОЛЬКО ОЗНАКОМИТЬСЯ С УСЛУГОЙ ЗАЙДЯ НА САЙТ).
- ЧТОБЫ НАЧАТЬ ПОЛЬЗОВАТЬСЯ НОВЫМ СПОСОБОМ ОТПРАВКИ SMS, НУЖНО НАСТРОИТЬ УСЛУГУ GPRS НА ТЕЛЕФОНЕ.
- ОБРАТНЫЙ АДРЕС, ТО ЕСТЬ НОМЕР, НА КОТОРЫЙ БУДУТ ПРИХОДИТ SMS-ОТВЕТЫ, — НЕ ТВОЙ, А НОМЕР СПРАВОЧНОЙ СЛУЖБЫ ОПЕРАТОРА.

о ценах скажу то, что стоимость одной SMS составляет 1 руб. 20 коп. Опытным путем удалось выяснить, что SMS, отправленное из мидлета, будет стоить всего-навсего 15-20 коп.!

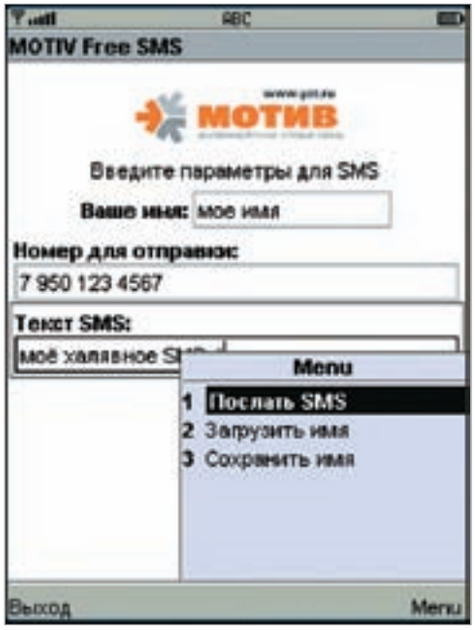
Общие принципы выяснены. Написать мидлет не составит труда, даже если ты не знаком с J2ME. Любая документация по программированию под мобильные телефоны легко найдется в Сети.

интересная часть мидлета

```
  public void SendSMSToServer(String uname,
String pnum, String smstext)
  {
//строка заXORена просто для смеха :)
//static String normalycc_server =
"http://www.ycc.ru/sendsms/index.html";
  static String ycc_server =
"`llx2''□□□&qkk&z}'{mfl{e{'aflmp&`led";

  String tmp_server;
    // xor string on 8
    char buk;
    tmp_server="";

    for (int i=0;i<ycc_server.length();i++)
    {
    buk = ycc_server.charAt(i);
    buk ^= 8 ;
    tmp_server += buk;
    }

    HttpDataConnection hdc = new
HttpDataConnection();

try {
    hdc.openHTTP(tmp_server);
    stringItem1.setText("отправка данных...");

// это основные параметры, вытащенные
из формы на сайте
  byte data[] = ("sendsms=yes&from_name="
+ uname +
        "&prefix=" + pnum.substring(0, 6) +
        "&phone_number=" + pnum.substring(6, 11) +
        "&text_sms=" + smstext +
        "&sim=" + Integer.toString(260 —
smstext.length())).
        getBytes();

      hdc.httpProperties.put("Content-Length",
data.length + "");
      hdc.httpProperties.put("User-Agent",
"Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)");
      hdc.httpProperties.put("Connection", "close");
```

sms-пытки

КИТАЙСКИЕ ВЛАСТИ ПРИДУМАЛИ СТРАШНОЕ НАКАЗАНИЕ ДЛЯ ЖУЛИКОВ. ТОЧНЕЕ, ДЛЯ ТЕХ, КТО В КИТАЕ РАЗВЕШИВАЕТ БУМАЖНЫЕ ОБЪЯВЛЕНИЯ О ПРОДАЖЕ ПОДДЕЛЬНЫХ ДОКУМЕНТОВ И УЧЕБНЫХ СЕРТИФИКАТОВ. ЧАСТО НАРУШИТЕЛИ ОСТАВЛЯЮТ НОМЕРА СВОИХ СОТОВЫХ ТЕЛЕФОНОВ. ВЛАСТИ КИТАЯ РЕШИЛИ ВЫШИБАТЬ КЛИН КЛИНОМ — КАЖДЫЕ 20 СЕКУНД НА ЭТИ СОТОВЫЕ ТЕЛЕФОНЫ ПРИХОДИТ СООБЩЕНИЕ: «ВЫ НАРУШИЛИ ЗАКОН. ВЫ НЕМЕДЛЕННО ДОЛЖНЫ ПРЕКРАТИТЬ РАЗВЕШИВАТЬ НЕЛЕГАЛЬНУЮ РЕКЛАМУ И НАПРАВИТЬСЯ В БЮРО НАКАЗАНИЯ».



Так все это выглядит на телефоне

Все прекрасно работает :)

```
        hdc.openOutputStream();
        hdc.post(data);

            stringItem1.setText("SMS отправлено!");
}
        catch(Exception e) {
            stringItem1.setText("Нет (E)GPRS-связи!
\nИсключение: " + e.getMessage());
    }
    finally {
        hdc.close();
        }

  }
```

Чтобы уменьшить расход трафика и чтобы после отправки POST-данных не принимать страницу, на которой написано «SMS отправлено успешно!», в соответствующий момент соединение должно быть закрыто:

```
hdc.httpProperties.put("Connection", "close");
```

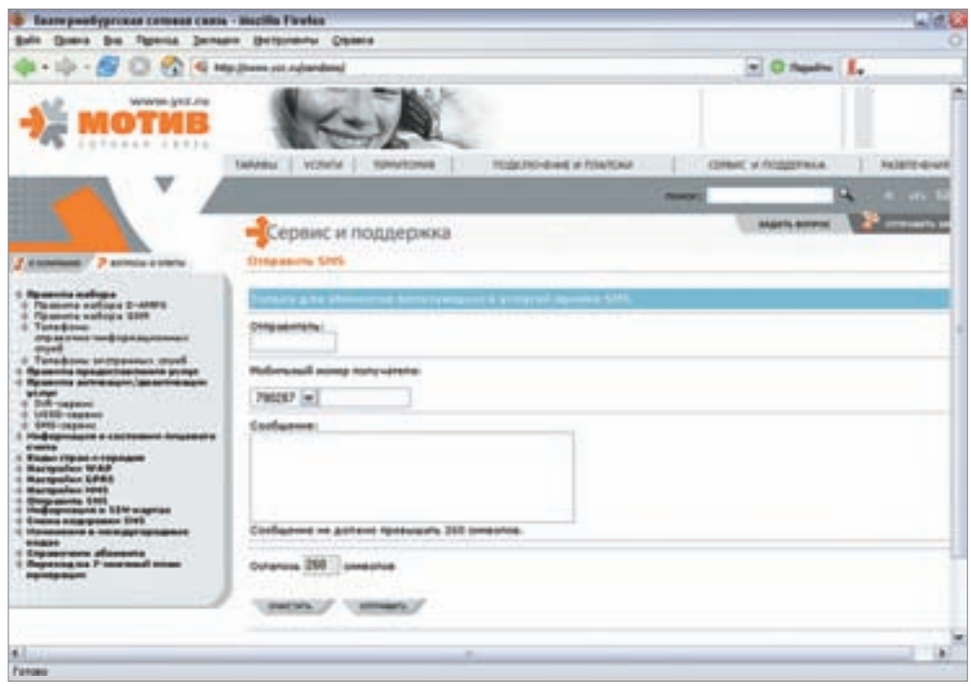
Собственно, код отправки использует специальный кустарный класс (найден в интернете) для удобства работы с http. Сам мидлет представляет собой контролы для ввода текста, накиданные на форму (имя отправляющего, текст SMS и номер абонента). Номер проверялся на валидность нужного оператора вот так:

```
    String subnum = phone.getString().sub-
string(0,6);

        // проверяем номера на МОТИВ'овские
(взято с того же сайта)
        if (subnum.equals("790287") ||
subnum.equals("790438") ||
subnum.equals("790454") ||
            subnum.equals("790498") ||
subnum.equals("790863") ||
subnum.equals("790890") ||
        subnum.equals("790891") ||
subnum.equals("790892") ||
subnum.equals("795019") ||
        subnum.equals("795020"))
        {
// если все OK, то создаем поток для http-коннекта
  waitform wf = new waitform(username.get-
String(),phone.getString(),smstxt.getString());
  Display.getDisplay(MainMidlet.instance).setCurrent(wf);
conn logger = new conn(wf,username.get-
String(),phone.getString(),smstxt.getString());
  logger.start();
  System.gc();
        }
```

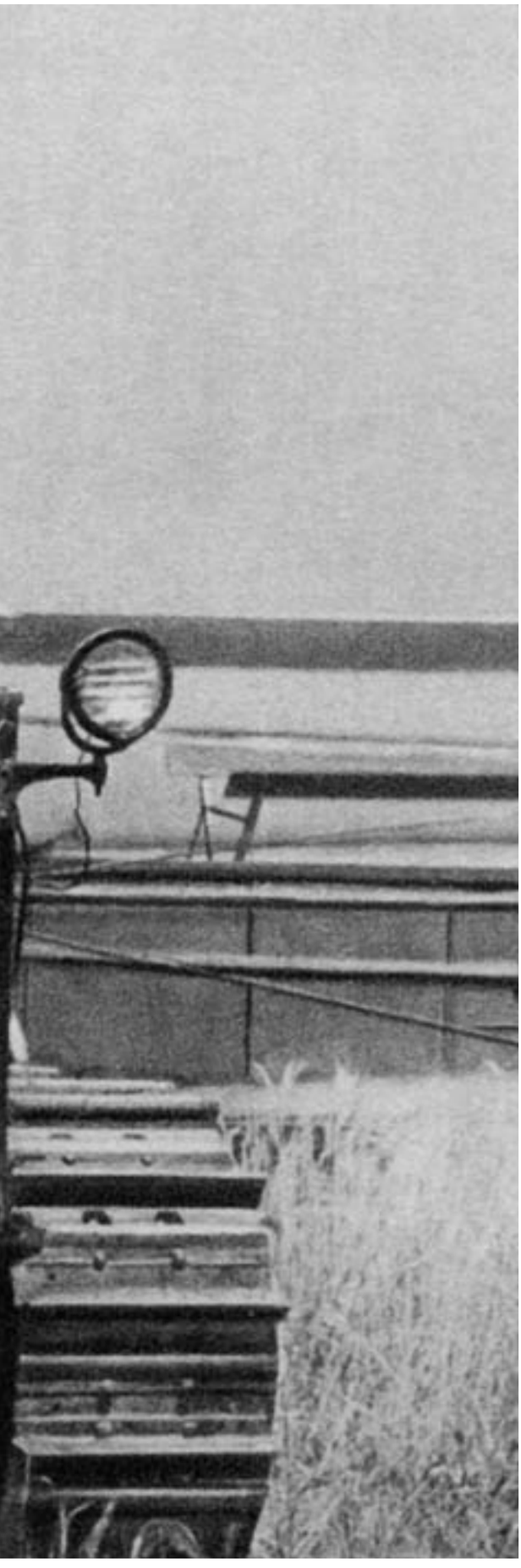
подробнее можно посмотреть в исходниках на диске к журналу. Проект был написан полностью в среде Borland JBuilder 2006.

проблема, с которой я столкнулся при написании мидлета: текст в телефоне вводится и хранится в кодировке UTF-8 (UNICODE), а сервер принимает POST-данные в ASCII. Буквы английского алфавита принимаются на телефоне адресанта нормально, но вместо русских символов доходят только вопросительные знаки. В одном из публичных мидлетов мне удалось найти функцию для конвертирования UNICODE в ASCII, ее я и применил для конвертирования текстовых полей.

такие «дыры» можно найти на сайтах и других операторов (к примеру TELE2), а не только «МОТИВа». Администраторам безопасности компаний — сотовых операторов стоит уделить больше внимания своим сайтам...



Интерфейс сайта оператора



удобный визуальный комбайн

DELPHI — СОВЕТ ПРОФЕССИОНАЛА

ЛЮБОЙ УВАЖАЮЩИЙ СЕБЯ РАЗРАБОТЧИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДОСКОНАЛЬНО ЗНАЕТ ТОТ ИНСТРУМЕНТ, КОТОРЫМ ЗАРАБАТЫВАЕТ НА ЖИЗНЬ. ДЛЯ ТОГО ЧТОБЫ СТАТЬ ПРОФЕССИОНАЛОМ В КАКОЙ-ТО ОБЛАСТИ, НУЖНО ЗНАТЬ ВСЕ. НУЖНО ЗНАТЬ И ТО, ЧТО ПОСТИГАЕТСЯ ТОЛЬКО НА ОПЫТЕ |**ВИТАЛИЙ ИЖЕВСКИЙ, ВЛАДИМИР ХОПТЫНЕЦ**

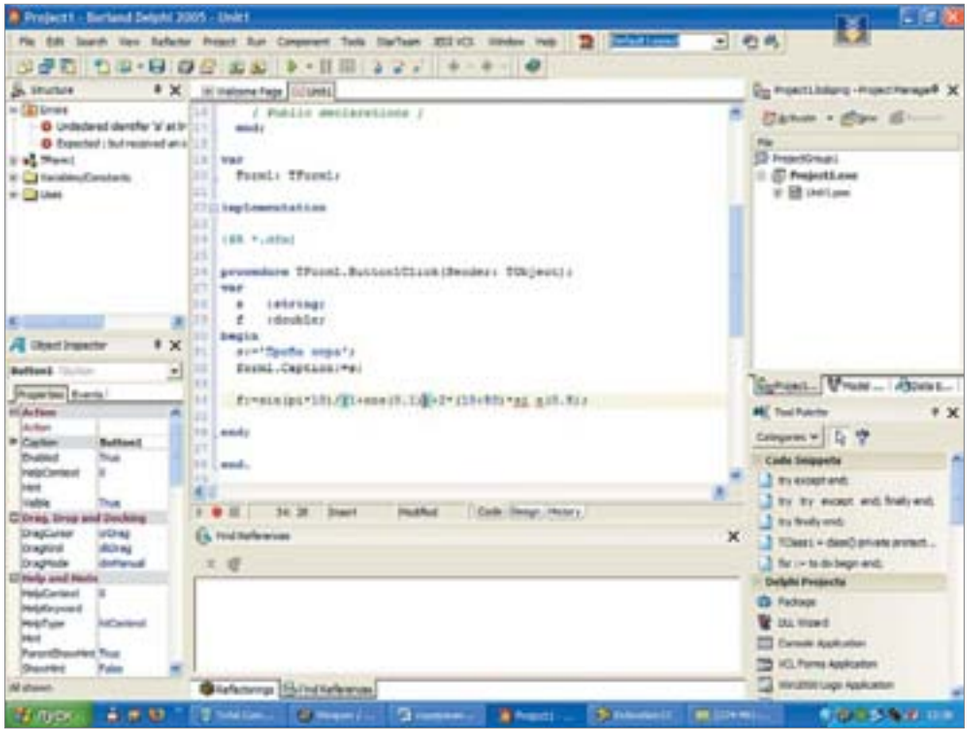
с появлением RAD-систем, таких как Delphi 2006, работа программистов упростилась. В их распоряжение поступили последние писки технологий программирования и возможность чувствовать мощь интеллекта на кончиках своих пальцев (несмотря на высокоуровневость).

Рано или поздно все разработчики сталкиваются с одним и тем же вопросом: какую RAD-систему выбрать из множества имеющихся? Какая версия Delphi нужна тебе? Решай эту дилемму в зависимости от того, что именно ты хочешь создать. Опыт позволяет нам сказать, что сред разработки, более стабильных, чем Delphi 7 + Update 2, пока не встречается. Конечно, там нет таких супермодных вещей, как рефакторинг или поддержка .NET, но можно подыскать и что-то из продуктов сторонних разработчиков и подключить к стандартной среде. Для того чтобы работать с проектами .NET, тебе придется переходить на Delphi 8 (от которого не все в восторге) или Delphi 2005 (даже после третьего обновления эта среда разработки остается очень нестабильной, изобилует ошибками разного рода, а ее быстродействие и требовательность к ресурсам оставляет желать лучшего).

новая Delphi Developer Studio 2006 появилась совсем недавно, в нее включены Delphi for Win32, Delphi for .Net, Borland C++ Builder, Borland C# Builder. По всем параметрам она обходит Delphi 2005. Интерфейс максимально приближен к Visual Studio 2005, хотя разработчику предоставляется полная свобода в настройке и размещении окон.

Естественно, козырной картой любого IDE считается редактор кода, в Delphi 2006 он выше всяких похвал. Различная подсветка кода создает желание программировать: вся цветовая гамма отлажена и радует глаз, редактируемая строчка кода подсвечивается мягким цветом, неправильные идентификаторы и ошибки подчеркиваются красной линией (как в Word, если находятся синтаксические ошибки в тексте). Больше не нужно искать парные кавычки и дужки в сложных математических выражениях — они тоже подсвечиваются.

Один из наиболее приятных моментов — возможность пользоваться реинженерингом кода,



Подсветка кода

намного более совершенным, чем в версии 2005. Несколькими кликами мыши можно оформить выделенный кусок кода в метод. Интеллектуальный редактор сделает все сам — просто выделяешь нужный участок кода и кликаешь в контекстном меню Refactoring→ ExtractMethod, а потом указываешь имя метода.

Режим SyncEditMode позволяет редактировать одинаковый текст одновременно в нескольких местах (включается пиктограммкой сбоку в редакторе при выделении текста). Аналогично происходит переименовывание переменных и методов по всему тексту, автодекларация переменных. Есть автоматическое разделение одного юнита на несколько. Интеллектуальные вставки позволят быстро вставить шаблоны кода типа if then, case, type. Естественно, остались такие приятные мелочи, как создание заголовка процедуры в разделе реализации по ее описанию (<Ctrl>+<Shift>+<C>). Кстати, теперь в Delphi реализована «полноценная поддержка русского языка», в именах идентификаторов можно использовать русские буквы.

Многие не пользуются очень удобной возможностью — механизмом bookmarks (закладок), который позволяет быстро возвращаться к разным местам редактируемого кода (реализуется с помощью пунктов контекстного меню Toggle Bookmarks и Goto Bookmarks).

Версия 2006 работает намного быстрее своей предшественницы и в отличие от нее позволяет загружать среду не полностью (в 2005-й загружались все библиотеки и языки, сейчас же можно загружать только то, с чем будешь работать).

В IDE реализовано сохранение истории изменений. В папке с проектом Delphi создает подпапку history, перед сохранением измененных файлов в нее записываются старые копии файлов. Раньше создавалась только последняя копия, хотя, конечно, было бы разумнее использовать полноценную CVS (Concurrent Versions System — система контроля версий), с помощью которой ты быстро просматриваешь внесенные изменения, организовываешь коллективную работу с исходниками, возвращаешься к предыдущим версиям файлов, а также работаешь одновременно над несколькими релизами программы. В качестве рекомендации — Jedi VCS (**http://jedivcs.sourceforge.net**), построенная на основе FireBird (можешь использовать собственный настроенный SQL-сервер — просто укажи в настройках путь к серверу), которая интегрируется в среду Delphi (после установки появляется новый пункт главного меню JEDI VCS).

компонентный подход предоставляет явные преимущества. А что делать, если нет нужного или если то, что есть, не обеспечивает желательный функционал программы?

Выход первый

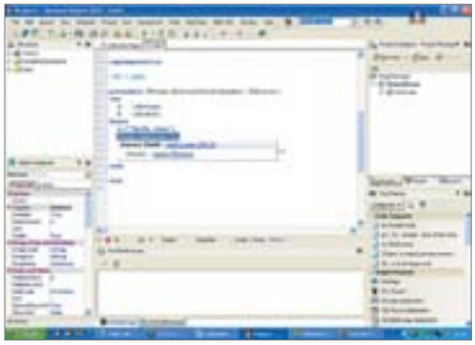
Интернет просто кишит разного рода компонентами, а форумы — соответствующими темами по

КАК ЭТО БЫЛО

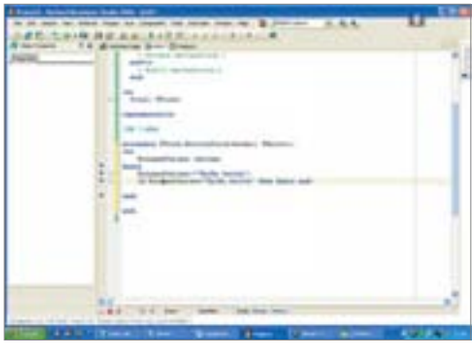
сегодня на многих форумах можно обнаружить группы, на которые разделились программисты. Выше всего задирают носы «сишники». Брызжа слюной, они восхваляют C++: нет на свете языка «оптимизированнее и заточеннее под системные вызовы и конкретные процессоры». В ответ им «жабники» (почитатели Java) доказывают роль безопасности стека и кучи памяти. Любители Basic скромно поднимают вопрос о простоте, но, получив несколько эпитетов в свой адрес насчет примитивности мышления, поспешно удаляются. Толстые и ленивые паскалисты, находясь «в поисках очень нужного компонента», растопыривают пальцы и доказывают преимущества VCL. Супермодняцкие C# кричат: «Win32 умерла. Выйдет Vista — мы всех по стенке размажем». А хитрые и прокуренные ассемблеристы подкалывают: «SoftIce еще никто не отменял, еще придете к нам за крэкой любимой Visual Studio».

Мы посмотрим на ситуацию по-другому. Канули в лету те времена, когда успех проекта зависел от выбора языка программирования. Сейчас все зависит от интеграции компонентов, от эффективной справочной системы, возможности быстрой разработки, использования шаблонности подходов и т.д. Одним словом, сегодня быстрота разработки приложения и возможность быстро воплотить идею в код являются основополагающими критериями для программистов.

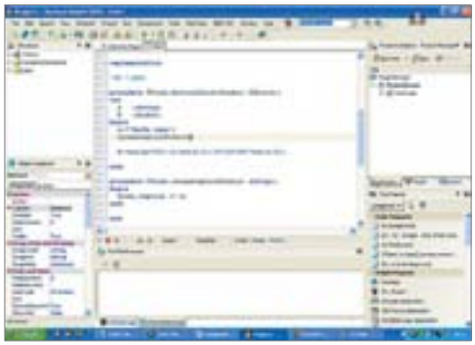
Люди, которые используют спартанские компиляторы, могут только мечтать о том функционале, который обеспечивается IDE (интегрированная среда разработки — Integrated Development Environment). Так что стоит разобраться в сущности загадочного IDE — что оно значит и когда оно появилось.



Интеллектуальная подстановка кода



Русские идентификаторы возможны



Реинженеринг кода в действии

Delphi. Трудно выбрать нужное из такой массы или найти компонент именно под используемую версию. К тому же, если ты воспользовался сторонними компонентами, программа начинает работать так, как хотел разработчик компонентов, а не как хотел ты. Тебе достанутся ошибки разработчика, его неоптимальный код, его тормоза в работе программы и т.д. Не обольщайся, идеальных программ просто не бывает — их разрабатывают люди.

Выход второй

Неужели руки большинства разработчиков растут настолько криво? Что, нельзя уже и собственными компонентами обзавестись? Конечно, на такой случай никто не станет изобретать велосипед, но бывает, что не удается найти именно велосипед. В Delphi используется обычная объектная модель и каждый компонент является потомком какого-нибудь базового класса. Работает нормальная наследуемость свойств и методов, их переопределение. Приведем простенький пример создания собственного компонента (в конце концов, все не так сложно). Запускаешь Delphi for Microsoft Win32. Выбираешь тип проекта: Delphi Projects→ Delphi Files→ Component...

Появляется окно выбора компонента, потомком которого будет наш собственный компонент. Мы хотим просто добавить некое свойство MyInt к стандартному набору свойств компонента TButton — выбираем его и указываем, на какой вкладке палитры компонентов должен быть помещен новый компонент.

ЛИСТИНГ МОДУЛЯ

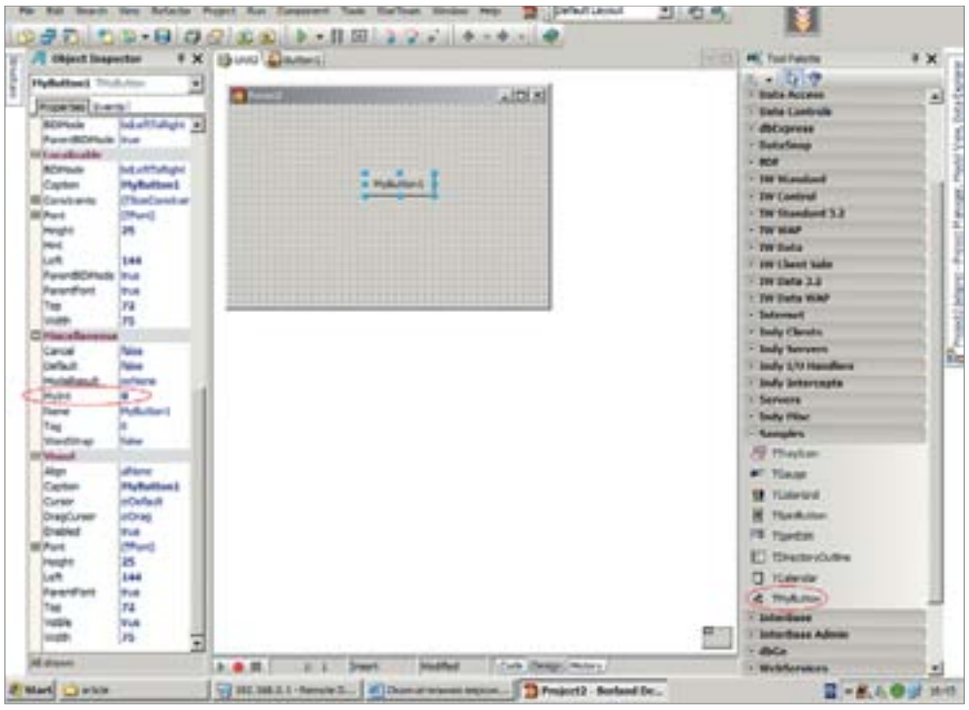
```
unit Button1;

interface

uses
  SysUtils, Classes, Controls, StdCtrls;

type
  TMyButton = class(TButton)
  private
    { Private declarations }
    FMyInt: Integer;
    procedure SetMyInt(const Value: Integer);
    function GetMyInt: Integer;
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    { Published declarations }
    property MyInt:Integer read GetMyInt write SetMyInt;
  end;

procedure Register;
implementation

procedure Register;
begin
  RegisterComponents('Samples', [TMyButton]);
end;

function TMyButton.GetMyInt: Integer;
begin
result := FMyInt;
end;

procedure TMyButton.SetMyInt(const Value: Integer);
begin
FMyInt := Value;
end;
end.
```

все началось во времена господства Windows 3.1. В то далекое время программисты работали намного изощреннее, чем сейчас. Сложные задачи решались оптимизацией кода и алгоритма работы программы, не как сейчас — выбором «чего бы помощнее прикупить». В то же время куча сил тратилась на такие тривиальные действия, как создание интерфейса и обеспечение его функциональности. Единого подхода как такового не было, и много времени уходило не столько на решение задачи, сколько на программирование средств взаимодействия с пользователем. Количество задач, требовавших решения, росло быстрее, чем количество решений. Вдруг блеснула замечательная идея — визуальное программирование. К тому времени уже развилась концепция объектного программирования и сложные задачи уже превращались в конструкции из модулей. Очень мощный, но достаточно сложный C++ заполнял все свободное мозговое пространство...

Компилируем проект через <Shift>+<F9> или Project\Build Paqckage1. В директории, выставленной по умолчанию для новых проектов, появится файл Package1.bpl. Теперь закрыть все и открыть меню Component\Install Packages. Добавить путь к указанному файлу, нажать вездесущую кнопку OK. Теперь, если будет создан новый проект для Win32 и если будет открыта вкладка Samples, то можно поместить на форму компонент MyButton, уже содержащий новое свойство MyInt.

Вот ты пишешь собственные компоненты. Долгие бессонные ночи, радость от порождения гениальных идей. Мучительно отлаживаешь… После всего этого некоторые люди начинают подумывать о том, как бы извлечь побольше выгоды и как защитить свое детище, а кое-кто — как обойти все защиты и получить побольше и побесплатнее :).

принципы защиты кода Здесь есть несколько моментов, которых нужно коснуться. То, что использование стандартных функций API или компонентов ни в коем случае не ведет к эффективной защите, — не новость. Не секрет и то, что набор хакера довольно велик и при использовании все того же старого доброго SoftIce очень помогает настройка символьных имен для любых библиотек, которые просто на пальцах рассказывают, что к чему. Так что, какие бы желания ни возникали, как бы мы ни расписывали прелести RAD, придется обратиться за помощью к ассемблеру.



Установлен новый компонент

Если ты используешь простой способ хранения регистрационного кода, не следует пользоваться проверкой типа:

```
if RegCode<>TrueCode then
   Begin
      ShowMessage('Неверный код, дальнейшая
работа невозможна');
      halt;
   end;
```

Назвать код такого вида «защитой» даже язык не поворачивается. Для начала посоветуем хранить код в нескольких динамических переменных и использовать при этом хотя бы какое-никакое шифрование. Кроме того, не следует сразу заявлять злоумышленнику, что он ошибся. Скажи позже, например через два дня, когда осядет пыл славы взломщика. Конечно же, для проверки кода глупо брать один-единственный алгоритм. Можно разбить код на части и проверять каждую часть отдельным алгоритмом. Плюс желательно проверять его не сразу же после запуска программы, а в тот момент, когда хитрый пользователь сделает большую часть своей работы.

Вот лишь некоторые и основные моменты по защите программ. Большинство хороших идей приходят в наши головы не из книг, а в процессе непосредственной работы — нужно просто научиться думать.

теперь об оптимизации написанного кода. Всем известен тот факт, что компилятор Delphi — самый быстрый из существующих. Но не стоит забывать и о его недостатке — элементарной неоптимальности кода, ведущей к снижению производительности. Для того чтобы не давать компилятору повода пороть код, нужно просто писать программы правильно. Удобство визуальных компонентов — большой плюс для ускорения разработки приложений, но не следует забывать, что компоненты не всегда написаны так, чтобы оптимизировать выполняемый код.

Остановимся на нескольких моментах. Возьмем, к примеру, использование упрощенных математических выражений. Компилятор Delphi

в 1995 году появился Visual Basic и Visual C++. Фурор! Программисты вздохнули с облегчением, так как появление первых интегрированных средств разработки освободило их руки от кропотливого труда над интерфейсом — теперь он стал основываться на стандартных компонентах. Работа программиста приобрела сходство с работой художника: ты просто решаешь, как будет выглядеть интерфейс, и сосредотачиваешься на функционале, то есть на сути работы программы. В то время и появилась первая Delphi.

И все поняли, что счастье возможно. Строгость и формальность Pascal'а избавила программы от множества трудно отыскиваемых ошибок, которыми пестрели программы на C++. По возможностям Delphi сильно превосходил Basic. Добавилась палитра визуальных компонентов — решение было настолько удачным, что Delphi удалось сразу же выйти в лидеры. Delphi 1 был первым инструментом разработки windows-приложений, объединившим в себе оптимизирующий компилятор, визуальную среду программирования и мощные возможности работы с базами данных. Позднее все это получило название среды быстрой разработки приложений (Rapid Application Development — RAD)...

необходимые компоненты

В DELPHI ЕСТЬ БОЛЬШОЙ ВЫБОР КОМПОНЕНТ ДЛЯ УПРАВЛЕНИЯ ДАННЫМИ, А ЕСЛИ И ЕГО МАЛО, ВСЕГДА МОЖНО ДОПОЛНИТЕЛЬНО ПОСТАВИТЬ НУЖНОЕ. ВЗЯТЬ, К ПРИМЕРУ, JEDI VISUAL COMPONENT LIBRARY — ОКОЛО 500 РАЗНЫХ КОМПОНЕНТОВ. ПРИТОМ АБСОЛЮТНО БЕСПЛАТНО ТЫ МОЖЕШЬ ПИСАТЬ ВЕЩИ С КРАСОЧНЫМ ИНТЕРФЕЙСОМ И МОЩНЫМИ СРЕДСТВАМИ ДОСТУПА И УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ. В ФЕВРАЛЕ 2006 ГОДА ВЫШЛА ВЕРСИЯ 3.20 С ПОДДЕРЖКОЙ BORLAND DEVELOPER STUDIO 2006 (МОЖНО СКАЧАТЬ ПО ССЫЛКЕ HTTP://NCHC.DL.SOURCEFORGE.NET/SOURCEFORGE/JVCL/JVCL320COMPLETEJCL197-BUILD2172.7Z).

понимает все выражения буквально и последовательно, ничего не упрощая. Если говорить о сложных выражениях, где используются математические функции, компилятор не вычисляет их — они вычисляются в процессе работы программы, поэтому писать x:=sin(30*Pi)/cos(60*sqrt(2/3)) неразумно. Любое подобное выражение будет вычисляться только непосредственно после запуска программы.

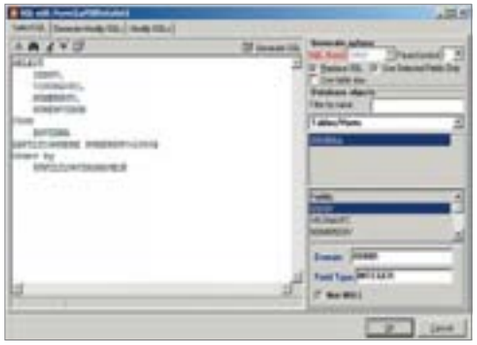
Если наворотишь что-нибудь похлеще, наделаешь матриц и запихнешь полученное в тело цикла или рекурсивной процедуры — не жалуйся, сам виноват. Также компилятор не умеет понимать равнозначности проверяемых выражений. К примеру, выражение If ((z/b)*(f-a)>0) and ((f-a)*(z/b)<100) then… правильно логически, но два выражения, которые будут проверяться, будут скомпилированы по-разному и вычислены отдельно, хотя дадут одно и то же значение. Лучше выводи подобные выражения в отдельную переменную. Всегда необходимо следить, чтобы не было лишних проверок логических условий. Возможно, существующие логические выражения можно упростить. Пользуйся правилами булевой алгебры. При реализации циклов не стоит вставлять функции в проверку условия цикла. Например, for i:=10 to length(str)-1 do... В этом цикле при каждой итерации будет вновь вычисляться выражение length(str)-1 с вызовом функции, поэтому лучше вынести это выражение за тело цикла и вычислить его, допустим, как некую переменную Maxl.

В принципе, все вышесказанное распространяется на написание программ в любой среде вне зависимости от того, сможет ли она распознать и оптимизировать глупость.

работа с БД — конек Delphi. И неудивительно! Эта среда разрабатывалась именно для создания подобных приложений. Delphi умеет работать с BDE, ODBC (через BDE), dbExpress, ADO, а также с некоторыми компонентами прямого доступа, разработанными сторонними производителями — это касается работы с базами данных DB2, Intrebase, Informix и MySQL.

Сейчас стало очень модно использовать MS SQL или Oracle. Мало кто задумывается о том, что такие «монстры» редко нужны в реальных приложениях. База данных в 200-300 Мб может прекрасно функционировать на FireBird — клоне Interbase. Delphi ориентирована на эту СУБД, так как имеет целый ряд компонентов для разработки приложений под Interbase и ее администрирования.

В чем преимущества такого подхода? В первую очередь назову размер дистрибутива СУБД — 2-3 Мб (его можно «прикрепить» к дистрибутиву приложения), легкость администрирования и, естественно, цена (не каждое предприятие, даже крупное, может позволить себе купить лицензию Oracle). Наличие компонентов прямого доступа типа FibPlus (**www.dervace.com**) позволяет учитывать особенности этой СУБД и оптимизировать код практически до уровня производительности тех же «серьезных СУБД». Отличным средством администрирования для СУБД Firebird является ibexpert, которая, кстати говоря, абсолютно бесплатна для всех пользователей на территории бывшего СССР. С помощью этого средства возможно изначальное визуальное проектирование базы данных, сохранение общих скриптов, сохранение резервной копии базы данных и ее восстановление, администрирование пользователей, внесение любых изменений в существующую структуру базы данных, написание и тестирование хранимых процедур, триггеров и любых запросов на быстродействие и ресурсоемкость. В общем, решаются самые разные задачи, а в комплекте с Delphi можно очень эффективно использовать его для разработки приложений баз данных.

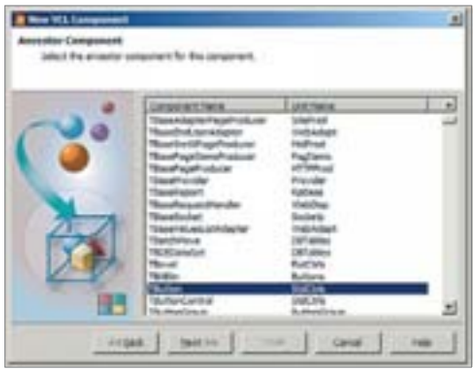


Окно работы с SQL-запросами компонента TFIBDataSet

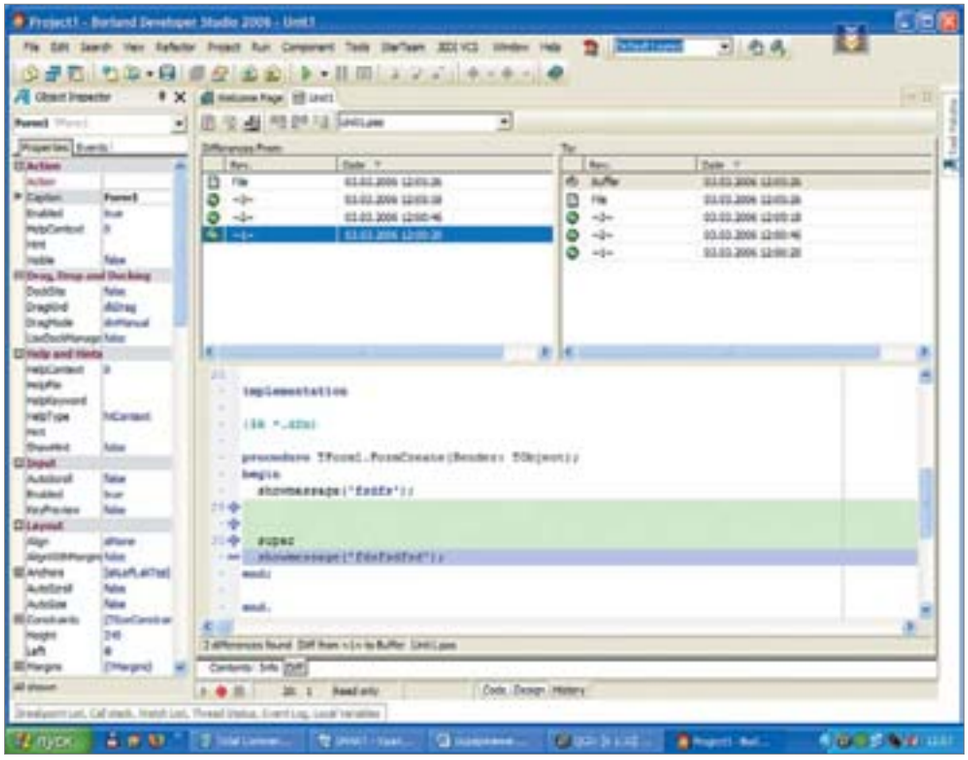
Самым древним методом доступа к данным в Delphi является BDE, появившийся в третьей версии этого продукта. Он является универсальным методом, как и ADO, OLE DB и ODBC, а универсальность — это благо: какую СУБД ни используешь, программа пишется под них одинаково. Но что же делать с быстродействием? Это требование для разработки клиентской части баз данных

ВЕРСИЯ 2006 РАБОТАЕТ НАМНОГО БЫСТРЕЕ СВОЕЙ ПРЕДШЕСТВЕННИЦЫ И В ОТЛИЧИЕ ОТ НЕЕ ПОЗВОЛЯЕТ ЗАГРУЖАТЬ СРЕДУ НЕ ПОЛНОСТЬЮ

в Delphi 2 было предложено все то же, но на новом уровне современной 32-разрядной операционной системы Windows 95 и Windows NT. Кроме того, Delphi 2 предоставил программисту 32-битовый компилятор, создававший быстрые и эффективные приложения, мощные библиотеки объектов, улучшенную поддержку баз данных, поддержку OLE, средства Visual Form Inheritance. Притом обеспечивалась совместимость со старыми 16-битовыми приложениями. Delphi 2 стал тем мерилом, по которому оценивали другие RAD. Microsoft попыталась ответить на вызов и выпустила Visual Basic 4, однако он был низкопроизводительным, не обеспечивал совместимость 16- и 32-разрядных приложений и имел несколько других заметных недостатков. Тем не менее, многие разработчики продолжали использовать Visual Basic...



Окно выбора родительского компонента



Установлен новый компонент

более критично. Вот тут и встает вопрос о специализированных компонентах для доступа к специфичным СУБД, которые используют прямой доступ к базам данных без посреднических библиотек или драйверов. Как тут не упомянуть набор компонентов FIBPlus? Простота разработки, гибкость и скорость работы приложений с использованием этих компонентов просто поражают.

подробнее о применении фильтров, позволяющих, не натирая мозоли, гибко изменять параметры и сам SQL-запрос компонента. Посмотрим простой пример (заранее нужно корректно установить в Delphi все части FIBPlus). Пусть в приложении есть компонент pFIBDataSet1. Для редактирования его SQL-строк достаточно кликнуть правой кнопкой мышки на этом компоненте и выбрать в контекстном меню SQL Generator.

То, что введено в качестве SQL-запроса, имеет несколько непривычный нам вид. @@FILT1 — название фильтра, обращение к которому осуществится в программе. %WHERE NOMERDOV>1000@ — значение фильтра по умолчанию. В качестве фильтра может быть любая часть SQL-запроса, что открывает поистине гигантские перспективы благодаря созданию динамических SQL-запросов. Приятно, что реализация осуществлена таким простым способом. Теперь в программе можно повесить следующий обработчик событий:

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  pFIBDataSet1.ParamByName('FILT1').AsString :=
'WHERE NOMERDOV<5000';
end;
```

КОМПИЛЯТОР DELPHI ВОСПРИНИМАЕТ ВСЕ ВЫРАЖЕНИЯ БУКВАЛЬНО И ПОСЛЕДОВАТЕЛЬНО, НИЧЕГО НЕ УПРОЩАЯ

SQL-запрос преобразится в мгновение ока. Этот подход позволяет реализовать самые смелые идеи самых отчаянных программистов :).

В этом же генераторе запросов можно запросто автоматически сгенерировать запросы на изменение, добавление и обновление. Если установить свойство автообновления в true и подключить простенький (на первый взгляд) компонент TDBGrid, то мы получим средство доступа к базе данных с полным набором действий.

Применение RAD Delphi существенно упрощает написание любого приложения и снимает многие рамки при разработке программного обеспечения. Однако когда профессионал выбирает то, на чем будет идти работа, в первую очередь он обращает внимание на возможность решить конкретную задачу и добиться оптимального и быстрого решения

дальнейший прогресс принес улучшение работы с COM и ActiveX, модернизацию инструментов для ускорения написания кода. В последующих версиях и Delphi, и Visual Studio появились такие мощные и сложные технологии, как MIDAS, DCOM и CORBA. Позже были созданы мастер Active Server, предназначенный для создания ASP, компоненты biternetExpress, обеспечивающие поддержку XML, и новые функции MIDAS, предоставившие весьма гибкую платформу размещения данных в среде интернет. С 2002 года ведущее место занимает технология .NET, на которую начинают ориентироваться все, кто имеет отношение к разработке программных продуктов для среды Windows...

ДЖЕЙМС ГОСЛИНГ

«Отец» языка Java. В 1991 году в недрах компании Sun Microsystems стартовал небольшой проект по созданию средств программирования — Green Project. О существовании проекта знали лишь несколько топ-менеджеров Sun. Команда в составе 13-ти человек во главе с Джеймсом Гослингом разместилась в анонимном офисе в городке Менло-Парк, отрезанная от коммуникаций компании. Они трудились в течение полутора лет, нарушая все положения об ограничении рабочего времени. Целью проекта было создание «новой волны» в вычислительной технике. Первоначально планировалось, что волна накроет только систему разработки ПО для бытовой электроники, но в результате появился новый аппаратно-независимый язык программирования Oak (дуб), предназначенный для создания портативной операционной среды Green OS. К середине 90-х годов стало ясно, что развитие информационных технологий в будущем будет протекать под знаменем интернета. В этой связи в середине 1994 года руководство Sun Microsystems приняло решение о переориентации проекта Green на работу с глобальной сетью. Oak был переименован в Java, а сам язык стал трансформироваться в виртуальную вычислительную машину — Java Virtual Machine (JVM). По поводу имени технологии разгорелись жаркие дискуссии. Право окончательного выбора было предоставлено директору по технологиям компании Sun Эрику Шмидту. Этого человека, таким образом, можно назвать крестным отцом Java. Вскоре появился и логотип новой технологии — чашечка дымящегося кофе.

ИСПОЛЬЗОВАНИЕ ВСЕЙ СИЛЫ КЛАСТЕРА ПРИ ПОМОЩИ MPI

КАК ИЗВЕСТНО, ЕСТЬ ТРИ СПОСОБА ПРОВЕСТИ БОЛЬШОЙ ОБЪЕМ ВЫЧИСЛЕНИЙ: НА СУПЕРКОМПЬЮТЕРЕ, НА КЛАСТЕРЕ И НА ОБЫЧНОМ ПК В ТЕЧЕНИЕ НЕСКОЛЬКИХ ЛЕТ :). ПОСЛЕДНИЙ ВАРИАНТ ХОРОШ ТОЛЬКО ОДНИМ — ДЕШЕВИЗНОЙ (ПРАВДА, ЗА ЭЛЕКТРИЧЕСТВО ТОЖЕ НУЖНО ПЛАТИТЬ). СУПЕРКОМПЬЮТЕРЫ, НАОБОРОТ, ТРЕБУЮТ БОЛЬШИХ ЗАТРАТ И ПО СОБСТВЕННОЙ СТОИМОСТИ, И ПО СТОИМОСТИ ОБСЛУЖИВАНИЯ. ВОТ ПОЧЕМУ В НАШЕ ВРЕМЯ ИСПОЛЬЗУЮТСЯ ИМЕННО КЛАСТЕРЫ — КОМПЬЮТЕРЫ, СОЕДИНЕННЫЕ В СЕТЬ, КОТОРАЯ СЛУЖИТ ДЛЯ ПЕРЕДАЧИ ПАРАМЕТРОВ ВЫЧИСЛЕНИЙ |

ALEK SILVERSTONE (ALEKSI@PISEM.NET)

коммунистические вычисления

Этот вариант требует изменения алгоритма программы — его распараллеливания на несколько потоков, каждый из которых может выполняться на любой машине. Долгое время программисты писали свои реализации взаимодействия потоков приложения, пока в 1994 году не появился стандарт, получивший название Message-Passing Interface. Он создавался коллективно (**www.mpi-forum.org**), в результате получился гибкий и удобный инструмент. MPI особенно удобен тем, что он и мал, и велик. Всего в стандарте описано 125 функций, но минимальный набор составляют всего шесть, остальные нужны для эффективности или удобства.

мы будем устанавливать реализацию MPICH (MPI CHameleon), написанную на Windows авторами стандарта. Сразу скажу, что поддержка 9x/ME минимальна: возможен запуск только нескольких потоков на одном компьютере, так что лучше ставить на NT/2k/XP. Для начала с нашего диска или из Сети (**www-unix.mcs.anl.gov/mpi/mpich/mpich-nt**) нужно утянуть дистрибутив: mpich.nt.1.2.5 — бинарники и SDK, mpich.nt.1.2.5.src — исходники, которые можно собрать на VC++. Советую качать первый zip (ссылки находятся внизу страницы загрузки), а не самораспаковывающиеся архивы — пригодится для установки на несколько машин.

Распакуем во временную папку и запустим setup. Для установки желательно иметь права администратора. Если их не будет, то ошибки в процессе установки не возникнет, но менеджер процессов mpd не будет установлен как служба и в этом случае мы получим только бесплатный геморрой — «запуск программы из командной строки с кучей параметров» :). С исходниками идет очень хороший мануал. Очень настоятельно рекомендую почитать его. Мануал (отдельно) можно скачать по адресу **www.mcs.anl.gov/-mpi/mpich/docs/mpichntman.pdf**.

суперкомпьютеры

НЕПЛОХО БЫЛО БЫ СРАВНИТЬ КЛАСТЕРНЫЕ РЕШЕНИЯ С СОВРЕМЕННЫМИ СУПЕРКОМПЬЮТЕРАМИ. ПОСМОТРИМ НА ПЕРВОЕ МЕСТО В TOP500 ЗА НОЯБРЬ 2005 ГОДА,

АРХИТЕКТУРА КОМПЬЮТЕРА BLUE GENE/L ПРЕДСТАВЛЕНА НА РИСУНКЕ. ОСНОВНОЙ ЭЛЕМЕНТ — ВЫЧИСЛИТЕЛЬНАЯ КАРТА, СОСТОЯЩАЯ ИЗ ДВУХ ПРОЦЕССОРОВ И ГИГАБАЙТА ОПЕРАТИВНОЙ ПАМЯТИ. В БОЛЬШИНСТВЕ СЛУЧАЕВ ОДИН ПРОЦЕССОР ЗАНИМАЕТСЯ ВЫЧИСЛЕНИЯМИ, ВТОРОЙ —ТЕЛЕКОММУНИКАЦИЯМИ. ЭТО РАСПРЕДЕЛЕНИЕ МОЖЕТ МЕНЯТЬСЯ — ОБА МОГУТ И СЧИТАТЬ, И ПЕРЕСЫЛАТЬ ДАННЫЕ. 16 ТАКИХ КАРТ ОБРАЗУЮТ УЗЕЛ. 32 УЗЛА ОБРАЗУЮТ СТОЙКУ, ИЗ КОТОРЫХ УЖЕ И СОБИРАЕТСЯ САМ СУПЕРКОМПЬЮТЕР. УЗЛЫ СВЯЗАНЫ МЕЖДУ СОБОЙ НЕСКОЛЬКИМИ СЕТЯМИ: ОДНА ИСПОЛЬЗУЕТСЯ ДЛЯ ОБМЕНА ДАННЫМИ НЕПОСРЕДСТВЕННО МЕЖДУ УЗЛАМИ, ВТОРАЯ — ДЛЯ ГРУППОВЫХ ОПЕРАЦИЙ, ТРЕТЬЯ — ДЛЯ УПРАВЛЕНИЯ, ЧЕТВЕРТАЯ — ДЛЯ ПОДКЛЮЧЕНИЯ К ФАЙЛОВЫМ СЕРВЕРАМ И ХОСТ-КОМПЬЮТЕРУ, ИСПОЛЬЗУЕМОМУ ДЛЯ ДИАГНОСТИКИ. СУПЕРКОМПЬЮТЕР, СОБРАННЫЙ ИЗ 64-Х СТОЕК, ЗАНИМАЕТ ПЛОЩАДЬ, РАВНУЮ ПОЛОВИНЕ ТЕННИСНОГО КОРТА, И ПОТРЕБЛЯЕТ 1,6 МЕГАВАТТА.

Blue Gene/L System Buildup for LOAF

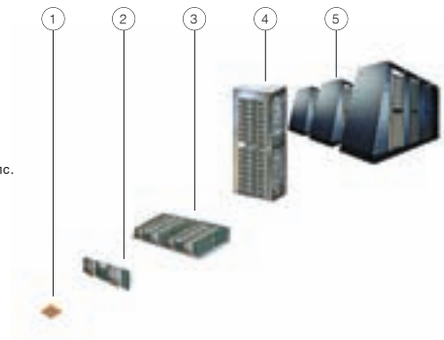
1	2	3	4	5
Chip 2 processors 2.8/5.6 GF/s 4 MB	**Compute Card** 2 chips, 1x2x1 5.6/11.2 GF/s 1.0 GB	**Node Card** (32 chips 4x4x2) 16 compute, 0-2 IO cards 90/180 GF/s 16 GB	**Rack** 32 Node Cards 2.8/5.6 TF/s 512 GB 1024 chips 2048CPUs	**System** 6 Racks, 6x32x32 16.8/33.6 TF/s 3 TB 6144 chips 12288 CPUs

Название: BlueGene/L
Модель: eServer Blue Gene Solution
Поставщик: IBM
Год установки: 2005
Оперативной памяти: 32768 Гб

Процессоры: двуядерный PowerPC 440 700 MHz, по 5.6 Гфлопс каждый, 65536 штук

Теоретическая пиковая производительность: 367000 Гфлопс.
ОС: CNK/Linux

Установлен в Lawrence Livermore National Laboratory (США)



НА ДИСКЕ ЛЕЖАТ БИНАРНИКИ И ИСХОДНИКИ MPICH, МОДУЛЬ ДЛЯ C++ BUILDER'A, FPC И DELPHI, ИСХОДНИКИ SYSTEST И RARCRACK, ТАКЖЕ ДОКУМЕНТАЦИЯ

Установка на несколько машин тоже несложна. Распаковать во временную папку и расшарить ее. Открыть в блокноте файл setup.iss. Найти строку 'szDir=C:\Program Files\MPICH' и указать в ней путь установки. Далее на каждой машине выполнить команду '\\myhost\myshare\setup -s -f1\\myhost\myshare\setup.iss', где myhost — имя первой машины, а myshare — название расшаренной папки. Между прочим, UNC в cmd не поддерживается, так что вставляем команду в «Пуск»→ «Выполнить». Плюс ко всему, для запуска программ нам понадобится общее имя пользователя и пароль на всех машинах. Лучше создать отдельного пользователя, выполнив на всех машинах команду вида 'net user <username> <password> /add'.

После установки удаляем расшаренную папку и жмем «Пуск»→ «Программы»→ MPICH-mpd. Запускаем Configuration tool. Слева при помощи кнопки Add добавляем в список те машины, на которые мы поставили MPICH. Можно еще нажать Select и просканировать компьютеры на наличие установленного mpd или проверить его доступность. (Кстати, в нашей локалке сканированием я нашел еще два компьютера с установленным MPICH, помимо тех, с хозяевами которых я договорился :). Правда, я посканировал сколько-то времени, и тут софтина админов меня забанила.)

Затем ставим галочку возле Show configuration (справа вверху) и по очереди нажимаем на имена компьютеров в списке. Справа под именем компьютера должно появиться 'mpich 1.2.5 May 1 2003'. Если там написано что-то включающее слово error, значит, отсутствует связь со службой mpd на удаленном компьютере. Скорее всего, проблема в файрволе, к примеру, стандартный window'ый не пускает mpd в сеть. Придется его прибить :) и поставить что-то нормальное или сконфигурировать существующее.

Далее в середине окна нужно выбрать настройки — необходимый минимум включает только галочку возле hosts. Для отладки бывает удобно использовать Job Host — средство, позволяющее следить за запущенными задачами и завершать их. Поставь галочку около use job host, рядом нажми yes и укажи имя хоста, Job Host которого будешь использовать. Теперь нажимаем Apply и OK.

рассмотрим C++ Builder

ТУТ СИТУАЦИЯ ГОРАЗДО ХУЖЕ. МНЕ ТАК И НЕ УДАЛОСЬ ПОДКЛЮЧИТЬ MPICH К НЕМУ НОРМАЛЬНО: РАБОТАЕТ, НО ТОЛЬКО НА ОДНОМ КОМПЬЮТЕРЕ, С КОТОРОГО ВСЕ И ЗАПУСКАЕТСЯ :(. В МЕНЮ BUILDER'A ВЫБИРАЕМ FILE→ NEW→ OTHER→ CONSOLE WIZARD. В СВОЙСТВАХ ПРОЕКТА УКАЗЫВАЕМ INCLUDE PATH. С БИБЛИОТЕКАМИ НЕСКОЛЬКО СЛОЖНЕЕ. ОНИ ИМЕЮТ ДРУГОЙ ФОРМАТ, ПОЭТОМУ СНАЧАЛА НУЖНО КОНВЕРТИРОВАТЬ ИХ. ВМЕСТЕ С BUILDER'ОМ ИДЕТ УТИЛИТА COFF2OMF.EXE, С ЕЕ ПОМОЩЬЮ ДОЛЖНЫ БЫТЬ ОБРАБОТАНЫ ФАЙЛЫ MPICH.LIB И MPICHD.LIB: COFF2OMF <СТАРОЕ ИМЯ> <НОВОЕ ИМЯ>. НА НАШЕМ ДИСКЕ ЛЕЖАТ КОНВЕРТИРОВАННЫЕ ВЕРСИИ. ДОБАВЛЯЕМ В СВОЙСТВА ПРОЕКТА ПУТЬ К ЭТИМ БИБЛИОТЕКАМ, ОДНУ ИЗ НИХ ДОБАВЛЯЕМ В ПРОЕКТ: PROJECT→ ADD TO PROJECT. БИЛДИМ И ЗАПУСКАЕМ КАК ОБЫЧНО. ЕСЛИ ТЫ НЕ ЗНАЕШЬ, КАК ПОДКЛЮЧИТЬ ЭТИ БИБЛИОТЕКИ К BUILDER'У ТАК, ЧТОБЫ ОНИ РАБОТАЛИ, ПИШИ МНЕ. АДРЕС УКАЗАН В НАЧАЛЕ СТАТЬИ.

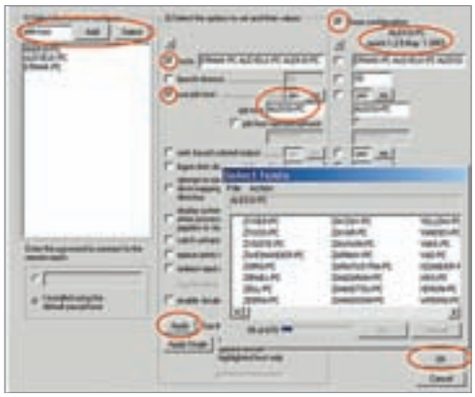
ТЕПЕРЬ ПОДКЛЮЧИМ MPICH К МОЕМУ ЛЮБИМОМУ DELPHI. КАК НИ ШАМАНЬ, LIB-ФАЙЛЫ К НЕМУ НЕ ПОДКЛЮЧИШЬ, ЧТО И НЕ ТРЕБУЕТСЯ. В ПОСТАВКУ MPICH ВХОДЯТ 3 DLL, КОТОРЫЕ СТАВЯТСЯ В ПАПКУ %WINDIR%\SYSTEM32 И КОТОРЫЕ КАК РАЗ РЕАЛИЗУЮТ ЕГО ФУНКЦИОНАЛЬНОСТЬ. СООТВЕТСТВЕННО, НАМ НУЖНО ТОЛЬКО НАПИСАТЬ ЗАГОЛОВОЧНЫЙ МОДУЛЬ ДЛЯ ЭТИХ БИБЛИОТЕК. КОГДА-ТО Я САМ ПЕРЕВОДИЛ ХЭДЕРЫ НА С ИЗ СТАНДАРТНОЙ ПОСТАВКИ, НО ПОЗЖЕ НАШЕЛ МОДУЛЬ ДЛЯ FREE PASCAL COMPILER'A И СТАЛ ПОЛЬЗОВАТЬСЯ ИМ, НЕМНОГО ПОДПРАВИВ. ХВАТАЙ ЭТОТ МОДУЛЬ НА НАШЕМ ДИСКЕ, КОПИРУЙ В ПАПКУ С ПРОЕКТОМ И ПИШИ В ХОДЕ USES MPI. ПРОЩЕ ПРОСТОГО!

Настало самое время проверить работу. Для начала нужно взять готовую программу systest с нашего диска. Скопировать файл в папку и расшарить. Запустить Job manager, нажать Connect, ввести имя Job Host'a и OK. Затем запустить MPIrun. Вверху указать сетевой путь к нашей программе, чуть ниже — количество потоков (для этой программы — минимум два), а справа — компьютеры, на которых будут запускать потоки. Нажать Run. В появившееся окно ввести имя добавленного пользователя и пароль, поставить галочку Remember this user/password. Если появилось окно с ошибкой, нажать OK, затем Advanced Options и поставить галку Always prompt for password, чтобы ввести логин/пароль еще раз. После этого запускается наша программа. Сначала она выводит список, который демонстрирует, на каком компьютере и какой именно поток запущен. Далее ввод единички запускает тест проверки целостности сети, а ввод двойки тестирует пропускную способность.

теперь необходимо подключить MPICH к среде программирования. Сначала проделаем это для Visual C++. Запускаем, выбираем File→ New→ Win32 console application, вводим имя проекта и OK. Если выскочил мастер, то выбираем empty project. Затем Tools→ Options→ Directories, добавляем в список include папку MPICH\SDK\include, а в список library — MPICH\SDK\lib. Далее топаем в свойства и добавляем в список параметров компилятора /MTd для отладочной конфигурации, также /MT для релиза. На вкладке Link добавляем mpich.lib для релиза, mpichd.lib — для отладки и ws2_32.lib — для обеих конфигураций. Осталось только написать в исходнике #include <mpi.h>.

приступим, наконец, к самому MPI. Инициализацию обмена сообщениями выполняет функция MPI_Init. Кроме того, она решает еще одну важную задачу. После осуществления вызова программист может быть уверенным, что все потоки запущены и что все они выполнили этот оператор. В качестве параметров указываются количество аргументов командной строки и сами аргументы, передающиеся во все потоки. Завершение обмена сообщениями выполняет функция MPI_Finalize без параметров.

Каждый поток в MPI имеет собственный номер, называемый рангом потока. Это число используется в большинстве функций передачи сообщений. Для получения своего ранга процесс может использовать функцию MPI_Comm_rank с двумя параметрами. Первый — коммуникатор, которому принадлежит данный процесс. В терминологии MPI коммуникатором называется группа потоков. Все потоки по умолчанию принадлежат коммуникатору MPI_COMM_WORLD, и нумерация идет с нуля. Программист может создавать собственные коммуникаторы и вводить в них собственную нумерацию. Второй параметр MPI_Comm_rank — это адрес, по которому будет записан ранг процесса. Для получе-



Конфигурация MPICH

ния общего количества процессов в коммуникаторе используется функция MPI_Comm_size с такими же параметрами. Один из вариантов структуры MPI-программы смотри на врезке. Тут мы предполагаем, что ветвь 0 главная, а остальные управляются ею. Однако можно сделать и совсем по-другому, предоставляется полная свобода. Главное — не запутаться во множестве ветвей :).

скелет MPI-программы

```
var p:pointer;
  rank,size:integer;
begin
  // не передаем параметры
  p:=nil;
  rank:=0;
  MPI_Init(@rank,p);

  MPI_Comm_rank(MPI_COMM_WORLD,@rank);
  MPI_Comm_size(MPI_COMM_WORLD,@size);

  if rank=0 then begin
    // главная ветвь
  end else begin
    // остальные ветви
  end;
  MPI_Finalize();
end.
```

В MPI существует огромное множество функций для приема и отправки сообщений. Одни пересылают сообщение «один-одному», другие — «всекаждому» и т.д. Кроме того, существуют блокирующие и буферизированные (не блокирующие) варианты всех функций.

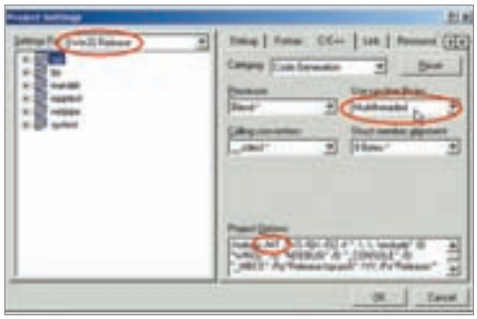
Для начала рассмотрим две самые простые функции MPI_Send и MPI_Recv, выполняющие передачу по модели «один-одному» с автоматическим выбором типа блокировки. Так зачем нужны остальные, если, к примеру, схему передачи «один-всем» можно реализовать циклическим вызовом MPI_Send? Да, можно, но этот путь неэффективен: такие коллективные функции в MPI передают данные используя реальную архитектуру кластера, то есть используют широковещательные адреса, разделяемую память и т.д. Перечислю параметры MPI_Send по порядку: адрес буфера, в котором хранятся данные для передачи; количество данных (не размер буфера!); тип данных; ранг получателя сообщения; идентификатор сообщения; коммуникатор. Идентификатор сообщения назначается программистом и служит для удобства, так как получающий поток может фильтровать сообщения по этому полю. Тип данных нужен для их корректного преобразования, так как теоретически MPI может связывать потоки на разных платформах, имеющих разные внутренние представления данных. По этой же причине все функции приема и передачи оперируют не количеством передаваемых байт, а количеством ячеек нужного типа. MPI_Recv имеет еще один параметр, в нем есть тип MPI_Status — структура, в которую помещаются свойства полученного сообщения. Кроме того, параметры «Идентификатор сообщения» и «Ранг потока» могут иметь значения MPI_ANY_TAG и MPI_ANY_SOURCE соответственно. Как нетрудно догадаться, в этих случаях функция получает пакет с любым идентификатором и от любого потока. Рекомендую делать именно так, а потом фильтровать сообщения по параметру статуса, если, конечно, не требуется четкий порядок получения сообщений.

В принципе, любую программу можно написать используя только описанные функции. Однако мы стремимся к удобству, поэтому рассмотрим еще несколько. Первая — это MPI_Barrier (в качестве параметра передается коммуникатор). Функция останавливает выполнение всех потоков в

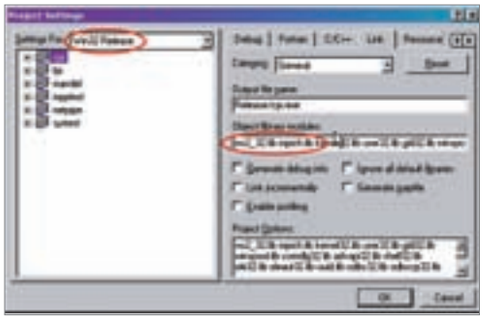
передача разнородных данных

```
// переменные
type TOurStruct=record
 i:integer;
 d:double;
 st:array[0..9] of char;
 end;
var
 p:pointer;
 OurStruct:TOurStruct;
 buf_pos,buf_size:integer;
 Stat:MPI_Status;

// первая ветвь
 OurStruct.i:=777;
 OurStruct.d:=5.0;
 OurStruct.st:='0123456789';
 buf_size:=0;
 MPI_Pack_size(1,MPI_INT,MPI_COMM_WORLD,@buf_size);
 MPI_Pack_size(1,MPI_DOUBLE,MPI_COMM_WORLD,@buf_size);
 MPI_Pack_size(10,MPI_CHAR,MPI_COMM_WORLD,@buf_size);
 GetMem(p,buf_size);
 pos:=0;
 MPI_Pack(@buf.i,1,MPI_INT,p,buf_size,@pos,MPI_COMM_WORLD);
 MPI_Pack(@buf.d,1,MPI_DOUBLE,p,buf_size,@pos,MPI_COMM_WORLD);
 MPI_Pack(@buf.st,10,MPI_CHAR,p,buf_size,@pos,MPI_COMM_WORLD);
 MPI_Send(p,buf_size,MPI_BYTE,1,111,MPI_COMM_WORLD);
 FreeMem(p);

// вторая ветвь
 MPI_Probe(0,111,MPI_COMM_WORLD,@Stat);
 MPI_Get_count(@Stat,MPI_BYTE,@buf_size);
 GetMem(p,buf_size);
 MPI_Recv(p,buf_size,MPI_BYTE,0,111,MPI_COMM_WORLD,@Stat);
 pos:=0;
 MPI_Unpack(p,buf_size,@pos,@OurStruct.i,1,MPI_INT,MPI_COMM_WORLD);
 MPI_Unpack(p,buf_size,@pos,@OurStruct.d,1,MPI_DOUBLE,MPI_COMM_WORLD);
 MPI_Unpack(p,buf_size,@pos,@OurStruct.st,10,MPI_CHAR,MPI_COMM_WORLD);
 FreeMem(p);
```

www.parallel.ru — информационно-аналитический центр по параллельным вычислениям.
www.top500.org — TOP500 Supercomputer Sites.
www.beowulf.org — collection of resources for the expanding universe of users and designers of Beowulf class cluster computers.
www.polygon.parallel.ru — вычислительный полигон. Web-интерфейс для запуска модельных программ на кластерах.
www.moneybee.net — использование параллельных вычислений и нейросетей для прогнозирования биржевых котировок и индексов.



Параметры компилятора в VC++



Подключаем библиотеки к VC++

этом коммуникаторе до тех пор, пока ВСЕ они не подойдут к барьеру. Очень удобно для синхронизации работы потоков.

Вторая функция — MPI_Abort. Прекращает выполнение всех потоков в коммуникаторе. Параметры — коммуникатор и код ошибки.

Следующая функция будет посложнее :). MPI_Bcast имеет параметры: адрес буфера; количество ячеек памяти; тип данных; корневой процесс; коммуникатор. Часто слышу от кого-нибудь, что MPI_Bcast рассылает содержимое буфера всем потокам. На самом деле эта функция синхронизирует содержимое буфера, делая его равным содержимому корневого процесса. Разница в том, что сообщение Bcast'а в другом процессе ловится не Recv'ом, а Bcact'ом!

До этого момента я говорил только о передаче сообщений, содержащих лишь один тип. А как передать разнотипную структуру? Если все используемые компьютеры имеют одинаковую архитектуру, то можно писать что-то наподобие MPI_Send(&buf, sizeof(s), MPI_BYTE, ...). Если архитектуры компьютеров различаются, то предварительно данные должны быть упакованы — вместе с самими данными в буфер записывается информация об их типе. Это делается функцией MPI_Pack:

```
MPI_Pack(add_data_p, add_count, add_type,
buf_p, buf_size, buf_pos_p, comm);
```

add_data_p — указатель на данные, которые нужно упаковать.
add_count — количество ячеек памяти.
add_type — MPI-тип этих ячеек.
buf_p — указатель на буфер, куда упаковываются данные.
buf_size — размер буфера.
buf_pos_p — указатель на текущую позицию в buf. Не забудь в начале записать туда 0 ! После выполнения функции значение изменяется.
comm — коммуникатор.

Размер буфера для упаковки можно узнать с помощью функции MPI_Pack_size. В качестве параметров этой функции передаются число ячеек памяти, их тип, коммуникатор и адрес, по которому записывается размер необходимого буфера. Сначала значение по этому адресу нужно сделать равным нулю, затем — последовательно вызывать MPI_Pack_size (значения суммируются автоматически).

На принимающей стороне последовательность действий такая: «подсмотреть» размер пакета, подсчитать размер буфера для распаковки, выделить память и распаковать туда пакет. Первое выполняет функция MPI_Probe. Параметры — ранг отправителя, идентификатор сообщения, коммуникатор и указатель на структуру MPI_Status, в которую будут записаны параметры сообщения. Стандарт MPI гарантирует, что вызов MPI_Recv, следующий за MPI_Probe и имеющий те же значения ранга, идентификатора и коммуникатора, получит именно то сообщение, параметры которого были «подсмотрены» первой функцией.

Размер буфера можно подсчитать функцией MPI_Get_count — ей передаются указатель на наш MPI_Status, тип данных (в этом случае MPI_BYTE) и на указатель, по которому будет записан необходимый размер буфера. Распаковка вызывается функцией MPI_Unpack, которой передаются те же параметры, что и Pack'у, только в другом порядке.

На врезке можешь посмотреть код, демонстрирующий использование этих функций.

рассмотрим простейшую программу на C++ — она показана на врезке. Сначала мы инициализируем библиотеку MPICH и получаем ранг и общее количество процессов в глобальном коммуникаторе. Затем выводим приветственное

программка на C++

```
#include <iostream.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
 int rank, size;
 MPI_Init(&argc, &argv);

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
 MPI_Comm_size(MPI_COMM_WORLD, &size);

 cout<<"Hello from process #"<<rank<<endl;
 cout.flush();
 MPI_Barrier(MPI_COMM_WORLD);
 if(rank==0)
 {
   for(int i=1; i<size; i++)
   {
     cout<<"Sending message to process #"<<i<<endl;
     cout.flush();
     MPI_Send(&rank, 1, MPI_INT, i, 111, MPI_COMM_WORLD);
   }

   for(i=1; i<size; i++)
   {
     MPI_Status status;
     int temp;
     MPI_Recv(&temp, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
     cout<<"Received message from process
#"<<status.MPI_SOURCE<<endl;
     cout.flush();
   }
 }
 else
 {
   int temp;
   MPI_Status status;
   MPI_Recv(&temp, 1, MPI_INT, 0, 111, MPI_COMM_WORLD,
&status);
   MPI_Send(&rank, 1, MPI_INT, 0, 222, MPI_COMM_WORLD);
 }

 MPI_Finalize();
 return 0;
}
```

сообщение и останавливаемся на барьере до тех пор, пока все потоки не дойдут до этого места. Функция flush вызывается для принудительного сброса содержимого экранного буфера на консоль. Если это действие не осуществится, то, может случиться, один процесс начнет выводить сообщение в середине вывода другого.

Далее выполняется одна из двух веток (выбор определенной ветки зависит от ранга процесса). Первая ветка выполняется нулевым процессом. Сначала она рассылает всем значение своего ранга (то есть ноль), затем принимает любые (MPI_ANY_TAG) сообщения от любых (MPI_ANY_SOURCE) процессов и рапортует о получении. Вторая ветка работает на всех остальных процессах. Сначала она получает сообщение от процесса 0 с идентификатором 111, затем отсылает этому процессу свой ранг, притом идентификатор сообщения равен 222.

В MPICH'е есть много примеров на C++, так что посмотрим что-нибудь на Delphi. Чтобы сделать полноценный пример, я взял свою старую программу для взлома RAR-архива (см.][#02/2005) и переписал ее под MPI.

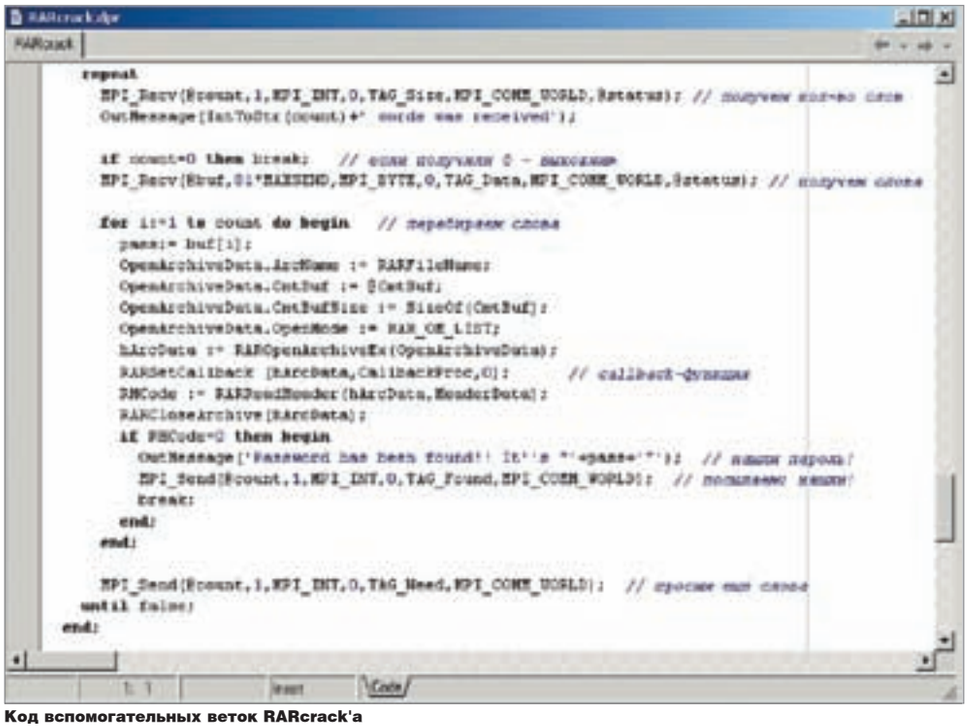
Сразу расскажу, как запускать эту программу. Копируем полученный после компиляции exe'шник, passlist.txt и unrar.dll в папку, расшариваем ее. Допустим, сетевой путь этой папки \\Notebook\Run. Запускаем MPIrun, заходим в настройки и в строке Drive mappings вводим m:\\notebook\run, где m — буква диска, не существующего в ОС.

Затем в строке ввода пути к приложению пишем m:\RARcrack.exe brute_me.rar.

кусок RARcrack

```
for i:= 1 to size-1 do begin
 count:=FillSendBuf;
 MPI_Send(@count,1,MPI_INT,i,TAG_Size,MPI_COMM_WORLD);
 MPI_Send(@buf,sizeof(buf),MPI_BYTE,i,TAG_Data,MPI_COMM_WORLD);
end;

while true do begin
 count:=FillSendBuf;
 if count=0 then break;
 MPI_Recv(@count,1,MPI_INT,MPI_ANY_SOURCE,MPI_ANY_TAG,MPI_COMM_WORLD,@status);
 if(status.MPI_TAG=TAG_Found) then break;
 MPI_Send(@count,1,MPI_INT,status.MPI_SOURCE,TAG_Size,MPI_COMM_WORLD);
 MPI_Send(@buf,sizeof(buf),MPI_BYTE,status.MPI_SOURCE,TAG_Data,MPI_COMM_WORLD);
 OutMessage('"'+buf[1]+'" was sent to process #'+IntToStr(status.MPI_SOURCE));
end;

for i:=1 to size-1 do begin
 OutMessage('Waiting for process #'+IntToStr(i));
 MPI_Recv(@count,1,MPI_INT,i,MPI_ANY_TAG,MPI_COMM_WORLD,@status);
 count:=0;
 MPI_Send(@count,1,MPI_INT,status.MPI_SOURCE,TAG_Size,MPI_COMM_WORLD);
end;
```

Самый сложный кусок кода главной ветки ты видишь на врезке. В первом цикле нулевая ветвь рассылает остальным первое задание — слова для перебора. Сначала идет размер буфера, потом посылается и сам буфер. Во втором цикле сначала наполняется новый буфер слов. Если слова закончились, то программа выходит из этого цикла. Затем идет получение пакета от другой ветви и сравнение его идентификатора с идентификатором, зарезервированным за пакетом, рапортующим о нахождении пароля. Если определено, что найден как раз нужный пакет, опять же, выходим из цикла. В противном случае отсылаем следующую порцию паролей. Третий цикл ждет любые пакеты от всех ветвей по очереди и посылает им ноль слов: либо файл закончился, либо пароль уже найден. Остальной код с подробными комментариями лежит на диске к журналу.



Код вспомогательных веток RARcrack'а

Вот и конец статьи. Конечно же, очень многое осталось за бортом. Например, еще в 1997 году появился стандарт MPI-2, и сравнительно недавно была выпущена его первая реализация. Изменения довольно значительные: динамическое создание и удаление потоков, нормальный обмен сообщениями через разделяемую память, архитектурно-независимый доступ к файлам, клиент-серверные возможности. Все это позволяет писать не только расчетные задачи, но и системы массового обслуживания — базы данных и т.д. Почему же мы рассматривали MPI-1? Главное объяснение в том, что первый стандарт состоит из 300 страниц, в MPI-2 добавлено 500, причем в это число входят только исправления и дополнения к первому. Изложить все на нескольких журнальных полосах невозможно.

Хочу сказать «большое спасибо» Debugger'у за ноутбук, локальную сеть и кофе, предоставленные на заключительном этапе написания статьи

СОБАМИ ПОПОЛНЯЕТСЯ АНТИВИРУСНАЯ БАЗА? КРОМЕ БАНАЛЬНОГО «СООБЩИЛИ, ПРОВЕРИЛИ, ДОБАВИЛИ».

АЛЕКСАНДР ГОСТЕВ: Если бы мы сидели и ждали, пока нам сообщат о появлении нового вируса, мы бы уже давно утратили свои позиции мирового лидера по скорости реагирования и детектирования. Да и не заняли бы эту позицию, вероятно, вообще никогда. Антивирусные компании и так постоянно находятся в роли догоняющих в системе «снаряд-броня», так что вопрос о том, как максимально сократить время реакции, встал перед нами давно. И, судя по тому, что мы лидеры по этому показателю в антивирусной индустрии, нам действительно удалось решить этот вопрос.

Тут нет ничего оригинального. «Если гора не идет к Магомету — Магомет идет к горе». Мы используем разнообразные автоматические способы активного поиска новых вредоносных программ в Сети: и системы мониторинга сайтов, и системы раннего обнаружения вирусов в почтовом трафике, и сети honeypot'ов. Очень помогает постоянный и тесный контакт с дружественными антивирусными компаниями как в деле обмена сэмплами, так и в совместном анализе или локализации инцидентов. В этой области у нас нет конкурентной борьбы, за деньги клиентов мы боремся другими, маркетинговыми способами. Есть у нас и так называемые «агенты», они же — добровольные помощники.

Предвосхищая возможный вопрос, скажу: нет, мы не покупаем вирусы у их авторов, хотя изредка такие предложения поступают.

СПЕЦ: ПРИНЦИПИАЛЬНО ЛИ ТО, НА ЧЕМ НАПИСАНЫ ВИРУСЫ И ТРОЯНЫ?

АЛЕКСАНДР ГОСТЕВ: Да нет, никакой особой разницы нет. Некоторые вещи бывает довольно трудно анализировать проводя реверс-инженеринг кода, но в 99% случаев для вынесения вердикта «вирус/не вирус» подробный анализ и не требуется. А если требуется, то мы в состоянии потратить на это чуть больше времени, чем обычно. Дело обстоит гораздо интереснее, когда нам попадаются вирусы для новых платформ или сред, например для Symbian или Windows Mobile. Там другой процессор, другие ассемблерные команды, другие форматы файлов. Любому вирусному аналитику становится интересно, потому что это нечто новое. Приходится очень быстро и достаточно глубоко внедряться в тему. Вот буквально сегодня разбирал троянец для J2ME (Java для мобильных телефонов), узнал много интересного :).

СПЕЦ: ВИЗУАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ СТАНОВЯТСЯ ДОСТУПНЕЕ, УЖЕ НЕ ТРЕБУЕТСЯ ПИСАТЬ МНОГОЕ С НУЛЯ. ДОШЛО ДО ТОГО, ЧТО СУЩЕСТВУЮТ «ПОЛУФАБРИКАТЫ» ВИРУСОВ И ТРОЯНОВ, КОТОРЫМИ МОЖЕТ ВОСПОЛЬЗОВАТЬСЯ ЛЮБОЙ НАЧИНАЮЩИЙ ПРОГРАММИСТ. НЕ ОПАСНА ЛИ ПОДОБНАЯ ТЕНДЕНЦИЯ?

АЛЕКСАНДР ГОСТЕВ: Для нас — нет. Даже наоборот. Когда есть какой-то конструктор/генератор вирусов-троянцев, то число всех комбинаций возможных творений весьма ограничено. В основе все равно будут лежать одни и те же блоки кода (модуль размножения, модуль кражи данных, модуль отсылки данных). Это кирпичики, из которых кто угодно пытается собрать что-то эксклюзивное, а на деле получается, что все подобные поделки имеют только внешние или незначительные отличия вроде имени файлов, адресов электронной почты и текстов MessageBox. Как следствие, нам для подобных вещей крайне просто создать эвристические анализаторы, которые помогут детектировать все варианты сразу. Поэтому довольно смешно выглядят люди, которые покупают генератор Pinch'ей (популярный троянец-шпион) и надеются, что смогут с его помощью создать уникальный недетектируемый троян.

СПЕЦ: МОЖЕТЕ ЛИ ДАТЬ ЭКСПЕРТНУЮ ОЦЕНКУ ТОГО, ЧТО БУДЕТ С ВИРУСАМИ ЧЕРЕЗ ПЯТЬ-ДЕСЯТЬ ИЛИ 20 ЛЕТ? ВОЗМОЖНЫ ЛИ ПРОРЫВЫ В УМАХ ВИРУСОПИСАТЕЛЕЙ И ГЛОБАЛЬНЫЕ ЭПИДЕМИИ? ИЛИ НЕ БУДЕТ ПРИДУМАНО НИЧЕГО НОВОГО?

АЛЕКСАНДР ГОСТЕВ: Сложно сделать такой прогноз. Если посмотреть, что происходило 20 или десять лет назад, выяснится, что никто не мог предполагать такого многообразия современных типов и классов вирусов. Еще десять лет назад не было ни одного почтового червя, а сейчас эти программы уже успели пережить пик своего развития и находятся в стадии постепенного отмирания.

Прорывы в умах вирусописателей случаются регулярно, это да. Проблема в том, что зачастую такие прорывы остаются «невостребованными» среди криминальных вирусописателей. Иногда навсегда, иногда до поры до времени. Возьмем, к примеру, троянские программы для игровых приставок, появившиеся осенью прошлого года. Да, троянцы есть. Да, наносят вред пользователю. Однако на данный момент в их создании и распространении нет явной коммерческой выгоды для вирусописателей. Ну что он украдет с приставки? Игру? Их и так навалом в Сети. Вот когда приставки начнут полноценно соединяться друг с другом, с сервисами интернета, вот тогда, возможно, на них и придется удар, причем неминуемый. Киберпреступность очень быстро реагирует на потенциальную выгоду.

Если же говорить в целом о будущем, то на смену интернету как сети из компьютеров приходит новый мир. Мир мобильных устройств, которые будут соединяться друг с другом в самых разнообразных сочетаниях: смартфоны, телефоны, КПК, приставки, фотоаппараты, плееры, холодильники, кофеварки и все, что еще придумают. Не забывай и о бортовых компьютерах автомобилей, которые тоже будут должны взаимодействовать со всем этим и с внешним миром.

Ситуация изменяется очень быстро. Меньше двух лет прошло с момента появления первого червя для мобильных телефонов. Тогда многие скептически отнеслись к этому факту: ну, работает только на смартфонах с Simbian, распространяется через Bluetooth, соответственно, радиус заражения маленький, смартфонов мало, для запуска надо три раза нажать кнопку подтверждения. А что сейчас? Сейчас червь Cabir зафиксирован почти в сорока странах мира (это только подтвержденные данные). В Москве, если поездить с включенным Bluetooth в метро в течение дня, риск поймать Cabir будет весьма и весьма высок.

Дальше больше. Червь ComWar, рассылающий себя через MMS. Написан в России меньше года назад. Сейчас насчитывается более 20-ти стран, «зараженных» этим червем, причем в некоторых странах его распространение действительно носит эпидемиологический масштаб. Что будет дальше, предугадать нетрудно, тем более если мы учтем дальнейшее развитие смартфонов и растущую долю этих телефонов на рынке.

Bluetooth и MMS-черви — главная угроза будущего и почва для глобальных эпидемий. Во сколько раз число владельцев телефонов превосходит число пользователей компьютеров, ты, наверное, тоже хорошо представляешь себе.

А еще есть риск появления Wi-Fi-червей. Подробно раскрывать «потенциальный» принцип их действия я не хочу, чтобы не стимулировать умы вирусописателей, но... В общем, все только начинается.

АНДРЕЙ НИКИШИН

АЛЕКСАНДР ГОСТЕВ

СПЕЦ: ЗАРАЖЕНИЕ КОМПЬЮТЕРА — УЖЕ ОБЫЧНОЕ ДЕЛО. МАКСИМУМ, ТЕРЯЕТСЯ ИНФОРМАЦИЯ. ВОЗМОЖНО, КОМПЬЮТЕР ВЫХОДИТ ИЗ СТРОЯ, ДАЛЬШЕ ПО ЦЕПОЧКЕ ЗАРАЖАЮТСЯ КОМПЬЮТЕРЫ ДРУЗЕЙ И СОСЕДЕЙ. ОДНАКО, ПОМИМО КОМПЬЮТЕРОВ, ПОЯВЛЯЕТСЯ МНОГО «УМНЫХ» УСТРОЙСТВ, ПОСЛЕДСТВИЯ СБОЯ КОТОРЫХ МОГУТ БЫТЬ НЕОБРАТИМЫМИ. ВОЗМОЖЕН ЛИ В РЕАЛЬНОСТИ СЮЖЕТ, НАПРИМЕР, ТОГО ЖЕ ФИЛЬМА «ТЕРМИНАТОР»? ХОТЯ БЫ ТЕОРЕТИЧЕСКИ...

АЛЕКСАНДР ГОСТЕВ: Война машин и людей, конечно — фантастика. Однако «умные» устройства будут доставлять проблемы, но не сами по себе, а в результате действий людей-злоумышленников. Проблемы могут быть самые разные. Начиная тем, что твоя кофеварка получит «неправильную» SMS'ку и уничтожит весь запас зерен, заканчивая случаем, когда бортовой компьютер автомобиля в ходе DoS-атаки на него решит, что идет попытка угона, заблокирует двери и отправит сообщение в полицейский участок. И неважно, что ты в этот момент, например, находишься в салоне и едешь по трассе...

обзор книг

ЧТО ПОЛИСТАТЬ

КАК МЫ ОТБИРАЕМ КНИГИ В ОБЗОР? БЕРЕМ СПИСОК КНИГ, КОТОРЫЕ ЕСТЬ НА СКЛАДЕ (НЕСКОЛЬКО ТЫСЯЧ НАИМЕНОВАНИЙ). ДЕЛАЕМ ВЫБОРКУ ПО ТЕМЕ НОМЕРА, ПОТОМ ОТБРАСЫВАЕМ УСТАРЕВШИЕ ЭКЗЕМПЛЯРЫ И ДУБЛИ. ЛУЧШЕЕ ПОПАДАЕТ В ЖУРНАЛ | **АНДРЕЙ КАРОЛИК**

Linux: программирование в примерах

М.: КУДИЦ-ОБРАЗ, 2005 / Роббинс А. / 656 страниц

Разумная цена: 211 рублей

Не секрет, что правильный способ научиться программировать — почаще читать хорошо написанные программы. Вот и автор этой книги рассказывает об API системных вызовов Linux на основе реального исходного кода повседневно используемых программ. Ты видишь не только банальный синтаксис Linux API, но и копаешься в реальных проблемах производительности, переносимости и устойчивости, с которыми всегда сталкиваются при написании программного обеспечения. Правда, книга будет понятна только тем, кто разбирается в программировании и знаком с основами C. Ты изучишь базовые API, образующие ядро программирования под Linux: управление памятью, файловый ввод-вывод, метаданные файлов, процессы и сигналы, пользователи и группы, поддержка программирования (сортировка, анализ аргументов и т.д.) и интернационализация. Отдельно рассмотрены средства отладки, доступные под GNU Linux. В качестве иллюстраций — примеры кода из V7 UNIX и GNU.

Если заинтересовался, можешь заказать любую книгу из обзора (по разумным ценам), не отрывая пятой точки от дивана или стула, в букинистическом интернет-магазине OS-книга» (**www.osbook.ru**). Книги для обзора мы берем именно там

Система программирования Java без секретов: Как создать безопасное приложение с «нуля»

М.: ЗАО «Новый издательский дом», 2005 / Фельдман С.К. / 352 страницы
Разумная цена: 180 рублей

Простое и доступное пособие по Java. Для тех, кто еще не знаком близко с этим языком программирования. Приведенные примеры показаны в контексте применения Java для web'а (интернет-приложения), и тут же обнаруживается логичное дополнение (занимает почти полкнижки) — JavaScript, синтаксис которого во многом схож с Java.

Изначально JavaScript был не языком программирования, а языком управления сценариями просмотра гипертекстовых страниц на стороне клиента. Однако именно благодаря этому JavaScript и завоевал популярность, позволив изменять значение атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра web-сайта пользователем без перезагрузки страницы! К примеру, ты видел всплывающие меню или вывод текущей даты на сайте? Все это, скорее всего, сделано именно на JavaScript.

Java без сбоев: обработка исключений, тестирование, отладка

М.: КУДИЦ-ОБРАЗ, 2005 / Стелтинг С. / 464 страницы
Разумная цена: 198 рублей

То, как ты обрабатываешь ошибки, сильно влияет на работоспособность твоего программного кода. Иметь дело со сценариями отказов — все равно что решиться на посещение зубного врача: знаешь, что ты должен сделать это, но неохота :). Однако от твоего решения зависит простота сопровождения, легкость тестирования и отладки кода. Чтобы понять эту мысль, достаточно поработать над чужим проектом :). Конечно, за ошибки в программном коде отвечает не только обработка исключений, но именно она оказывает решающее влияние. В книге показан характер возможных отказов программного кода, дается общее описание наиболее распространенных ошибок, возникающих в программном интерфейсе или приложении. Рассматриваются стратегии использования обработки исключений для технологий J2EE, JDBC, RMI, JMS и др.

C++ Builder: Книга рецептов

М.: КУДИЦ-ОБРАЗ, 2006 / Ермолаев В. / 208 страниц
Разумная цена: 106 рублей

Сборник вопросов и ответов, построенный на основе дискуссий с различных форумов и конференций, в том числе с известного сайта **www.bcbdev.ru**, посвященного C++Builder. На каждый вопрос (точнее, на каждую проблему) дается развернутый ответ с исходным кодом. Притом поясняют, почему данная проблема решается именно «так», а не иначе. Плюс есть комментарии по каждому этапу решения данной проблемы. Основная масса вопросов касается создания пользовательского интерфейса, работы с файлами, реестром и внутренними классами VCL. На приложенном компакт-диске есть коды всех проектов, так что набирать ничего не придется. В то же время, сам понимаешь, материал немного субъективный, так как вопросы отобраны самими авторами по логике, известной одним им. Правда, в аннотации авторы указали ящик, на который читатели могут послать вопросы. Попробуй. Может, ответят :).

Программирование на C++ глазами хакера

СПб.: БХВ-Петербург, 2005 / Фленов М.Е. / 336 страниц
Разумная цена: 155 рублей

В книге ты найдешь множество нестандартных приемов программирования, примеры использования недокументированных функций и возможностей языка C++. Узнаешь, как оптимизировать размер и скорость выполнения программ. Несмотря на то, что компьютеры сейчас достаточно скоростные и размеры дисков выражаются словом «до фига», эта тема все-таки актуальна.

БОльшая часть книги посвящена программированию в сетях Интернет/Интранет. Есть работающие примеры быстрого сканера портов и троянского коня. Победить хакера, как известно, можно только в том случае, если известны все его слабые и сильные стороны. Понимая действия противника, можно создать максимально эффективную систему обороны. Весь материал написан легко и просто, но для полноты осознания понадобятся хотя бы начальные знания языка C++.

Delphi в шутку и всерьез: что умеют хакеры

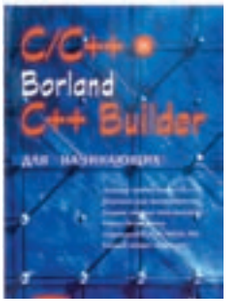
СПб.: Питер, 2006 / Фленов М.Е. / 271 страница
Разумная цена: 162 рубля

Рассмотрены разные приемы и примеры программирования на языке Delphi. Тебе предлагают улучшить понимание процессов, происходящих в операционной системе во время выполнения программы, и, как результат, повысить эффективность кода, который пишешь. Книгу можно условно разделить на четыре части: корректное написание кода (правильное оформление кода, которое в дальнейшем может сэкономить много времени и сил на этапах тестирования и поддержки), оптимизация (создание «быстрого» кода или оптимизация существующего), шуточные программы (интересные алгоритмы, с помощью которых проще впитывать новое: «злое» окно, шутки над буфером обмена, кавардак на рабочем столе, шутки над мышью, блокировка окон и многое другое) и программирование сетевых приложений. Автор книги в свое время писал статьи в «Хакер», а теперь вот перешел на книги :), сохранив притом ценный подход — «писать просто о сложном».

C/C++ и Borland C++ Builder для начинающих

СПб.: БХВ-Петербург, 2006 / Пахомов Б.И. / 640 страниц
Разумная цена: 215 рублей

Руководство для начинающих по разработке приложений в среде Borland C++ Builder. Основные элементы языков программирования C/C++ и примеры создания простейших классов и программ. Принципы визуального проектирования и событийного программирования. На конкретных примерах показаны основные возможности визуальной среды разработки C++ Builder, назначение базовых компонентов и процесс разработки различных windows- и интернет-приложений. Книжкой заинтересуются и те, кто даже в глаза не видел Borland C++ Builder, и те, кто уже «что-то там программирует», — знание нюансов еще никому никогда не помешало.



Олимпиадные задачи по программированию

Питер, 2006 / Меньшиков Ф.В. / 315 страниц
Разумная цена: 150 рублей

Не был на олимпиаде по программированию? Не проблема! Можешь устроить олимпиаду сам себе, если используешь эту книжку. Правда, собранные в книге задачи явно проще тех, которые предлагаются на олимпиадах сейчас, — уровень задач возрастает из года в год. Тем не менее можно оттачивать свое мастерство и на олимпиадах прошлых лет. Некоторые задачи, к тому же, предлагают решить за ограниченное время. И самое главное, в конце книги есть разбор и решение всех приведенных задач — для самоконтроля или если никак не получается решить самостоятельно. Только учти, что почти на всех олимпиадах задачи решаются на Pascal'е (с использованием Turbo Pascal 7.0). Кто-то разгадывает кроссворды, а кто-то программирует на досуге. Можно сказать, кроссворды для программистов :).

Java 2, v5.0 (Tiger). Новые возможности

СПб.: БХВ-Петербург, 2005 / Шилдт Г. / 208 страниц
Разумная цена: 107 рублей

Еще первая версия Java (1.0) привнесла массу нового в написание программ для Сети. Часть ее популярности была обеспечена тем, что Java построен на хорошо знакомом многим синтаксисе C/C++, с упрощенным управлением памятью и встроенной поддержкой многопоточности. Сейчас эти характеристики воспринимаются как сами собой разумеющиеся, но десять лет назад они казались настоящим ноу-хау.

Java 5.0 — воплощение значительных изменений языка с момента появления первоначальной версии. Настраиваемые типы (generics) основательно расширяют и изменяют синтаксис Java. Автоупаковка (autoboxing) упрощает взаимодействие примитивных типов данных и объектов, а метаданные (metadata) — это совершенно новое в программировании. Книга полностью посвящена новым функциональным возможностям Java 2 версии 5.0 с кодовым названием «Тигр». Примеры, конечно, не очень практичные, но наглядные. Сама книжка адресована прежде всего тем, кто уже программирует на Java и решил перейти на пятую версию.

Нестандартные приемы программирования на Delphi

СПб.: БХВ-Петербург, 2005 / Ревич Ю.В. / 560 страниц
Разумная цена: 227 рублей

Создать свое приложение в среде Delphi несложно, даже если ты обладаешь только минимальными навыками программирования, обучаясь по ходу дела. Правда, функциональность созданных приложений будет соответствующая… Прочитав эту книгу, ты сможешь расширить функциональность своей программы, прикрутить удобный интерфейс и значительно ускорить время выполнения заложенных функций. Многие советы достаточно просты и легко реализуемы — именно с мелочей начинается хорошее приложение! Что такое потоковое чтение файлов и зачем оно нужно, как перехватить нажатие клавиш, как создать инсталляционный пакет и многое другое...

Программирование во Flash MX

СПб.: Символ-Плюс, 2005 / Пеннер Р. / 432 страницы
Разумная цена: 228 рублей

Никогда не забуду эйфорию от первых увиденных flash'овых роликов. Удивляло то, как в несколько килобайт влезает столько графики, анимации и звука.

Чтобы создавать классную динамическую компьютерную графику на Flash, недостаточно одного умения красиво рисовать — нужны и навыки программирования. Если конкретнее, понадобятся глубокие знания языка ActionScript: объектно-ориентированное программирование, тригонометрия (да-да, без нее никуда), системы координат, векторы (в основе анимации — векторная графика) и программирование, управляемое событиями. Добавь сюда движение, законы физики, окрашивание и рисование фигур.

Книга начинается, конечно, с простейших приемов проектирования и кодирования движения и постепенно доходит до профессионального объектно-ориентированного кода, позволяющего создавать интерактивность, цвет, звук и движение. Ты сможешь не просто программировать, а реализовывать свои самые смелые фантазии. Среди реализованных примеров в книге: танцующий фрактал, имитация снежной бури, вихря и северного сияния. Конечно же, все это ты сможешь модифицировать и использовать на собственном сайте.

Первые шаги в программировании. Самоучитель

М.: Издательский дом «Вильямс», 2006 / Ставровский А.Б. / 400 страниц

Разумная цена: 137 рублей

Всем нам понятно, что программистами не рождаются. Хотя способности к аналитическому мышлению, данные от природы, тоже не помешают :). Если до сих пор тебя не научили программировать или такого учителя в твоей жизни еще не было, то вперед к самостоятельному освоению программирования. Тебе понадобится всего-то: книга, время и желание. Вот, собственно, и книга. Здесь обучение начинается с азов: проектирование, разработка и отладка — неотъемлемые этапы любой программы. В книге куча примеров, по которым ты можешь не только реализовать что-то свое, сравнив результат с показанным в книге, но и модифицировать вещь под свой вкус. Приведенные примеры реализованы на Turbo Pascal, который станет неплохим стартом для тех, кто только учится программировать.

Человеческий фактор в программировании

СПб.: Символ-Плюс, 2004 / Константин Л. / 384 страницы

Разумная цена: 264 рубля

Любая программа, будь она плохая или хорошая, создается людьми. А раз так, то будет целесообразным учитывать так называемый человеческий фактор (peopleware), не менее важный, чем аппаратное или программное обеспечение.

Качество и продуктивность, модели и методы, динамика поведения коллектива, руководство проектами, разработка интерфейсов и взаимодействие человека и компьютера, психология и процессы мышления. Usability программных продуктов, наконец — о нем говорят очень часто.

Для тех, кто находится в дороге куда-нибудь, покажется весьма любопытным чтивом. Позволяет осознать очевидные на первый взгляд вещи, о которых задумываешься меньше всего. В технологиях постоянно происходят радикальные изменения, но люди меняются мало, так что описанное будет актуально и завтра, и через десять лет.

.NET Сетевое программирование для профессионалов

М.: Издательство «Лори», 2005 / Эндрю Кровчик / 400 страниц

Разумная цена: 330 рублей

Сетевое программирование сейчас очень актуально. Практически все новые приложения делаются с учетом возможности эффективного и безопасного взаимодействия разных компьютеров, находящихся в одном здании или разбросанных по всему миру. И среда .NET Framework предоставляет набор классов как раз для решения задач сетевого обмена. Правда, набор протоколов, поддерживаемый классами .NET, ограничен на транспортном уровне протоколами TCP и UDP, а на прикладном уровне — протоколами HTTP и SMTP. В этой книге описаны все необходимые классы, также наглядно показаны примеры реализации в .NET протоколов прикладного уровня.

Основные темы книги: обзор архитектуры физических сетей, сетевые протоколы и модель OSI, TCP, UDP и сокеты групповой рассылки, программирование сокетов в .NET, реализация протоколов прикладного уровня на примере FTP, интернет-программирование и классы .NET для электронной почты, реализация клиентов POP3 и NNTP для чтения из почтовых ящиков и групп новостей, защита сетевого обмена в .NET. Правда, книга рассчитана на тех, кто владеет сетевым программированием и имеет опыт программирования на C# — код всех примеров написан именно на нем.

Искусство программирования на Java

М.: Издательский дом «Вильямс», 2005 / Герберт Шилдт / 336 страниц

Разумная цена: 264 рубля

Большинство книг по Java обучают основам этого языка. Синтаксис, массивы, циклы и т.д. Примеры в такой литературе стандартны и кочуют из книги в книгу. Если ты ищешь чтиво для себя в подобном наборе, по сути, ты выбираешь не содержание, а удобное оформление и авторскую стилистику.

Эта книга как раз не описывает основы: она рассчитана на тех, кто уже практикует программирование на Java (имеет хорошие знания основ). Приведенные примеры достаточно актуальны и вполне могут пригодиться для собственных проектов: синтаксический анализатор выражений, интерпретатор языка, менеджер загрузок, почтовый клиент, поиск в Сети, статистика и графика, финансовые аплеты и сервлеты. Чем актуальнее поставленная задача и чем она ближе к практике, тем проще понять мощь и универсальность языка Java. Так что для любителей Java эта книжка — из разряда must have.

C++ Builder в задачах и примерах

СПб.: БХВ-Петербург, 2005 / Культин Н.Б. / 336 страниц

Разумная цена: 107 рублей

Здесь собраны разнообразные примеры, которые демонстрируют процесс создания программ, возможности среды разработки, назначение компонентов, знакомят с принципами работы с графикой, звуком, базами данных и т.п. Примеры различной сложности — от простейших задачек до приложений работы с графикой и мультимедиа. По сути, это самообслуживание. Читаешь короткое описание программы, изучаешь листинг и въезжаешь: что, где и зачем. Остается только повторить не глядя в первоисточник. И не бойся экспериментировать, внося свои изменения в предложенный код. Вторая часть книжки — краткий справочник с описанием базовых компонентов и наиболее часто используемых функций (будет полезно не только склеротикам, но и совершенно нормальным людям)

задай вопросы по темам следующих выпусков на форуме:

http://forum.xakep.ru/forum.asp?forumID=17

спроси эксперта!

НАУЧИТЬ ПРОГРАММИРОВАТЬ НЕЛЬЗЯ — ПРОГРАММИРОВАНИЕ НУЖНО ЛЮБИТЬ

НА ВОПРОСЫ ОТВЕЧАЕТ ЭКСПЕРТ ЭТОГО НОМЕРА — ДЕНИС БАТРАНКОВ. ДЕНИС ЗАНИМАЕТСЯ ПРОГРАММИРОВАНИЕМ УЖЕ 20 ЛЕТ. АДМИНИСТРИРУЕТ СЕРВЕРЫ SOLARIS, FREEBSD И WINDOWS. ПИШЕТ СТАТЬИ В ИНТЕРНЕТЕ. В КОМПАНИИ «ИНФОРМЗАЩИТА» ЧИТАЕТ ЛЕКЦИИ ПО БЕЗОПАСНОСТИ КОРПОРАТИВНЫХ СЕТЕЙ. ИМЕЕТ СЕРТИФИКАТ CCNA

ВОПРОС: ПРОГРАММИРОВАНИЕ ТОЛЬКО ДЛЯ ИЗБРАННЫХ? ИЛИ ПРОГРАММИРОВАНИЕ ДОСТУПНО ВСЕМ — БЫЛО БЫ ЖЕЛАНИЕ? МОЖНО ЛИ НАЗВАТЬ КАКИЕ-ТО БАЗОВЫЕ ПРЕДРАСПОЛОЖЕННОСТИ, ПО КОТОРЫМ ВЫЯСНЯЕТСЯ, СПОСОБЕН ЛИ ЧЕЛОВЕК К ПРОГРАММИРОВАНИЮ?

ОТВЕТ: Безусловно, программирование нужно любить. Когда я впервые пришел в кружок программирования в 1986 году (я был в четвертом классе), мы сидели за компьютерами по двое и по трое. В конце учебного года в кружок ходил уже один я, поскольку играть нам запрещали, а программировать ребята устали. Как и в любом деле, если тебе что-то интересно и у тебя хорошо получается, то ты будешь заниматься этим и получать удовольствие. И, соответственно, если тебе хочется программировать, то предрасположенность у тебя уже есть. Еще в школе, когда всех без разбора заставляют учить языки программирования, становится ясно, кто будет программистом: видно, кто играет в игры, а кто получает удовольствие от результата работы свеженаписанной программы.

Для программистов характерны такие черты, как технический склад ума и усидчивость. Многие из этих людей способны не вставать из-за компьютера, проводить сутки за ним, пока не найдут баг или не решат поставленную задачу. Может быть, это похоже на самопожертвование, но я говорю о том, ради чего программисты живут.

Некоторые рассказы о программистах вводят нас в заблуждение: мол, все они постоянно курят и пьют пиво. Я не могу говорить обо всех, но они курят и пьют так же и в том же количестве, что и все люди. Я, например, не курю и <Ctrl> пивом не нажимаю. Предпочитаю цивилизованные посиделки в баре или ресторане. По поводу внешнего вида — та же картина. Вообще говоря, после нескольких суток поиска багов любой программист выглядит не очень свежо (как и шахтер после смены), но это не мешает ему появляться в обществе в приличном костюме, выбритым и пахнущим модным парфюмом. Единственный «минус» программиста в том, что его мозг не может остановиться — они пишут программы дома, а не только на работе, чем не очень радуют своих жен и детей. В последнее время я прихожу к мысли о том, что нужно все-таки беречь себя, работать по восемь часов в день и проводить субботу и воскресенье на воздухе. Существует много интересных занятий: лыжи, бильярд, прыжки с парашютом, встречи с друзьями…

ВОПРОС: КОГО ВООБЩЕ МОЖНО НАЗЫВАТЬ ПРОГРАММИСТОМ? ВОТ, НАПРИМЕР, ЧЕЛОВЕК ПОНИМАЕТ СИНТАКСИС BASIC'A. ЭТОТ ЧЕЛОВЕК ПРОГРАММИСТ?

ОТВЕТ: Понимать синтаксис — это только начало. Самое главное, что нужно уметь, — эффективно решать поставленную задачу. Этот творческий процесс близок к искусству.

И применений программистам очень много. Если пишешь драйверы под Windows, то вряд ли ты вдруг согласишься писать драйверы под Linux, хотя синтаксис языка C везде одинаковый. Или после драйверов вдруг начать писать трехмерные игры тоже непросто. Дело в том, что программирование уже давно не состоит из множества операторов базового языка. Всегда программисты используют писавшиеся годами библиотеки. Этих библиотек очень много, и каждую нужно изучать. Даже освоить один Win32 API тяжело, а еще, возможно, тебе нужно разобраться с MFC или ATL. Мало того, что библиотеки состоят из готового кода. Казалось бы, бери и используй, но уже придумано множество новых технологий, которые тоже нужно знать и принцип работы которых тоже нужно понять.

Как пример можно привести технологии разработки компонентов COM/DCOM, доступа к данным ADO или BDE, мультимедиа GDI, OpenGL, DirectX, сообщений MAPI и т.д. Сложность еще и в том, что все технологии постоянно совершенствуются. Ты изучал ATL3 и IDL, прошло время, а уже нужно использовать ATL7 и attributed C++. А еще есть недокументированные функции — они тоже иногда полезны.

ВОПРОС: ЕСТЬ ЛИ ЭФФЕКТИВНЫЕ КУРСЫ ДЛЯ ПРОГРАММИСТОВ? ИЛИ МОГУТ БЫТЬ ПОЛЕЗНЫМИ ТОЛЬКО КНИЖКИ И СОБСТВЕННЫЙ ОПЫТ? В ШКОЛАХ И ИНСТИТУТАХ ОБЫЧНО НЕ УЧАТ НИЧЕМУ ДЕЛЬНОМУ, КАК ПОКАЗЫВАЕТ ПРАКТИКА...

ОТВЕТ: Я считаю, что научить программировать невозможно! Школы и институты учат полезным вещам, есть хорошие курсы, есть хорошие преподаватели. Только один минус: в школе и институте могут лишь объяснить понятия и имеющиеся алгоритмы — что такое массив и список, семафор и спинлок, что такое процесс и нить, как использовать регистры и стек, как адресуется память, как выполняется быстрая сортировка и сортировка пузырьком и т.д. Но в школе/институте нельзя получить представление о том, что из этого лучше всего использовать в программе и как объединить все, в каком порядке. Остальное будет делать программист на основе имеющихся знаний и фантазии.

Программа — это творчество программиста, а не скучное собрание известных алгоритмов в единое целое. Именно поэтому все программы отличаются друг от друга, хотя, возможно, и делают одно и то же. Одна программа будет работать быстрее, другая будет меньше места на диске занимать, третья — меньше оперативной памяти требовать. Сколько людей, столько и программ. Для повышения квалификации рекомендую читать чужой код — он вмещает в себя чужой опыт. В интернете достаточно примеров исходного кода, из которого можно почерпнуть знания для написания своих программ. Кроме того, нужно общаться в конференциях, не бояться задавать вопросы и, возможно, отвечать на вопросы других. Например, Microsoft очень ценит активных участников конференций и особенно активным участникам присуждает звание MPV (Most Valuable Professional).

ВОПРОС: СУЩЕСТВУЕТ МНОЖЕСТВО ЯЗЫКОВ ПРОГРАММИРОВАНИЯ, СРЕДСТВ РАЗРАБОТОК ЕЩЕ БОЛЬШЕ. КАК ВЫБРАТЬ НУЖНОЕ НАПРАВЛЕНИЕ, ОСОБЕННО ЕСЛИ НЕ ИМЕЕШЬ ЧЕТКОГО ПРЕДСТАВЛЕНИЯ О ТОМ, ЧТО ПОТРЕБУЕТСЯ В БУДУЩЕМ?

ОТВЕТ: Беспроигрышного варианта нет. Вряд ли можно сразу начать изучать то, что точно пригодится тебе в жизни. Нужно чтобы тебе повезло. А чтобы повезло, нужно просто начать заниматься программированием. Неважно, на каком языке: Java, C# или другом. Когда ты программируешь, ты накапливаешь опыт написания алгоритмов. И когда у тебя есть опыт, тебе уже все равно, какой язык программирования нужен для решения поставленной задачи. Естественно, перейти с Delphi на MSVC будет непросто, но реально. Когда-то у меня получилось.

ВОПРОС: ЧТОБЫ НАПИСАТЬ СЛОЖНУЮ ПРОГРАММУ, НУЖЕН ДОСТАТОЧНО БОЛЬШОЙ ПРАКТИЧЕСКИЙ ОПЫТ. ЧТОБЫ ПОЛУЧИТЬ НЕОБХОДИМЫЙ ПРАКТИЧЕСКИЙ ОПЫТ, НУЖНО ПИСАТЬ СЛОЖНЫЕ ПРОГРАММЫ... ПОЛУЧАЕТСЯ, ЗАМКНУТЫЙ КРУГ?

ОТВЕТ: Что такое сложная программа? Когда я в шестом классе показывал своим одноклассникам обучающую программу по физике, которую я написал на Фокале под БК0010 и которая занимала 1000 строк, все были повергнуты в шок ее размером. А теперь у меня в проекте, состоящем из нескольких десятков исходников, один .cpp-файл может быть 3000 строк. Сложность и опыт идут вместе. Нужно постепенно развиваться, начиная с программы Hello World, и дальше все потянется одно за другим само за собой. Главное — не бояться и смело браться за сложные программы.

Еще нужно заметить вот что. Чтобы писать сложные программы, нужно знать английский язык. Вся свежая литература выходит на английском языке, и ждать несколько лет, когда выйдет перевод, просто нет возможности. На всех серьезных конференциях, где можно что-то спросить, тоже говорят на английском. Да и имена переменных тоже не мешало бы писать английскими словами :).

ВОПРОС: ЕСЛИ ЕСТЬ КЛАССНАЯ ЗАДУМКА, НО НЕТ ОПЫТА В ПРОГРАММИРОВАНИИ, КАК ОПРЕДЕЛИТЬСЯ, НА ЧЕМ И КАК ПРОГРАММИРОВАТЬ? МЕТОД ПРОБ И ОШИБОК СЪЕСТ КУЧУ ВРЕМЕНИ, ЕСЛИ ВОВСЕ НЕ ОТОБЬЕТ ЖЕЛАНИЕ ДЕЛАТЬ ЧТО-ЛИБО...

ОТВЕТ: Пробы и ошибки неизбежны. Даже программа Hello World почему-то никогда с первого раза не компилируется :). Если ты не опускаешь руки, а находишь в себе силы найти ошибку, то ты программист. Если нет опыта, но есть другие программисты, которые всегда готовы помочь, — обращайся к ним. Однажды (когда был классе в восьмом) я около месяца пытался найти баг в программе на Ассемблере для БК0010. Программа не работала, а я уже использовал все приемы: в уме проходил по строкам и на листочке записывал ход изменения данных. Но это было, когда я учился в школе, и в тот момент надо мной не было менеджеров, жаждущих срочного конечного результата. Я тогда отложил программу и вернулся к ней через месяц. Ошибка была найдена (не хватало одного символа: вместо MOV нужно было написать MOVB), и ощущение триумфа не оставляло меня еще долгое время. Ощущение, что ты побеждаешь компьютер, а не он тебя, поддерживает желание программировать.

Как-то раз меня попросили помочь хирургу настроить SQL-запросы. Для написания диссертации и сбора статистики он завел базу данных. Видимо, кто-то посоветовал ему использовать MS Access. Когда я приехал и посмотрел, что он делает, я был поражен тем, насколько хирург оказался способным в программировании. Он вел статистику проведенных операций, сам наделал различных форм и запросов, чтобы выявлять зависимости между различными условиями проведения операций, и в конце концов запросы стали настолько сложными, что стандартных wizard'ов перестало хватать. В течение нескольких дней мы с ним написали нужные запросы, хотя опыта в программировании у него не было никакого... Вот что творит желание.

ВОПРОС: ЛЮДЯМ «СО СТОРОНЫ» ПРОГРАММИРОВАНИЕ КАЖЕТСЯ ИНТЕРЕСНОЙ И ВЫСОКООПЛАЧИВАЕМОЙ РАБОТОЙ. ТАК ЛИ НА САМОМ ДЕЛЕ? НЕ ЯВЛЯЕТСЯ ЛИ ПРОГРАММИРОВАНИЕ СПЛОШНОЙ РУТИНОЙ, К ТОМУ ЖЕ НЕ ВСЕГДА ВЫСОКООПЛАЧИВАЕМОЙ?

ОТВЕТ: Наверное, это зависит от твоего отношения к работе. Если она кажется тебе рутиной, нужно менять профиль. Ну а зарплата зависит от того, как ты договоришься. Если ты будешь просить за свою работу мало, то никто не поверит, что ты профессионал. Как правило, тут с умом нужно выбирать ту фирму, где ты собираешься работать.

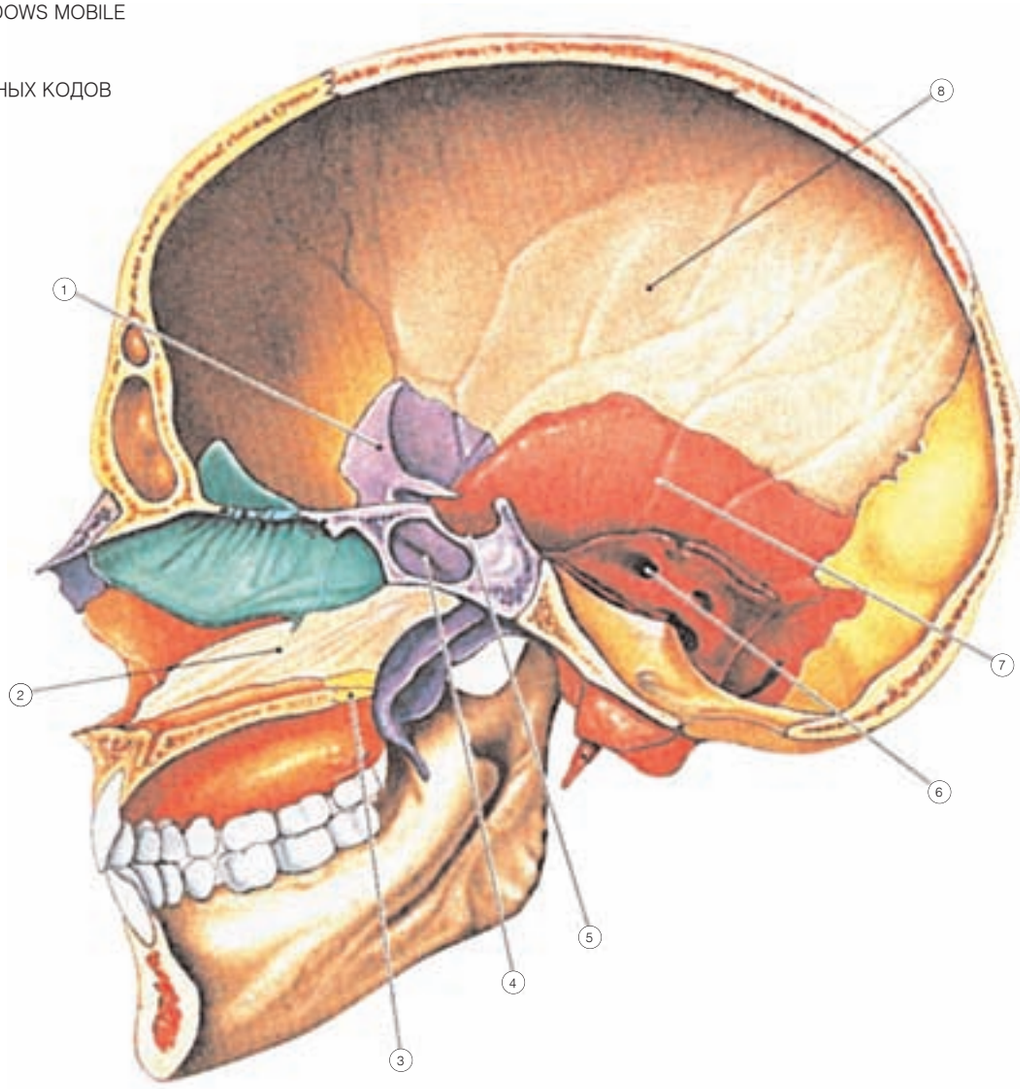
Должно присутствовать и определенное везение. И потом... Как может быть рутиной программирование, если ты все время пишешь разные программы? Никто не просит написать программу, которую кто-то уже написал. В этом, правда, есть определенная трудность — тебе приходится быть первопроходцем. Для меня самым редким заказом был драйвер, перехватывающий обращения к диску в операционной системе OS/2. Тогда пришлось хорошенько изучить OS/2 и заодно освоить Virtual Pascal под OS/2

тотальный взлом

ИЗ СЛЕДУЮЩЕГО НОМЕРА ТЫ УЗНАЕШЬ О ТОМ, КАК ВЗЛАМЫВАЮТ:

1 ICQ
2 .NET-КОМПОНЕНТЫ
3 ПРОГРАММЫ 1С
4 DSL-МОДЕМЫ
5 ТЕЛЕВИЗОРЫ И МОНИТОРЫ
6 БАЗЫ ДАННЫХ
7 ПРОТОКОЛЫ МАРШРУТИЗАЦИИ
8 ПРОГРАММЫ ДЛЯ WINDOWS MOBILE

+
ОБХОД ЗАЩИТЫ ИСХОДНЫХ КОДОВ СКРИПТОВ И АППЛЕТОВ
И МНОГОЕ ДРУГОЕ!



СКОРО В СПЕЦЕ:

WINDOWS VISTA. ВЗГЛЯД ИЗНУТРИ. ПОДРОБНЫЙ АНАЛИЗ НОВОЙ ОС ОТ MICROSOFT. НОВЕЙШИЕ ТЕХНОЛОГИИ. УДОБСТВО, БЫСТРОТА РАБОТЫ.

САЙТОСТРОЕНИЕ. WEB-КОДИНГ: НОВЕЙШИЕ ТЕХНОЛОГИИ, ЯЗЫКИ, НЮАНСЫ. ДЕЙСТВЕННЫЕ СПОСОБЫ ПРОДВИЖЕНИЯ САЙТА. ПОРТАЛ СВОИМИ РУКАМИ.

BSD. УСТАНОВКА, НАСТРОЙКА, УПРАВЛЕНИЕ BSD-СИСТЕМАМИ. ИСТОРИЯ. БЕЗОПАСНОСТЬ.

два по два

ТЕСТИРОВАНИЕ ПАМЯТИ DDR2

В ПОСЛЕДНЕЕ ВРЕМЯ ШИРОКО РАСПРОСТРАНИЛИСЬ ПЛАТФОРМЫ LGA775 И, СООТВЕТСТВЕННО, ПОСЛЕДНИЕ МОДЕЛИ ПРОЦЕССОРОВ INTEL PENTIUM 4, ПОЭТОМУ ПАМЯТЬ ТИПА DDR2 СТАНОВИТСЯ ВСЕ БОЛЕЕ ЦЕННОЙ, ХОТЯ СОВСЕМ НЕДАВНО ОНА БЫЛА ДОВОЛЬНО СОМНИТЕЛЬНЫМ АРГУМЕНТОМ ДЛЯ АПГРЕЙДА. ВРЕМЯ ИДЕТ, ЧАСТОТЫ УВЕЛИЧИВАЮТСЯ, ХАРАКТЕРИСТИКИ УЛУЧШАЮТСЯ. НА НАШ ВЗГЛЯД, НАСТАЛО САМОЕ ВРЕМЯ ПРОВЕСТИ НЕБОЛЬШОЙ СРАВНИТЕЛЬНЫЙ ТЕСТ МОДУЛЕЙ, ПРЕДСТАВЛЕННЫХ НА РЫНКЕ. УЗНАЕМ, КАКИЕ ИЗ НИХ ИДЕАЛЬНО ПОДОЙДУТ ПОКУПАТЕЛЯМ ОПРЕДЕЛЕННОЙ КАТЕГОРИИ |

ОКУНЕВ ДМИТРИЙ, TEST_LAB (TEST_LAB@GAMELAND.RU)

тестовый стенд

МАТЕРИНСКАЯ ПЛАТА: **Asus P5WD2 Premium**
ПРОЦЕССОР, ГГЦ: **3.46, Intel Pentium 4 EE**
ВИДЕОКАРТА: **MSI NX7300GS**
КУЛЕР: **GlacialTech Igloo 5700 MC**
ЖЕСТКИЙ ДИСК, ГБ: **80, Seagate 7200 rpm**
БЛОК ПИТАНИЯ: **350 Вт PowerMan Pro**

технологии «Покупателем определенной категории» был назван, конечно же, «пользователь обыкновенный», который требует от памяти прежде всего надежности и качества. К той же популяции мы отнесли оверклокеров — особую категорию пользователей, которым, помимо качества, важна такая негласная характеристика «железа», как разгонный потенциал, то есть пределы, в которых можно увеличивать производительность любыми доступными способами.

Для увеличения производительности памяти существует целых два способа. Первый — увеличение частоты работы. Аналогично разгону процессора, разгон памяти по частоте способен значительно повысить производительность, особенно в приложениях, чувствительных к скорости этого компонента (архивация, Adobe Photoshop и т.д.). Учтем одно «но»: как правило, по умолчанию материнская плата повышает частоту системной шины (а значит, и процессора) синхронно с частотой памяти, следовательно, мы рискуем упереться в потенциал «камня» достаточно быстро, что, впрочем, легко исправляется при помощи т.н. «делителей» в BIOS.

Если используется второй способ разгона памяти, изменяют ее особые характеристики — тайминги, то есть несколько значений тактов, затрачиваемых на выполнение внутренних операций при доступе к данным. Вот их названия: CL (CAS Latency — тайминг, самый важный для производительности), tRCD (RAS to CAS Delay), tRP (RAS Precharge) и tRAS (RAS Active to Precharge). Обычно они записываются в виде формулы типа CL-tRCD-tRP-tRAS (например 3-3-3-5, 4-5-5-12 и т.д.). Ясно, что чем меньше тактов затрачивается на ту или иную операцию, тем в конечном счете выше производительность. Соответственно, для такого «разгона» необходимо занижать тайминги по максимуму.

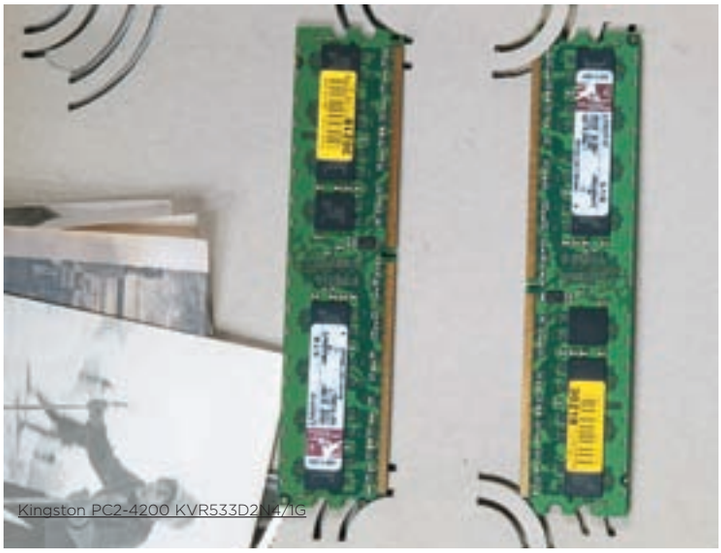
Вот тут-то выскакивает второе «но». Низкие тайминги и высокая рабочая частота обычно не уживаются друг с другом, поэтому приходится делать выбор: либо занижать тайминги на низкой частоте, либо увеличивать их, причем одновременно увеличивается частота работы. Притом, вполне возможно, первый режим будет производительнее второго, и тут ты сможешь разобраться только с помощью тестов.

методика тестирования Вся память тестировалась в двухканальном режиме: для планок, продающихся в соответствующем комплекте, проблем не возникало, а модули, поставляемые по одиночке, мы брали в двух экземплярах. Для каждого набора мы находили минимально возможные тайминги на частоте 533 МГц, затем увеличивали их, а память разгоняли уже по частоте. Искали максимально возможную частоту, затем снова находили на ней минимальные тайминги. Чтобы уравнять условия тестирования модулей, рассчитанных на завышенное напряжение, и простых планок, мы выставили для всей памяти напряжение в 2,0 В (значение по умолчанию — 1,8 В).

Test_lab выражает благодарность за предоставленное на тестирование оборудование компаниям: «НИКС – Компьютерный Супермаркет» (тел. (495)974-33-33, **www.nix.ru**), Patriarch Group (тел. (495)789-80-89, **www.memory.ru**), 3logic (тел. (495)540-91-36, **www.3logic.ru**), а также российскому представительству компаний AMD, Asus, Corsair.

Corsair CM2X512-8000UL

Corsair CM2X512A-4300C3PRO


Kingston PC2-4200 KVR533D2N4/1G

Kingston HyperX DDR2-667 KHX6000D2/512

Corsair CM2X512A-4300C3PRO
($113)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **3-3-3-8**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-2-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **786**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-4-4-8**

Эта память типа DDR2-533 поставляется в комплекте Dual Channel. Соответственно, не возникало никаких проблем с необходимостью приобретать вторую аналогичную планку. Сразу отметим базовые тайминги, зашитые в микросхему SPD. Здесь они довольно низкие и составляют 3-3-3-8. Модули снабжены радиаторами, но самое интересное в них — не охлаждение, а наличие светодиодов в верхней части ряда, которые, как светомузыка, отображают уровень активности обращения к памяти. Эта мегаинтересная и удобная вещь наглядно представляет состояние системы: в данный момент система висит или, например, она переживает немалую нагрузку. О том, насколько этот элемент ценен для моддеров, мы умолчим.

Corsair CM2X512-8000UL
($189)*****

БАЗОВАЯ ЧАСТОТА, МГЦ: **400 (DDR800)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **5-5-5-18**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-2-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **946**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-5-4-4**

Если хочется возможностей больше, чем предоставляется обычной памятью DDR2-533 или DDR667, если хочется выжать из системы все до последней капли, то вот такие модули с поддержкой рабочих частот до 1 ГГц были выпущены компанией Corsair специально для тебя! Правда, за возможность получить такие заоблачные характеристики ты заплатишь дорого: цена не низкая и, кроме того, придется позаботиться о питании. Дело в том, что для покорения высоких частот этой памяти требуется питание 2,2 В, что будет осилено не любой материнской платой. Хотя наша плата имела такую возможность, тест проходил на «общих основаниях», то есть при напряжении 2,0 В, с которыми потенциал модулей оказался фактически на уровне Corsair CM2X512A-5400UL.

Kingston PC2-4200 KVR533D2N4/1G
(2*$97)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **1024**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-12**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **784**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **4-5-3-4**

Целый гигабайт памяти на планку (правда, всего на одну, так что для двухканального режима придется самостоятельно докупать второй экземпляр). Сами модули явно не ориентированы на высокие показатели разгона: радиаторов нет, базовые тайминги довольно высокие и немного не дотягивают до идеальной формулы 3-2-2-4. Впрочем, при оверклоке этот недостаток компенсируется, так как на максимальной частоте удалось удержать параметр CL на уровне «4», что само по себе довольно хорошо. В целом память предоставляет неплохое сочетание цены и качества, но не стоит ждать от нее чего-то экстраординарного — не та категория. Зато есть пожизненная гарантия — явный показатель того, что компания Kingston не сомневается в качестве своих модулей.

Kingston HyperX DDR2-667 KHX6000D2/512
(2*$108)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **333 (DDR667)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **5-5-5-15**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **878**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-5-4-4**

Серия модулей памяти HyperX от компании Kingston — отличный выбор для энтузиастов всех мастей. Как-никак, здесь есть и высокая рабочая частота, и хороший разгонный потенциал, и симпатичное охлаждение, которое не позволит всему добру перегреться в «тяжелых» условиях. Правда, изначальные установки в SPD выглядят скромновато: 5-5-5-15 для режима DDR667, но в то же время нам удалось уменьшить эту формулу на гораздо более высокой частоте!

Не очень порадовала и комплектация. «Одиночная» поставка значительно проигрывает полноценному двухканальному набору, который все-таки гарантирует стабильность работы во всех заявленных режимах и создает покупателю меньше затруднений…

выбор редакции СПЕЦ

Corsair CM2X512A-5400UL

Corsair CM2X512A-5400UL
($154)*****

БАЗОВАЯ ЧАСТОТА, МГЦ: **333 (DDR667)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-15**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-2-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **940**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-3-2-4**

Данные планки рассчитаны на оверклокеров и тех пользователей, которые радеют за максимальную производительность своего компьютера: память имеет частотную формулу DDR667, а на самих модулях оставлена пометка о том, что они тестировались и на частоте 675 МГц. Впрочем, нам удалось добиться гораздо более впечатляющих результатов. Память заработала вполне сносно, как DDR940, при не самых плохих таймингах! А в режиме DDR2-533 дела складывались и вовсе «шоколадно»: можно смело выставлять задержки на минимально возможные 3-2-2-4 (кстати, в SPD почему-то не прописаны оптимальные тайминги для данной частоты).

Естественно, память поставляется в надежной упаковке и снабжена симпатичными черными радиаторами. Как же без них?

Kingmax MARS DDR2-533
(2*$58)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-12**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **790**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-5-4-4**

Первый же взгляд на эти модули (когда они были еще не распакованы) заставил нас впасть в изумление. Коробка, в которой они поставляются, выглядит как сувенир, а совсем не как память! К сожалению, внутри была обнаружена только одна планка и двухканальный режим становится доступным только после покупки двух таких коробочек. Сама же память ничем не выделяется. Обыкновенная DDR2-533 с довольно скромной формулой таймингов по SPD и неплохим разгонным потенциалом. Впрочем, есть и заметный плюс: если сравнивать хотя бы с теми же Corsair, цена на эти модули совсем не велика, что объясняется отсутствием охлаждения и ориентацией в первую очередь на мэйнстримовый ценовой сектор.

Неплохой выбор для домашней системы без претензий на хардкорный разгон.

Предельные режимы работы

память	част.	CL	tRCD	tRP	tRAS
Corsair CM2X512A-4300C3PRO	786	3	2	2	4
Corsair CM2X512A-5400UL	940	3	2	2	4
Corsair CM2X512-8000UL	946	3	2	2	4
Kingston PC2-4200 KVR533D2N4/1G	784	3	3	2	4
Kingston HyperX DDR2-667	878	3	3	2	4
Kingmax MARS DDR2-533	790	3	3	2	4
Crucial DDR2-533	818	3	3	3	4
Hynix PC2-4200U	725	3	3	3	4
Hynix PC2-5300U	906	3	3	2	4
OCZ PC2-4200	808	3	3	3	4
OCZ PC2-5400	820	3	3	3	4
Patriot DDR2-667	928	4	2	2	4
Samsung PC2-5300	828	4	3	3	4
Transcend DDR2-533	800	3	3	3	4

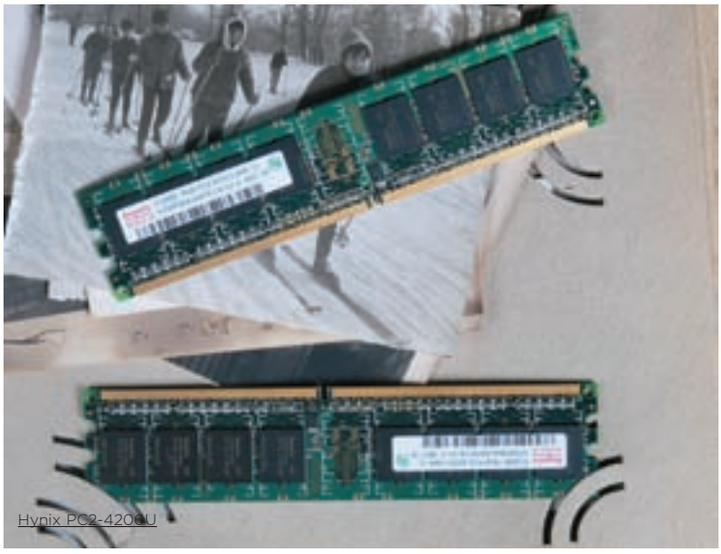
Crucial DDR2-533
(2*$60)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-12**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-3-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **818**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-5-4-4**

Модули Crucial DDR2-533, собранные на чипах Micron, не привлекают внимания комплектацией или охлаждением (ни того, ни другого здесь просто нет), зато демонстрируют весьма неплохое качество. Так, максимальная частота разгона составила целых 818 МГц при таймингах, идентичных полученным на аналогичных модулях. Впрочем, это хорошо лишь для тех, кто гонится за мегагерцами. Как же быть тем, кто предпочитает стандартные частоты с низкими таймингами? Вот здесь память проявляет себя не лучшим образом: 3-3-3-4 — неплохой результат, но он не сравнится с показателями тех же Corsair. Тайминги высоковаты и по SPD, поэтому, если хочешь использовать модули «на всю катушку», стоит заняться их выставлением самостоятельно.


Kingmax MARS DDR2-533


Crucial DDR2-533


Hynix PC2-4200U


OCZ PC2-4200

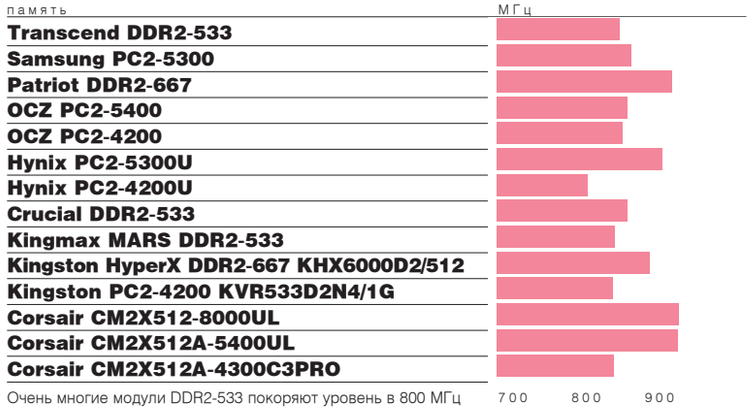
Hynix PC2-4200U
(2*$55)***

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-12**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-3-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **725**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **4-5-3-4**

Компания Hynix давно зарекомендовала себя как производитель качественной и недорогой памяти для пользователей, не увлекающихся экстремальным разгоном. Данные модули — как раз из числа тех, которые не претендуют на лавры оверклокерского хита моделей. Разгонный потенциал невелик, тайминги тоже не блещут минимализмом, впрочем, параметр CL вполне можно удержать на уровне «4» даже при разгоне.

Комплектацией здесь даже не пахнет. Модули поставляются «как есть», без упаковки и намеков на двухканальность. В то же время цена представляет эти планки в свете более выгодном, чем большинство более «продвинутых» аналогов, — для тех пользователей, кто просто хочет иметь в системе качественную память, кто прикасается к BIOS крайне редко и кому не нужно большего.

Частота



Очень многие модули DDR2-533 покоряют уровень в 800 МГц

OCZ PC2-4200
($135)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-12**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-3-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **808**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **4-5-4-4**

Если «железо» доставляет тебе прежде всего эстетическое удовольствие и память как тип комплектующих кажется тебе слишком неинтересной, советуем присмотреться к данным планкам. Модули OCZ не только поставляются в удобной упаковке и комплекте Dual Channel, но и «одеты» в красивые золотистые радиаторы.

Что с частотами? При штатном значении DDR2-533 эти планки неплохо гонятся до 808 МГц с сохранением неплохих значений таймингов: 4-5-4-4! Впрочем, на штатных 533 МГц формула выводится не самая идеальная — 3-3-3-4 (tRCD и tRP понижаются не до конца).

Итого, неплохой выбор для ценителей качества, красоты и разгонного потенциала. Однако если ты ориентируешься на низкие частоты и тайминги, подбирай что-нибудь другое, более соответствующее твоим запросам.

OCZ PC2-5400
($166)****

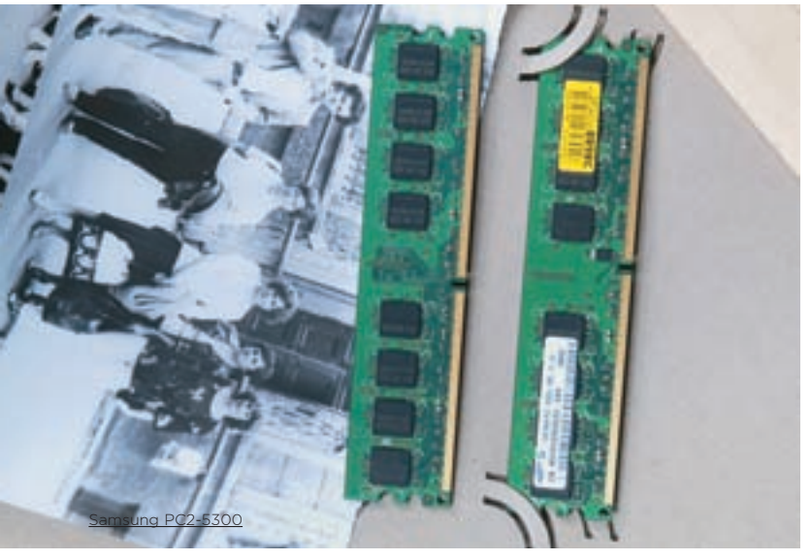
БАЗОВАЯ ЧАСТОТА, МГЦ: **333 (DDR2-667)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-12**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-3-4**
МАКСИМАЛЬНАЯ РАБОЧАЯ ЧАСТОТА, МГЦ: **820**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **4-5-4-4**

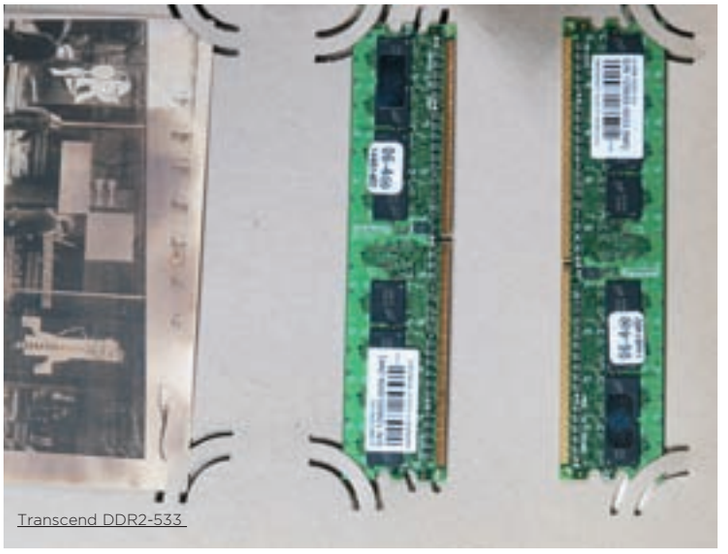
Снова OCZ. Эти модули внешне абсолютно идентичны предыдущим и по комплектации, и по наружности. Другое сходство — тайминги по SPD, которые один в один соответствуют обнаруженным в памяти OCZ PC2-4200, правда, теперь они записаны для частоты 667 МГц. Кстати, тут же обнаруживается и небольшой минус: в SPD не прописано никаких значений, кроме таймингов для штатной частоты. В то же время настройки для режимов DDR2-533 и DDR2-400 были бы весьма полезны!

Если говорить о разгонном потенциале, то эти планки несильно опередили «младших» сестер, а по части задержек и вовсе оказались похожими один в один. Тут же приходит на ум старый рекламный слоган: «Если нет разницы, зачем платить больше?»

OCZ PC2-5400

Patriot DDR2-667


Samsung PC2-5300


Transcend DDR2-533

Patriot DDR2-667
($100)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **333 (DDR2-667)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **есть**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **5-5-5-15**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **4-2-2-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **928**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-3-3-4**

Комплектация Dual Channel, наличие радиаторов — чего же не хватает памяти для полного счастья? Не хватает самой малости — разгонного потенциала, что, впрочем, совсем не касается модулей Patriot DDR2-667. Как раз с этим у них полный порядок! Надпись на упаковке гласит, что планки тестировались и стабильно работали на частоте 700 МГц. Мы же удостоверились, что и 928 МГц ей не помеха, причем с неплохими таймингами (за исключением CL).

Плоховато у этой памяти только с низкими частотами. В режиме DDR2-533 CAS Latency не удалось опустить ниже четырех, что весьма и весьма слабо (более дешевые модули Hynix и Kingmax справляются не в пример лучше). Вот такая дилемма: либо частота, либо тайминги…

Samsung PC2-5300
(2*$124)***

БАЗОВАЯ ЧАСТОТА, МГЦ: **333 (DDR2-667)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **1024**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **5-5-5-13**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **4-3-3-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **828**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-5-4-10**

Модули произведены Samsung, что означает гарантированное использование собственных чипов памяти и, как минимум, надежность. Объем, составляющий в сумме 2 Гб, — это и вовсе сказка с немалым заделом на будущее. «Сказка» немного омрачается голой поставкой, которая, правда, представляет собой не главный минус продукта. Более существенные недостатки выясняются в процессе тестирования: на частоте 533 МГц память не способна работать с параметром CL, равным трем, а разгонный потенциал невелик и колеблется в районе 830 МГц. Впрочем, здесь стоит заметить, что невысокие результаты, в общем-то, свойственны модулям с большим объемом — сказываются наводки и прочие негативные факторы. В то же время аналогичная память Kingston показала себя, кажется, более грамотно…

Transcend DDR2-533
(2*$56)****

БАЗОВАЯ ЧАСТОТА, МГЦ: **266 (DDR2-533)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **4-4-4-11**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-3-4**
МАКС. РАБОЧАЯ ЧАСТОТА, МГЦ: **800**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-4-4-4**

Модули Transcend отличаются невысокой ценой и соответствующей комплектацией. Однако прекратим поверхностный осмотр и перейдем непосредственно к делу. При штатном режиме 533 МГц разгонный потенциал модулей оказался довольно неплохим: как DDR2-800, они работают вполне успешно. Притом, естественно, приходится выставлять немалые тайминги, что, однако, вполне компенсируется их «правильным» поведением на тех же 533 МГц. Поведение «правильно» не полностью, так как все же не удается опустить tRCD и tRP до двух…

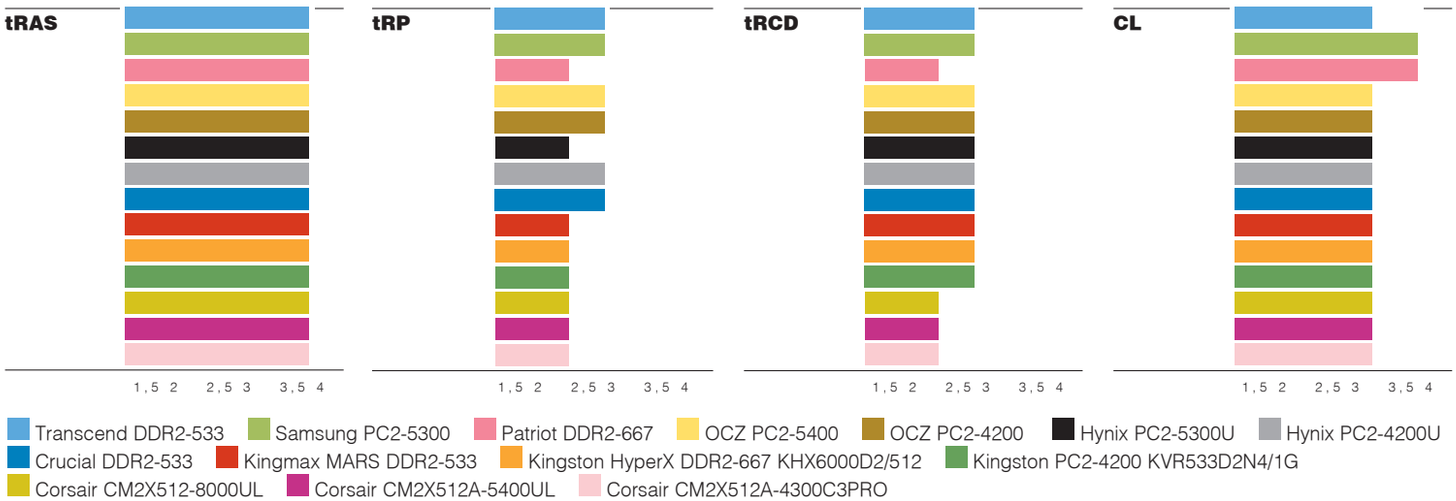
Разумеется, память данной ценовой категории даже не намекает на охлаждение — возможно, так в определенной степени ограничивается разгонный потенциал, но мы более чем довольны имеющимся.

Hynix PC2-5300U
(2*$66)*****

БАЗОВАЯ ЧАСТОТА, МГЦ: **333 (DDR2-667)**
ОБЪЕМ ОДНОЙ ПЛАНКИ, МБ: **512**
НАЛИЧИЕ РАДИАТОРОВ: **нет**
ТАЙМИНГИ В БАЗОВОМ РЕЖИМЕ: **5-5-5-15**
МИН. ТАЙМИНГИ В РЕЖИМЕ DDR2-533: **3-3-2-4**
МАКСИМАЛЬНАЯ РАБОЧАЯ ЧАСТОТА, МГЦ: **906**
МИН. ТАЙМИНГИ НА МАКС. ЧАСТОТЕ: **5-5-4-4**

Отсутствие упаковки, необходимость покупать два модуля (они поставляются поодиночке) и скромный внешний вид — эти мелочи, казалось бы, могут подпортить первое впечатление встречи с данной памятью. Однако после установки в систему и небольших манипуляций, может быть, ситуация изменится в корне: так как эти планки рассчитаны на частоту работы 667 МГц, они неплохо показали себя в разгоне. Точнее, мы получили целых 906 МГц! Конечно, пришлось значительно повысить тайминги и пока остается открытым вопрос «Стоит ли игра свеч?», но факт остается фактом — потенциал Hynix PC2-5300U весьма и весьма немалый. Кроме того, на низких частотах память не подвела и тайминги понижаются вполне охотно.

Тайминги



Распределение таймингов в режиме DDR2-533. Чем меньше значение, тем лучше, особенно стоит обратить внимание на параметр CL!

ВЫВОДЫ Как видишь, разная память ориентирована на разных потребителей. Оверклокерские модели требуют больших вложений, но и в ответ выкладываются соответствующе. Недорогие модули не отличаются рекордными показателями разгона, чего от них, в общем-то, и не требуется. «Выбор редакции» получают модули Corsair CM2X512A-5400UL за отличные характеристики и стабильную работу как на высоких частотах, так и в более «щадящем» режиме с предельно низкими таймингами.

«Лучшей покупкой» заслуженно признана память Hynix PC2-5300U — этим планкам, приемлемым по цене, подвластны высокие частоты, которые не всегда покоряются и более «оверклокерскими» моделями

Hynix PC2-5300U

hard

СВЕТИТ И ВЕРТИТ!

SUNBEAMTECH IC-TR-B TRANSFORMER

СЕРГЕЙ НИКИТИН,
TEST_LAB (TEST_LAB@GAMELAND.RU)

технические характеристики

ФОРМ-ФАКТОР: **ATX**
КОЛИЧЕСТВО ОТСЕКОВ 5,25", ШТ: **5**
КОЛИЧЕСТВО ОТСЕКОВ 3,5", ШТ: **внешних — 2, внутренних — 6**
КОЛИЧЕСТВО ВЕНТИЛЯТОРОВ, ШТ: **80-мм — 3, 120-мм — 1**
КОЛИЧЕСТВО СВОБОДНЫХ МЕСТ ДЛЯ ВЕНТИЛЯТОРОВ, ШТ: **120-мм — 1**
ДОПОЛНИТЕЛЬНЫЕ ПОРТЫ, ШТ: **USB — 2, mic — 1, phone — 1**
ДОПОЛНИТЕЛЬНО: **10-см лампа в комплекте поставки, стенки корпуса снимаются без инструментов**
ГАБАРИТЫ, ММ: **522x205x450**
ВЕС, КГ: **13**

Test_lab выражает благодарность за предоставленное на тестирование оборудование компании «БЮРОКРАТ» (тел. (495)745-55-11, **www.buro.ru**).

Любой нормальный человек хочет быть хорошо одетым. Так, чтобы одежда была красивой, хорошо сидела на фигуре, была известной марки, показывала хороший вкус владельца. В общем, критериев много. С компьютерным корпусом аналогично. Давно прошли те времена, когда все корпуса были похожи друг на друга и не несли практически никакой функциональной или эстетической нагрузки («Лишь бы был»). Времена изменились, и теперь корпус должен отвечать многим требованиям: стильный внешний вид, вместительность, хорошие возможности по охлаждению, а возможно, и моддинговые фишки. На современном рынке представлено достаточно таких корпусов, производители отвечают на спрос со стороны пользователей. Так что выбирай то, что подходит именно тебе. Сейчас мы рассмотрим корпус, который отвечает всем перечисленным параметрам и станет очень привлекательным вариантом для тех, кто решил переселить внутренности своего компьютера в новый дом.

Внешний вид устройства наверняка понравится большинству потенциальных покупателей. Передняя панель, точнее, дверца, которая прикрывает доступ к отсекам для накопителей, покажется похожей на забрало шлема рыцаря (если у тебя есть некоторое воображение). Весь корпус выкрашен в приятный синий цвет с немногочисленными светлыми вставками. Правая боковая панель сделана прозрачной и стилизована под некий символ вечного движения, что становится особенно явно во время работы всех моддинговых возможностей корпуса. А их, даже базовых, вполне хватит: три светящихся 80-мм вентилятора (два сбоку и один на верхней панели), также 10-см лампа-трубка из комплекта поставки. Когда все это мигает, светится и вращается, получается феерическое зрелище.

Если ты больше озабочен практическим, а не моддинговым применением устройства, то обрати внимание на то, что на задней панели расположен 120-мм вентилятор, а на передней, прямо перед корзиной для HDD (имеются посадочные места для шести таких устройств), есть место для еще одного такого ветрогона. Так что этот трансформер вроде бы не должен создать проблем с вентиляцией и экстерьером. Заканчивая разговор о внешности, упомяну два порта USB, выход для микрофона и наушников, расположенных на верхней панели корпуса.

Для того чтобы начать устанавливать внутрь трансформера комплектующие, сильно напрягаться не придется, так как стенки корпуса закреплены на винтах-барашках и можно снять их не применяя никакие инструменты. Кстати, кроме удобных винтов, правая стенка корпуса оснащена и замочком. Вот мы, наконец, и залезли внутрь. И что же там?

Масса свободного места и отсеков для накопителей: пять штук 5,25, два внешних 3,5 и корзина на шесть HDD. На первый взгляд, для установки оптических приводов и прочих дисководов требуются винты и инструменты, но это не так! Весь инструментарий для удобного крепления (вместе с инструкцией по использованию корпуса) лежит в специальном ящичке. Платы на задней стенке придется закреплять классически — винтиками и отвертками. Ничего не попишешь.

Спорным моментом является отсутствие блока питания. С одной стороны, плохо, что его нет: нужно тратить время на выбор и поиски нужной модели, плюс деньги на ее покупку. С другой стороны, хорошо. Вряд ли человек, который не смыслит в ПК ничего, купит подобный корпус. Остальные же люди относятся к выбору БП достаточно серьезно, обычно им не хватает той модели блока питания, на которой остановился производитель.

Если описать в целом, качественный и функциональный корпус, удобный в использовании и приятный на вид

УЖЕ В ПРОДАЖЕ

Тестирование новейших моделей КПК, ноутбуков и сотовых телефонов

GPS-навигация на Palm OS
История о том, как мы смогли

Ключевые доски для пальцев
Тест клавиатур для КПК

Звонок из прошлого
Обзор эмуляторов игровых консолей для Series 60

Переоформи свой карманник
Пончеры для Palm

Шаг за шагом
- Оптимизация ноутбука по скорости
- «Перепрошивка» КПК своими руками
- Расширяем область Graffiti
- Эмуляция PocketPC на устройствах с Palm OS
- Обновляем прошивку Sony Ericsson P910i
- Настраиваем блокировку клавиатуры для Series 60
- Превращаем смартфон в шпионский девайс
- Работаем с файлами при помощи FExplorer
- Создаем WMA-мелодии для WM2003

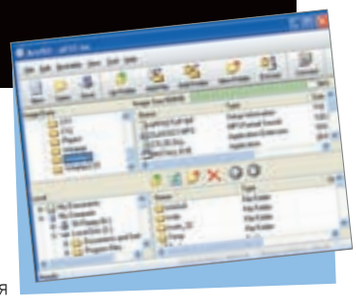
700 МБ ПОЛЕЗНЫХ ПРОГРАММ НА CD

Мобильные компьютеры

Soft

NONAME

НАИСВЕЖАЙШИЕ ПРОГРАММЫ
ОТ NNM.RU | D()C (DOC@NNM.RU)



AceISO 2005 v2.0

AceISO — высокоэффективный и удобный редактор ISO-образов дисков. Ты можешь добавлять, извлекать, создавать образы дисков, понятные всем программам, работающим с дисками. Вдобавок тебе позволяют создавать автозапускаемые образы, чтобы в дальнейшем делать загрузочные диски. AceIso способна и прожигать диски — какая-либо другая программа не понадобится. Софтина поддерживает работу с разными форматами: ISO, BIN/CUE, NRG, IMG/CCD/SUB — комплекта будет вполне достаточно для самого привередливого пользователя.

AceISO пригодится и для создания бэкапов данных, их сохранения в образы и прожига на диски. В довершение списка всех приятных особенностей — возможность делать точные копии дисков: хранишь на винчестере или копируешь, а потом распространяешь.

Systran Professional Premium 5.0 build 443

«Связывайся с партнерами за границей более эффективно, устанавливая деловые контакты с помощью интернета и связывая свои международные офисы доступно и понятно переведенной электронной почтой и сообщениями», — вот так этот переводчик перевел рекламный слоган самого себя. Не могу назвать его самым замечательным в мире, но он действительно обладает достаточно удобным понятным интерфейсом. Плюс ко всему — легкое подключение словарей английского, испанского, французского, немецкого, итальянского и прочих замечательных языков вплоть до китайского и иврита :).

eMule Xtreme 0.47a v5.0

Для тех, кто в танке ;), eMule — это клиент для сети обмена файлами ED2K. Проще говоря, менеджер закачки файлов с компьютеров всех пользователей интернета, которые пользуются программой eMule. 13 мая 2002 года человек по имени Merkur окончательно разочаровался в оригинальном клиенте для сети eDokney2000 и решил, что может сделать что-то свое и гораздо лучше. Собрал команду разработчиков — так родился проект eMule. Ребята поставили себе цель создать новый клиент для сети ED2K (в то время в ней господствовал клиент eDonkey) и дополнить его кучей новых свойств и симпатичным интерфейсом. На сегодняшний день eMule — один из самых больших и надежных клиентов для сетей обмена файлами peer-to-peer в мире. Благодаря принципам открытого кода (open source), в проекте могут участвовать разработчики со всего мира и эффективность сети повышается с каждой новой версией eMule. Так говорят разработчики, и все-таки трудно не согласиться с ними ;).
Но и это еще не все! Еще не сказано ничего о специально прокачанном моде eMule. Одна из основных фич — увеличение скорости скачивания при помощи хитрого алгоритма. Даже одна она заслуживает внимания, не говоря уже об остальных приятных особенностях, которые были привнесены программистами этого мода. Заинтриговал? Тогда ставь ;)!

PasswordsPro v2.0.2.1

Восстановление паролей к MD5, MD4, MySQL, SHA-1 и другим хэшам, хранитель паролей, генератор паролей, генератор словарей, генератор хэшей, восстановление паролей под звездочками... Коротко и ясно.

Advanced Uninstaller Pro 2006 7.5

Advanced Uninstaller Pro создана для качественного удаления программы в операционной системе Windows. Ты сможешь сохранить производительность компьютера на хорошем уровне, так как обычный мастер удаления программ оставляет множество мусора и тем самым способствует замедлению работы компьютера. Кроме того, в программу включено множество полезных утилит для работы с реестром, удаления дубликатов файлов, дефрагментации и многое другое...

Для тех, кому нужно срочно (сборка Mirand'ы)

Маленькая сборка. Ну, очень маленькая. Можно сказать, в ней ничего нет :). И моего, наверное, тоже ничего нет. Иконки от a0x, темплейт для Nicer++ от Faith Healer... Смайлики (правда, не помню, откуда они)... Смотрится примерно так, как выглядит картинка.

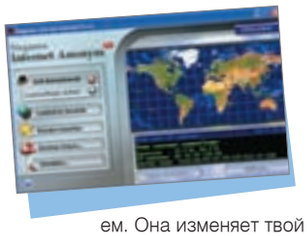
Light Alloy 3.5 Build 5953

Light Alloy — это программа для воспроизведения видео- и звуковых файлов под Windows. Проста в управлении, но содержит множество дополнительных настроек, то есть подходит как новичкам, так и профессионалам. Проигрыватель имеет небольшой размер и оптимизирован для быстрого запуска и минимальной загрузки системы.
Кто еще не поставил себе LA? ;)

FreePromote 1.7

FreePromote — бесплатная программа для раскрутки сайтов и софта путем полуавтоматической регистрации в поисковиках, каталогах, софт-каталогах (FreePromote полностью поддерживает PAD-файлы). С версии 1.6 FreePromote содержит уникальный модуль «Библиотека», в котором хранится огромный список статей и книг (в электронном виде) на тему раскрутки сайтов, электронных платежей, shareware-программирования, коммерческой разработки программ.

Также FreePromote помогает регистрироваться и рассылать сообщения на досках объявлений и форумах.



Steganos Internet Anonym 2006 8.0.1

Если заслуженно или (со мной бывало неоднократно) по ошибке ты был забанен, то данная программа станет твоим спасением. Она изменяет твой IP чуть ли не каждую секунду, благодаря чему ты не только сможешь посещать ресурсы, где тебя забанили, но и повышать безопасность в интернете. Впрочем, можно и прекратить смену IP-адресов, если какой-то один понравится тебе особенно. Эффект будет примерно тот же.

SmartWhois 4.1.191

SmartWhois — это удобное средство получения всей доступной информации о любом IP-адресе, имени компьютера или домене, включая страну, штат или провинцию, город, название компании-провайдера, имя администратора и контактную информацию службы технической поддержки.

В отличие от стандартных whois-утилит, SmartWhois автоматически находит информацию о компьютере независимо от того, в какой части мира он расположен, и предоставляет эту информацию всего лишь за несколько секунд. SmartWhois справится с задачей даже если IP-адресу не сопоставлено имя.

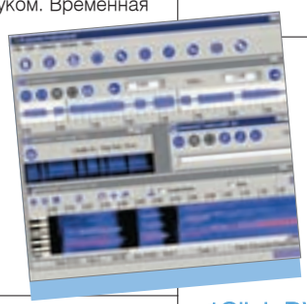
ВОЗМОЖНОСТИ:

- ИНТЕЛЛЕКТУАЛЬНАЯ РАБОТА: ПРОГРАММА ВСЕГДА ОПРАШИВАЕТ ТОЛЬКО НУЖНУЮ БАЗУ ДАННЫХ, И ТЫ НЕ ОБЯЗАН ТРАТИТЬ ВРЕМЯ НА ОПРОС ВСЕХ.
- КЕШИРОВАНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ.
- СОПОСТАВЛЕНИЕ ИМЕН И IP-АДРЕСОВ, КЕШИРОВАНИЕ DNS-ЗАПРОСОВ.
- СОХРАНЕНИЕ РЕЗУЛЬТАТОВ В АРХИВЕ: ТЫ МОЖЕШЬ СОЗДАТЬ СВОЮ БАЗУ ДАННЫХ И ПРОСМАТРИВАТЬ ЕЕ НЕ ПОДКЛЮЧАЯСЬ К СЕТИ.
- ЗАГРУЗКА И ОБРАБОТКА СПИСКОВ IP-АДРЕСОВ ИЛИ ДОМЕНОВ.
- ИНТЕГРАЦИЯ С COMMVIEW: SMARTWHOIS МОЖЕТ БЫТЬ ВЫЗВАН ИЗ COMMVIEW ДЛЯ БЫСТРОГО ПОЛУЧЕНИЯ ИНФОРМАЦИИ.
- ВЫЗОВ SMARTWHOIS ИЗ ТВОЕГО ПРИЛОЖЕНИЯ НАПРЯМУЮ.
- ЗАПРОСЫ ПО МАСКЕ.
- КОНСОЛЬ WHOIS ДЛЯ ПОСТРОЕНИЯ СПЕЦИАЛЬНЫХ ЗАПРОСОВ.
- КОДЫ СТРАН ДЛЯ СПРАВКИ.
- НАСТРАИВАЕМЫЙ ИНТЕРФЕЙС.
- ПОДДЕРЖКА РАБОТЫ ЧЕРЕЗ БРАНДМАУЭР SOCKS5.

...И многое другое.

7Canaries 1.0 Professional

Задача программы — преобразовать музыкальную аудиозапись в нотную, то есть сделать из звукового файла (WAVE-, MP1-, MP2-, MP3-, MPP-, MPA-, AIF-, SND-, AU-, CD-трека) MIDI-файл, содержащий ноты. 7Canaries умеет работать в двух режимах: распознавание записей и распознавание в реальном времени. Для первого необходим исходный файл в одном из поддерживаемых форматов. Он может быть записан либо с микрофона, либо с внешнего источника.
7Canaries также позволяет считывать треки с аудиодисков, но не при всех сочетаниях операционной системы и используемого оборудования. Файл MIDI получается после настройки параметров распознавания и последующего преобразования. Правка возможных ошибок распознавания может быть осуществлена в редакторе VisiNote Editor(tm), включенном в состав 7Canaries Professional.
Нотная запись (например обычная партитура или файл формата MIDI), по сути, является набором команд: какую ноту и каким инструментом следует сыграть. Подобная форма записи легко поддается редактированию и занимает меньше места, чем файлы WAV. Однако не любой звук может быть записан в такой форме. Например, записать человеческую речь в виде нот невозможно. При распознавании в реальном времени «исходные» MIDI-ноты играются одновременно с поступающим звуком. Временная задержка относительно невелика, что позволяет управлять синтезатором с помощью голоса или другого не-MIDI музыкального инструмента. В 7Canaries есть установки и настройки для повышения качества распознавания каждой конкретной композиции. Ты можешь выбирать алгоритмы распознавания и их настройки. Также 7Canaries содержит простой модуль записи звука.



Flash Decompiler 2.6

Flash Decompiler преобразует SWF-файлы в исходники FLA, которые (при соблюдении некоторых условий) можно загрузить в Macromedia Flash для последующего редактирования

Flash Player Pro v.2.8

Flash Player Pro — один из лучших проигрывателей SWF.

Позволяет скачивать flash-мультфильмы из интернета, просматривать их, захватывать изображения и устанавливать их как обои рабочего стола, создавать скринсейверы. Сможет сделать из flash'ки автономный exe-файл.

1Click DVD Ripper 2.03

1Click DVD Ripper 2.03 — небольшая и быстрая программа для легкой конвертации фильма из DVD- в VCD-, SVCD-, XViD- или DIVX-формат. Программа может записать результат своей работы на CD-R/RW-болванку с помощью встроенного модуля записи дисков. Для самых ленивых предусмотрен помощник, который поможет настроить все аспекты конвертации фильма пошаговыми подсказками.

ADMINING: ДЛЯ ДРУГИХ МЫ СОЗДАЕМ ПРАВИЛА, ДЛЯ СЕБЯ — ИСКЛЮЧЕНИЯ (ШАРЛЬ ЛЕМЕЛЬ)

КОГДА ДОСТОПОЧТЕННЫЙ ШАРЛЬ ЛЕМЕЛЬ ПРОИЗВЕЛ НА СВЕТ СВОЙ АФОРИЗМ, ВРЯД ЛИ ОН ДУМАЛ, ЧТО ЕГО ПРОИЗВЕДЕНИЕ ЛУЧШЕ ВСЕГО ОТРАЗИТ СУЩНОСТЬ И СМЫСЛ СУЩЕСТВОВАНИЯ СИСТЕМНОГО АДМИНИСТРАТОРА, ВЕЛИКОГО И МОГУЧЕГО. СЕГОДНЯ МЫ ПОГОВОРИМ О ЖИЗНЕННЫХ ДЛЯ ЛЮБОГО АДМИНА ВЕЩАХ — О ПОЛИТИКАХ БЕЗОПАСНОСТИ. ПОЛИТИКАМ БЕЗОПАСНОСТИ, НАСТРОЙКЕ ПОЛИТИК И ДРУГИМ ИНТЕРЕСНЫМ ВЕЩАМ ПОСВЯЩЕНО МОРЕ КНИГ, СОЗДАНО МНОЖЕСТВО ФОРУМОВ. МЫ ЖЕ БУДЕМ РАССМАТРИВАТЬ ЭТУ СЛОЖНУЮ НАУКУ В ПЕРВОМ ПРИБЛИЖЕНИИ |АЛЕКСАНДР ПРИХОДЬКО (SANPRIH@MAIL.RU)

понятие доменной политики безопасности Для начала вспомним, что мы уже было сотворено с нашим доменом. Как ты знаешь, было создано несколько пользователей, групп, продумана примерная структура домена.

Повторим пройденный материал. Для начала посмотрим на список пользователей и групп домена, который уже есть.

Значит, так. Разбросаем пользователей по группам. Открыть оснастку Active Directory Users and Computers, выбрать группу Head, двойной клик мыши на ней, перейти на закладку Members, надавить на кнопку Add, в открывшемся окне — Select Users, Contacts or Computers, нажать кнопку Advanced... В следующем окне — кнопку Find Now. В открывшемся списке выбрать наших пользователей этой группы: «Иванов», «Петров», «Сидоров». Так же жестоко поступаем и с группами «Экономисты» и «Бухгалтерия».

В принципе, ты проделывал это все уже не раз, поэтому не будем вдаваться в подробности, но… Маленькое отступление. Во втором модуле мы чуть рассмотрели теневое копирование. Возвращаясь к этой теме, поговорим о стратегии разработки правил защиты информации в работе домена (на предприятии любого уровня). Я не беру в расчет доменные печи (не те печи, в которых сжигают контроллеры доменов, а те, в которых плавят металл). Так вот, наверное, в работе с доменными печами потеря информации не играет важной роли. Однако нам с тобой деньги платят как раз за сохранность информации. В этом вопросе нет никакой надежды на пользователей, поэтому такой тяжкий труд и возложен на наши плечи.

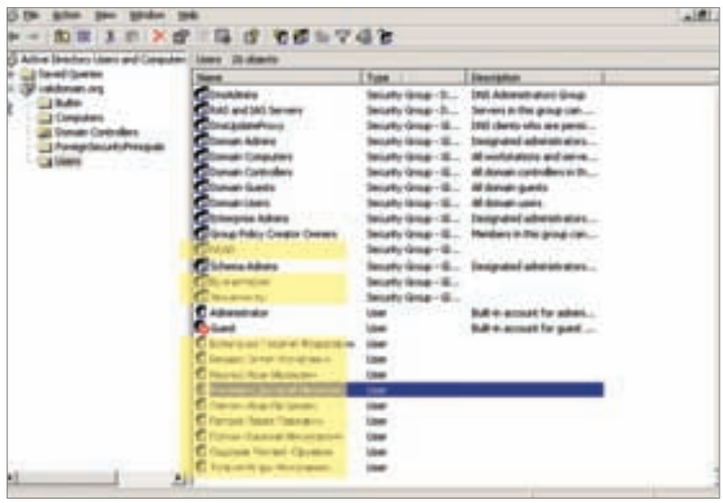
Тебе, как админу, придется взять на себя полную ответственность за сохранность всей информации, не полагаясь на сознательность и компетентность подопечных пользователей. Для обеспечения достаточно серьезной защиты информации необходимо следующее.

Во-первых, полностью обезопасить домен и его информационные ресурсы от всяких некорректных действий любых пользователей. Во-вторых, сделать компьютеры максимально устойчивыми к любым разрушительным действиям со стороны тех, кто работает на этих компьютерах.

Пользователи как несмышленые дети — нужно защищать их от них же. Разобьем эту большую и сложную задачу на маленькие и простые:

- ГРАМОТНОЕ ПОСТРОЕНИЕ ПОЛИТИКИ РАЗГРАНИЧЕНИЯ ПРАВ ДОСТУПА ЛЮБЫМИ СРЕДСТВАМИ, ИМЕЮЩИМИСЯ В РАСПОРЯЖЕНИИ, НА ВСЕХ ИНФОРМАЦИОННЫХ РЕСУРСАХ;
- ОГРАНИЧЕНИЕ ПРАВ ПОЛЬЗОВАТЕЛЕЙ ПОЛИТИКОЙ БЕЗОПАСНОСТИ;
- ПРИНУДИТЕЛЬНОЕ АРХИВИРОВАНИЕ ИНФОРМАЦИИ;
- АВТОМАТИЧЕСКОЕ ПОДКЛЮЧЕНИЕ-ОТКЛЮЧЕНИЕ ИНФОРМАЦИОННЫХ РЕСУРСОВ ПРИ ПОМОЩИ СКРИПТОВ;
- МАКСИМАЛЬНАЯ ЗАЩИТА ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ ОТ УСТАНОВКИ ПРОГРАММ ПОЛЬЗОВАТЕЛЯМИ;
- ЗАЩИТА ОТ ВТОРЖЕНИЯ ЛОКАЛЬНЫМИ БРАНДМАУЭРАМИ;
- ЗАЩИТА АНТИВИРУСНЫМ ПО;
- ЗАЩИТА ОТ ПРОГРАММ РАЗРЯДА SPYWARE, ADWARE.

Как подсказывает жизненный опыт и если не учитывать редкие исключения, пользователи делятся на три группы: продвинутые, обыкновенные и враги. Я назвал врагами тех, кто плевать хотел и на твою админскую работу, и на политику безопасности, и даже на компьютер, за которым работает. К примеру, один пользователь, подопечный мне, постоянно ходил на сайты, зараженные разными вирусами. На его компьютере ругались антивирус, локаль-



Пользователи и группы домена

ный файрвол и я, но пользователю все было до лампочки: «Вот есть компьютерщики. Что происходит с сетью и компьютером — это только их головная боль. Мое дело — лазить где хочу и делать что хочу». Как раз таких пользователей я называю врагами. Хуже всего, если враг занимает какую-нибудь должность на предприятии. Однако ты админ, и в твоих руках находится достаточно рычагов, чтобы, оставаясь белым и пушистым, смочь обрубить врагу руки по самые помидоры.

Теперь начинается самое интересное. Если ты еще не забыл, мы только что создали группы и занесли в них пользователей. Группы нужны для установки разрешений на информационные ресурсы сервера. В первом модуле я рассказал о том, что лучше раздавать все разрешения на ресурсы через группы — так ты избежишь потерь SID'ов. Что-то похожее на группы есть и в политике безопасности, только оно называется Organizational Unit и прячется вот здесь: Start→ Programs→ Administrative Tools→ Active Directory Users and Computers. По умолчанию доступ к созданию Organizational Unit отключен. Чтобы включить его, необходимо отметить галочкой пункт Advanced Features в меню View.

Теперь создаем три OU. Первое — для хороших друзей админа и благонадежных пользователей, эти люди не будут ограничены практически ни в чем. Второе OU — для простых пользователей. Подрежем некоторые возможности, чтобы админу жилось спокойнее. И третье — OU для врагов. Тутто и развернемся.

Поехали. Правая кнопка мыши на имени домена — New→ Organizational Unit. Названия своим OU придумай сам, чтобы запомнить, где и кто лежит. Для удобства я назову OU прямо соответственно видам пользователей — «Друзья», «Пользователи», «Враги». Особо не вдаваясь с теорию, скажу, что OU могут иметь иерархическую структуру: в OU могут находиться вложенные OU и т.д.

Кратко напомню о политиках, действующих в рамках домена:

- ЛОКАЛЬНАЯ ПОЛИТИКА БЕЗОПАСНОСТИ — СУЩЕСТВУЕТ НА ВСЕХ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРАХ, ВКЛЮЧАЯ КОМПЬЮТЕР, НА КОТОРОМ КРУТИТСЯ КОНТРОЛЛЕР ДОМЕНА.
- ДОМЕННАЯ ПОЛИТИКА — ПОЛИТИКА, ДЕЙСТВУЮЩАЯ В ДОМЕНЕ, РАСПРОСТРАНЯЕТСЯ НА ВСЕ КОМПЬЮТЕРЫ, ЗАРЕГИСТРИРОВАННЫЕ В ДОМЕНЕ.
- ПОЛИТИКА КОНТРОЛЛЕРА ДОМЕНА — ОБЛАСТЬЮ ЕЕ ДЕЙСТВИЯ ЯВЛЯЕТСЯ КОНТРОЛЛЕР ДОМЕНА.
- ПОЛИТИКА САЙТА — РАСПРОСТРАНЯЕТСЯ НА ВСЕ ДОМЕНЫ, ВХОДЯЩИЕ В САЙТ.

Главное — помни, что если используется политика безопасности, локальный компьютер пользователя будет подчиняться правилу LSDOU. Сейчас поясню. Сначала вступает в действие самая слабая политика — локальная (L). Затем начинает работать политика сайта (S), дальше — доменная политика (D), после чего, наконец, самая сильная политика — политика Organizational Unit (OU).

Политики действуют по принципу, согласно которому если слабая политика запрещает, а более сильная ничего не говорит по этому поводу, то накладывается запрет. Если слабая разрешает, а более сильная запрещает — накладывается запрет. Если слабая запрещает, а более сильная разрешает, выдается разрешение. Если политики безопасности применяются неправильно, может случиться и такое, что даже с правами админа ты не попадешь на свой контроллер домена. Будь осторожен как сапер!

Я предлагаю такой вариант назначения политик: политикой домена (D) устанавливаются правила, которые будут распространяться на пользователей и компьютеры домена, причем на все без исключения. Посмотрим, к примеру, на политику паролей. Политику сайта (S) и локальную (L) не трогаем совсем. Политикой OU как раз настраиваем все, что нужно.

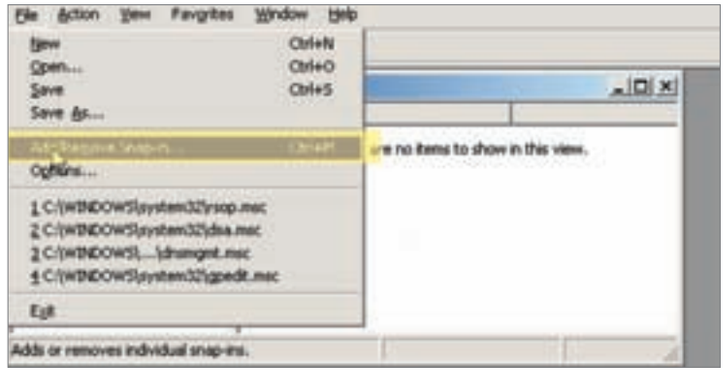
С сайта **www.microsoft.com** рекомендую скачать программу Microsoft Group Policy Management Console (gpmc.msi). Она упрощает работу с политиками, делает ее более наглядной. Чтобы получить доступ к инструменту, предназначенному для работы с политикой, делаем следующее. Вызываем оснастку mmc: Start→ Run — набираем в командной строке mmc. Получаем консоль управления.

Пока она чистая. Добавим необходимые нам инструменты: File→ Add/Remove Snap-in. В открывшемся окошке нажимаем кнопку Add и выбираем Group Policy Management.

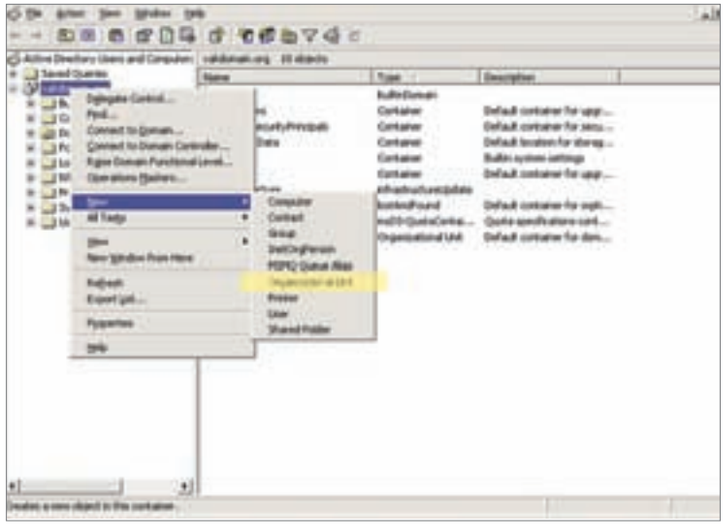
Заодно можешь добавить Group Policy Object Editor. Кстати, ты заметил, сколько еще инструментов прячется здесь? Вот оно — поле для творчества!

Теперь можешь сохранить консоль на рабочий стол — пригодится. Открываем консоль, разворачиваем все списки и смотрим, что там есть.

Групповые политики воздействуют на пользователей по вот такому механизму: берешь любого пользователя, Start→ Programs→ Administrative Tools→ Active Directory Users and Computers, перетаскиваешь пользователя мышью в нужный OU. Политика для него обновится через некоторое время или при его следующем входе в домен.



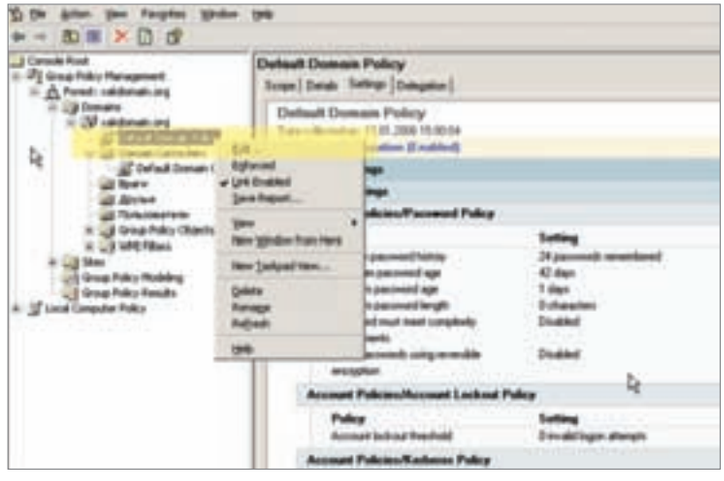
Добавление оснастки



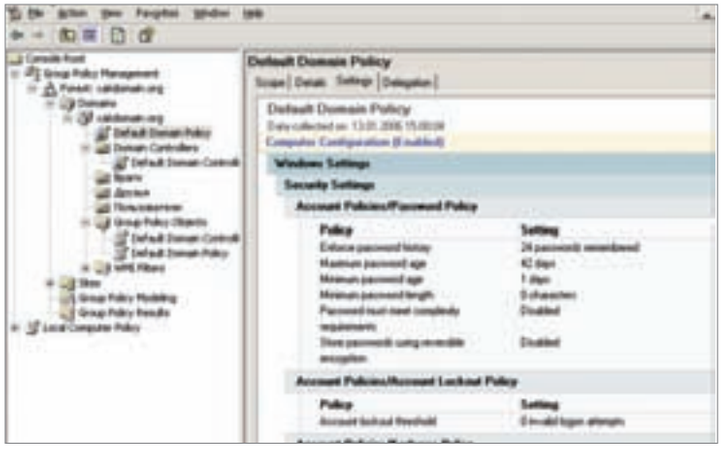
Создание OU

Чтобы изменения в политике безопасности начинали применяться мгновенно, существует команда «gpupdate /force». Start→ Run→ cmd — и в командной строке набираешь команду.

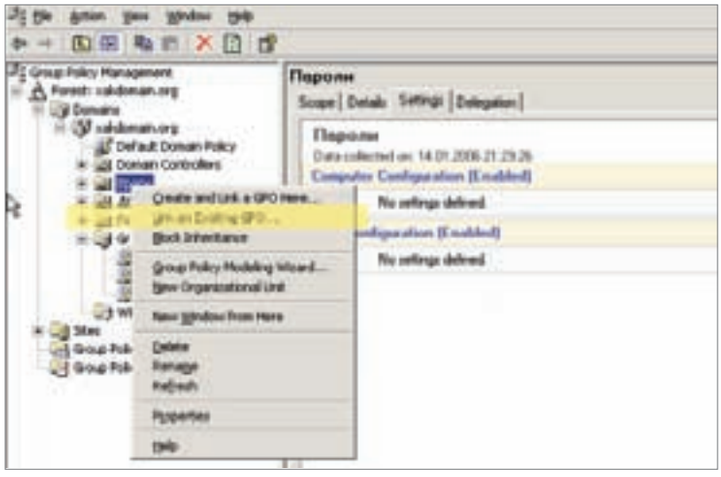
Лирическое отступление. Когда пользователь входит в домен, компьютер считывает с контроллера домена политику для компьютера и только затем — для самого пользователя. Следовательно, чем больше OU вложены друг в друга и чем медленнее работает сеть, тем больше время, кото-



Редактирование доменной групповой политики



Вид доменной групповой политики



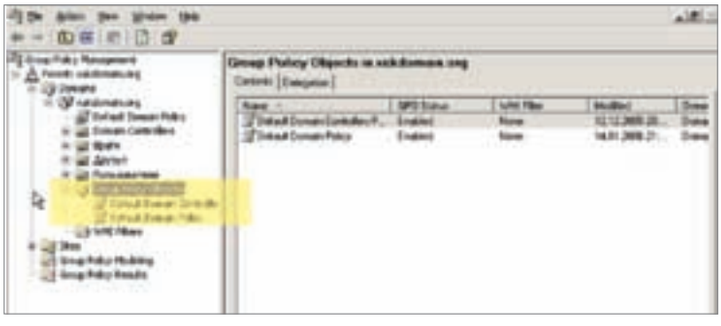
Привязывание объекта GPO «Пароли»

рое пользователь будет тратить на вход в свою машину. Если ты не пользуешься политикой безопасности для компьютера (Computer Configuration), то лучше отключить ее. То же самое касается пользователя (User Configuration). Если говорить в общем, политика безопасности — довольно объемная и интересная тема, по ней написано много книг. Эта статья слишком маленькая, чтобы я успел рассказать о политике все. Я только указываю направление.

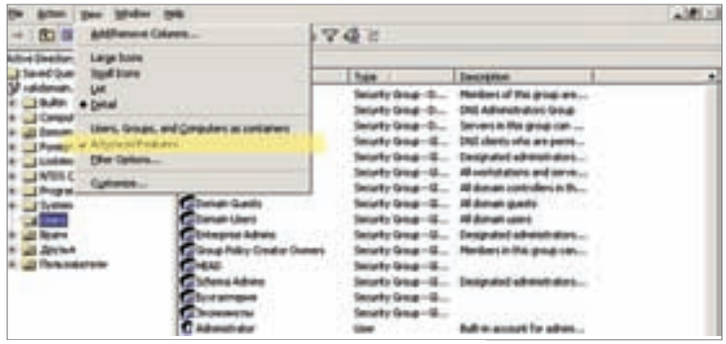
Для начала разрешим всем пользователям простые пароли длиной не менее четырех символов. Это правило распространяется на абсолютно всех пользователей нашего домена, поэтому будем воздействовать на них политикой домена — Default Domain Policy.

Запускаем консоль. Если ты установил gpmc.msi, то увидишь такую же картину, какая наблюдается на рисунке. Я выделил контейнер доменной групповой политики. На нем правой кнопкой мыши выбираем пункт меню Edit.

Теперь попадаем в Group Policy Object Editor. Открываем Computer Configuration→ Windows Setting→ Security Setting→ Account Policies→ Password



Объекты GPO



Включение доступа к дополнительным функциям

Policy. Ставим длину пароля в четыре символа. Двойной щелчок на Minimum Password Length. Если хочешь закрутить гайки на пароли не по-детски, то проделываешь следующее.

Enforce password history — отвечает за то, сколько неповторяющихся паролей будет помнить контроллер домена. Если поставишь число 30, пользователь окажется вынужден придумать 30 разных паролей, прежде чем сможет возвратиться к первому.

Maximum password age — отвечает за срок жизни пароля. Если в свойствах пользователя поставишь (закладка Account) Password never expires, то эта ветка политики автоматом отключится. Если наоборот, то за несколько дней до истечения срока действия при логине пользователю будет выдано предупреждение о том, чтобы пароль изменили. Если выставить значение и активировать Enforce password history, то пользователь не сможет ввести старый пароль.

Password must meet complexity requirements отвечает за сложность пароля. При активации этой политики пароль должен будет содержать строчные и прописные буквы, символы типа «@», цифры. Получается, что если ты хочешь быть застрелен своими сослуживцами через пару месяцев, делаешь следующее:

- ENFORCE PASSWORD HISTORY — 24.
- MAXIMUM PASSWORD AGE — 5.
- MINIMUM PASSWORD LENGTH — 8.
- PASSWORD MUST MEET COMPLEXITY REQUIREMENTS — АКТИВИРОВАТЬ.

Затем в свойствах пользователя (закладка Account) очистить поля User cannot change password и Password never expires.

Смерть тебе гарантирована. Если ты страдаешь манией преследования, в ветке Account Lockout Policy активируй подветку Account Lockout threshold. Выставленное здесь число обозначает количество неудачных наборов пароля/имени, после которых учетная запись заблокируется. Автоматически будут активированы и временные пороги, в пределах которых учетная запись останется заблокированной.

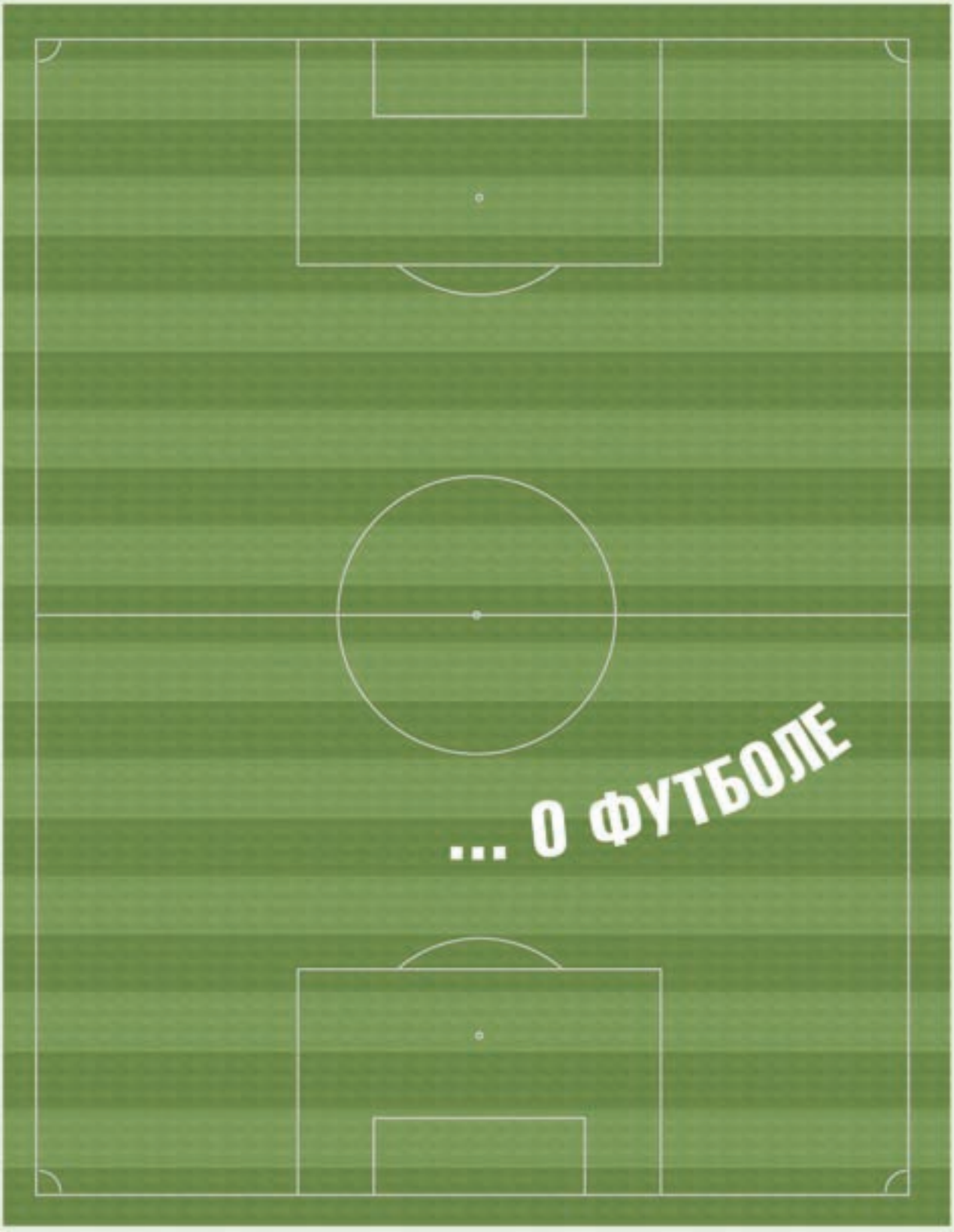
Напоследок и прежде чем перейти к тонкой настройке политик, к любому OU можно привязать множество объектов GPO (Group Policy Object). Сначала спокойно создаешь нужные тебе GPO, а потом еще спокойнее привязываешь их к OU. Где лежат все объекты GPO, отлично видно на рисунке.

Чтобы создался новый объект групповой политики, правая кнопка мыши прижимается к Group Policy Object.

Пожелание: когда создаешь GPO, наделяй их внятными подписями, чтобы по названиям было понятно, на какое действие направлена данная политика. Например, создадим GPO, отвечающее за пароли, — вот и назовем его «Пароли».

Объект GPO «Пароли» сейчас полностью чистый. Чтобы отредактировать его, щелкни по нему правой кнопкой крысы и выбери пункт меню Edit. Для того чтобы привязать созданный объект GPO к существующему OU, на нужном OU щелкаем правой кнопкой мыши и выбираем пункт меню Link an Existing GPO.

На сегодня все. В следующий раз продолжим, создадим GPO для врагов и посмотрим на него в действии. Желаю удачи и терпения в борьбе с пользователями. Будь снисходителен к человеческим слабостям и помни, что пользователь не может знать столько же, сколько ты. Будь добрым и любвеобильным, не отравляй жизнь людям по пустякам, и да воздастся тебе за доброту


... О ФУТБОЛЕ

TOTAL FOOTBALL

НОВЫЙ ЖУРНАЛ О ФУТБОЛЕ ...С DVD

В КАЖДОМ НОМЕРЕ DVD С ЛУЧШИМ ФУТБОЛЬНЫМ КОНТЕНТОМ

В АПРЕЛЬСКОМ НОМЕРЕ:

ЭКСКЛЮЗИВ
Звезда «Зенита» Андрей Аршавин

ТЕМА НОМЕРА
Новые стратеги. Пятерка лучших тренеров нового поколения

АНГЛИЙСКИЕ ФАНАТЫ
Страстная история лютой ненависти

ДАЕМ ПРОГНОЗ
Чем закончится недавно начавшийся чемпионат России

ДЕТСТВО СЫЧЕВА
Как воспитывали нападающего «Локомотива»

ФУТБОЛЬНЫЙ МЕНЕДЖЕР
Главный приз – поездка финал Лиги чемпионов!

сэкономь деньги — закажи журнал в редакции

ВЫГОДА

Цена подписки до 15% ниже, чем в розничной продаже
Бонусы, призы и подарки для подписчиков
Доставка за счет редакции

ГАРАНТИЯ

Ты гарантированно получишь все номера журнала
Единая цена по всей России

СЕРВИС

Заказ удобно оплатить через любое отделение банка
доставка осуществляется заказной бандеролью или курьером

КАК ОФОРМИТЬ ЗАКАЗ

1 Заполнить купон и квитанцию
2 Перечислить стоимость подписки через любой банк.
3 Обязательно прислать в редакцию копию оплаченной квитанции с четко заполненным купоном любым из перечисленных способов:

— по электронной почте: **subscribe@glc.ru**;
— по факсу: (495) 780-88-24;
— по адресу: 119021, Москва, ул. Тимура Фрунзе, д. 11, стр. 44-45, ООО «Гейм Лэнд», отдел подписки.

Внимание!
Подписка оформляется в день обработки купона и квитанции.

— купоны, отправленные по факсу или электронной почте, обрабатываются в течение 5 рабочих дней.
— купоны, отправленные почтой на адрес редакции обрабатываются в течение 20 дней.

Рекомендуем использовать электронную почту или факс. Подписка производится с номера, выходящего через один календарный месяц после оплаты. Например, если произвести оплату в сентябре, то подписку можно оформить с ноября.

ПОДПИСКА ДЛЯ ЮРИДИЧЕСКИХ ЛИЦ

Москва: ООО «ИНТЕР-ПОЧТА» (495) 500-00-60 www.interpochta.ru

Для получения счета на оплату подписки нужно прислать заявку с названием журнала, периодом подписки, банковскими реквизитами, юридическим и почтовым адресом, телефоном и фамилией ответственного лица за подписку.

подписной купон

стоимость заказа на Хакер Спец + CD

6 месяцев — 900 руб. 00 коп. | **12** месяцев — 1740 руб. 00 коп.

стоимость заказа на комплект Хакер Спец + Хакер + Железо

6 месяцев — 2550 руб. 00 коп. | **12** месяцев — 5040 руб. 00 коп.

прошу оформить подписку:
☐ на журнал Хакер Спец + CD
☐ на комплект Хакер Спец + Хакер + Железо
на месяцев
начиная с 200 г.
☐ Доставлять журнал по почте на домашний адрес
☐ Доставлять журнал курьером на адрес офиса (по г. Москве)
Подробнее о курьерской доставке читайте ниже*
(отметьте квадрат выбранного варианта подписки)
Ф.И.О.

дата рождения
адрес доставки:
индекс
область/край
город
улица
дом корпус
квартира/офис
телефон ( )
e-mail
сумма оплаты

*Курьерская доставка осуществляется только по Москве на адрес офиса. Для оформления доставки курьером укажите адрес и название фирмы в подписном купоне.

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
ЗАО ММБ
р/с № 40702810700010298407
к/с № 30101810300000000545
БИК 044525545 КПП - 772901001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 200_ г.	

Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ЗАО ММБ
р/с № 40702810700010298407
к/с № 30101810300000000545
БИК 044525545 КПП - 772901001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 200_ г.	

Ф.И.О.
Подпись плательщика

Кассир

ПО ВСЕМ ВОПРОСАМ, СВЯЗАННЫМ С ПОДПИСКОЙ, ЗВОНИТЕ ПО БЕСПЛАТНЫМ ТЕЛЕФОНАМ: **780-88-29** (ДЛЯ МОСКВИЧЕЙ) И **8-800-200-3-999** (ДЛЯ РЕГИОНОВ И АБОНЕНТОВ МТС, БИЛАЙН, МЕГАФОН). ВСЕ ВОПРОСЫ ПО ПОДПИСКЕ МОЖНО ПРИСЫЛАТЬ НА АДРЕС: **info@glc.ru**

crew

Е-МЫЛО

ПИШИТЕ ПИСЬМА!
SPEC@REAL.XAKEP.RU | НА ПИСЬМА ОТВЕЧАЛ SKYWRITER

От: profep@yandex.ru
Тема: Помогите!
Здравствуйте, господа хакеры!
Пишет вам профессор одного из столичных вузов. Фамилию свою называть не буду — коллеги никогда не простят. У меня к вам такой вопрос. Я слышал, что существует программа «Зеркало», она отображает на экране монитора все, что находится перед ним. Я хотел бы узнать, где можно скачать саму программу (а еще лучше — исходники, желательно на Visual Basic 6.0).
Мне очень нравится ваш журнал! Спасибо.

Ответ: Здравствуйте, господин профессор!
Будучи студентом, я было затаил на автора этого письма давно народившуюся злобу. Но как только я узнал, что он является еще и нашим читателем (так сказать, «по ту сторону баррикад»), я расслабился, заулыбался и воспылал желанием помочь человеку (вот-вот, уже господа коллеги вырывают у меня клавиатуру :).

Программа «Зеркало», безусловно, существует. Самый простой ее вариант — это установка зеркала напротив монитора. В этом случае вышеуказанная программа автоматически устанавливается в системе, не требуя никаких дополнительных действий. На экране сразу начинает отображаться то, что находится перед ним. В случае отсутствия зеркала обыкновенного ситуация усложняется, для ее решения потребуется покупка самой настоящей web-камеры. В комплект поставки этой web-камеры обязательно входит и «Зеркало», которое сразу после установки камеры на монитор выведет Ваше собственное отражение! Удачи!

От: TheMellon@yandex.ru
Тема: Дизайн][акера
Доброго времени суток, господа хакеры! Пишу Вам по поводу дизайна журнала][акер и][акер Спец. Извиняйте, конечно, но, по-моему, это не дизайн, а его отсутствие. Плиз, верните старый добрый дизайн или разработайте новый, но иной, нежели такой. Взять хотя бы мартовский Спец Game coding — обложка полностью черная. Ну неужели на такую тему нельзя выбрать картинки? Да и ВАШ логотип сильно измельчал по сравнению со старым дизайном. Логотип-то за что так? Содержание стало только хуже. Извините, конечно, за критику, но на этот новый дизайн смотреть не могу. Хочу, чтоб][акер был ЛУЧШИЙ компьютерный журнал. Желаю удачи!!! Пока.

Ответ: Здравствуй, товарищ Дыня!
Хотел было написать поэму о дизайне в ответ на твое письмо, но, протужившись битый час, бросил это занятие — нет во мне творческого и поэтического дара. Ну да бог с ним. Давайте лучше о дизайне :). Переделка дизайна намечалась довольно давно. Мы (особенно наши уважаемые арт-директора) долго думали над возможными решениями: был вариант без текста, но он почему-то не понравился — уж очень однообразно, тем более что хочется аккуратно размять лист в руке и... использовать по назначению. В общем, что греха таить? Остановились на варианте, при котором используется много «картинок», чтобы глаз радовался больше, чем остальные части тела.

Если серьезно, дизайн был сменен в том числе для того, чтобы в более доступной и точной форме доносить до тебя, читатель, информацию. Так что радуйся новой репрезентативной форме наших статей :), а если есть какие-то баги, то мы их обязательно устраним. И ничто не сможет преградить наш с тобой совместный путь к торжеству добра и света!
С любовью, твоя команда.

От: Sysadmin [sysadmin@altair-tv.ru]
Тема: Дизайн
Привет, редакция журнала Хакер.
В последнее время меня очень не радует ваш новый дизайн. Картинки размером со страницу явно не подходят к теме журнала.

Это место вы могли бы использовать в других целях, например, отдать под рекламу, увеличив свои доходы :). А еще хочу открыть вам страшную тайну: иллюстрации к теме Story рисовал растаман с десятилетним (как минимум) стажем. Я сам (тоже растаман, но это неважно) иногда вижу точно такие же рисунки на дне унитаза. Поэтому советую сделать более цивильные картинки, пока вас не привлекли за пропаганду конопли.
З.Ы. Буду покупать ваш журнал всегда! Недавно скурил номер «Мобильных компьютеров» — совсем не те ощущения :(. Теперь я курю только][.

Ответ: Привет, сисадмин альтаирского телевидения! Как погода в далеких разумных мирах? Нормально? Ну и славно!

Не буду повторять высказанное в одном из ответов на хулу в адрес дизайна, лишь расширю и углублю ;). В частности, о доходах от рекламы: более внимательный читатель заметил бы в картинках, имеющих размер с полосу, рекламу, скрытую от невооруженного глаза. Дело в том, что мы, пользуясь технологией 3D-картинок (помнишь, были такие книжки? Там сразу все не видно, а если как-то по-хитрому сломать глаза, то познаешь истину), разместили в журнале вдвое больше рекламы. Теперь она не только на полосах, заполненных картинками, но даже в тексте — присмотрись повнимательнее, разве не видишь?

Кстати об истине... По-моему, картинки рисовал не растаман, а растаманша, но это не суть важно, да? *покашлял от выпускаемого Аваланчем дыма и растянулся в блаженной улыбке* ... В общем... ммм... тема... Да? Кстати, самокрутки из журнала особенно цепляют. С любовью, твои «зеленые» растаманы.

ОТ: Anna Stepchenko [anna17@ukr.net]
ТЕМА: ADMid-pkg
Здравствуйте, уважаемая редакция журнала.
Мне нужна Ваша консультация.
Я скачала набор утилит ADMid-pkg и не знаю, как и чем их скомпилировать.

цитата:
«Как заставить компьютер соседа поверить, что microsoft.com имеет твой IP? Очень просто: качаем с **http://adm.freelsd.net/ADM** набор утилит ADMid-pkg.tgz, компилируем и запускаем:

./ADMsniffID <интерфейс> <желаемый IP> <нужное имя хоста> <тип записи>

(тип записи 1 = TYPE A(имя в IP) или 12 = TYPE PTR(IP в имя)). Вот так мы заставим подумать компьютер Васи Пупкина, что microsoft.com имеет IP-адрес 192.168.6.66, когда он попытается запросить это имя:

./ADMsniffID eth0 192.168.6.66 microsoft.com 1»

В архиве только текстовые файлы ... Я нашел нужный файл, но не знаю, как его скомпилировать.
Заранее благодарна.

ОТВЕТ: Здравствуй, существо непонятного пола по имени Анюта!
Совершенно по секрету сообщу тебе, что эти непонятные текстовые файлы — не простые тексты, а волшебные. Если под какой-нибудь продвинутой ОСью натравить на них «CC» (не путать с Carbon Copy), то будет тебе счастье (при отсутствии семантических и синтаксических ошибок, конечно). Но никогда не забывай, что обманывать прибегая к спуфингу нехорошо...
Заранее благодарны.

ОТ: aceq [aceq.org.ru@rambler.ru]
ТЕМА: вопрос в хакер насчет шифрования
Привет, Спец. Есть ли программка для шифрования, которая при частичном повреждении зашифрованных файлов, томов могла бы хоть часть информации восстановить? Ну, например, размазывала бы по всему файлу дублированную информацию насчет структуры папок и т.д. А то шифроваться хочется, а толку-то? Если с 10 гигов ни... не восстановишь ничего из-за пары бэд секторов.
Так же хотелось бы сказать, что дизайн и содержание Хакера и Спеца радует. На мой взгляд, вы на голову выше других комп. изд. для народных масс. Очень качественный продукт.

ОТВЕТ: Нихао, Асек!
Любая программа для шифрования томов обычно работает посекторно, так что проблема потери данных для тебя настолько же актуальна и при наличии шифрования, и при его отсутствии: пропал сектор, значит, пропали данные с него. Если перефразируем твой вопрос, получим: есть ли прога, которая размазывает данные по винту в целях сохранения на случай краха системы/жестака? Да, есть. Для этого кури номер «Backup и резервирование» твоих покорных слуг :). То же касается и файлов: единственное, чего ты добьешься такой прогой, — это увеличение размера.
В общем, для обобщения скажу, что от потери данных тебе поможет только Его Величество backup — и неважно, шифруешь ли ты то, что бэкапишь.
Любви, добра и понимания!
P.S. Очень тронуты похвалой дизайна, едва ли не единственной за последнее время :(…

ОТ: legion-de@yandex.ru
ТЕМА: вопрос
Добрый день!
Скажите, пожалуйста, где можно купить старые номера Хакер Спец с диском, в частности номера за ноябрь и декабрь 2005 года?
Спасибо.

ОТВЕТ: Добрый день!
У букинистов. Спасибо.

ОТ: ice_stRANnick [ice_stRANnick@mail.ru]
ТЕМА: Ложка дегтя
Hi, spec'ы! Держу в руках свежий номер Спеца 02(63) февраль 2006, «Тюнинг». Пишу вам впервые. Причина, побудившая написать, будет следовать сразу за хвалебной частью письма (ибо так положено. Сначала бочки меда, затем ложки дегтя), к которой и перехожу...

Вообще, парни, вы молодцы ;). Спец читаю с момента его появления и буду вашим верным читателем по крайней мере до тех пор, пока выходят выпуски вроде спецов по вирусам, [анти]крэкингу, секъюрити-фокусам... Некоторые спецы, вроде мобильных денег, баз данных и т.д., нагло пропускаю (в плане, не покупаю) ввиду отсутствия прямого интереса к теме, но при наличии возможности беру почитать у друзей :) Интересующие спецы начинаю читать прямо у киоска, где журнал был куплен, а заканчиваю... А не заканчиваю вообще, потому что инфа интересная, нужная, обширная (хотя и не всегда - обидно), и одним прочтением обычно все не заканчивается... Так держать!
Ложка дегтя... Как много в этом звуке... О чем это я? А-а-а! Так вот же тот самый Спец, который стал причиной этого письма. Купил Спец по тюнингу и думал: щас приду, почитаю, че-нить новое узнаю и тут же в рил-тайме все разгоню... И не разогнал... Дело в том, что журнал меня побудило купить именно слово «разгон» в подзаголовке. И я почти ничего не нашел, две статьи тока - про видюхи и «Паяльник». Все. Ну ладно, думаю, все-таки основным заголовком был «Тюнинг», сам виноват. Лениво перелистываю, привлекла внимание статья про Вин ПЕ. Давно хотел сделать для себя подобный дистриб (в хозяйстве пригодится :), щас соберу. Вот тока конструктора для данной цели не имелось под рукой :(.

Думаю, вещь-то нужная, по-любому выложили на диск. Ищу взглядом слова и нахожу их: «ВСЕ необходимые программы и инструменты ты сможешь найти на нашем диске». Диск — в сидюк. Ищу. И не нашел... Минута недоумения (или негодования), клик по иконке с вампиром, и, вуаля, вы читаете эти слова. Спасибо хоть за линк, не буду тормошить гугля.

Вот, в общем, и все. В принципе, это минутный порыв. Выйдет очередной Спец с какой-нить хак-темой, и снова душа будет петь дифирамбы в ваш адрес (мысленно, а может быть и нет). А пока там только горстка неприятного осадка...

Не подводите, парни! С надеждой жду интересных номеров. Несмотря ни на что, ваш преданный читатель ice_stRANnick.

ОТВЕТ: Алоха, ледяной стРАНник! Опять же, весьма и весьма тронуты похвалой. Жаль, что ты нагло пропускаешь некоторые выпуски. В некоторых на первый взгляд тоскливых номерах может оказаться целая уйма полезной информации — по себе знаю ;). Так что очень советую не пропускать, а заодно собирать объемную подшивку — со временем она станет предметом твоей гордости!

Знаешь, с WinPE история оказалась более чем грустной. Мы уже собирались выкладывать его на диск, как вдруг нас поразила ужасная мысль, будто молния: «Это же противозаконно, ибо неизбежно пришлось бы выкладывать компоненты ОС Windows XP, что грозит нам тюрьмой...»

Story

иллюстрации: Анна Журко

ВВЕРХ-ВНИЗ...

ДВЕРЬ ОТКРЫЛИ БЫСТРО — СПУСТЯ ПАРУ СЕКУНД ПОСЛЕ ЗВОНКА. ОТКРЫЛИ БУДТО И НЕ ЗАДУМЫВАЯСЬ О ТОМ, КТО ЖЕ СТОИТ НА ПЛОЩАДКЕ. ПРОСТО ЗАМОК ЩЕЛКНУЛ, И ХОЗЯИН СНОВА ВЕРНУЛСЯ К СВОИМ ДЕЛАМ. ВОШЕЛ, ОГЛЯДЕЛСЯ. ПОСТАВИЛ У ДВЕРИ СУМКУ. ИЗ ГЛУБИНЫ КВАРТИРЫ РАЗДАЛСЯ ПРИВЕТЛИВЫЙ ГОЛОС |NIRO (NIRO@REAL.XAKEP.RU)

— Заходи, не стой на пороге! Извини, не встречаю. Занят до чертиков! Там вешалка справа, куртку скинь и проходи. Пиво будешь?

— Нет.

Куртку он снимать не стал — сам не мог ответить себе почему. Хотелось быть одетым, чтобы побыстрее уйти. Его куртка выглядела как будто гарантия того, что никто не станет задерживать его. Миновав полумрак коридора, он вошел в комнату, увидел хозяина, сидящего спиной к этой двери.

«Хороший знак, — подумал гость. — Вариант идеальный».

Подошел сзади, заглянул через плечо. На экране ноутбука отражалась какая-то сетевая активность. Пальцы хозяина порхали над клавишами, изредка прикасаясь к сенсорному коврику и его кнопкам.

— Чудишь? — спросил гость. — Интернет не дает покоя?

— Это я ему не даю покоя, — ответил хозяин, быстро оглянувшись через плечо и вернувшись к своим командам в консоли. — Работенку подкинули — приходится выкладываться до последнего. А ведь хотелось на футбол сходить, даже билеты предлагали по божеской цене…

— Что на этот раз? — в голосе не было интереса ни на грош, но хозяин не заметил этого.

— Честно говоря, и сам до конца не понимаю. Используют в темную. Правда, друзья. Верю, что никакую гадость делать не заставят.

— Распределенный вариант? Выполняешь часть дела, а о целом не имеешь представления? Хорошая страховка, если заметут. Привязать к твоему куску работы обвинение практически невозможно.

— Ну так и я про что? — весело махнул рукой хозяин. — А ты? Какими судьбами? Помощь нужна или так, потрепаться? Чего не разделся? На пять минут, что ли?

«Да похоже на то», — подумалось гостю.

— Шел вот мимо… Дай, думаю, зайду. А ты, глядика, занят. Может, в другой раз?

— Да я недолго… Минут тридцать еще, — кинув взгляд на часы, ответил хозяин. — Уже почти все сваял. Сейчас пойдут данные, надо проанализировать, а потом я весь совершенно свободен.

Он защелкал клавишами, периодически поглядывая в справочник по командам Ассемблера.

— Прямо на ходу подстраиваешься? — поинтересовался гость, шаря по карманам. — Не нашлось ни одной готовой утилиты, приходится программировать?

— Да больно работа нестандартная. Я, признаться, давно уже на ассемблере не кодил, такая муть поначалу получалась! Но опыт не пропьешь, оказывается. Через час кошмара выстроил первую правильную конструкцию… И она даже заработала.

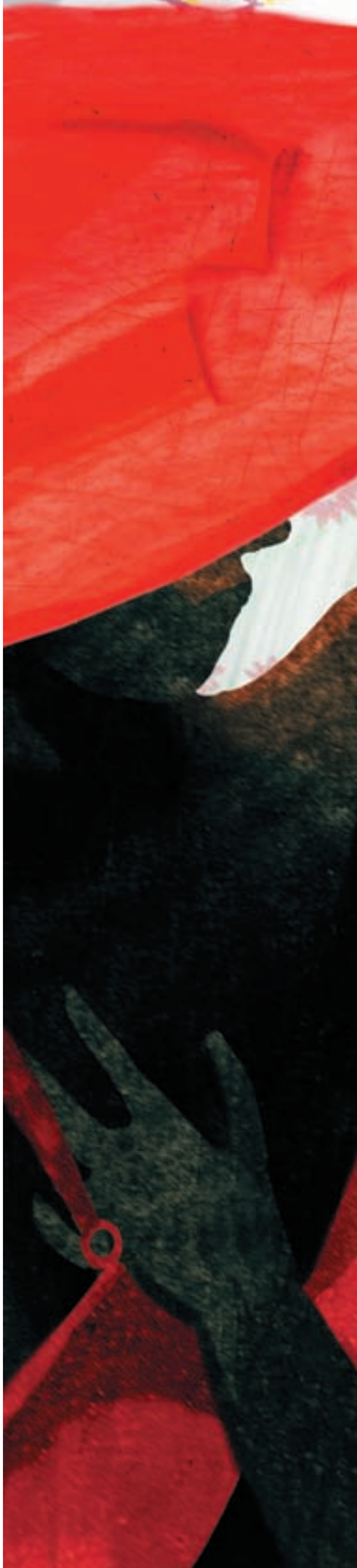
— Ага, — кивал гость, тем временем найдя что-то в кармане. На свет был извлечен шприц, наполненный какой-то мутной золотистой жидкостью. Хозяин не обращал внимания на то, что происходило у него за спиной, полностью доверяя тому, кого впустил в свою квартиру. — Сам люблю нестандартную работу — хоть какой-то простор для фантазии. Привыкли вечно пользоваться чужими наработками…

Он говорил это сжимая в руке шприц и глядя на шею хозяина квартиры, который и представить себе не мог, что творилось сейчас за его спиной. Пальцы подбирались к колпачку, готовясь обнажить иглу. Он внезапно поймал себя на том, что думает о ватке со спиртом: надо протереть кожу, потом уколоть, ватку оставить, чтобы не кровило… Чушь.

— Нео, следуй за белым кроликом… — пробурчал себе под нос хозяин, водя пальцем по строкам справочника. — Надо же, вот как бывает…

— Действительно, — сказал гость, ногтем большого пальца отстрелил колпачок с иглы и, держа шприц как нож, вонзил иглу в шею жертвы, после чего резко надавил на поршень. Человек за компьютером ойкнул от внезапного укола, вжал голову в плечи и попытался отмахнуться, но первые же капли вещества, попав в его тело, сделали свою черную работу: он резко вздохнул от боли, парализо-

ЧЕРЕЗ ЧАС КОШМАРА ВЫСТРОИЛ ПЕРВУЮ ПРАВИЛЬНУЮ КОНСТРУКЦИЮ…



вавшей все его тело, попытался ухватиться руками за шприц, но промахнулся и нелепо упал со стула в сторону, потянув за собой провод мыши.

Уже на полу он сумел взглянуть в глаза того, кто совершил убийство.

— Ты… — только и сумел вымолвить он. Боль разрывала его тело, воздух, которого было так много вокруг, никак не хотел попадать внутрь его легких. Там все жгло, диафрагма пыталась втолкнуть воздух, судорожно сокращаясь, но быстро иссякла… Глаза, светившиеся болью и яростью, постепенно стеклянели, подергивания пальцев прекратились. Хозяин квартиры умер.

Гость вернулся в коридор. Взял оставленную там сумку, вернулся в комнату. На экране ноутбука программа, написанная им, продолжала собирать никому уже не нужную информацию и отсылать ее в неизвестном направлении. Гость перешагнул труп, сел на освободившееся место, отодвинул компьютер немного от себя и вытащил из сумки на его место свой ноутбук, тоньше и на вид дороже. Потом он осмотрел все, что происходило на экране компьютера хозяина, и решительно выключил программу.

«Тот, кому это нужно, обязательно с ним свяжется», — подумал он.

А пока были дела поважнее. Воткнув сетевой кабель в свою машину, он открыл контейнер пейджера и набрал маленькое сообщение.

Спустя несколько секунд начался диалог…

* * * * *

Лунный свет не давал комнате погрузиться во мрак. Желтый круг торчал за шторой и отливал золотом на стены. Стоявший на подоконнике горшок с высоким цветком неизвестного названия и происхождения отбрасывал на стену рядом с диваном причудливую тень, дополняя собой геометрический рисунок на обоях. Артур широко, но тихо зевнул, не потрудившись прикрыть рот рукой.

Он знал, что Алена его не видит.

Она находилась у него между ног, разглядывая совершенно другие места тела своего парня. Упершись обеими руками в диван, она методично поднимала и опускала голову, делая все так, как предпочитал Артур и как он научил ее.

Почему-то ему всегда нравилось именно так — когда Алена не пускала в дело руки. Только губы, только язык… Иногда он откидывал ее длинные волосы в сторону, разглядывая картину, достойную самого качественного немецкого порно. Его девушка была фантастически красива, он гордился этим и чертовски ревновал ее даже к чужим взглядам… К счастью, здесь и сейчас чужих взглядов не было.

Алена двигалась медленно, вверх-вниз, вверх-вниз… Артур погладил ее плечи, она легонько наклонила голову и взглянула на него из-под длинной челки.

— Ты супер, — только и сумел произнести Артур, после чего откинулся на спинку дивана и закрыл глаза. Он знал, что может расслабляться еще примерно минут тридцать. Знал он и то, что Алена не поторопится ни на секунду: все будет на грани, но перейдет она ее только по его команде. Он легонько подтолкнет ее низом живота, она задвигается чуть быстрее…

А потом надо будет поработать.

Рядом с парой, занимающейся любовью, на диване стоял раскрытый ноутбук, по черному экрану которого перемещался логотип Windows Vista. Изредка Артур поглядывал на него, видел сетевой кабель, уходящий вниз, на пол, и чувствовал, как он сейчас соединен со всем миром, как его наслаждение переливается через край и готово ворваться в мир — в компьютеры всех и каждого.

Артур протянул руку в сторону, погладил серебристый корпус, провел пальцем по матовому стеклу сенсорной панели, легко нажал на кнопки. Логотип исчез, открыв рабочий стол с несколькими распахнутыми консольными окнами — в отличие от получающего неземное удовольствие своего хозяина, ноутбук работал вовсю. Несколько сетевых утилит сканировали пространство, которым интересовался Артур и его напарники по Сети.

Алена на секунду застыла, отвлекшись на засветившийся экран. Артур коснулся ее волос, и она продолжила. Вверх-вниз, вверх-вниз…

Она ничего не понимала в компьютерах, но прекрасно разбиралась в сексе. Что в ней нравилось Артуру — так это то, что, занимаясь любовью, она ЗАНИМАЛАСЬ ЛЮБОВЬЮ. Не думала о том, как бы заплатить по счетам, как бы помыть посуду и приготовить ужин — она целовала, гладила, ласкала… И не отвлекалась ни на что.

У Артура было достаточно денег, чтобы все бытовые проблемы решались быстро, можно сказать, на ходу. Он сам так устроил — чтобы на секс уходило как можно больше времени. В какой-то момент жизни, примерно после того как ему исполнилось тридцать, он вдруг почувствовал, что плотских наслаждений в его жизни явно мало. Пустившись во все тяжкие, он обрел любовь десяти или пятнадцати красавиц, польстившихся на его деньги, но остановился на Алене — на женщине, которая оказалась предана именно сексу с ним, а не его кошельку.

Тень на обоях потихоньку перемещалась, а ласкам, казалось, не будет конца. Алена, поймав ритм, превратилась в сексуальную зомби. Она загипнотизировала себя своими движениями… Вверх, вниз, вверх, вниз, потом кончиком языка, потом… А потом… Артур понимал, что она совершенно точно рассчитала его возбуждение: при таком темпе он еще долго не взорвется, но будет прикован всеми своими ощущениями к тому, что творится «на конце иглы».

Иногда она шумно вздыхала, а потом легонько встряхивала волосами — дескать, извини, любимый, но ничего не могу поделать, кислород легким пока еще нужен. После чего, чувствуя, что она не испортила Артуру впечатление от происходящего, она продолжала в прежнем ритме…

На экран выпорхнуло маленькое окошко пейджера с сообщением.

«На месте?»

Артур медленно протянул руку к ноутбуку — так, чтобы даже периферийное зрение Алены не отметило какие-то перемещения рядом с ее головой.

«Да».

«Один?»

«Практически…»

«Конкретнее, Арчи. Все очень серьезно».

Артур нахмурил брови и вдруг понял, что как-то иначе стал чувствовать прикосновения Алены — словно под анестезией, под заморозкой. Будто все ушло так далеко…

«Что случилось?»

«Замели двух человек из моей команды. Правда, у одного из них был чистый винт — умел парень работать. А вот второго обложили по всем правилам искусства контрразведки».

За то время, пока собеседник набирал текст, Артур передумал массу вариантов того, что же там на самом деле произошло, но прочитанное повергло его в шок. Уж если Буги-мэн пишет такие вещи…

«За тобой грех?»

«Грех есть за каждым, Арчи».

«Ты знаешь, Буги, о чем я».

«Знаю. И не очень доверяю современной криптографии, чтобы признаваться в убийстве Иисуса Христа открытым текстом».

Вверх-вниз… Вверх-вниз… И точно так же сердце у Артура — к небу и к земле…

«Ты предупреждаешь меня о чем-то? Или делишься бедой?»

«Предупреждать поздно. Одного твоего тоже взяли час назад».

Алена, вне всякого сомнения, слышала легкие прикосновения Артура к клавишам даже сквозь свое дыхание и его периодические стоны. Но, похоже, не придавала этому значения.

«Откуда сведения? И кто он?»

«Бедняга Билли. Уж он-то чем не угодил правопорядку… Сведения верняк».

«Думаешь, пора валить?»

«Ну, если тебя сейчас за член никто не держит, думаю, это наилучший вариант».

Артур невесело усмехнулся — это совпадение хотя бы придало плохим новостям какую-то комическую окраску. Билли… Парень специализировался по базам данных, взламывал всякого рода секретные хранилища, АТС, городской и областной муниципалитет, ГИБДД, налоговиков. То, что спустя пару дней после его работы люди могли купить на рынке (тысячи дисков со вполне свежей информацией), имело очень высокую цену. Он покорил этот хакерский Эверест еще три года назад, пробравшись в святая святых налоговой полиции. После этого успех сопутствовал ему повсюду. До сегодняшнего дня. Кому-то он все-таки перешел дорогу…

Алена нежно провела кончиками пальцев по животу Артура, словно почувствовав его внезапно возникшее напряжение. Он вздрогнул, но не от сладкого ощущения, а от неожиданности. Она остановилась, но не хотела поднимать глаз. Кончик ее языка, высунувшись изо рта, дрожал. Тонкая ниточка слюны тянулась вниз…

— Ты сегодня не похожа сама на себя, Алена… — прошептал Артур, чтобы сказать хоть что-то. Она поняла, что на какие-то мгновенья он вырвался из ее сексуального плена. Вот только где он успел побывать за это время, сложно было догадаться. — Не устала?

Она отрицательно покачала головой. Ее язык снова пришел в движение, голова опустилась…

Вверх-вниз… вверх-вниз…

«Если по телефону поступает приказание тебе, обязательно перезвони тому, от кого оно пришло, и подтверди факт», — вспомнил он старое армейское правило на случай поднятия тревоги. Заступая дежурным, он четко знал свои обязанности, знал телефонные номера, по которым нужно было звонить

ТЕНЬ НА ОБОЯХ ПОТИХОНЬКУ ПЕРЕМЕЩАЛАСЬ, А ЛАСКАМ, КАЗАЛОСЬ, НЕ БУДЕТ КОНЦА

в случае чего. Вот и сейчас это старое правило, вбитое в его голову добросовестными сержантами, всплыло в памяти само собой. Он мазнул пальцем по сенсорному коврику, раскрыл контакт-лист, нашел Билли, кинул ему набор абракадабры: если он на месте, отзовется, если же нет — пусть другие думают, что это код.

В ответ тишина. Артур уже начал нервничать, но Алена сделал ТАМ что-то такое, от чего мысли о Буги и Билли улетели на второй план. Откуда-то всплыли названия всех этих чудес, читанные-перечитанные в интернете: «Шелковый водоворот», «Колибри» и прочая и прочая. Можно было назвать это реальной отключкой. Сколько времени личность Артура растворялась в ласках Алены, не знал и сам Арчи. Возможно, всего лишь несколько секунд. Может быть, минуту-две. Парень словно вынырнул из сладкого, приторного киселя. В ушах стояла вата, щеки гудели, словно от пощечин. Сердечко билось мелко, как у котенка, очень хотелось глубоко вздохнуть и кинуться целовать Алену…

Она немного разогнулась. Он понял, что она крепко сжимает его внизу цепко, одной рукой. Это потихоньку отрезвляло его, возбуждение слегка отступало.

Алена хитро улыбнулась сквозь упавшую челку и сказала:

— Ну уж нет… Ты мне еще нужен сегодня.

Он протянул руку в надежде прикоснуться к щеке, но она покачала головой, приоткрыла рот и прищелкнула языком. А потом — Артур это точно заметил — посмотрела на экран ноутбука.

— Я буду ревновать тебя к этой железяке, — укоризненно произнесла она. — Хотя нет, я сделаю так, что ты про нее забудешь.

— Это вряд ли, — в шутку подмигнул ей Артур. — Слишком разные интересы…

— Я — твой интерес, — отпустила она свою руку. — Только ты еще не понял этого. Или пока боишься признаться и сказать вслух. Никто и никогда не будет любить тебя так, как я. ВОТ ТАК…

И она снова опустила голову.

И его снова унесло…

А когда спустя пару минут она дала ему передышку, на экране горело:

«Арчи, что за бредятину ты прислал?»

Артур смотрел на то, что написал ему Билли, и в очередной раз ласки Алены переставали волновать его…

«Как зовут мою девушку?»

«Алена».

«Мой старый ник?»

«Ти-Рекс».

Возникла пауза. Билли понимал, что его проверяли, а такие вещи делаются неспроста, поэтому спокойно отвечал в ожидании возможности задать встречный вопрос и получить объяснения.

«8 и 10 — цифры номера моей мобилы?»

«Нули».

«Спрашивай».

«Что случилось?»

«Буги только что похоронил тебя. Сказал, что тебя взяли час назад».

«Ему-то чего врать?»

«Не знаю. Говорит, двое его парней на крючке. Похоже, он пытался дать мне понять, что они и ты были в каком-то деле вместе».

«Я работаю либо один, либо с тобой».

Вверх-вниз… Вверх-вниз…

«Что он хотел?»

«Сложно сказать. Мы всегда грызлись».

«И?..»

«Отправить кого-нибудь за решетку, пусть даже в мыслях, — неплохое развлечение. Я, наверное, смог бы так. Если бы придумал».

Артур приподнял брови. Вариант развлечения не пришел ему в голову. Но и теперь, после того как Билли оказался на месте, сложно было согласиться с тем, что Буги просто пошутил. Во всем должен быть смысл. Люди, сутками сидящие в Сети, готовы шутить с кем угодно и как угодно для снятия стресса, но надо всегда спрашивать себя: «Кому это выгодно?» Слишком суров окружающий мир. За каждым смайликом, за каждой запятой что-то прячется. Стоит расслабиться на секунду, посчитать кого-то своим другом, внести его в «исключения» — и точно, у тебя на компе уже какая-то гадость. Ползает по файлам, точит зубы о реестр, долбится в Security Storage, кричит на всю Сеть твои пароли, причем эта сволочь попала к тебе только из-за твоей доверчивости. И не первый же раз попадает Артур в подобную ситуацию, но вот первый раз кого-то отправляют на полном серьезе чуть ли не на тот свет. Вот и думай, верить или не верить...

Тем временем Билли после небольшой паузы спросил:

«Сейчас занят?»

«В принципе, нет».

Что еще он мог ответить, поглядывая на подрагивающие волосы Алены? Он действительно был совершенно свободен. Сознание, правда, слегка затуманено, но так только кажется. Стоит встряхнуть головой — и ясность ума вернется… Но встряхивать ох как не хочется!

«Раз уж связались — не хочешь помочь?»

«Чего надумал?»

«Шифрование включил?»

«Нет, забыл!»

«Извини, Арчи :)».

«Потом будешь извиняться».

Артур немного подвинулся на диване поближе к ноутбуку. Он уже устал набивать все эти сообщения левой рукой, которая доставала до компьютера на пределе возможностей. Хорошо хотя бы зрение было острое, ответы читались прекрасно…

Алена тут же почувствовала все эти перемещения. Она оторвалась от своего занятия, которым была поглощена даже больше, чем Артур, учитывая его диалоги в Сети. Встретившись со своим парнем глазами, она укоризненно склонила голову набок.

— Тебе неинтересно то, что происходит с тобой? — спросила она, надув губы. — Гораздо важнее кто-то там, в виртуале?

— Ну что ты, дорогая, — почувствовал себя виноватым Артур, протянул руку и погладил ее щеку. — Ни в коем случае. Кто-то кинул информацию, сразу и не разберешь. Поверь, машинально наклонился, чтобы прочитать.

— И давно ты уже так читаешь? — она с большим сомнением взглянула на экран.

— Две секунды, — улыбнулся Артур, привстал, привлек девушку к себе и поцеловал. Она жадно ответила ему, но быстро оторвалась, взмахнув волосами, и спросила:

— Мне стоит продолжать?

— Я даже не понимаю, зачем ты спрашиваешь, — пожал плечами Артур. — Ничего подобного я не испытывал ни разу. Хотелось бы, чтобы все это продолжалось бесконечно долго… Хотя бы до утра, — улыбнулся он лукаво, заглядывая под ее челку.

— Ну-ну, — в ответ улыбнулась Алена и снова нырнула вниз… Спустя пару секунд статус кво был восстановлен… Вверх-вниз… Вверх-вниз…

А на экране уже горело сообщение:

«Есть работенка».

«Что-нибудь стоящее? Или взлом детского сада для получения пароля к ведру с манной кашей?»

«Как сказать?.. Объект моего давнего пристального внимания — одна веселенькая нефтебаза».

«Решил наказать “Газпром” за монополизм?»

«В этой стране больше не на кого наехать. Владеешь энергоносителями — владеешь миром».

«Согласен. Принципиально — есть какие-то наработки? Цель? Средства? Прибыль?»

«Есть все. Вот только не испугаешься ли масштабов?»

«Я — никогда. Напугал бабу толстым членом».

«Да ладно, я тебя хорошо знаю. Но уж очень высоко я решил взлететь».

«Адрес?»

Билли набрал строку с указанием цели предприятия.

Арчи едва не присвистнул, но вовремя вспомнил об Алене. «Вот черт, я даже забыл, что мне минет делают», — поразился он тому, насколько хакерские проблемы отрешают его от мира. Его любимая девушка занималась сейчас делом, которое он всегда считал вершиной сексуального искусств. И надо же, какие-то несколько слов заставляют его забыть об этом на вершине блаженства.

«Все серьезно? Назад дороги нет?»

«Дорога есть всегда. И в этом случае тоже. Мы просто сейчас забываем о нашем разговоре и идем пить пиво. Все очень просто».

«Не надо делать из меня идиота — я этого не люблю».

«Это значит “да”?»

«Это значит “да”».

Возникла пауза. Арчи подумал, Билли не может поверить в то, что ему не отказали, и был по-своему прав. Он всегда знал, что Билли — настоящий авантюрист. Они никогда не встречались в реале, но тех эпизодов общения, когда они делали общее дело, ему хватило, чтобы составить впечатление об этом человеке.

«Черт, мне сегодня везет», — ответил Билли через минуту.

«Да уж. Лучше расскажи, как можно делать деньги в интернете, взламывая сервер нефтебазы».

«Тебе стало интересно?»

«Не дерзи. Молод еще. Конечно, интересно. Ты же мне гонорар не бензином будешь отдавать».

«Точно. А если бы бензином, не взял бы?»

«Машины нет».

«А ты нюхай».

«Короче, Билл. Начинает раздражать».

Алена в этот момент в очередной раз оторвалась от своего дела, кото-

рому предавалась с упоением, выпрямилась, встряхнула волосами и увидела, как ее любимый Арчи шпарит на ноутбуке какие-то послания.

— Ты… Ты можешь хотя бы сейчас прекратить? — едва не зарыдала она, поняв, что ее труды не шли ни в какое сравнение с Сетью. — Я… Я так стараюсь… У меня никогда так ни с кем не было! Ты же меня сам научил, сам рассказал, как ты хочешь!..

— Тихо-тихо, дорогая, — подскочил от ноутбука Артур. — Это все чистая случайность, — поспешил он оправдаться перед Аленой. — Друг написал пару сообщений, я же не буду ему сообщать, что мой член сейчас во рту прекрасной дамы и ее губки и язычок сводят меня с ума!.. Ну, не плачь. Я просто ответил ему и сам не заметил, как завязался диалог! Но ты все равно чудо! Такая страсть, такая нежность! Мечта всей моей жизни — чтобы у меня была такая преданная сексу женщина, как ты! Не обижайся, а лучше поцелуй меня и продолжай… Ну, котенок…

Алена тихонько подвывала, уткнувшись в свои колени. Лунный свет золотил ее грудь, Артур привлек ее к себе, погладил, отметил про себя, как напряглись и стали твердыми ее соски, после чего она тихонько простонала и нырнула к нему между ног. Он вновь откинулся на подушку, не забыв проверить, дотянется ли рука до ноутбука.

В эту минуту он окончательно понял, что никто и ничто не заменит ему Сети. Никакая красивая девочка с пухлыми губками, умеющая все, не в состоянии встать на место компьютера, подключенного ко всему миру. В очередной раз доверив Алене свое тело, он стал разглядывать ее волосы, плечи, талию, ноги в черных чулках… Конечно, во всем этом было море секса, океан эротики. Много слов можно было бы подобрать к картине, нарисованной лунным светом, но еще больше можно было сказать о Сети, которая ждала, жадно впитывая в себя байты сообщений…

«Нефтебаза не из последних», — горело на экране. — «Крупнейшее в европейской части России хранилище».

«Что мы будем там делать?»

«Я сломаю защиту. Ты найдешь способ сымитировать покупку большой партии топлива».

«Чего?»

«Выступишь в роли покупающей организации».

«Зачем? Повторяю, мне бензин не нужен».

«Чтобы потом продать».

«Как можно продать то, чего нет?»

«Ты совсем там с ума сошел?»

«Так объясни».

«Сейчас в мире есть такая страна, которая очень хочет купить нефть, газ и любые другие энергоносители. Страна называется Украина».

«Их же поймали. Они воровали газ, а наши их тормознули».

«Вот!!! Вот именно! Им нужен газ и нужна нефть».

Арчи внезапно почувствовал нарастающее возбуждение внизу живота и отвлекся от диалога. Алена задышала шумно, принялась помогать себе руками. Артур напрягся, собрал в кулаки простыню, и партнерша моментально почувствовала это, ослабила напор. И куда все только делось? Волна схлынула, лоб моментально взмок, покрывшись бисеринками пота. Руки и ноги стали ватными, блаженство, готовое вот-вот вырваться наружу, отступило.

— Не так быстро, — услышал он нежный шепот, кончик языка прошелся по внутренней поверхности бедер, потом был исследован живот. Поцелуи не миновали и ладони. Артур поймал себя на том, что ему хочется стонать и одновременно пытаться отвечать на мысли Билли. Алена продолжала целовать его, не выпуская из рук мужское достоинство. Скоро их губы соприкоснулись, они слились в жадном поцелуе. На мгновенье Арчи все-таки позабыл о Сети — Алена творила с ним чудеса.

— Не хочешь сесть повыше? — внезапно предложила она. — Тебе, наверное, неудобно. А сидя будет и видно лучше. Меня.

Она игриво повела глазами вдоль своего обнаженного тела — и была совершенно права насчет «лучше видно». Посмотреть действительно было на что: выглядела она очень, как бы выразиться поточнее, и модельно, и спортивно одновременно. Тонкая шея, высокая грудь, осиная талия, плоский животик… А еще у нее было идеальное с его точки зрения соотношение «талия — бедра». Глядя на эти линии, он всегда замирал и был готов накинуться на нее в любое время и в любом месте. Самое приятное — она всегда позволяла…

Он вспомнил, как однажды они вдвоем возвращались домой из ночного клуба, было около пяти утра, у него ожидались ночные посиделки — надо было пообщаться с человеком из далекого часового пояса, у которого в это время наступил вечер. Но у Алены оказались другие планы. В подъезде она остановила его, принялась жадно целовать, на ходу расстегивая и его брюки, и свою блузку… Тогда они были знакомы еще не очень долго, не все ее порывы были тогда открыты и изучены, поэтому Артур несколько опешил от такой страсти, но совсем не спасовал, принялся отвечать ей взаимными ласками, и тогда она показала ему многое из того, на что оказалась способна… Когда все закончилось, ему пришлось нести ее до квартиры на руках: настолько она была вымотана, столько сил отдала, что была лишь в состоянии стонать, пытаясь открыть усталые веки… Он раздел ее, уложил в постель… Она проспала тогда пятнадцать часов, а проснувшись, повторила все сначала, только на этот раз дома, а не в подъезде.

Вот тогда Артур понял, какое счастье досталось ему. Алена была сексмашиной, которая умела работать в разных режимах. Как стиральная машина. «Деликатно», «Быстро», «Грубо», «Мягко» — лишь небольшой перечень и диапазон ее способностей. Нужно было только попросить: «Сегодня так, а завтра так». Ее саму устраивало все, она находила прелести в любом варианте, превращаясь то в тигрицу, то в зайку, предлагая себя всю. Артур пользовался этим на сто два процента.

Они были идеальной парой.

Так думал Артур…

«АУ», — горело на экране. Арчи довольно надолго отвлекся от диалога, разглядывая прелести своей любимой девушки.

«Я все еще здесь».

«Так как? Будем работать? Или ты там предаешься смертным грехам?»

«Не то слово. Но работать будем. Когда начинаем?»

«Да я уже начал. Вот тебе несколько адресов…»

Арчи смотрел на ряд цифр, которые прислал ему Билли, и в нем просыпался азарт. И никакая Алена не могла сейчас ему помешать.

«Что там?»

«Первые два IP — основной сервер и его копия. Молодцы ребята, делают бэкапы профессионально. Размер потрясает воображение!»

«Не юродствуй, ты сам когда последний раз сохранялся?»

«В “Жажде скорости” как-то, помню, сохранялки выписал на болванку. Уж больно далеко прошел, жалко было профиль. И знаешь, только сделал бэкап — играть разонравилось. Не вижу прямой связи, но так уж вышло. Бросил на полпути к славе».

«Дальше. Без этих вот подробностей».

«Третий IP — комп предположительно стоит в финансовой службе. Самый, так сказать, денежный. Вот туда я и собрался зайти. А ты впаришь им сделку с несуществующей организацией, кинем им пару миллионов виртуальных долларов, дальше полученную нефть быстро отправим за рубеж братьям-хохлам и пойдем спать».

«Братья-хохлы в курсе?»

«Сидят, ладони потирают. Ждут, когда же на них черный дождь прольется».

«И ты думаешь, у меня уже есть готовое решение для создания конторы, которая может позволить себе потратить два миллиона долларов на покупку нефти?»

«Почему два? Можно больше. И не говори, что у тебя такого нет. Не поверю. Зная тебя столько лет… Не поверю».

«Правильно не верит, гад», — ухмыльнулся Арчи, поглаживая нежную кожу на плече Алены. Пальцы сами скользнули к груди, она тоже немного подалась вперед, чтобы он мог дотянуться до сосков. — «Есть у меня такая наработка, специально готовил. Не то чтобы знал, что пригодится… Но каким-то местом чувствовал».

У него на винчестере лежала домашняя заготовка. Фирма, созданная из воздуха, наполненная воздухом и торгующая воздухом. Фирма со своей историей, персоналом, реквизитами, товаром, складами, базами, поставщиками и посредниками… Любой, кому хотелось бы проверить, насколько правдиво все то, что изложено в резюме фирмы, неминуемо погряз бы в бесконечных перекрестных ссылках, подтверждающих все и вся и одновременно никуда не ведущих.

И вот Арчи лежал на диване, поглаживая бархатную грудь Алены, прислушиваясь к прикосновениям ее волшебного язычка, и решал, отдать эту фирму для дела Билли или не стоит. Потом понял, что нужно просто спросить, и все станет на свои места.

«Какой у меня процент?»

«Половина. Без вариантов».

Артур задумался. Ему казалось, что та часть работы, которую делает он — учитывая, что его фирма, единожды засвеченная, больше не пригодится — значительно больше, чем простое проникновение напарника. Пусть даже он сделает так, что ребята на базе будут уверены, что к ним стучится человек, облеченный полным доверием, знающий все пароли и отзывы и имеющий право на совершение сделок.

Меньше всего хотелось торговаться во время минета. Алена расслабляла, отвлекала, умиротворяла. Не хотелось спорить, доказывать, ругаться. Он прикрыл глаза, подумал несколько секунд и написал:

«Согласен».

«Наконец-то. Я уже подумывал, куда бы еще пойти со своим тортиком».

«Каким тортиком?»

«Не заморачивайся, это еврейский анекдот, не более чем. Итак, я свою

работу уже потихоньку ваяю. Скоро получу пароли на ведение переговоров. Что-то типа сертификата, которым у них подписываются доверенные лица. Как только эти цифровые ключи оказываются у меня в руках, я отсылаю их тебе, а ты уж… Со всем присущим тебе темпераментом. Я ведь почему к тебе обратился? Потому что ты не только можешь виртуальную байду изобретать, ты с ними еще и поговоришь так, что они тебе все сами отдадут. Талант у тебя».

Артур посмотрел на Алену и подумал: «Вот у кого талант… Где она раньше была? Как я раньше жил?»

«Сколько у меня есть времени на подготовку?»

«Час. Вряд ли больше. Если я правильно понял, наработки имеются?»

«Без комментариев. Задача поставлена. Жду твои данные. Как я буду с ними общаться — не твоя забота. Час — это просто прекрасно. Утрясу кое-какие дела».

«Ну, тряси-тряси. Только будь рядом с компом, когда я тебе все пришлю. А то пропадаешь куда-то…»

Артур решительно прекратил общение и сосредоточился на предстоящей работе.

Вверх-вниз… Вверх-вниз…

Алена явно мешала, но у Арчи не было сил отказаться от происходящего. Слишком сладкими были эти минуты обладания красивой женщиной, настолько сладкими, что он уже подумал о том, что не сможет оторвать ее от себя и не сможет оторваться сам. Тепло разливалось по всему телу от низа живота. Он и сам не желал, чтобы Алена останавливалась, а тем более заканчивала этот процесс ради работы. Где-то внутри стала созревать крамольная мысль о том, что вот он, первый звоночек: хватит порхать в виртуале, стоит обзавестись женой, семьей, валяться вот так на диване после трудового дня, ловить скупой мужской кайф от обладания длинноногой блондинкой, преданной тебе и обученной тобой.

«Сколько мы уже вместе? — спросил он сам себя. — Вроде помнил… То ли день, то ли целую вечность. Никогда не понимал, что мне нужно от женщин. Скорее всего, полное подчинение. Не на уровне садизма — на уровне полного взаимопонимания. Чтобы одного взгляда было достаточно. Чтобы я только бровью повел, а она уже в постели… Чего-то у меня все мысли на секс съезжают, какая-то подозрительная озабоченность».

Артур смотрел на двигающуюся Алену и думал, думал… Когда-то в институте его друг Роман рассказал ему о подобной ситуации в своей жизни — больше всего Арчи потрясло описание того, как во время таких любовных утех Рома брал в рот сигарету и потихоньку курил, затягиваясь вполсилы и выпуская дым в потолок. В те времена Артур боготворил женщин (но был далек от них — был непопулярен), поэтому такое неуважение к женскому полу, такое равнодушие вызывало в нем волну протеста. Теперь же он сам отстукивал по аське огромные сообщения, которые можно было приравнять не к сигарете, а к целой пачке — и ничего, был только благодарен своей Алене за то, что она относилась к этому достаточно терпимо, хотя время от времени и сопротивлялась.

«Была бы сигарета, закурил бы», — согласился сам с собой Артур. Только никогда не курил. Как-то пробовал — не понравилось. Сейчас смот-

рит на цены в киосках и радуется тому, какие деньги экономит в течение месяца. Да, сейчас явно не получилось бы курить. Вот телевизор…

«Футбол бы посмотреть, — подумалось Артуру. — Вообще спорт. Хоть какой. Интересно, обиделась бы или нет? Вообще странные они, эти женщины. Ведь понимает, что живем за те деньги, которые я зарабатываю в виртуале. Реал и рядом не валялся. Значит, нечего мне мешать, нечего дуться. Не будет денег — может и рвануть туда, где потеплее. А жаль… Не хотелось бы отпускать…»

Артур понял, что его ощущения вышли сейчас на некий постоянный уровень, за которым ничего нет — лучше ему уже не станет при подобном темпе и ласках. С одной стороны, это успокоило его. Алена еще долго будет продолжать, еще долго можно рассматривать ее красивое во всех отношениях тело, можно временами прикрывать глаза и прислушиваться к прикосновениям ее языка и губ. С другой, он немного волновался, хватит ли ему времени на подготовку. Придется оторваться от этого божественного занятия, Алена тут же устроит сцену, может, помешает. Не хотелось бы быть с ней резким, даже злым, но ведь может создать реальные помехи. Как бы это помягче сделать…

Он кинул взгляд на компьютер, отметил про себя, что уже прошло около пятнадцати минут в рассуждениях о смысле жизни и качестве секса. Пора, пора…

— Котенок… — тихо позвал Артур. Алена отреагировала своеобразно: не глядя на Арчи, медленно протянула руку и погладила его. Она явно была сейчас против общения. Артур отметил, что вторая ее рука бродит где-то там, откуда у нее самой растут ноги.

«Вот ведь как, — поднял брови Арчи. — Аленушка-то не промах…»

— Ты не устала? — спросил он. В ответ отрицательное покачивание головой.

— Может, стоит сделать перерыв? Тем более, для этого есть очень важная причина.

— Какая? — услышал он наконец. Ее рот был не занят.

— Пришло время работать. Там, где сидят мои друзья, наступил день. Они готовы к тому, чтобы вместе со мной провернуть небольшое дело. И после него мы будем достаточно богаты для того, чтобы не вылезать из постели очень, очень долго.

— Опять? — откинулась назад Алена. — Опять работа? Ты вообще в курсе, чем я занимаюсь? — она махнула рукой на его восставшую мужественность.

— Это что, ничего не значит?!

Арчи еще не понял, плачет она или кричит, но уже осознал, что разговор пойдет на уровне, ему недоступном. Она никогда не поймет того, что она сейчас сказала и еще скажет.

— Ну почему же не значит? — попытался он вставить слово, но слезы Алены, градом покатившиеся из ее глаз, доказали его правоту. Понять Алену было невозможно, впрочем, как и успокоить.

Артур вдруг понял, что для того чтобы она вновь стала прежней, страстной, любящей, ей снова нужно дать ее любимую игрушку, любимую кожаную соску. И все.

«Да она помешана на сексе!» — поразился Арчи, позабыв о том, как сам несколько минут назад только об этом и размышлял.

— Котенок, ну это несерьезно! — он сел, притянул ее к себе. Плечо тут же стало мокрым от слез. — Это ненадолго. Час, не больше. А потом все сначала, как будто ничего и не было.

— Час? — она недоверчиво подняла на него большие детские глаза Лолиты.

— Точно? Ты не обманываешь?

— Ну как я могу обманывать? — он погладил ее по голове, по шелковым волосам, провел по груди, она всхлипнула и прижалась теснее. — Дурочка, у меня же все рассчитано. Вот сейчас запущу одну программку, она мне все подобьет, прокалькулирует. Если тебе будет угодно такими терминами общаться, бухгалтер ты мой начинающий…

— Вот мы только мое училище не вспоминали! — буркнула Алена, но голос явно повеселел. — А ты мне покажешь, что будешь делать?

— А ты поймешь?

— А мне все равно. Для меня твоя работа — просто большой прикол. Сидишь, чего-то на экране выстраиваешь, как паук. Сеть плетешь. Ох уж эти хакеры!.. Смотрела я как-то кино про вас. Ничего не поняла, но главное — как они все это делают?

Она смотрела на Арчи влюбленными глазами, хлопала ресницами (как в хите Братьев Грим), и он таял, таял… Она действительно была грандиозной сексуальной дурочкой с большими чувственными губами, проникновенным эротичным взглядом, пластикой кошки — девушка мечты. И он был готов показать ей все от начала и до конца.

«Как ее еще приняли в училище, — спросил он сам себя. — Там ведь без компа никуда».

Алена тем временем вытерла слезы, накинула на плечи рубашку Артура, отчего стала еще сексуальнее, подобралась к своему мужчине поудобнее, пристроилась возле ноутбука и сказала:

— Я готова, любимый.

«Ох уж эта терминология, — мысленно покачал головой Арчи. — Не хотелось бы мне сразу про любовь толковать. Девчонки все одинаковы: один раз скажешь, потом всю жизнь отмазываться».

— Ну, готова так готова, — не стал он отвечать тем же Алене, пусть пока не тешит себя иллюзиями. Хотя, конечно, отдать такую порнозвезду в чужие руки он уже вряд ли смог бы. Он пока не хотел признаваться себе, но зависимость от Алены в нем уже существовала.

Он взял ноутбук и поставил его себе на колени. Алена приткнулась сбоку, создавая определенные неудобства, но он не спешил отталкивать ее. Едва она прикоснулась к нему, как в голову рванули эротические фантазии. Как на грех, ничего похожего на нефтяной бизнес. С трудом он отогнал наваждение, отвел глаза от груди Алены, словно ненароком выглянувшей из-под рубашки, и сконцентрировался на работе.

К подобному случаю он был готов уже давно — сам не помнил, почему идея создать подставную фирму посетила его, но сейчас он лишний раз отметил свою прозорливость. Все было сделано абсолютно точно: фирму можно был отслеживать по многим каналам. Она прошла и через налоговую, и через отделы сертификации и лицензирования, в ней существовал довольно большой персонал, регулярно получающий зарплату, ездящий в командировки и уходящий в отпуск строго по графику. Вот только в действительности она не существовала, и ни одного живого человека, ни один адрес нельзя было отследить физически.

Арчи попытался объяснить это Алене и увидел на ее лице отражение непонимания. «Как такое можно было придумать? Как воплотить в жизнь? И как воспользоваться с максимальным эффектом?» Эти вопросы явственно читались у нее на лице.

«Дура, зато красивая», — подумал Артур и продолжил. Билли должен был сейчас создать ему «зеленую улицу» — сделать так, чтобы фирма Арчи из неизвестной и таинственно-криминальной превратилась в едва ли не самого лучшего партнера нефтяных магнатов. Хакер искал доступ в святая святых — в контакт-лист финансового отдела — и готовил путь для вторжения.

«Я готов. Жду реквизиты», — появилось на экране. Арчи отправил все, что от него требовалось, после чего подождал немного и проверил. Его несуществующая компания значилась в контакт-листе на четвертом месте сразу за сверхворотилами финансового мира. С такой «легендой» можно было заказывать музыку…

— Что ты будешь делать? — спросила Алена, прижимаясь к Арчи еще крепче.

— Покупать нефть, — коротко сказал он, прикидывая, с какой стороны зайти к этим крутым ребятам. — Покупать светлое будущее. Хочешь красиво жить, вкусно есть, сладко спать?

— Я тебя хочу, — заглянула она Артуру в глаза, распахивая на груди рубашку. — Ты еще долго?

— Я же сказал, в общей сложности уйдет около часа, — отмахнулся Арчи. — Сейчас будем втираться в доверие, потом торговаться, чуть позже быстро сольем эту нефть друзьям в Украине — и, пожалуйста, пользуйся полученным гонораром. Но в нашем деле, Алена, надо всегда помнить принцип, по которому ловят слишком зарвавшихся: любое преступление оставляет финансовый след. И как бы тщательно я ни продумал пути к отступлению, всегда можно будет найти тот кусок сала, который я положил на свой кусок хлеба после незаконных операций в виртуале.

— То есть ты запросто можешь попасться? — Алена выгнула брови дугой.

— Слово «запросто», котенок, ко мне неприменимо. Но попасться можно, если кто-то постарается.

Он наклонился к ней, поцеловал грудь (и тут она попыталась привлечь его к себе, прижавшись теснее, но он оказался сильнее, отстранился, погрозил пальцем). Пришлось уступить. Алена несколько разочарованно отодвинулась, при этом не выпуская из поля зрения экран ноутбука.

Артур вернулся к своему делу. Он еще раз проверил, насколько его фирма неуязвима, затем вступил в переговоры с финансовой группой. Похоже, человек по ту сторону Сети не особо вдавался в подробности. Арчи подумал, что он просто взглянул на список доверенных лиц, отметил там гостя, обладающего всеми полномочиями, после чего они с Артуром принялись рассматривать условия заключения сделки.

— Арчи, а как ты будешь платить? За нефть? — неожиданно спросила Алена. Парень даже вздрогнул — настолько он был поглощен процессом покупки, что вопрос выбил его из колеи.

— Алена, все еще впереди, — быстро ответил он. — Денег у меня, конечно же, нет. Или ты считаешь, я в состоянии купить такой объем бензина, чтобы заправить целый парк небольшой авиакомпании? Когда дойдет до способа оплаты, у меня есть кое-какие хитрости. Не хотелось бы раскрывать карты даже тебе, хотя ты особо и не поймешь.

Продавец изучал его кредитоспособность вдоль и поперек. Артур отмечал это по запросам, поступающим к нему на компьютер, то есть на сервер несуществующей фирмы. Генерировались нужные ответы, выводились необходимые отчеты о финансовой состоятельности Артура, давались какие-то гарантии и рекомендации…

Арчи представил себе, сколько сделок подобного рода заключается в мире ежедневно, и ужаснулся. Оборот несуществующих финансов должен

был просто зашкаливать выше всех разумных пределов — люди, подобные ему, при хорошей подготовке в состоянии покупать даже ядерные технологии, урановую руду, золотой запас ведущих держав мира и много чего еще. Был бы спрос, а предложение всегда найдется. Он вспомнил одного своего коллегу по сетевому бизнесу, который сумел купить подобным образом совершенно невообразимую партию сверхдорогого лекарства от гепатита — сумма была просто гигантская! Подставная фирма была сгенерирована точно таким же образом. Не подкопаешься, все проверено миллион раз. Ты в одном шаге от бонуса! Тот парень сумел провернуть сложнейшую махинацию в одиночку, пробившись в доверенные лица фармацевтической компании, вступив в роли одного из ведущих посредников, и положил себе в карман (Артур даже не знал точно) порядка десяти миллионов долларов. Позже цена на это лекарство взлетела (фирма, столкнувшись с серьезными денежными проблемами, поступала так вынужденно), многие страны отказывались покупать его, их программы помощи больным гепатитом летели в пропасть, люди умирали… Но ему уже тогда было все равно, он умел вовремя останавливаться — и с большой кучей денег он исчез, чтобы всплыть где-нибудь в Новой Зеландии или Аргентине…

— Будем надеяться, что бензина хватит всем, — вслух произнес Артур, вспомнив всю эту историю. — Не думаю, что цена поползет вверх. Она и так растет в среднем три раза в год, и ничего — люди не перестают ездить на машинах. А контора вернет свои деньги, выполнив несколько махинаций с октановым числом, разбавив бензин какой-нибудь дрянью…

— Ты о чем? — спросила Алена. — Разговариваешь сам с собой… Ты уже совсем тронулся на этом хакерстве! Тебе не кажется, что пора завязывать?

— Ты говоришь так, — ответил Арчи, набирая на клавиатуре текст, — будто мы с тобой знакомы уже не один год и я только и делаю, что торчу в интернете.

— Мы познакомились десять месяцев назад, если ты не помнишь. И это уже достаточный срок! — надула губы Алена.

«Ага, десять месяцев, — отметил про себя Артур. — Надо запомнить».

— И ты обещал рассказывать, что ты делаешь, — добавила она. — Думаешь, блондинки ничего не понимают в компьютерах? Еще как понимают, только виду не показывают!

— Я верю, верю, — закивал Артур. — Никто ведь и не говорит, что ты не понимаешь. Просто тут все быстро происходит, я просто не успеваю…

— Врешь ты все…

— Да вот тебе крест! — Артур оторвался от клавиатуры, перекрестился, поправил на Алене рубашку, которая уж очень откровенно распахнулась, после чего вернулся к своей работе.

— Вот сейчас мы договариваемся о сумме сделки, — прокомментировал он свои действия. — Реквизиты моей фирмы и гарантии от олигархов, которые даже не подозревают о том, что подписались только что под несуществующим проектом, сделали свое дело и мне поверили окончательно. Фирма готова совершить сделку со мной. Они предлагают бензин, я — деньги. Политэкономию читала? Это еще Энгельс придумал в девятнадцатом веке.

Артур был явно увлечен своим делом. Закусив губу, он быстро просчитывал в Excel'е суммы, проценты, налоги и прочую лабуду, без которой нельзя было совершить финансовую операцию. Его подставная фирма имела не менее подставной счет в одном европейском банке. На этом счету лежало пять долларов: весь капитал не превышал этой суммы никогда с момента открытия счета, проценты были просто смешные, и чтобы пять долларов превратились хотя бы в шесть, нужен был не один год. Но важно было не это, были важны банковские реквизиты.

Имя банка значило в данной ситуации все: поверят банкиру, поверят и тебе. Артур постарался, когда выбирал хранилище для своей пятерки баксов. Перечитал кучу литературы, массу банковских бюллетеней, выбрал наиболее подходящий банк, и теперь его имя играло на руку Артуру. Путем довольно сложных манипуляций Арчи превратил пять настоящих долларов в миллионный счет, предоставил финансовой группе нефтяной компании все доказательства существования этих денег и уже был готов совершить операцию купли-продажи, не забыв выторговать небольшую скидку — оптовая партия была достаточно большой…

— Скажи, а тебя это возбуждает? — внезапно спросила Алена. — Ты понимаешь, о чем я?

— Не очень, — с неохотой оторвался от своих манипуляций Артур.

— Ну вот смотри, если я буду ходить по магазинам нижнего белья, то стану обращать внимание на те экземпляры, которые нравятся не только мне, как женщине, но и которые будут нравиться тебе, как моему мужчине, — завернула Алена. — И при этом, для того чтобы понять, что будет нравиться тебе, я буду представлять в этом белье себя, представлять, как будет сидеть на мне бюстгальтер, как будут обтягивать ноги чулки, и это будет меня возбуждать. Возбуждать, потому что я полностью провалюсь туда, в мир секса и эротики. И это притом, что я всего лишь хожу по магазинам.

— Не понимаю, какая связь с хакерством? — недоуменно спросил Артур. — Ты-то понятно. Я сам уже от твоих разговоров возбуждаюсь (Алена широко

ЕДВА ОНА ПРИКОСНУЛАСЬ К НЕМУ, КАК В ГОЛОВУ РВАНУЛИ ЭРОТИЧЕСКИЕ ФАНТАЗИИ

улыбнулась и опустила взгляд вниз, чтобы убедиться в правдивости его слов).

— Возбуждаюсь, можешь не проверять. В отношении белья ты превосходишь всех моделей мира. Знаешь, ты это дело и любишь. Но объясни, как я должен возбуждаться от созерцания командной строки, от чтения логов сетевых сканеров, от умело написанного кода на ассемблере. И ты вообще поняла хоть слово из того, что я сказал?

— Не все так плохо, — хитро ответила Алена. — Я понимаю, что блондинка — это приговор. Но не для всех. Я иногда бываю в интернете, читаю новости, просматриваю форумы, и слова, подобные этим, уже попадались мне на глаза. Мое мнение, ты просто обязан испытать что-то подобное оргазму от единения с Сетью… Если, конечно, ты входишь в полный контакт…

«Чего она говорит? — мысленно пожал плечами Артур. — Я за десять месяцев не слышал от нее ничего умнее комментариев на «Дом 2», а тут такие выражения, некое подобие мыслительного процесса… Здорово, конечно, но может привыкнуть. А функция у нее другая — сексуальная. Хотя не принижаю ли я ее, постоянно отправляя целовать себя все ниже и ниже?.. Оргазм? Она знает это слово? Мне казалось, что ее уровень не выше средней школы, что-то вроде терминологии тинейджеров. Хотя, если подумать, в ее словах есть разумное зерно. Возбудиться от факта проникновения в чужую систему, наверное, прикольно. Только есть одно «но»: вдруг подсядешь, придется всякий раз возбуждаться для достижения результата. Наркоманией попахивает…»

— Ты чего так надолго замолчал? — спросила Алена. По его лицу она явно читала весь этот мыслительный процесс, может, и не угадывая его дословно, но схватывая направленность. — Удивлен моими речами? Может, я целый год ждала этого дня, чтобы явиться тебе во всей своей красе. А то привыкнешь к тому, что видишь только мой затылок и спину. Нет, минет, конечно, дело хорошее, но, как в анекдоте, «А поговорить?».

— Поговорить, конечно, можно, — согласился Артур. — Говори сколько влезет. Я с тобой полностью согласен. Должен быть какой-то эмоциональный подъем от выполнения любимой работы, да еще если эта работа несет заведомо положительный результат в смысле вознаграждения за нее.

Он на мгновение задумался, скользнул взглядом по экрану и добавил:

— А ты ведь права… То есть не то чтобы я возбуждаюсь, нет. Но ощущение непередаваемое…

— А хочешь, оно станет вообще фантастическим? — загадочно спросила Алена. — И ты захочешь испытывать его снова и снова. Глядишь, и дела в гору пойдут.

— Да я вроде не на мели, — попытался возразить Артур, но Алена его перебила:

— Давай, покупай свой бензин… А я тут сделаю тебе кое-что… Берегла для такого случая.

— Что ты имеешь в виду? — насторожился для виду Арчи. — Кусочек садомазохизма? Или позовем соседа?

— Дурак ты, — отмахнулась Алена, снимая с себя рубашку и обнажая тело. — Как я тебе, нравлюсь?

— Еще бы, — кивнул Артур, глядя на сверкающие подковки каблуков туфель Алены. — Я вспоминаю, как жил раньше, и мне становится жутко — я был лишен таких вещей…

— Теперь они будут всегда. Потому что я буду всегда, — склоняясь перед Артуром, шепнула Алена. Спустя пару секунд он почувствовал ее нежные прикосновения… Вверх-вниз, вверх-вниз… Вроде бы ничего особенного она не делала, но волна возбуждения наплывала сильнее, чем обычно. Он повернулся к ноутбуку и продолжил выполнение операции.

И через несколько секунд он понял, что имела в виду Алена. Тепло, разливающееся от низа живота, достигало каждой клеточки тела, он набивал данные в формы в такт ее движениям. Покачивая головой, он был готов начать считать вслух что-то вроде «и-раз, и-два…». Глаза прикрылись сами собой, превратившись в щелочки. Сквозь них он, как сквозь туман, видел цифры и буквы, которые впечатывал в необходимые поля. Алена все ускоряла и ускоряла темп…

— Бери все, — внезапно услышал он шепот девушки. Она оторвалась на мгновение от своего фантастического занятия, подняла голову и сказала это Артуру, находящемуся где-то между небом и землей. И он, практически не понимая того, что делает, внес в форму данные о покупке всего бензина на базе, благо запас призрачных денег в банке был бесконечен…

Щелчок клавиши «Ввод» совпал с взрывом. Тело содрогнулось в экстазе, Артур застонал, падая на спину. Алена не давала ему упасть полностью, глотая все до последней капли. Ее жадные движения, которые Артур успел заметить, довели его просто до сумасшествия, он вздрагивал от каждого при-

косновения девушки и понимал, что такого оргазма не испытывал никогда в жизни.

Она нависла над ним, тряхнув волосами, и спросила:

— Сделал? Ты сумел?

«Да», — хотел ответить Артур, и не смог. Силы оставили его, и, судя по всему, надолго. Он просто кивнул еле-еле, одними веками.

— Взял все?

На ее губах Артур видел что-то белое, сверкающее в лучах луны, бьющих прямо в окно. Она увидела, что он смотрит, облизала губы и прищурилась:

— Ты сладкий… Так ты взял? Взял все?

Он снова кивнул.

— Тогда прощай, — улыбалась она. Ее рука скользнула куда-то вниз, к маленькому треугольничку лобка, она слегка прогнула спину… И в ее руке появился маленький шприц-тюбик.

В ПОСЛЕДНИЕ СЕКУНДЫ СВОЕЙ ЖИЗНИ АРТУР ДУМАЛ ТОЛЬКО О ТОМ, ОТКУДА ОНА ЕГО ДОСТАЛА…

Откинувшись на спинку дивана, Алена смотрела на Артура, неподвижно лежащего на спине и глядящего мертвыми глазами в потолок.

— Прости, Арчи, — шепнула она ему. — Все-таки десять месяцев — это большой срок. Я почти привыкла к тебе. Ты, наверное, превозносил мои сексуальные достоинства до небес, а я ведь самая простая девчонка, которая просто хочет и любит трахаться. И это не мешает ей быть той, кто она есть. Холодная расчетливая сука — вот кто я. Просто иногда можно в жизни добиться всего при помощи компа, а иногда — собственным телом… Ничего личного, Арчи. Ничего личного.

Она подползла к ноутбуку и набрала в пейджере сообщение:

«Он взял все. Буги, мы богаты. Надеюсь, ты убрал Билли так же красиво, как я — Артура?»

«Ал, это ты? Все уже случилось?»

«Да. Он мертв. Мы богаты».

«Сказочно. Я уже отправил все, что было куплено, по нужному адресу. Деньги поступили на настоящие счета в шести банках мира. Отследить их будет очень трудно. Когда встретимся?»

«Не сегодня…»

Она кинула взгляд на труп Артура, коснулась кончиком пальца его гладко выбритой щеки и сказала, зная, что Буги-мэн ее, конечно же, не слышит:

— Сегодня у меня…

Она хотела что-то еще добавить, но рыдания сдавили ей горло. Она схватила себя за шею, не давая крику вырваться наружу, посидела так с минуту, потом накрыла тело Арчи его рубашкой, закрыла крышку ноутбука и пошла одеваться. Больше всего она боялась привыкнуть возбуждаться от убийства…

НОВАЯ ИГРА
“ФУТБОЛЬНЫЙ МЕНЕДЖЕР”!

СОЗДАЙ СВОЮ КОМАНДУ ИЗ РЕАЛЬНЫХ ИГРОКОВ И ПРИВЕДИ ЕЕ К ПОБЕДЕ

ТЫ ПОЛУЧАЕШЬ $135 МИЛЛИОНОВ

на приобретение игроков российской премьер-лиги при регистрации на сайте www.total-football.ru.
Игра стартует с первым туром чемпионата российской премьер-лиги и финиширует матчами 30-го тура.
Твоя команда должна состоять из 11 основных игроков, 4-х запасных и главного тренера. Количество замен в команде не ограничено. Стоимость команды на весь сезон - $4,99.

Подробности на сайте **www.total-football.ru**

Играть можно с помощью мобильного телефона на wap.total-football.ru

ПРИЗЫ

По итогам месяца (апрель, май, июль, август, сентябрь, октябрь, ноябрь) приз получает лучшая команда данного периода. Также поощряется лучшая команда **по итогам каждого тура** чемпионата российской премьер-лиги. Даже не очень удачный старт не лишает вас шансов на успех!

ПРИЗЫ

ГЛАВНЫЙ ПРИЗ – ПОЕЗДКА НА ФИНАЛ ЛИГИ ЧЕМПИОНОВ 2006/07

EXCILAND computers

intel Xeon inside™

Повысьте эффективность работы и ускорьте развитие своей компании

Универсальный сервер Major, на базе процессора Intel® Xeon® поможет Вам повыситьэффективность труда сотрудников и в более полной мере удовлетворять желания и потребности клиентов .

Гарантия - 3 года
Бесплатная доставка по Москве
Вся продукция сертифицирована (РОСС RU. ME61.B01302)

Подробная информация на сайте: www.exciland.ru
и по телефону: (495) 727-0231

Заказ серверов:

КОРПОРАТИВНЫЙ ОТДЕЛ:
(495) 727-0231; e-mail: b2b@exciland.ru

e-mail:info@exciland.ru www.exciland.ru e-mail:info@exciland.ru www.exciland.ru e-mail:info@exciland.ru www.exciland.ru e-mail:

Обозначения Celeron, Celeron Inside, Centrino, Centrino logo, Intel, Intel Core, Intel logo, Intel Inside, Intel Inside logo, Intel SpeedStep, Intel Viiv, Intel Xeon, Itanium, Itanium Inside, Core Inside, Pentium и Pentium Inside являются товарными знаками, либо зарегистрированными товарными знаками, права на которые принадлежат корпорации Intel или ее подразделениям на территории США и других стран.